Optimisation non convexe en finance et en gestion de production : modèles et méthodes

Duc Quynh Tran

THÈSE

en vue de l'obtention du titre de

DOCTEUR DE L'UNIVESITÉ DE PAUL VERLAINE-METZ

(arrêté ministériel du 30 October 1993)

Spécialité Informatique

présentée par

Tran Duc Quynh

Titre de la thèse :

Optimisation non convexe en finance et en gestion de production : Modèles et Méthodes

Date de soutenance : le 14 Octobre 2011

Composition du Jury :

Président	PHAM DINH Tao	*Professeur, INSA-Rouen*
Rapporteurs	El-Houssaine AGHEZZAF	*Professeur, Université de Gand, Belgique*
	Van-Dat CUNG	*Professeur, INP de Grenoble*
Examinateurs	Alain BILLIONNET	*Professeur, ENSIIE-Paris*
	Abdel LISSER	*Professeur, Université de Paris Sud*
	ADJALLAH Kondo Hloindo	*Professeur, ENIM-Metz*
Directeur de thèse	LE THI Hoai An	*Professeur, Université de Paul Verlaine-Metz*

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

Remerciements

La préparation de cette thèse, sous la direction de Madame le Professeur LE THI Hoai An - Mathématicienne renommée en Optimisation DC, a été réalisée au sein du laboratoire LITA de l'Université de Paul-Verlaine Metz que je tiens vivement à remercier tous ceux qui m'ont accordé cette ambiance de travail.

Je remercie en premier lieu Madame le Professeur LE THI Hoai An, ma directrice de thèse, Directrice du laboratoire LITA, pour son aide inestimable, ses précieux conseils ainsi que les encouragements qu'elle m'a donnés durant la préparation de la thèse.

Je tiens ensuite à remercier tout spécialement Monsieur le Professeur PHAM DINH Tao - Mathématicien de renommée internationale, pionier de l'Optimisation DC et créateur de DCA (DC Algorithm), Directeur de l'équipe Modélisation et Optimisation Appliquée de l'INSA de Rouen pour ses conseils et son attention constante, sa sympathie et les discussions très intéressantes qu'il a menées pour me suggérer les voies de recherche.

Je souhaite également exprimer ma gratitude à Monsieur Van-Dat CUNG, Professeur à INP de Grenoble et Monsieur El-Houssaine AGHEZZAF, Professeur à l'Université de Gand, Belgique de m'avoir fait l'honneur d'accepter la charge du rapporteur de ma thèse.

Je tiens aussi à remercier Monsieur Abdel LISSER, Professeur à Université de Paris Sud, Monsieur Alain BILLIONNET, Professeur à ENSIIE-Paris et Monsieur ADJALLAH Kondo Hloindo, Professeur à ENIM-Metz pour avoir participé à juger mon travail.

Je n'oublie pas de remercier toute l'équipe du personnel de l'Université Agronomique de Hanoi pour m'avoir donné de soutien. Je tiens à remercier particulièrement Monsieur NGUYEN Hai Thanh pour son soutien et son encouragement.

Je voudrais exprimer ma gratitude à tous mes collègues et mes amis français, vietnamiens, algériens, iraniens rencontrés à Metz pour les moments agréables lors de mon séjour en France. Je remercie particulièrement Minh, Phuong, Thuan, Phuc, Son, Linh pour le partage dans le travail et dans la vie.

Je témoigne tout mon affection et reconnaissance à mes parents, mes soeurs, ma copine pour avoir pu supporter mes longs moments d'absence. Je leur remercie aussi pour m'avoir donné de soutien inconditionnel ainsi que pour m'avoir transmis l'énergie malgré la distance géographique.

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 Outils de base **19**

1 Introduction à la programmation DC et DCA **21**

1.1 Éléments de base de l'analyse DC 22

1.1.1 Notations et propriétés 22

1.1.2 Fonctions convexes polyédrales 24

1.1.3 Fonctions DC 25

1.2 Optimisation DC 26

1.2.1 Dualité DC 27

1.2.2 Optimalité globale en optimisation DC 28

1.2.3 Optimalité locale en optimisation DC 29

1.3 DCA 31

1.3.1 Principle de DCA 31

1.3.2 Existence des suites générées 32

1.3.3 Calcul des sous-gradients 33

1.3.4 Optimisation DC polyédrale 34

1.3.5 Interprétations de DCA 35

1.4 Pénalité exacte en Programmation DC 36

2 Méthode par Séparation et Evaluation (SE) **39**

2.1 Méthode de résolution et convergence 40

2.2 Réalisation 43

2.2.1 Stratégie de division 43

2.2.2 Règle de sélection 47

2.2.3 Estimation de borne 48
2.3 SE combinée avec DCA 49

II Optimisation en Gestion Financière 51

3 Résolution du problème min max continu en gestion de portefeuille en présence des contraintes de cardinalité 57

3.1 Introduction 57
3.2 Description et formulation 58
3.3 Programmation DC et DCA pour la résolution du problème 61
3.3.1 Reformulation 61
3.3.2 Résolution de (P_4) par DCA 65
3.4 Résultats numériques 66
3.5 Conclusion 70

4 Résolution d'une classe des problèmes d'optimisation à deux niveaux et une application en gestion de portefeuille 71

4.1 Introduction 71
4.2 Résolution du problème (4.1) 73
4.2.1 Méthode de résolution par DCA 73
4.2.2 Algorithme combiné de DCA et SE (SE-DCA) 76
4.3 Résultats numériques 77
4.4 Application en gestion de portefeuille 80
4.5 Conclusion 82

III Optimisation en Gestion de Production 89

5 Minimisation du coût de maintenance comprenant le temps de séjour et la pénalité du retard 95

5.1 Introduction 95
5.2 Description du problème 97
5.3 Formulation mathématique 98

5.4 La résolution basée sur la programmation DC et DCA 101
5.4.1 DCA appliqué au problème (P_1) 101
5.4.2 DCA appliqué au problème (P_2) 103
5.5 Les résultats numériques 104
5.6 Conclusion 107

6 Minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement 109
6.1 Introduction 109
6.2 Modèle mathématique 111
6.3 Résolution basée sur la programmation DC et DCA 113
6.3.1 Résolution de (P_u) par DCA 113
6.3.2 Algorithme combiné de DCA et SE 116
6.4 Résultats numériques 119
6.5 Conclusion 123

7 Détermination des prix de transfert et des politiques de stockage pour une chaîne d'approvisionnement de deux entreprises 133
7.1 Introduction 133
7.2 Formulation mathématique 136
7.3 Méthode de résolution basée sur la programmation DC et DCA 139
7.3.1 Reformulation DC et l'algorithme DCA 139
7.3.2 Algorithme combiné DCA et SE pour le problème (P) 141
7.4 Résultats numériques 144
7.5 Conclusion 146

Table des figures

2.1 Exemple de la frontière efficiente. 56

3.1 Les valeurs de la fonction objectif dans le cas où α=0.5. 67
3.2 Les valeurs de la fonction objectif dans le cas où $Card = 20$ 68
3.3 La frontière efficiente ($Card = 20$) 69

4.1 Gap de tous le trois algorithmes dans le cas 1 79
4.2 Gap de tous le trois algorithmes dans le cas 2 79
4.3 Gap de tous le trois algorithmes dans le cas 3 80

5.1 Le schéma de la relation entre la date d'exécution, la date limite, la date de terminaison 99
5.2 Résultats dans le cas où l'horizon H=60 105
5.3 Résultats dans le cas où l'horizon H=90 105
5.4 La comparaison entre (P_1) et (P_2) 106
5.5 Les coûts moyens dans tous les trois cas 107
5.6 les couts moyens relatifs dans tous les trois cas 108
5.7 Les proportions d'utilisation d'entité dans tous les trois cas 108

6.1 Comparaison entre SEDCA, SE et COUENNE dans le cas où $n = 50$. 123
6.2 Comparaison entre SEDCA, SE et COUENNE dans le cas où $n = 100$. . . . 124

7.1 Cas 1 : I=5, K=10, T=12 145
7.2 Cas 2 : I=10, K=10, T=12 145
7.3 Cas 3 : I=15, K=10, R=6, T=12 146
7.4 Cas 4 : I=20, K=10, R=6, T=12 146

7.5 Cas 5 : I=30, K=10, R=6, T=12 147

Liste des tableaux

3.1 Résultats obtenus dans le cas où α=0.5, le symbole * signifie que CPLEX a échoué à résoudre un sous-problème généré dans le schéma de "Branch and Cut" 67

3.2 Résultats obtenus dans le cas où $Card = 20$, le symbole * signifie que CPLEX a échoué à résoudre un sous-problème généré dans le schéma de "Branch and Cut" 68

4.1 Résultat dans le cas où n=50, m=100 (cas 1) 83

4.2 Résultats dans le cas où n=100, m=150 (cas 2) 84

4.3 Résultats dans le cas où n=150, m=150 (cas 3) 85

4.4 Résultats avec les données de France 86

4.5 Résultats avec les données d'Angleterre 86

4.6 Résultats avec les données des Etats-Unis 87

4.7 Résultats avec les données de Luxembourg 87

6.1 Les notations utilisées 112

6.2 Les variables utilisées 112

6.3 Les résultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 5$ 120

6.4 Les résultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 10$ 121

6.5 Les résultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 20$ 121

6.6 Les réesultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 50$ 122

6.7 Les résultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 100$ 122

6.8 Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 5$ 125

6.9 Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 10$ 126

6.10 Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 20$ 127

6.11 Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 50$ 128

6.12 Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 100$ 129

6.13 Les résultats de SEDCA, SE et COUENNE au moment où SEDCA s'arrête dans le cas $n = 20$ 130

6.14 Les résultats de SEDCA, SE et COUENNE au moment où SEDCA s'arrête dans le cas $n = 50$ 130

6.15 Les résultats de SEDCA, SE et COUENNE au moment où SEDCA s'arrête dans le cas $n = 100$ 131

7.1 Résultats dans le cas où $I = 5, K = 10, T = 12$. 148

7.2 Résultats dans le cas où $I = 10, K = 10, T = 12$. 149

7.3 Résultats dans le cas où $I = 15, K = 10, R = 6, T = 12$. 150

7.4 Résultats dans le cas où $I = 20, K = 10, R = 6, T = 12$. 151

7.5 Résultats dans le cas où $I = 30, K = 10, R = 6, T = 12$. 152

Liste des Publications et Conférences

Tran Duc Quynh

Article avec comité de lecture

- Le Thi Hoai An, Tran Duc Quynh, *Solving continuous min max problem for single period portfolio selection with discrete constraints by DCA*, to appear in Optimization.

- Le Thi Hoai An, Tran Duc Quynh, Pham Dinh Tao, *A DC programming approach for a class of bilevel programming problems and application in portfolio selection*, accepted to appear in Numerical Algebra, Control and Optimization (NACO) with minor revisions.

- Tran Duc Quynh, Le Thi Hoai An, Kondo Hloindo Adjallah, *"DCA for minimizing the cost and tardiness of preventive maintenance tasks under real-time allocation constraint*, LNCS, Volume 5991, pp 410-419.

- Tran Duc Quynh, Le Thi Hoai An, *A fast and scalable algorithm for a multi-stage manufacturing problem*, in the proceeding of the conference IESM 2011, 10 pages.

- Le Thi Hoai An, Tran Duc Quynh, *Transfer prices for two-enterprise supply chain optimization by DCA,* in the proceeding of the internet conference IPROMS 2010, 15-26 November 2010, 6 pages.

- Le Thi Hoai An, Tran Duc Quynh, Kondo Hloindo Adjallah, *Minimization of preventive maintenance cost with unequal release dates and tardiness penalties, under real-time and resource constraints, using DCA*, submitted to Applied Mathematics & Computation.

- Le Thi Hoai An, Pham Dinh Tao, Tran Duc Quynh, *Optimizing a multi-stage production/inventory system with bottleneck by DC programming based approaches*, submitted to IEEE-SMC.

- Le Thi Hoai An, Tran Duc Quynh, *A DC Programming approach for multi-period*

problem of fair transfer prices and inventory holding policies in two-enterprise supply chains, submitted to Computer & Operation Research.

Communications aux colloques internationaux avec actes publiés

- Duc Quynh Tran, Hoai An Le Thi, *DCA for solving continuous min max problem for single period portfolio selection,* invited session on Novel opportunities of DC programming and DCA for Industry and Finance, 23rd European Conference on Operational Research, Bonn, July 5 - 8, 2009.

- Tran Duc Quynh, Le Thi Hoai An, *DCA for solving the problem of fair transfer price and inventory holding policies in two-enterprise supply chains,* International Conference on Computational Management Science. Vienna, Austria. July 28-30, 2010."

- Tran Duc Quynh, Le Thi Hoai An, Pham Dinh Tao *DC programming approach for a class of bilevel programming problems and application in portfolio selection*, The 8th International Conference on Optimization : Techniques and Applications (ICOTA8), December 10-13, 2010 Shanghai, China.

Introduction générale

Contexte général et problématique

L'optimisation est une branche importante des mathématiques. En général, elle consiste à déterminer les meilleurs éléments d'un ensemble au sens d'un critère donné. On peut rencontrer des problèmes d'optimisation dans divers domaines, par exemple l'économie, la gestion de portefeuille, la théorie de jeux, la gestion de production, la télécommunication,... Pour résoudre ces problème, l'optimisation offre un cadre algorithmique très riche. On peut distinguer deux branches de l'optimisation déterministe : 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 des contraintes convexes. Lorsque la double convexité chez l'objectif et les contraintes n'est pas vérifiée, on est en face 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 on peut résoudre tout programme convexe, mais encore faut-il bien étudier la formulation du programme convexe en question - la reformulation constitue d'ailleurs un thème de recherche d'actualité - et bien adaptée, aux structures spécifiques des problèmes traités, pour proposer des variantes performantes peu coûteuses et donc capables d'atteindre des dimensions réelles très importantes. 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 l'on dispose 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 conduits à une extension logique et naturelle de la non convexité et 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 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. On peut distinguer deux approches différentes mais complémentaires en programmation non convexe :

i)Les approches globales combinatoires qui sont basées sur les techniques combinatoires de la Recherche Opérationnelle. Elles consistent à localiser les solutions optimales à l'aide des méthodes d'approximation, des techniques de coupe, des méthodes de décomposition, de séparation et évaluation. Elles ont connu de très nombreux développements importants au cours de ces dernières années à travers les travaux de H. Tuy (reconnu comme le pionnier) ([3]), R. Horst, P. Pardalos ([12, 14]). L'inconvénient majeur des méthodes globales est leur lourdeur (encombrement en places mémoires) et leur coût trop important. Par conséquent, elles ne sont pas applicables aux problèmes d'optimisation non convexes réels qui sont souvent de très grande dimension.

ii) Les approches locales et globales d'analyse convexe qui sont basées sur l'analyse et l'optimisation convexe. Ici la programmation DC (Différence de deux fonctions Convexes) et DCA (DC Algorithmes) jouent le rôle central car la plupart des problèmes d'optimisation non convexe sont formulés/reformulés sous la forme DC. Sur le plan algorithmique, l'essentiel repose sur les algorithmes de l'optimisation DC (DCA) introduits par Pham Dinh Tao en 1985 et développés intensivement à travers de nombreux travaux communs de Le Thi Hoai An et Pham Dinh Tao depuis 1993 pour devenir maintenant classiques et de plus en plus utilisés par des chercheurs et praticiens de par le monde, dans différents domaines des sciences appliquées (voir [17]-[50], [58]-[63] et [77]-[80]).

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.

Un programme DC est de la forme

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

où $g, h \in \Gamma_0(\mathbb{R}^n)$, le cône convexe de toutes les fonctions convexes semi-continues inférieurement et propres sur $\mathbb{R}^n$. Une telle fonction f est appelée fonction DC et g et h des composantes DC de f. La programmation DC est une extension de la Programmation Convexe : cette extension est assez large pour couvrir la quasi-totalité des programmes non convexes dit réalistes mais pas trop pour pouvoir utiliser l'arsenal puissant de la Programmation Convexe. DCA est une approche locale qui travaille avec les deux fonctions convexes (dont la différence est la fonction objectif elle-même du programme DC) et non avec cette fonction objectif. 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), 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 la méthode de résolution des problèmes d'optimisation non convexes en deux domaines principaux : gestion financière, plus précisément gestion de portefeuille et gestion de production. Ces deux domaines ont une relation cohérente : toutes les activités de gestion de portefeuille et de gestion de production ont pour objectif de maximiser la performance d'une entreprise.

En gestion de portefeuille, le but d'investisseurs est de maximiser le rendement avec un niveau maximal fixé de risque ou de minimiser le risque avec un rendement minimal fixé. En 1952, H. Markowitz a proposé le premier modèle mathématique dans lequel il a utilisé la variance pour mesurer le risque et la moyenne pour mesurer le rendement. Grâce à cette 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, nombreux chercheurs et praticiens s'intéressent à l'analyse de moyenne-variance, beaucoup d'extensions du modèle MV ont été étudiées. Pour adapter à la réalité, les chercheurs ont ajouté les termes non convexes comme la fonction non convexe de coût de transaction ou des variables binaires/entières. En outre, il existe des modèles pour analyser le pire des cas dans lesquels l'on considère multi scénarios de risques et de rendement au lieu d'un seul vecteur de rendement et une seule matrice de covariance. Ce sont des modèles très importants dans le contexte où l'investisseur veut se protéger contre le risque. Malheureusement, ces problèmes se ramènent souvent à un problème d'optimisation avec des contraintes de complémentarité/des variables mixtes qui sont aussi non convexes et donc difficiles à résoudre.

Dans le domaine de gestion de production, on rencontre souvent les problèmes en variables mixtes et la fonction objectif est non linéaire/ non convexe. En outre, beaucoup de problèmes n'ont pas encore un modèle déterministe et les méthodes de résolution sont heuristiques. Ces méthodes heuristiques ne nous permettent de trouver qu'une solution réalisable mais il manque souvent des outils pour évaluer la qualité des solutions obtenues. Par conséquent, le développement des modèles déterministes et des approches efficaces pour des problèmes en gestion de production est un enjeu important.

Dans cette thèse, nous développons les approches basées sur la programmation DC et DCA pour certains problèmes d'optimisation en gestion financière et en gestion de production. Du point de vue mathématique, les problèmes étudiés sont classifiés dans différentes catégories :
– La programmation linéaire/quadratique convexe en variables mixtes.
– La programmation non linéaire/non convexe en variables mixtes.

– La programmation à deux niveaux.
La plupart des problèmes considérés sont combinatoires (variables 0-1 et/ou entières). L'épine dorsale de notre méthodologie est de reformuler ces problèmes dans le cadre continu pour pouvoir utiliser DCA. L'avantage de l'approche DCA est sa capacité de traiter une classe très large des problèmes qui couvre presque tous les problèmes d'optimisation en réalité. En outre, DCA peut résoudre des problèmes d'optimisation difficiles avec la fonction objectif non convexe, non différentiable. De plus, pour ces problèmes, la relaxation DC est prometteuse et la combinaison DCA avec l'algorithme par séparation et évaluation (SE) nous permet d'avoir une résolution globale. Nous pouvons reformuler des problèmes d'optimisation en variables mixtes, des problèmes d'optimisation à deux niveaux sous la forme d'une programmation DC. En appliquant DCA, nous pouvons résoudre des problèmes d'optimisation de grande taille en gestion de production ou des problèmes d'optimisation à deux niveaux/en variables mixtes difficiles en gestion de portefeuille.

Notre travail en Programmation DC et DCA pour la modélisation, la conception et la réalisation des DCA bien adapté 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.
- Mise en oeuvre des schémas de DCA correspondant.
- Combinaison de DCA et SE pour avoir une résolution globale ou pour prouver la globalité des solutions obtenues par DCA.
- Implémentations et simulations numériques comparatives.

Plus précisément, les modèles traités et les contributions au cours de la thèse sont :
- *Résolution du problème min max continu en gestion de portefeuille en présence des contraintes de cardinalité :* c'est une extension de modèle MV classique pour analyser le pire des cas. Le modèle min max continu a été présenté par N.Gulpinar et al. [82] mais ils n'ont pas considéré la contrainte de cardinalité, une contrainte importante en gestion de portefeuille. Dans nos travaux, nous considérons le problème dans le cas de présence des contraintes de cardinalité. La présence de ces contraintes rend le problème plus adapté à la réalité mais plus difficile à résoudre en raison des variables mixtes. Pour la résolution, nous l'avons reformulé sous la forme d'un programme DC et développé DCA pour le résoudre. Nous avons également présenté un problème d'optimisation quadratique convexe en variables mixtes qui est équivalent au problème original. Les résultats donnés par DCA ont été comparé avec ceux de CPLEX en résolvant le problème quadratique convexe en variables mixtes.
- *Résolution d'une classe des problèmes d'optimisation à deux niveaux et une application en gestion de portefeuille :* les problèmes d'optimisation à deux niveaux ont beaucoup d'applications en économie, en gestion financière et en théorie de jeux. Ils sont non convexes même quand toutes les fonctions objectifs et toutes les contraintes sont linéaires. En outre, ils sont prouvés NP-difficile dans nombreuse articles [141, 142, 143]. Concernant à notre travail, nous présentons une approche basée sur la programmation DC et DCA pour une

classe des problèmes d'optimisation à deux niveaux où la fonction objectif du premier niveau est quadratique convexe, la fonction objectif du second niveau et toutes les contrainte sont linéaires. La comparaison sur les jeux de donné de grande dimension ont également été effectué. Finalement, nous présentons une application en sélection de portefeuille. Les résultats numériques sur les données réelles qui sont les prix des actifs sur le marché de la France, Luxembourg, Angleterre, Etats-Unis ont aussi été rapportés.

- *Minimisation du coût de maintenance comprenant le temps de séjour et la pénalité du retard :* c'est un problème de distribution des tâches de maintenance aux réparateurs dans l'objectif de minimiser le total du temps de séjour et la pénalité de retard. Il n'y a eu aucun modèle déterministe pour ce problème avant notre travail. Une nouvelle méthode de résolution existante est heuristique, basée sur la règle de temps de séjour (FTR) ("flow-time rule" en anglais). En discrétisant l'horizon de temps, nous avons modélisé le problème comme un programme linéaire en variables mixtes, ensuite reformulé le problème sous la forme d'une programmation DC et utilisé DCA pour le résoudre. DCA peut résoudre un problème avec 9000 variables binaire et environs 18000 variables continues. Les résultats numériques fournis par DCA ont également été comparés avec ceux donnés par FTR.
- *Minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement :* il s'agit d'un problème d'optimisation non convexe en variables mixtes entières. La fonction objectif non convexe rend le problème très difficile au point de vue de l'optimisation déterministe. Une méthode existante est heuristique mais l'on ne sait pas la qualité des solutions obtenues car il n'y a pas d'outil pour évaluer la borne inférieure. Dans notre travail, nous avons structuré le problème et développé un algorithme DCA explicite. L'algorithme obtenu est donc très rapide. En outre, nous avons proposé un problème relaxé convexe en basant sur la relaxation DC pour calculer la borne inférieure. Une méthode par séparation et évaluation (SE) et un algorithme combiné SE et DCA ont été développés. La comparaison sur les résultats numériques donnés par tous les trois algorithmes et COUENNE, un logiciel pour la programmation non convexes, a également été effectuée.
- *Détermination des prix de transfert et les politiques de stockage pour une chaîne d'approvisionnement de deux entreprises :* on considère une chaîne d'approvisionnement de deux entreprises. L'entreprise A fabrique et livre les produits à l'entreprise B et l'entreprise B les distribue au client. En réalité, le prix de transfert de chaque produit est sélectionné parmi des niveaux fixés par le siège social. L'objectif est de déterminer le prix de transfert de chaque produit, la politique de stockage/transport pour maximiser le bénéfice total de A et B en assurant que la distribution de bénéfice entre A et B est équilibrée. Ici, nous nous intéressons à l'approche basée sur la maximisation de la fonction de négociation Nash qui est prouvé meilleure que les autres. En 2002, Jonatan a modélisé le problème considéré comme un programme non linéaire/non convexe en variables mixtes (PNLM). Dans notre travail, nous avons étudié l'algorithme DCA pour le problème PNLM considéré. Un problème relaxé convexe a également été proposé dans le but de construire un schéma SE. Après, nous avons combiné DCA avec la méthode SE où DCA est utilisé pour trouver une bonne solution réalisable.

L'approches basées sur la programmation DC et DCA présentés dans cette thèse nous

montrent que DCA peut être appliquée dans nombreuse classes des problèmes réels. En outre, nous pouvons combiner DCA avec des méthodes globales, surtout SE pour obtenir un algorithme global efficace.

Organisation de la thèse

La thèse est divisée en trois parties et composée de sept chapitres

i) Dans la première partie intitulée "Outils de base" nous présentons des outils théoriques et algorithmiques servant des références aux autres. Après une présentation non exhaustive mais complète de la programmation DC et DCA dans le Chapitre 1, nous explorons la technique de Séparation et Evaluation dans le Chapitre 2.

ii) La seconde partie concerne les problèmes d'optimisation non convexe en gestion de portefeuille. Elle comprend 2 chapitres. Le Chapitre 3 est consacré à la résolution du problème min max continu en gestion de portefeuille et en présence des contraintes de cardinalité. La résolution d'une classe des problèmes d'optimisation à deux niveaux et une application en sélection de portefeuille est présentée dans le Chapitre 4.

iii) La troisième partie est intitulée "Optimisation non convexe en gestion de production". Elle est composée de trois chapitres. Dans le Chapitre 5, nous étudions le problème de minimisation du coût de maintenance comprenant le temps de séjour et la pénalité du retard. Le Chapitre 6 est consacré à la minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement. Finalement, la détermination des prix de transfert et les politiques de stockage pour une chaîne d'approvisionnement de deux entreprises est développée dans le dernier chapitre.

Première partie

Outils de base

Chapitre 1

Introduction à la programmation DC et DCA

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

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

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

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

Le pionnier de cette approche est H. Tuy dont le premier travail remonte à 1964. Ses travaux sont abondants, citons les livres de Horst-Tuy ([3, 4]) qui présentent la théorie, les algorithmes et les applications de l'optimisation globale. Viennent ensuite les principales

contributions de l'Ecole Américaine (P. M. Pardalos, J. B. Rosen,...), Allemande (R. Horst, N.V. Thoai,...), Française (Le Thi Hoai An, Pham Dinh Tao,...) et l'École Vietnamienne (Phan Thien Thach, Le Dung Muu, ...).

La seconde approche repose sur l'arsenal puissant d'analyse et d'optimisation convexe. Le premier travail, dû à Pham Dinh Tao (1975), concerne le calcul des normes matricielles (problème fondamental en analyse numérique) qui est un problème de maximisation d'une fonction convexe sur un convexe. Le travail de Toland (1978) ([75]) sur la dualité et l'optimalité locale en optimisation DC généralise de manière élégante les résultats établis par Pham en maximisation convexe. La théorie de l'optimisation DC est ensuite développée notamment par Pham Dinh Tao, J. B. Hiriart Urruty, Jean - Paul Penot, Phan Thien Thach, Le Thi Hoai An. Sur le plan algorithmique dans le cadre de la seconde approche, on dispose actuellement que des DCA (DC Algorithms) introduits par Pham Dinh Tao (1986), qui sont basés sur les conditions d'optimalité et de dualité en optimisation DC. Mais il a fallu attendre les travaux communs de Le Thi Hoai An et Pham Dinh Tao (voir [17]-[50] et [58]-[63]) pour qu'il s'impose définitivement en optimisation non convexe comme étant un des algorithmes les plus simples et performants, capable de traiter des problèmes de grande taille.

Nous reportons dans ce chapitre les principaux résultats relatifs à la programmation DC et DCA qui nous seront les plus utiles pour nos travaux. Ces résultats sont extraits de ceux présentés dans H. A. Le Thi 1994 ([17]), H. A. Le Thi 1997 ([18]). Pour une étude détaillée nous nous référons à ces deux références (voir également [17]-[50] et [58]-[63]).

1.1 Éléments de base de l'analyse DC

1.1.1 Notations et propriétés

Ce paragraphe est consacré à un rapide rappel d'analyse convexe pour faciliter la lecture de certains passages. Pour plus de détails, on pourra se référer aux ouvrages de P.J. Laurent ([15]), de R.T. Rockafellar ([64]) et d'A. Auslender ([69]), J.B.H. Urruty et al.([72]). 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 $+\infty - (+\infty) = +\infty$ ([64]). Etant donnée une fonction $f : S \longrightarrow \overline{\mathbb{R}}$ définie sur un ensemble S convexe de X, on appelle domaine effectif de f l'ensemble

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

et épigraphe de f

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.1.2 Fonctions convexes polyédrales

Une partie convexe non vide 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 *([64])*
- *Soit f une fonction convexe polyédrale. f est partout finie si et seulement si $C = X$,*
- *Si f est polyédrale alors $f^\star$ l'est aussi. De plus si f est partout finie alors*

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

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

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

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

1.1.3 Fonctions DC

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

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

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

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

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

Ces résultats se généralisent aux cas des fonctions à valeur dans $\mathbb{R} \cup \{+\infty\}$ ([18]). Il en résulte que l'ensemble des fonctions DC sur Ω est un espace vectoriel ($DC(\Omega)$) : c'est le plus petit espace vectoriel contenant l'ensemble des fonctions convexes sur $\Omega(Co(\Omega))$.

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

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

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

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

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

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

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

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

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

1.2 Optimisation DC

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

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

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

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

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

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

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

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

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

peut souligner que, comme on l'a vu en haut, la classe des fonctions DC est remarquablement stable par rapport aux opérations fréquemment utilisées en optimisation. En outre, on dispose d'une élégante théorie de la dualité ([55, 56, 75, 70, 17, 18, 29]) qui, comme en optimisation convexe, a de profondes répercussions pratiques sur les méthodes numériques.

Les algorithmes de l'optimisation DC (DCA) dus à Pham Dinh Tao ([57, 58]) 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 ([55, 57]). Cependant, il a fallu attendre les travaux communs de Le Thi et Pham au cours de ces dix dernières années (voir [17]-[50] et [58]-[63]) pour que la programmation DC les DCA deviennent 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 ([64]). 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 [29]).

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

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

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

et le problème dual

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

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

Ce résultat de dualité DC défini à l'aide des fonctions conjuguées donne une importante relation en optimisation DC ([75]).

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

(i)

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

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

Preuve :
(i)

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

(ii) cf. Toland ([75]).

□

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 ([71]) 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 [18] est plus simple et convient bien au cadre de l'optimisation DC : elle exprime tout simplement que cette condition d'optimalité globale est une traduction géométrique de l'égalité des valeurs optimales dans les programmes DC primal et dual.

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

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

Remarque 1.3 –

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

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

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

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

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

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

1.2.3 Optimalité locale en optimisation DC

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

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

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

où V_x désigne un voisinage de x.

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

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

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

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

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

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

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

Remarquons que pour un certain nombre de problème DC et en particulier pour h polyédrale, la condition nécessaire (1.9) est également suffisante, comme nous le verrons un peu plus loin. On dit que $x^\bullet$ est un point critique (ou un point KKT généralisé) de $g-h$ si $\partial h(x^\bullet) \cap \partial g(x^\bullet)$ est non vide ([75]). 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 ([18, 29]))*** *Si $x^\star$ admet un voisinage V tel que*

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

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

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

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

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

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

Preuve : Ce résultat généralise le premier obtenu par C. Michelot dans le cas où $g, h \in \Gamma_0(X)$ sont finies partout et h convexe polyédrale (cf. ([18, 29])). □

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 ([18, 29]))*** *Supposons que $x^\bullet \in dom(\partial h)$ soit un minimum local de $g-h$, soient $y^\bullet \in \partial h(x^\bullet)$ et $V_{x^\bullet}$ un voisinage de $x^\bullet$ tel que $g(x) - h(x) \geq g(x^\bullet) - h(x^\bullet), \quad \forall x \in V_{x^\bullet} \cap dom(g)$. Si*

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

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

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

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

1.3 DCA

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

1.3.1 Principle de DCA

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

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

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

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

Schéma de DCA simplifié

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

On construit ainsi :

Algorithme 1. [DCA]

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

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

1.3.2 Existence des suites générées

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

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

Lemme 1.1 *([29]) Les suites $\{x^k\}$, $\{y^k\}$ dans DCA sont bien définies si et seulement si*

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

La convergence de l'algorithme est assurée par les résultats suivants ([29]) :

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

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

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

(ii)

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

où $dx^k = x^{k+1} - x^k$

Corollaire 1.5 *([29])**(Convergence)***
1.

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

2.

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

3.

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

4.

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

Corollaire 1.6 *([29]) Si les égalités ont lieu, il vient :*
1. $(g-h)(x^{k+1}) = (h^\star - g^\star)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1})$
2. $(g-h)(x^{k+1}) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^\star(y^k), \quad y^k \in \partial h(x^{k+1})$
3. $(h^\star - g^\star)(y^k) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^\star(y^k)$
4. $(h^\star - g^\star)(y^{k+1}) = (h^\star - g^\star)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1}), \quad x^{k+1} \in \partial g^\star(y^{k+1})$

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

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

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

1.3.3 Calcul des sous-gradients

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

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

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

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

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

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

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

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

Et similairement, par dualité

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

1.3.4 Optimisation DC polyédrale

L'optimisation DC polyédrale survient lorsque l'une des composantes convexes g ou h est convexe polyédrale. A l'instar des problèmes d'optimisation convexe polyédrale, cette classe de problèmes d'optimisation DC se rencontre fréquemment en pratique et possède d'intéressantes propriétés. Nous allons voir que la description de DCA y est particulièrement simple ([17, 18, 29]).

Soit le programme DC

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

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

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

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

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

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

1.3.5 Interprétations de DCA

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

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

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

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

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

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

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

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

qui coïncide en x^k avec f,

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

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

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

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

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

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

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

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

Pour une étude complète de la programmation DC et DCA, se reporter aux [17]-[50] et [58]-[63] et références incluses. Le traitement d'un programme non convexe par une approche DC et DCA devrait donc comporter deux tâches : la recherche d'une décomposition DC adéquate et celle d'un bon point initial. Pour un programme DC donné, la question de décomposition DC optimale reste ouverte, en pratique on cherche des décompositions DC bien adaptées à la structure spécifiques du programme DC étudié pour lesquelles les suites $\{x^k\}$ et $\{y^k\}$ sont faciles à calculer, si possible explicites pour que les DCA correspondants soient moins coûteux en temps et par conséquent capables de supporter de très grandes dimensions.

1.4 Pénalité exacte en Programmation DC

Les techniques de pénalité sont très souvent utilisées pour reformuler un problème d'optimisation DC sous contraintes anti-convexes en un problème d'optimisation DC sous contraintes convexes.

Considérons le problème d'optimisation

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

où K est un ensemble convexe non vide borné dans $\mathbb{R}^n$ et $p(x) \geq 0\ \forall x \in K$, f est une fonction DC, finie sur K. Le problème (P) est un problème d'optimisation non convexe. La

non convexité de (P) est due au fait que la fonction $f(x)$ est DC et que la contrainte $p(x) \leq 0$ est non convexe.

Etant un paramètre $\eta > 0$, on définit le problème de pénalité par

$$(P_\eta) \quad \alpha_\eta = \min\{f(x) + \eta p(x) : x \in K\}.$$

On obtient le théorème suivant :

Théorème 1.6 *[147] Soit une suite $\{\eta_k\}$ telle que $\eta_{k+1} > \eta_k$ et $\eta_k \to +\infty$. Suppose que x_k soit une solution optimale du problème (P_{η_k}). Alors tous les points limites de la suite $\{x_k\}$ est une solution optimale du problème (P).*

Preuve : La démonstration est donnée dans [147]. □

Ce théorème nous montre que la solution du problème de pénalité (P_η) est très proche d'une solution du problème (P) quand η est suffisamment grand. Dans certains cas, il existe un nombre η pour que le problème (P) et le problème de pénalité (P_η) aient le même ensemble des solutions optimales.

Théorème 1.7 *[45] Soient K un polyèdre convexe borné non vide, f une fonction DC finie sur K et p une fonction finie concave non négative sur K. Il existe $\eta_0 \geq 0$ tel que si $\eta > \eta_0$ alors deux problème ci-dessous ont la même valeur optimale et le même ensemble des solutions optimales.*

$$(P_\eta) \qquad \alpha(\eta) = \min\big\{f(x) + \eta p(x) : x \in K\big\},$$
$$(P) \qquad \alpha = \min\big\{f(x) : x \in K, p(x) \leq 0\big\}.$$

Preuve : La démonstration est donnée dans [45]. □

Le Théorème 1.7 nous permet de reformuler un programme non linéaire/non convexe en variables mixtes en un programme DC.

Considérons le problème d'optimisation

$$\min\{f(x, y) : (x, y) \in S,\ x \in \{0, 1\}^n\}, \tag{1.18}$$

où $f(x, y)$ est une fonction DC et S est un polyèdre convexe borné non vide dans $\mathbb{R}^{n+m}$.

On considère $K = \{(x, y) : (x, y) \in S, x \in [0, 1]^n\}$ et la fonction $p : \mathbb{R}^{n+m} \to \mathbb{R}$ qui est définie par $p(x, y) = \sum_{i=1}^{n} x_i(1 - x_i)$ ou $p(x, y) = \sum_{i=1}^{n} \min\{x_i, 1 - x_i\}$. Il est clair que $p(x, y)$ est finie, concave, non négative sur K et le problème (1.18) s'écrit comme

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

En utilisant le Théorème 1.7, nous obtenons, pour un nombre η suffisamment grand, le problème d'optimisation DC équivalent au problème (1.19).

$$\min\{f_\eta(x,y) = f(x,y) + \eta p(x,y) : (x,y) \in K\}.$$

Dans le cas où la fonction objectif est linéaire, nous pouvons appliquer le résultat suivant qui est un cas particulier du Théorème 1.7 (voir Le Thi et al. [45]).

Théorème 1.8 *[48] Soit K un polyèdre non vide, borné de $\mathbb{R}^n$. Soit f une fonction concave finie sur K et soit p une fonction concave finie et non négative sur K. Alors il existe un nombre fini $\eta_0 \geq 0$ tel que les deux problèmes suivants sont équivalents avec $\eta \geq \eta_0$(dans le sens où ils ont le même ensemble de solutions optimales) :*

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

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

De plus, si l'ensemble de points extrêmes $V(K)$ de K est contenu dans $\{x \in K, p(x) \leq 0\}$, alors $\eta_0 = 0$, sinon $\eta_0 = \min\{\frac{f(x)-\alpha(0)}{\xi}\}$, où $\xi := \min\{p(x) : x \in V(K), p(x) > 0\} > 0$.

Preuve : Voir Le Thi *et al.*[48]. □

Considérons le problème d'optimisation linéaire

$$\min\{f(x,y) = a^T x + b^T y : (x,y) \in S,\ x \in \{0,1\}^n\}, \tag{1.19}$$

où $a \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et S est un polyèdre convexe borné non vide dans $\mathbb{R}^{n+m}$.

En utilisant le Théorème 1.8, nous obtenons, pour un nombre η suffisamment grand, le problème d'optimisation DC équivalent au problème (1.19).

$$\min\{f_\eta(x,y) = a^T x + b^T y + \eta p(x,y) : (x,y) \in K\}. \tag{1.20}$$

Remarque 1.6 *Le problème (1.20) est un problème d'optimisation DC polyhédrale.*

Chapitre 2

Méthode par Séparation et Evaluation (SE)

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

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

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

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

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

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

Définition 2.1 *Soit M un compact dans $\mathbb{R}^n$ et soit I un ensemble fini d'indices. Un ensemble $M_i : i \in I$ de sous-ensembles compacts est appelé partition de M si*

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

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

2.1 Méthode de résolution et convergence

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

Prototype SE ([8, 73])

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

- **Itération** k

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

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

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

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

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

k.5 Poser

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

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

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

est la partition actuelle.

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

k.6 Calculer

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

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

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

et retourner à l'itération $k+1$.

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

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

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

Conditions de la convergence

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

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

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

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

Définition 2.2 *Une estimation de borne est dite cohérente (consistent) si, pour une suite décroissante quelconque $M_{k_q,i_{k_q}}$ générée par la procédure de séparation, i.e.*

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

on a

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

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

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

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

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

Définition 2.3 *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" (bound improving), si au moins après chaque nombre fini d'itérations, on a

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

Théorème 2.1 *([8, 13]) Supposons que S est fermé et que* $\min f(S)$ *existe. Soit, dans le prototype, l'opération d'estimation de borne est cohérente. On a :*
(i) Si la sélection est complète, alors

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

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

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

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

Preuve : Voir [8, 13]. □

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

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

pour tout l'ensemble M.

2.2 Réalisation

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

2.2.1 Stratégie de division

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

2.2.1.1 Subdivision simpliciale

M_0 et tout élément de subdivision sont de n-simplexes. Ce n'est pas difficile de contruire le premier simplexe M_0 contenant S. Soit $M = \text{conv}\{v^0, v^1, \ldots, v^n\}$ un simplexe avec des sommets $v^0, v^1, \ldots, v^n$. Alors, un point quelconque $s \in M$ peut être representé 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 remplacant un sommet x^j tel que $\lambda_j > 0$ par s on obtient un simplexe

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

et ainsi on peut construire une subdivision, appelée *radiale*, de M. Très souvent, on choisit s comme le milieu de la plus longue arrête de M et M est divisé ainsi en deux simplexes. Dans ce cas, on a une bissection de M. Il est démontré que la bissection est exhaustive. Pourtant, on constate que les procédures exhaustives de division (en particulier bisection) ne sont pas très efficaces ; la convergence de la méthode est assez lente. On suggère alors d'utiliser comme s un point ω obtenu dans la procédure d'estimation de borne (par exemple ω est le point correspondant à $\beta(M)$). On va appeler cette procédure ω-subdivision. le problème c'est qu'on ne peut plus assurer que la procédure de division soit exhaustive. Récemment, certaines strategies plus flexibles sont etudiées. L'idée est d'utiliser le plus souvent possible la ω-subdivision et de faire intervenir la bissection pour empêcher la dégradation. Une stratégie heuristique mais pratique, a été proposée dans [10]. Considérons un simplexe M et un point $\omega \in M$, qui s'ecrit comme $\omega = \sum_{i=0}^{n} \lambda_i v^i$. Etant 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 bissection.

Les expériences ont indiqué que $\delta = \frac{1}{2}n^2$ est un choix convenable.

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

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

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

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

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

Les processus de subdivision rectangulaire jouent un rôle important dans des méthodes de Séparation et Evaluation. L'approche de Phillips et Rosen [67] (voir également Kalantari et Rosen [66]) emploie la subdivision exhaustive, i.e. toutes les suite décroissantes des rectangles générées par l'algorithme tendra vers un point. Une bissection rectangulaire adaptative prétendue proposée dans Muu L.D. [16] semble être plus efficace parce que l'exhaustivité n'est pas nécessaire pour la convergence. Dans Horst et Tuy [12], un concept de la subdivision rectangulaire normale (normal rectangular subdivision - NRS) a été présenté pour la classe des problèmes concaves séparables de minimisation qui inclut l'approche de Kalantari-Rosen et une subdivision proposées plus tôt dans Falk et Soland [68]. Intuitivement, la variante des algorithmes rectangulaires en utilisant w-subdivision et subdivision adaptative devrait converger plus rapidement que ceux qui emploient la subdivision exhautive, parce qu'ils tiennent compte des conditions du sous-problème relaxé courant.

Nous détaillerons maintenant la subdivision rectangulaire normale aussi bien que sa construction quand l'utilisant un algorithme de Séparation et Evaluation (SE) pour résoudre un type de problème important dans la programmation DC.

Considérons le problème suivant

$$\min\{f(x) = g(x) - h(x) \mid x \in R\} \tag{2.16}$$

où $R = \prod_{i=1}^{n}[l_i, L_i]$ est un rectangle dans $\mathbb{R}^n$ et $h(x)$ une fonction séparable, i.e. $h(x) = \sum_{i=1}^{n} h_i(x_i)$ où $h_i(x_i) = \frac{1}{2}\lambda_i x_i^2 - d_i y_i$ $(\lambda_i > 0)$.

Une méthode standard de borne estimation dans un schéma SE est à utiliser une minorante convexe de la fonction objectif. La meilleure minorante convexe de $f(x)$ est son enveloppe convexe, dénotons par $\text{conv}_R(f)$. Cependant, dans le cas général, au lieu de calculer $\text{conv}_R(f)$, nous contruisons $\text{conv}_R(-h)$ et utilisons $g(x) + \text{conv}_R(-h)(x)$ comme la minorante convexe pour borne estimation.

Puisque la fonction $h(x)$ est séparable, l'enveloppe convexe $\text{conv}_R(-h)$ sur le rectangle R est simplement la somme des fonctions affines $\phi_{R_i}(x_i)$ qui coïncide avec $-h_i$ au sommets du segment $[l_i, L_i]$ ([66], [65], [67]), i.e. la fonction

$$\phi_R(x) = \sum_{i=1}^{n} \phi_{R_i}(x_i) \tag{2.17}$$

où $\phi_{R_i}(x_i)$ est donnée explicitement par

$$\phi_{R_i}(x_i) = [d_i - \frac{1}{2}\lambda_i(l_i + L_i)]x_i + \frac{1}{2}\lambda_i l_i L_i. \tag{2.18}$$

Par conséquent, la solution du programme convexe

$$\min\{g(x) + \phi_R(x) \mid x \in R\} \tag{2.19}$$

fournit un point x^R tel que

$$g(x^R) + \phi_R(x^R) \leq \min\{f(x) \mid x \in R\} \leq f(x^R) \tag{2.20}$$

Subdivision Rectangulaire Normale (NRS)

Nous rappelons maintenant le concept d'une subdivision rectangulaire normale comme présentée par Tuy (voir par exemple Horst et Tuy [12])

Soit $R = \{x : l_i \leq x_i \leq L_i\}$ un rectangle et soit $\phi_R(x)$ la minorante convexe définie au-dessus de $-h(x)$ sur R. Dénotons par x^R et $\beta(R)$ une solution optimale et la valeur optimale, respectivement, du programme convexe (2.19).

Considérons un processus de subdivision rectangulaire dans lequel un rectangle est subdivisé à sous-rectangles. Un tel processus génère une famille de rectangles qui peut être représentée par un arbre avec la racine R_0. Un chemin infinie dans cet arbre correspond à une séquence contractée de rectangles R_h, $h = 0, 1, \ldots$. Pour chaque h soit $x^h = x^{R_h}$, $\phi_h(x) = \phi_{R_h}(x)$.

Définition 2.4 *Une séquence contractée R_h est appelée être normale si*

$$\lim_{h\to\infty} | - h(x^h) - \phi_h(x^h)| = 0 \tag{2.21}$$

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

Nous discuterons quelques méthodes pour construire un processus de subdivision rectangulaire normale dans ce qui suit. Supposons maintenant qu'un processus de NRS a été défini. En utilisant ce processus en même temps que la procédure de borne estimation en haut nous pouvons construire un algorithme B&B pour résoudre (2.16).

Algorithme ([73])

- **Initialisation**
  Prendre $R_0 \leftarrow R$. Calculer ϕ_{R_0} et résoudre le programme convexe

$$\min\{g(x) + \phi_{R_0}(x) : x \in R_0\}$$

  pour obtenir une solution optimale x^{R_0} et la valeur optimale $\beta(R_0)$.
  Poser $\mathcal{R} = \{R_0\}, \beta_0 = \beta(R_0), \alpha_0 = f(x^{R_0})$ et $x^* = x^{R_0}$.
- **Itération** k
  k.1 Supprimer tout $R \in \mathcal{R}_k$ tel que $\beta(R) \geq \alpha_k$. Soit $\mathcal{P}_k$ l'ensemble de rectangles restants.
  Si $\mathcal{P}_k = \emptyset$, S'arrêter. x^* est une solution optimale globale.
  k.2 Sinon, sélectionner $R_k \in \mathcal{P}_k$ tel que

$$\beta_k := \beta(R_k) = \min\{\beta(R) : R \in \mathcal{P}_k\}$$

  et subdiviser R_k en R_{k1}, R_{k2} d'après le processus de subdivision rectangulaire normale choisie.
  k.3 Pour chaque R_{k1}, R_{k2} calculer $\phi_{R_{ki}}$ et résoudre

$$\min\{g(x) + \phi_{R_{ki}}(x) : x \in R_{ki}\}$$

  pour obtenir $x^{R_{ki}}$ et $\beta(R_{ki})$
  k.4 Mettre à jour x^* la meilleure solution réalisable et α_{k+1}.
  k.5 Poser $\mathcal{R}_{k+1} := (\mathcal{P}_k \setminus R_k) \cup \{R_{k1}, R_{k2}\}$ et aller à l'itération prochaine.

Théorème 2.2 *[74] (i) Si l'Algorithme termine à itération k alors x^* est une solution optimale globale du problème (2.16).*

(ii) Si l'Algorithme est infinie alors il génère une séquence bornée x^k dont tout point d'accumulation est une solution optimale globale du (2.16), et

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

Construction d'une NRS

Comme présenté dans [12], il y a plusieurs façons de construire un processus de NRS. Supposons qu'un rectangle $R_k = \{x \mid l_i^k \leq x_i \leq L_i^k\}$ est sélectionné en étape k.2. La règle de bisection suivante de R_k a utilisé dans Kalantari-Rosen [66] et Phillips-Rosen [67] :

(i) **Bisection exhautive**

En général, i_k est choisi comme l'indice du côté le plus long de R_k, i.e., tel que

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

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

$$R_{k1} = \{x \in R_k : x_{i_k} \leq \delta\}, \quad R_{k2} = \{x \in R_k : x_{i_k} \geq \delta\}$$

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

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

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

Une règle alternative a été proposée plus tôt par Falk et Soland [68] pour la programmation non convexe séparable :

(ii) w**-bisection** Il suit de la description de l'algorithme que, pour R_k choisi, $\beta(R_k) < f(x^k)$, alors,

$$-h(x^k) - \phi(x^k) > 0$$

Choisissons un indice i_k satisfaisant

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

et subdivisons R_k en sous-rectangles

$$R_{k1} = \{x \in R_k : x_{i_k} \leq x_{i_k}^k\}, \quad R_{k2} = \{x \in R_k : x_{i_k} \geq x_{i_k}^k\}$$

Dans Muu L.D. et al. [16], pour résoudre une classe de programmation non convexe concernant des fonctions quadratiques et bilinéaires, les auteurs ont introduit la règle de subdivision suivante :

2.2.2 Règle de sélection

Naturellement, on peut choisir à chaque itération

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

qui satifait (2.13), i.e. cette sélection améliore des bornes. Pourtant, il y a bien d'autres règles qui n'utilisent pas explicitement cette propriété. Par exemple (cf. Tuy et al. [6]) :

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

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

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

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

2.2.3 Estimation de borne

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

Définition 2.5 *Soit $\{T_k\}$ une suite d'ensembles de $\mathbb{R}^n$. Alors*

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

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

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

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

Lemme 2.1 *([8]) Soit $S \in \mathbb{R}^n$ un compact et soit $f : \mathbb{R}^n \to \mathbb{R}$ continue. Alors l'estimation de borne est cohérente si, pour toute suite décroissante de compacts M_k, on a*
(i) $M_k \to M$ compact, $M \cap S \neq \emptyset$;
(ii) Il existe une suite de compacts T_k telle que

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

(iii)

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

(iv)

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

L'estimation de borne présente toujours un dilemme entre la convergence et l'efficacité. Un algorithm SE va converger plus vite si on peut estimer des bornes d'une façon plus précise. Or, cela devrait coûter plus cher ce qui peut rendre l'algorithme moins efficace. L'utilisation de T_k donne une certaine souplesse dans l'estimation des bornes. Surtout, elle permet de combiner la technique de plans de coupe, en particulier AE, avec la technique SE. Cette approche paraît très prometteuse. Le premier algorithme de ce type a été proposé par Horst et al. [9] pour la minimisation concave. Les résultats numériques ont montré la supériorité de cet algorithme par rapport à ceux qui sont purement AE ou SE. Un de ses avantages, c'est que T_k peut être construit à l'aide de la programmation linéaire et l'algorithme SE se réduit ainsi à la résolution d'une suite de programmes linéaires.

2.3 SE combinée avec DCA

Considérons le problème

$$\alpha = \min\{f(x) = g(x) - h(x) : \ x \in S \subset M\} \tag{2.22}$$

où $g(x)$ et $h(x)$ sont deux fonctions convexes et M est un ensemble convexe.

Quand nous appliquons la méthode SE pour résoudre le problème (2.22), la borne inférieure est souvent obtenue en résolvant un problème relaxé convexe

$$\alpha = \min\{f'(x) : \ x \in M\} \tag{2.23}$$

où $f'(x)$ est un minorant convexe de la fonction $f(x)$.

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

Pour la description de l'algorithme combiné SE-DCA, nous utilisons les notations ci-dessous :

- M_k est un sous-ensenble de M,
- α, γ sont respectivement la meilleure borne supérieure, la meilleure borne inférieurs,
- $x_{M_k}^{DCA}$ est la solution fournie par DCA sur l'ensemble M_k,
- x_{M_k} est la solution du problème relaxé convexe sur l'ensemble M_k,
- $\gamma_k = f'(x_{M_k})$ est la borne inférieure sur l'ensemble M_k,
- x^* est la meilleure solution.

L'algorithme combiné SE-DCA est présenté comme suit :

Algorithme 2.1 *(l'algorithme combiné SE-DCA)*

Initialisation :
- *Choisir l'ensemble* $M_0 = M$.
- *Choisir un nombre positif suffisamment petit* ϵ.
- *Choisir* $\alpha = +\infty$, $\gamma = -\infty$.

Étape 1 :
- *Résoudre le problème relaxé convexe (2.23) pour obtenir la solution optimale* x_{M_0}.
- **Si** $x_{M_0} \in S$ *Alors ARRÊTER avec la solution optimal* $x^* = x_{M_0}$
  Sinon
  Nous obtenons la première borne inférieure $\gamma := \gamma_0 = f'(x_{M_0})$.
  Résoudre le problème (2.22) par DCA avec le point initial x_{M_0} *pour obtenir la solution* $x_{M_0}^{DCA}$.
  Si $x_{M_0}^{DCA}) \in S$ **Alors**
  Mettre à jour la meilleure solution x^* *et la meilleure borne supérieure* α.
  FinSi
  FinSi

– **Si** $(\alpha - \gamma) \leq \epsilon(1 + \alpha)$ **Alors** *ARRÊTER*
Sinon
$\mathcal{M} \longleftarrow \{M_0\}$
FinSi

Étape 2 :

– *Sélectionner un ensemble* $M_k \in \mathcal{M}$.
– *Diviser l'ensemble* M_k *en deux sous-ensembles* M_{k_0}, M_{k_1}
– *Résoudre le problème relaxé convexe*

$$\gamma_{k_i} = \min\{f'(x) : \ x \in M_{k_i}\}, \ i = 0, 1$$

pour obtenir la solution $x_{M_{k_i}}$.
– **Si** $x_{M_{k_i}} \in S$ **Alors**
Mettre à jour la meilleure borne inférieure α *et la meilleure solution* x^* **FinSi**.
– *Relancer DCA pour résoudre le problème (2.22) et obtenir la solution* $x^{DCA}_{M_{k_i}}$.
– **Si** $x^{DCA}_{M_{k_i}} \in S$ **Alors**
Mettre à jour la meilleure borne inférieure α *et la meilleure solution* x^* **FinSi**.
– $\mathcal{M} = \{M_k \in \mathcal{M} : \gamma_k < \alpha\} \cup \{M_{k_i} : \gamma_{k_i} < \alpha, i = 0, 1\} \setminus \{M_k\}$.

– *Mettre à jour* $\gamma = \min\{\gamma_k : M_k \in \mathcal{M}\}$.

Étape 3 :

– **Si** $(\alpha - \gamma) < \epsilon(1 + \alpha)$ **Alors** *ARRÊTER*
Sinon
Retourner à l'étape 2.
FinSi

Deuxième partie

Optimisation en Gestion Financière

Introduction et les notations préliminaires

Imaginez un investisseur qui dispose un capital et il a un certain nombre d'actifs financiers en face. Typiquement, un tel investisseur est en face d'une décision importante. Comment partitionner son capital parmi les actifs ? La gestion financière, plus précisément gestion de portefeuille a pour objectif de répondre à cette question. On s'intéresse aux problèmes de choix d'actifs financiers en présence de risque et de rendement. En général, le but de l'investisseur est de maximiser le rendement pour un niveau fixé du risque ou de minimiser le risque pour un rendement fixé. Alors, deux notations fondamentales d'un investissement financier sont le rendement et le risque. Le rendement est la variation de la valeur accumulé d'un actif ou d'un portefeuille sur une période donnée. Alors qu'il est simple d'évaluer le rendement, il existe de nombreuses mesures pour le risque.

Rendement

Définition 2.6 *(Le rendement ou la rentabilité)*

Le rendement mesure l'appréciation relative de la valeur d'un actif financier ou d'un portefeuille d'actif financier entre deux instants successifs [91].

Soient $t-1$ et t deux instants successifs, P_{t-1} et P_t les valeurs (prix) de l'actif aux instants $t-1$ et t, respectivement. Le rendement dans l'intervalle $[t-1, t]$ est calculé par

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}}.$$

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

$$r_t = \frac{P_{t+1} - P_t + D_{t+1}}{P_t}.$$

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 qu'on adopte consiste au calcul des rentabilités de sous-périodes, ensuite on utilise la formule ci-dessous :

$$\overline{r} = \frac{1}{T}\sum_{t=1}^{T} r_t.$$

Risque

Quant au risque, beaucoup de mesures ont été proposées. On peut citer la variance, la semivariance, le "value at risk" , le "mean absolute deviation" , le "tracking error", le "betas

bull", le "beta bear", la perte maximale historique (maximum drawdown), ect...Parmi nombreuses mesures de risque considérées, la variance Var de rendement, introduite par Harry Markowitz en 1952 [81], est la plus connue et la plus utilisée en gestion de portefeuille. Mathématiquement, la variance d'une variable aléatoire r est définie comme suit :

Définition 2.7 *Soit f la fonction de densité d'une variable aléatoire r, la variance de r est*

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

Où r est une variable continue et $E(r)$ est l'espérence mathématique de r. Dans le cas où r est une variable discrète, la variance de r se définit par

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

En pratique, la variance de rendement est calculée par

$$Var(r) = \frac{1}{T} \sum_{t=1}^{T} (r_t - \overline{r})^2.$$

En outre de la variance, la perte maximale historique est également utilisée comme une mesure de risque.

Définition 2.8 *(La perte maximale historique ou "Maximum Drawdown" [140])*
La perte maximale historique d'un portefeuille $x = (x_1, x_2, ..., x_N)^T$ durant une période [0,1,..,T] est donnée par

$$MDD[x] := - \min_{t \in \{0,1,..,T\}} x^T r_t = \max_{t \in \{0,1,...,T\}} -x^T r_t$$

où r_t est le vecteur des rendements à l'instant t.

Modèle MV classique de Markowitz

Basant sur deux notions de probabilité, l'espérance mathématique des rendements et la variance des rendements, en 1952, Markowitz a introduit le modèle moyenne-variance ("mean variance" en anglais) (MV) pour le problème de sélection de portefeuille [81]. Dans son modèle, Markowitz suppose que le rendement de chaque actif soit une variable aléatoire dont la moyenne soit l'espérance de rendement alors que le risque est mesuré par la variance.

Pour établir le modèle moyenne-variance, on considère souvent un ensemble des actifs $i, i \in \{1, 2, ..., N\}$ dans un horizon $[0, 1, ..., T]$. Le rendement de l'actif i à l'instant t est r_{it}. Le rendement espéré r_i de l'actif i est donné par la moyenne des valeurs r_{it}, c'est à dire

$$r_i = \frac{1}{T} \sum_{t=1}^{T} r_{it}.$$

Soient $x_1, x_2, ..., x_N$ les proportions du capital investies dans les actifs i qui forment le portefeuille. On suppose que $x_i \geq 0$ et $\sum_{i=1}^{N} x_i = 1$. Alors, le rendement du portefeuille $x = (x_1, x_2, ..., x_N)^T$ est $R = r^T x = \sum_{i=1}^{N} r_i x_i$ et le risque dédié au portefeuille est défini par $\sigma^2 = x^T A x$ où A est la matrice de variance-covariance dont l'élément a_{ij} est calculé par

$$a_{ij} = \frac{1}{T} \sum_{t=1}^{T} (r_{it} - r_i)(r_{jt} - r_j).$$

Si l'investisseur veut minimiser le risque pour un rendement fixé R, on a le problème d'optimisation suivant :

$$\begin{cases} \min x^T A x \\ s.t. \quad \sum_{i=1}^{N} r_i x_i \geq R, \\ \qquad \sum_{i=1}^{N} x_i = 1, \\ \qquad x_i \geq 0 \ \forall i = 1, 2, ..., N. \end{cases} \tag{2.24}$$

Si l'investisseur veut maximiser le rendement pour un risque fixé ν, le problème se modélise comme suit :

$$\begin{cases} \max \sum_{i=1}^{N} r_i x_i \\ s.t. \quad x^T A x \leq \nu, \\ \qquad \sum_{i=1}^{N} x_i = 1, \\ \qquad x_i \geq 0 \ \forall i = 1, 2, ..., N. \end{cases} \tag{2.25}$$

En pratique, on utilise souvent le modèle

$$\begin{cases} \min\{\alpha . x^T A x - r^T x\} \\ s.t. \quad \sum_{i=1}^{N} x_i = 1, \\ \qquad x_i \geq 0 \ i = 1, 2, ..., N. \end{cases} \tag{2.26}$$

Ici, α est le paramètre d'aversion au risque $0 \leq \alpha < +\infty$. Si l'on résout le problème (2.26) en variant α entre 0 et $+\infty$, à partir des résultats obtenus on peut tracer une courbe qui présente la relation entre le rendement et le risque du portefeuille. On l'appelle la frontière efficiente (voir le Figure2.1).

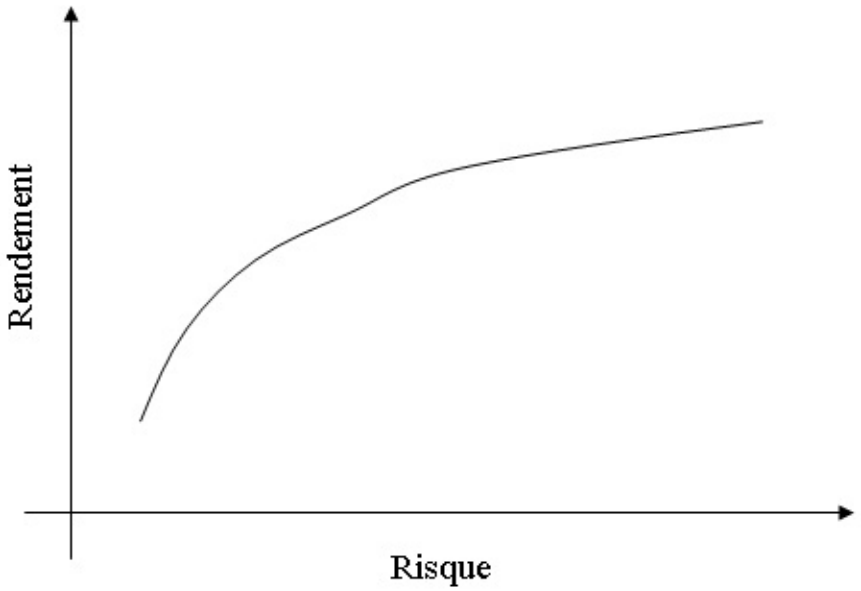


FIG. 2.1 – Exemple de la frontière efficiente.

Les problèmes proposés sont sous la forme d'un problème d'optimisation quadratique convexe qui maximise une fonction linéaire sous les contraintes quadratique ou qui minimise une fonction quadratique convexe sous les contraintes linéaires. Il n'est pas difficile de résoudre ces problèmes. Cependant, ces modèles sont rarement rencontrés en réalité car ils ne couvrent pas toutes les contraintes qui viennent du monde réel. Concernant notre travail, nous traitons deux extensions du modèle MV classique :

– Le problème min max continu en présence des contraintes de cardinalité. Ce problème a pour l'objectif d'analyser le pire des cas. On traite le problème de choix de portefeuille en comptant multi-scénarios de risques et de rendements au lieu d'un seul. En outre, les contraintes de cardinalité sont ajoutées pour diversifier le portefeuille et alors diminuer le risque.
– Dans la deuxième extension, nous considérons le problème dans le cas où une sous-compagnie base sur la décision du siège social pour sélectionner le portefeuille. La perte maximale historique est également considérée comme une mesure de risque. Ce problème se ramène à celui à deux niveaux.

Chapitre 3

Résolution du problème min max continu en gestion de portefeuille en présence des contraintes de cardinalité

Résumé *La plupart des modèles utilisés en gestion de portefeuille sont sensibles aux erreurs des entrées et des paramètres alors que l'investisseur veut souvent se protéger contre les risques dans le pire des cas. Ainsi, nous avons besoin de proposer les modèles où multi-scénarios de risque et de rendement sont considérés. Dans ce chapitre, nous présentons un modèle min-max continu qui contient les contraintes de cardinalité. La méthode DCA est utilisée pour résoudre le problème présenté. L'efficacité de DCA est comparée avec celle du CPLEX.*

3.1 Introduction

On sait que la gestion de portefeuille a pour but de maximiser le rendement pour un niveau fixé du risque ou de minimiser le risque pour un rendement fixé. Le modèle MV classique est un problème quadratique convexe. Aujoud'hui, c'est facile à résoudre un tel problème avec les logiciels très robustes, par exemple Cplex. Cependant, le modèle MV classique ne couvre pas toutes les contraintes qui viennent du monde réel, et n'est guerre rencontré en réalité. De plus, l'investisseur veut souvent se protéger dans le pire des cas. Alors, on a besoin d'étudier les modèles plus généraux aux quels on ajoute certaines contraintes et on traite simultanément multi-scénarios de risque et de rendement.

Une extension du modèle moyenne-variance classique est le modèle min-max qui a été présenté par Nalan et Berç Rustem [82, 84, 87]. Ce nouveau modèle consiste à étudier l'analyse de pire des cas en tenant compte de plusieurs matrices de covariance et plusieurs vecteurs de rendement espréré au lieu d'une seule matrice et un seul vecteur. De plus, il offre la possibilité d'avoir les portefeuilles de benchmark (l'investisseur cherche toujour à y parvenir).

Il y a deux types de modèle min-max : discret et continu. Dans le premier cas, la stratégie

d'investissement optimale est déterminée en considérant simultanément un nombre fini de scénarios de risque et de rendement. Contrairement, dans le deuxième cas, on considère multi-matrices de variance-covariance et le rendement espéré appartient à un intervalle. Sûrement, le modèle min-max continu est plus difficile à résoudre que le modèle min-max discret. La difficulté est dûe à l'infinité de nombre de scénarios. Pour la résolution, Berç Rustem et Nalan ont reformulé le modèle min-max continu sous la forme d'une programmation non convexe et puis utilisé le logicien E04UCF, Nag Library [85] pour le résoudre. Mais les auteurs ont ignoré une contrainte importante dans gestion de portefeuille, c'est celle de cardinalité. La cause est peut être la présence des variables binaires. Ces variables rendent le modèle plus difficile à résoudre.

Dans ce chapitre, nous présentons le modèle min-max continu en présence des contraintes de cardinalité. Ce sont les contraintes qui nous obligent de choisir un nombre limité d'actif disponible [88] . Elles ont été déjà étudiées dans nombreux articles [83, 88, 89, 90]. Ces recherches ont montré que les contraintes de cardinalité ont un impact important sur la stratégie de choix de portefeuille.

Le modèle original est écrit comme un problème de minimisation d'une fonction convexe sous les contraintes linéaires, les contraintes de complémentarité, et encore en variables mixtes. Puis, on prouve que les contraintes de complémentarité sont ignorées et le problème est équivalent à celui qui minimise une fonction linéaire sous les contraintes linéaire, quadratiques et en variables mixtes. En utilisant un résultat de pénalité exacte introduit par Le Thi et al. [45] on a reformulé ce problème comme un modèle de programmation DC. Finalement, la méthode DCA a été utilisée pour le résoudre.

Le reste du chapitre est organisé de façon suivante, la section 3.2 concerne la description et la formation de modèle. La section 3.3 est consacrée à la méthode de résolution. Les résultats numériques sont présentés dans la section 3.4 et la dernière section est réservée à la conclusion.

3.2 Description et formulation

Dans cette section, nous présentons le modèle moyenne-variance (MV) en présence des contraintes de cardinalité. Nous introduisons le modèle min-max continu où nous considérons multi-scénarios de risque alors que le rendement espéré est déterminé par une borne inférieure et une borne supérieure. Pour la présentation, nous utilisons des notations suivantes :

- N : le nombre d'actifs,
- $r \in \mathbb{R}^N$: le vecteur des rendements,
- r^l : la borne inférieure de rendements,
- r^u : la borne supérieure de rendements,
- $A_j \in \mathbb{R}^{N\times N}$: les matrices de variance-covariance associé aux scenarios de risque j, $j = 1, 2, ...J$,
- w : le vecteur des variables de décision,

- $\overline{w}$: la vecteur de portefeuille benchmark du marché,
- $p \in \mathbb{R}^N$: la position actuelle de l'investisseur,
- $c_b, c_s \in \mathbb{R}^N$: les vecteurs qui montrent les coûts de transaction pour les achats et les ventes, respectivement
- b, s : les variables de décision d'achat et de ventes,
- τ : le coût total de transaction (achat et vente),
- ν : la pire valeur de risque,
- α : le paramètre d'aversion au risque,
- l_i, u_i : les bornes inférieure et supérieure de la proportion du capital investi dans l'actif i,
- $Card$: le paramètre de cardinalité qui montre le nombre souhaité des actifs composant du portefeuille,
- $e = (1, 1, ..., 1)^T$: la transpose du vecteur (1,1,....,1).

Nous considérons un portefeuille qui est composé de N actifs. La proportion du capital investie dans l'actif i est w_i, $i = 1, 2, ..., N$. La somme de toutes les proportions doit être égale à 1. Alors, nous avons la première contrainte

$$\sum_{i=1}^{N} w_i = 1.$$

Si l'investisseur détient actuellement des actifs $1, ..., N$, alors le vecteur p (tel que $\sum_{i=1}^{N} p_i = 1$) présente sa position actuelle. Par contre, s'il ne possède aucun actif alors p=0. La répartition du budget initial peut être représentée par la contrainte suivante :

$$p + b - s = w.$$

Soit τ le coût total de transaction que l'investisseur doit payer pour son investissement. Notons que les coûts correspondant aux b et s sont c_b, c_s. La contrainte concernant le coût total de transaction est exprimée comme ci-dessous :

$$c_b^T b + c_s^T s = \tau.$$

Nous introduisons maintenant les contraintes de cardinalité. Ce sont celles qui ont pour objectif de contrôler le nombre d'actifs dans le portefeuille. Soient l_i et u_i les bornes inférieure et supérieure sur l'actif i. En introduisant les variable binaires z_i qui sont égales à 1 si l'actif i est investi et 0 si non. Les contraintes de bornes sont :

$$l_i z_i \leqslant w_i \leqslant u_i z_i, \text{ et } z_i \in \{0, 1\}, \ i = 1, 2, ..., N.$$

Le paramètre de cardinalité est noté $Card$ qui est le nombre souhaité des actifs dans le portefeuille. La somme des variables binaires doit être égale à $Card$. La contrainte de cardinalité est donc écrite comme suivante :

$$\sum_{i=1}^{N} z_i = Card.$$

Le rendement espéré du portefeuille, noté $E[R_p]$, est calculé par la formule $E[R_p] = (w-\overline{w})^T r$. Le risque relatif au portefeuille benchmark $\overline{w}$ est mesuré par $(w-\overline{w})^T A(w-\overline{w})$ où A est la matrice de variance-covariance. Dans le cas où on considère une seule matrice de variance-covariance, le problème de sélection de portefeuille optimal peut être formulé comme une programmation quadratique en variables mixtes.

$$\min\{-[(w-\overline{w})^T r - \tau] + \alpha.(w-\overline{w})^T A(w-\overline{w})\}$$

tel que

$$\begin{aligned}
&e^T w = 1,\\
&p + b - s = w,\\
&c_b^T b + c_s^T s = \tau,\\
&l_i.z_i \leqslant w_i \leqslant u_i.z_i,\ \forall i = 1,2,...,N,\\
&\sum_{i=1}^{N} z_i = Card,\\
&z_i \in \{0,1\},\ \forall i = 1,2,...,N,\\
&w, b, s \geqslant 0,
\end{aligned}$$

où α est le paramètre d'aversion au risque. Quand on varie α de 0 à $+\infty$ on obtient l'ensemble de toutes les stratégies d'investissement efficace.

Dans le modèle ci-dessus, on fixe un seul scénario de rendement et de risque. C'est rarement rencontré en pratique. Par ailleurs, la solution optimale de ce modèle est très sensible aux paramètres. Une petite erreur d'estimation dans la matrice de variance-covariance et/ou dans le vecteur de rendements entraine une solution erronée. De plus, le gérant veut souvent se protéger dans le pire des cas. Alors, la nécessité d'étude des modèles qui contiennent multi-scénarios de risque et de rendement est évidente. En considérant simultanément plusieurs scénarios on peut trouver une solution optimale qui est moins sensible aux erreurs éventuelles.

On considère le problème dans le contexte où l'investisseur est indécis entre plusieurs matrices de variance-covariance. Autrement dit, il existe un ensemble de matrices de variance-covariance. Concernant le rendement espéré, on prévoit qu'il soit dans un intervalle déterminé par une borne inférieure et une borne supérieure. Supposons que le nombre de matrices de variance-covariance soit J, r^l et r^u soient la borne inférieure et celle supérieure du rendement espéré respectivement. De cette façon, nous avons une extension de modèle moyenne-variance classique qui est nommée le modèle min-max continu et elle est présentée comme suit :

$$(\mathrm{P}) \quad \min\{\max_{r^l \leqslant r \leqslant r^u} -\{(w-\overline{w})^T r - \tau\} + \alpha.\max_j\{(w-\overline{w})^T A_j(w-\overline{w})\}\}$$

tel que

$$\begin{aligned}
&e^T w = 1,\\
&p + b - s = w,\\
&c_b^T b + c_s^T s = \tau,\\
&l_i.z_i \leqslant w_i \leqslant u_i.z_i \ \forall i = 1, 2, ..., N,\\
&\sum_{i=1}^{N} z_i = Card,\\
&z_i \in \{0, 1\} \ \forall i = 1, 2, ..., N,\\
&w, b, s \geqslant 0.
\end{aligned}$$

La présence des variables binaires rend le problème (P) non convexe et donc difficile à résoudre. Dans la section suivante, nous allons introduire une approche basée sur la programmation DC et DCA pour la résolution de ce problème.

3.3 Programmation DC et DCA pour la résolution du problème

3.3.1 Reformulation

Tout d'abord, on reformule le problème (P) sous la forme d'un problème qui contient les variables binaires et les contraintes de complémentarité mais sa fonction objectif est plus simple et convexe. Pour cela, on ajoute les variables x, y telles que :

$$x - y = w - \overline{w},\ x, y \geqslant 0 \text{ et } x^T y = 0.$$

Par conséquent, c'est facile à vérifier que

$$\begin{cases} x_i = w_i - \overline{w}_i \geqslant 0 \text{ et } y_i = 0 \text{ si } w_i - \overline{w}_i \geqslant 0; \\ y_i = -(w_i - \overline{w}_i) \geqslant 0 \text{ et } x_i = 0 \text{ si } w_i - \overline{w}_i \leqslant 0. \end{cases}$$

Alors,

$$\min_{r_i^l \leq r_i \leq r_i^u} (w_i - \overline{w}_i) r_i = \begin{cases} r_i^l (w_i - \overline{w}_i) = r_i^l x_i = r_i^l x_i - r_i^u y_i \text{ si } w_i - \overline{w}_i \geq 0, \\ r_i^u (w_i - \overline{w}_i) = r_i^u (-y_i) = r_i^l x_i - r^u y_i \text{ si } w_i - \overline{w}_i \leq 0. \end{cases}$$

et

$$\max_{r^l \leqslant r \leqslant r^u} \{-(w - \overline{w})^T r\} = - \min_{r^l \leqslant r \leqslant r^u} \{(w - \overline{w})^T r\} = -(r^l)^T + (r^u)^T y.$$

Ainsi, le problème (P) se transforme sous la forme :

$$(\mathrm{P}_1) \quad m_1 = \min \Big\{ -(r^l)^T x + (r^u)^T y + \tau + \alpha . \max_j \{(w - \overline{w})^T A_j (w - \overline{w})\} \Big\}$$

tel que

$$e^T w = 1, \tag{3.1}$$

$$p + b - s = w, \tag{3.2}$$

$$c_b^T b + c_s^T s = \tau, \tag{3.3}$$

$$x - y = w - \overline{w}, \tag{3.4}$$

$$x^T y = 0, \tag{3.5}$$

$$l_i.z_i \leqslant w_i \leqslant u_i.z_i, \ \forall i = 1, 2, ..., N, \tag{3.6}$$

$$\sum_{i=1}^{N} z_i = Card, \tag{3.7}$$

$$z_i \in \{0, 1\}, \ \forall i = 1, 2, ..., N, \tag{3.8}$$

$$w, b, s, x, y \geqslant 0. \tag{3.9}$$

La contrainte (3.5) est celle de complémentarité. Elle est une contrainte difficile. Néamoins, le résultat suivant nous permet de l'ignorer

Proposition 3.1 *i) Si $r_i^u > r_i^l \ \forall i$ alors le problème (P_1) et le problème ci-dessous sont équivalents, c.à.d. ils ont le même ensemble de solutions optimales*

$$(P_2) \quad m_2 = \min \Big\{ f(x, y, w, b, s, z, \tau) = \ -(r^l)^T x + (r^u)^T y + \tau \\ +\alpha. \max_j \{(w - \overline{w})^T A_j (w - \overline{w})\} \Big\}$$

sous les contraintes (3.1)-(3.4) et (3.6)-(3.9) (i.e. on obtient (P_2) en supprimant la contrainte de complémentarité (3.5) de (P_1)).

$$\begin{array}{l} e^T w = 1, \\ p + b - s = w, \\ c_b^T b + c_s^T s = \tau, \\ x - y = w - \overline{w}, \\ l_i.z_i \leqslant w_i \leqslant u_i.z_i \ \forall i = 1, 2, ..., N, \\ \sum_{i=1}^{N} z_i = Card \\ z_i \in \{0, 1\} \ \forall i = 1, 2, ..., N, \\ w, b, s, x, y \geqslant 0. \end{array}$$

ii) S'il y a un indice i tel que $r_i^u = r_i^l$, alors une solution optimale de (P_1) peut être déduire à partir de celle du problème (P_2).

Preuve : i) Soient $\mathcal{A}$ et $\mathcal{B} \subset \mathbb{R}^{6n+1}$ les ensembles des solutions réalisables de (P_1) et (P_2) respectivement. A partir de la définition de $\mathcal{A}$ et $\mathcal{B}$, il est clair que m_2 est plus petit que m_1. Pour la démonstration de (i), nous allons montrer que l'ensemble des solutions optimales de (P_2) est contenu dans $\mathcal{A}$.

Par l'absurde, nous supposons qu'il existe une solution optimale de (P_2), notée $(x^*, y^*, w^*, b^*, s^*, z^*, \tau^*)$, qui n'est pas dans $\mathcal{A}$. Cela signifique que la contrainte (3.5) n'est pas satisfaite, i.e. $(x^*)^T y^* > 0$. Donc, il existe un indice $i \in \{1, , 2, .., N\}$ tel que $x_i^* y_i^* > 0$. Nous considérons les valeurs

$$t_i = min\{x_i^*, y_i^*\}\ \forall i = 1, 2, .., N,\ t = (t_1, t_2, ..., t_N)$$

et le point $(x^1, y^1, w^*, b^*, s^*, z^*, \tau^*)$ qui est déterminé par

$$x^1 = x^* - t, y^1 = y^* - t.$$

C'est facile à vérifier que $(x^1, y^1, w^*, b^*, s^*, \tau^*) \in \mathcal{B}$. De plus, nous avons :

$$\begin{aligned} f(x^*, y^*, w^*, b^*, s^*, z^*, \tau^*) = & -(r^l)^T(x^1 + t) + (r^u)^T(y^1 + t) \\ & +\tau^* + \alpha. \max_j\{(w^* - \overline{w})^T A_j(w^* - \overline{w})\} \\ = & -(r^l)^T x^1 + (r^u)^T y^1 + \tau^* + \{(r^u)^T - (r^l)^T\}t \\ & +\alpha. \max_j\{(w^* - \overline{w})^T A_j(w^* - \overline{w})\} \\ > & f(x^1, y^1, w^*, b^*, s^*, z^*, \tau^*). \end{aligned}$$

ce qui est une contradiction à l'hypothèse que $(x^*, y^*, w^*, b^*, s^*, z^*, \tau^*)$ est une solution optimale de (P_2).

ii) Inversement, il est évident que si $(x^*, y^*, w^*, b^*, s^*, z^*, \tau^*)$ est une solution optimale de (P_2) alors $(x^1, y^1, w^*, b^*, s^*, z^*, \tau^*)$ déterminé de même manière dans la démonstration de (i) est une solution optimale du problème (P_1).

□

Remarque 3.3.1 *Le problème (P_2) peut être reformulé sous la forme d'une programmation quadratique convexe en variables mixtes.*

$$(P'_2) \quad m_2 = \min\{f(x, y, w, b, s, z, \tau, \nu) = -(r^l)^T x + (r^u)^T y + \tau + \alpha.\nu\}$$

tel que (3.1)-(3.4), (3.6)-(3.9) et

$$(w - \overline{w})^T A_j(w - \overline{w}) \leqslant \nu\ \forall j = 1, 2, ...J.$$

Pour résoudre le problème (P_2) par DCA, on le reformule comme une programmation DC. L'idée principale est d'utiliser le Théorème 1.7 introduit par Le Thi et al. (voir Section 1.4).

Soit $K := \Big\{(x, y, w, b, s, z, \tau) \in \mathbb{R}^{6N+1}$ tel que $(3.1 - 3.4), (3.6 - 3.7), (3.9)$ et $0 \leq z_i \leq 1\Big\}$. L'ensemble des solutions réalisables de (P_2) peut être formulé comme suit :

$$S = \Big\{(x, y, w, b, s, z, \tau) : (x, y, w, b, s, z, \tau) \in K \text{ et } z \in \{0, 1\}^N\Big\}.$$

Nous considérons la fonction $p : \mathbb{R}^{6N+1} \to \mathbb{R}$ qui est définie par

$$p(x, y, w, b, s, z, \tau) := \sum_{i=1}^{N} z_i(1 - z_i).$$

Clairement, $p(x, y, w, b, s, \tau)$ est concave, finie sur K et

$$p(x, y, w, b, s, z, \tau) \geq 0 \ \forall (x, y, w, b, s, z, \tau) \in K.$$

Par conséquent,

$$S = \Big\{(x, y, w, b, s, z, \tau) \in K,\ p(x, y, w, b, s, z, \tau) \leqslant 0\Big\}.$$

Ainsi, le problème (P_2) peut être réécrit comme suivant

$$\min \Big\{f(x, y, w, b, s, z, \tau, \nu) = -(r^l)^T x + (r^u)^T y + \tau + \alpha. \max\{(w - \overline{w})^T A_i (w - \overline{w})\}\Big\}$$

tel que

$$\begin{cases} (x, y, w, b, s, \tau) \in K, \\ p(x, y, w, b, s, \tau) \leqslant 0. \end{cases}$$

En appliquant le Théorème 1.7, nous obtenons, pour un nombre suffisamment grand η, le problème de minimisation équivalent à (P_2)

$$\begin{aligned} (P_3)\ \min \Big\{f_\eta(x, y, w, b, s, z, \tau) = \ & -(r^l)^T x + (r^u)^T y + \alpha. \max_j\{(w - \overline{w})^T A_j (w - \overline{w})\} \\ & + \tau + \eta. \sum_{i=1}^{N} z_i(1 - z_i)\ :\ (x, y, w, b, s, \tau) \in K\Big\}. \end{aligned}$$

Finalement, le problème (P_3) est equivalent au problème ci-dessous :

$$(P_4)\ \min \Big\{f_\eta(x, y, w, b, s, z, \tau, \nu) = -(r^l)^T x + (r^u)^T y + \eta. \sum_{i=1}^{N} z_i(1 - z_i) + \tau + \alpha.\nu\Big\}$$

tel que

$$\begin{cases} (x, y, w, b, s, \tau) \in K, \\ (w - \overline{w})^T A_j (w - \overline{w}) \leqslant \nu,\ j = 1, 2, .., J. \end{cases}$$

3.3.2 Résolution de (P_4) par DCA

Soit $X \subset \mathbb{R}^{6N+2}$ l'ensemble des solutions réalisables du problème (P_4) et les variables sont notées ζ, c'est à dire $\zeta := (x, y, w, b, s, z, \tau, \nu)$. Par conséquence, la fonction objectif du (P_4) a une décomposition DC comme suit :

$$f_\eta(\zeta) := -(r^l)^T x + (r^u)^T y + \eta. \sum_{i=1}^{N} z_i(1 - z_i) + \tau + +\alpha.\nu = g(\zeta) - h(\zeta),$$

où

$$h(\zeta) = (r^l)^T x - (r^u)^T.y + \eta. \sum_{i=1}^{N} z_i(z_i - 1) - \tau - \alpha.\nu$$

et $g(\zeta) = \chi_X(\zeta)$ est la function indicatrice de X, i.e.

$$\chi_X(\zeta) = \begin{cases} 0 & \text{si } \zeta \in X, \\ +\infty & \text{si non.} \end{cases}$$

Selon la description de DCA, la résolution de (P_4) par DCA consiste en la détermination de deux suites $\{u^k\}$ et $\{\zeta^k\}$ telles que

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

A partir de la definition de h, il est facile de constater que la fonction h est différentiable et son gradient $u^k = \nabla h(\zeta^k)$ est calculé de manière suivante :

$$\begin{cases} u_i^k = r_i^l,\ u_{i+N}^k = -r_i^u \quad \forall i = 1, 2, .., N, \\ u_i^k = 2\eta z_{i-5N} - \eta \quad \forall i = 5N + 1, ..., 6N, \\ u_i^k = -1 \quad i = 6N + 1, \\ u_i^k = -\alpha \quad i = 6N + 2, \\ u_i^k = 0, \quad \text{si non.} \end{cases} \tag{3.10}$$

Le calcul de ζ^{k+1} se ramène à la résolution du problème convexe suivant :

$$\min\{-\langle \zeta, u^k \rangle : \zeta \in X\}. \tag{3.11}$$

Algorithme 3.1 *Schéma de DCA appliqué à (P_4).*

Initialisation
- *Choisir la tolérance ϵ positive suffisamment petite.*
- *Choisir $\zeta^0 \in R^{6N+2}$ et $k = 0$.*

Répéter
- *Calculer $u^k = \nabla h(\zeta^k)$ via (3.10).*
- *Résoudre le problème (3.11) afin d'obtenir ζ^{k+1}.*

– $k \leftarrow k + 1$.
Jusqu'à $\|\zeta^k - \zeta^{k+1}\| \leq \epsilon$ *ou* $|f_\eta(\zeta^{k+1}) - f_\eta(\zeta^k)| \leq \epsilon$.

Grâce aux théorèmes sur la convergence de DCA [18, 62], nous avons les propriétés suivantes :

Théorème 3.1 *(Propriété de convergence de l'Algorithme 3.1)*

i) L'Algorithme 3.1 génère une suite $\{\zeta^k\}$ *telle que la suite* $\{f_\eta(\zeta^k)\}$ *est décroissante.*

ii) La suite $\{\zeta^k\}$ *converge vers une solution* ζ^* *qui satisfait la condition nésessaire d'optimalité locale*

Preuve : Ces propriétés sont les conséquences directes des propriétés convergentes de la programmation DC (voir par exemple [62]) □

3.4 Résultats numériques

Afin d'évaluer l'efficacité de DCA, nous l'avons implémenté en langage C++. A chaque itération, le logiciel CPLEX version 11.2 est utilisé pour résoudre le sous-problème quadratique convexe. Les données sont les prix hebdomadaires des 135 actifs durant 291 semaines.

Nous considérons 3 scénarios de risque. Dans le premier scénario, la matrice de variance-covariance A_1 est calculée à partir de tous les prix. La deuxième et troisième scénario de risque corespondent à la période de 145 semaines premières et celle de 146 semaines dernières.

Le rendement espéré est déterminé autour de la moyenne historique et la borne est de $\pm 10\%$ de la déviation standard. Nous considérons un seul portefeuille benchmark ($\overline{w}_i = 1/N$ pour tous les actifs $i = 1, 2, ..., N$). Aucun portefeuille initial est considéré, c'est-à-dire $p = 0$. Nous avons fixé le coût de transaction à 0.1% de chaque transaction qui est soit l'achat, soit la vente. Les bornes inférieure et supérieure sur la proportion du capital investi dans chaque l'actif sont fixées à 0.01 et 1.0, respectivement.

L'algorithme a été exécuté sur un ordinateur Core 2Duo 3.0GHz et 2.0 GB RAM. Les résultats fournis par DCA sont comparés avec ceux fournis par CPLEX en résolvant le problème quadratique convexe en varibales mixtes (P_2'). Nous constatons que CPLEX trouve une solution raisonnable après une heure. C'est pour cela nous avons fixé la limite d'une heure sur le temps d'exécution de CPLEX.

Dans la première expérimentation, nous étudions l'impact de la contrainte de cardinalité sur la performance de DCA. Nous avons fixé le paramètre d'aversion au risque α à 0.5 et faisons varier le paramètre $Card$ entre 10 et 25. Les résultats sont donnés dans le Tableau 3.1 où nous présentons, pour chaque paramètre $Card$, le temps d'exécution ("CPU time" en anglais) et la valeur de la fonction objectif fournis par tous les deux DCA et CPLEX.

	CPLEX				DCA		
$Card$	$time_{CPLEX}$	Val_{CPLEX}	LB	Gap_{CPLEX}	$time_{DCA}$	Val_{DCA}	Gap_{DCA}
10	3614.562	0.004295	0.004153	0.000142	4.282	0.004296	0.000143
11	3609.813	0.004248	0.004049	0.000199	4.375	0.004254	0.000205
12	3610.898	0.004203	0.003909	0.000294	4.313	0.004216	0.000307
13	3623.120	0.004160	0.003911	0.000249	4.313	0.004170	0.000259
14	3615.172	0.004121	0.003831	0.000290	3,969	0.004123	0.000292
15	115.172*	0.004081	0.003680	0.000401	4.062	0.004083	0.000403
16	3644.442	0.004044	0.003689	0.000353	4.151	0.004046	0.000357
17	103.172*	0.004020	0.003522	0.000489	4.360	0.004008	0.000486
18	87.078 *	0.004054	0.003499	0.000555	4.016	0.003969	0.000470
19	3638.226	0.003928	0.003478	0.000450	3.891	0.003935	0.000457
20	94.590*	0.004036	0.003375	0.000661	4.203	0.003905	0.000530
21	3609.969	0.003854	0.003318	0.000536	6.563	0.003856	0.000538
22	108.859*	0.003975	0.003271	0.000704	3.922	0.003823	0.000552
23	76.938*	0.003859	0.003209	0.000650	3.937	0.003749	0.000576
24	47.125*	0.003828	0.003158	0.000670	3.797	0.003749	0.000591
25	28.422*	0.003826	0.003052	0.000774	3.969	0.003716	0.000664

TAB. 3.1 – Résultats obtenus dans le cas où α=0.5, le symbole * signifie que CPLEX a échoué à résoudre un sous-problème généré dans le schéma de "Branch and Cut".

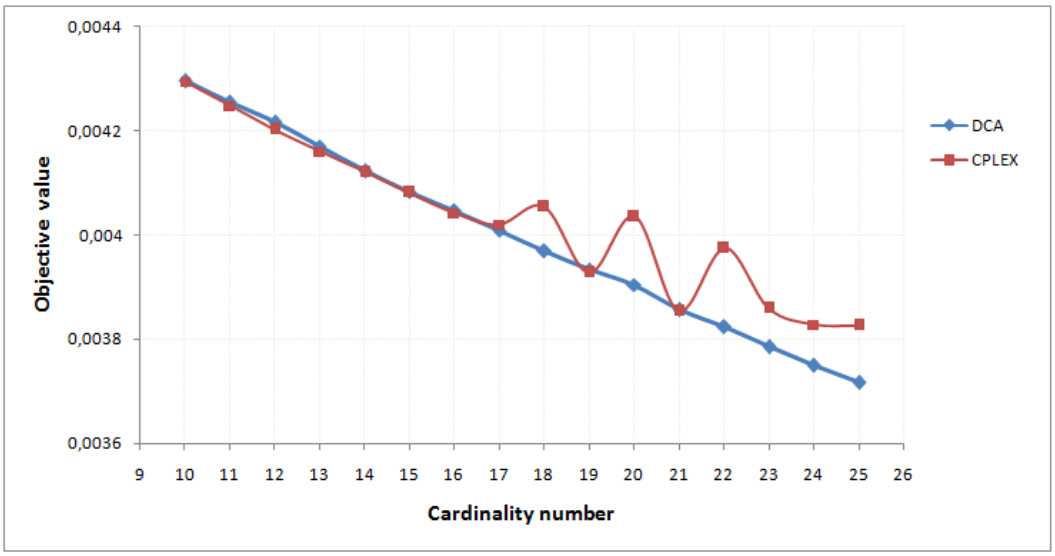


FIG. 3.1 – Les valeurs de la fonction objectif dans le cas où α=0.5.

Dans la deuxième expérimentation, nous étudions l'impact du paramètre d'aversion au risque α. Le Tableau 3.2 montre les résultats obtenus quand on fixe $Card = 20$ et varie α de 0.1 à

	CPLEX				DCA		
α	$time_{CPLEX}$	Val_{CPLEX}	LB	Gap_{CPLEX}	$time_{DCA}$	Val_{DCA}	Gap_{DCA}
0.1	3612.672	0.003581	0.003163	0.000418	5.547	0.003629	0.000466
0.2	43.125*	0.003761	0.003216	0.000545	4.297	0.003721	0.000505
0.3	129.984*	0.003791	0.003278	0.000513	3.756	0.003758	0.000480
0.4	30.125*	0.003862	0.003325	0.000537	3.719	0.003833	0.000508
0.5	94.590*	0.004036	0.003375	0.000661	4.25	0.003906	0.000531
0.6	135.797 *	0.004109	0.003421	0.000688	4.047	0.003970	0.000549
0.7	102.118*	0.004182	0.003466	0.000716	4.172	0.004063	0.000597
0.8	34.041*	0.004136	0.003511	0.000625	4.578	0.004111	0.000600
0.9	22.687*	0.004203	0.003551	0.000652	4.562	0.004181	0.000630
1.0	153.625*	0.004399	0.003595	0.000804	3.984	0.004251	0.000656

TAB. 3.2 – Résultats obtenus dans le cas où $Card = 20$, le symbole * signifie que CPLEX a échoué à résoudre un sous-problème généré dans le schéma de "Branch and Cut".

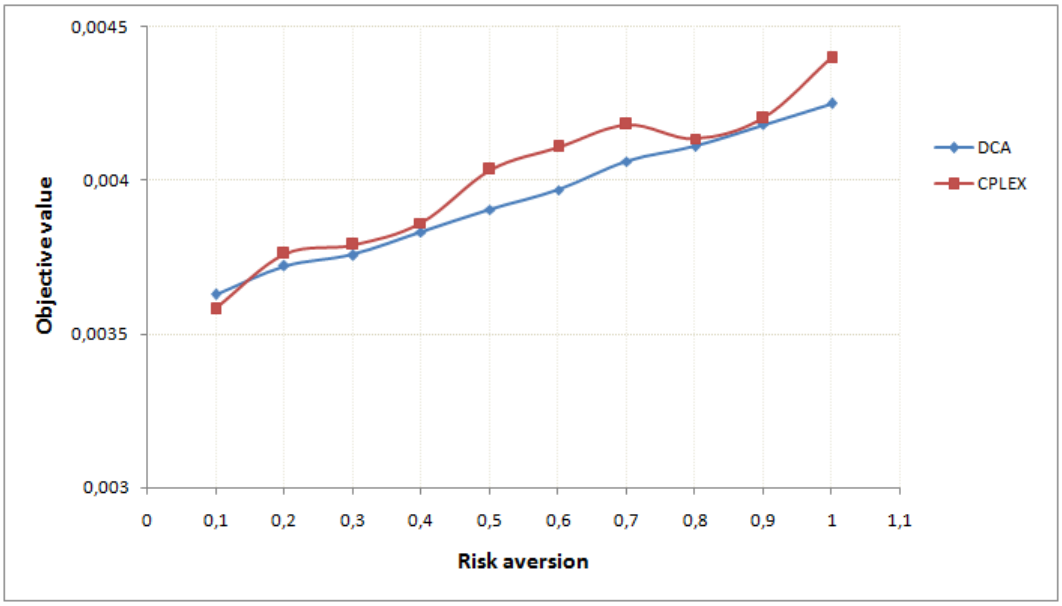


FIG. 3.2 – Les valeurs de la fonction objectif dans le cas où $Card = 20$

1.0.

Nous utilisons les notations suivantes :

♢ $time_{CPLEX}$: le temps d'exécution de CPLEX.
♢ $time_{DCA}$: le temps d'exécution de DCA.
♢ Val_{CPLEX} : la valeur de la fonction objectif fournie par CPLEX.
♢ Val_{DCA} : la valeur de la fonction objectif fournie par DCA.
♢ LB : la borne inférieure de la valeur de la fonction objectif fournie par CPLEX.

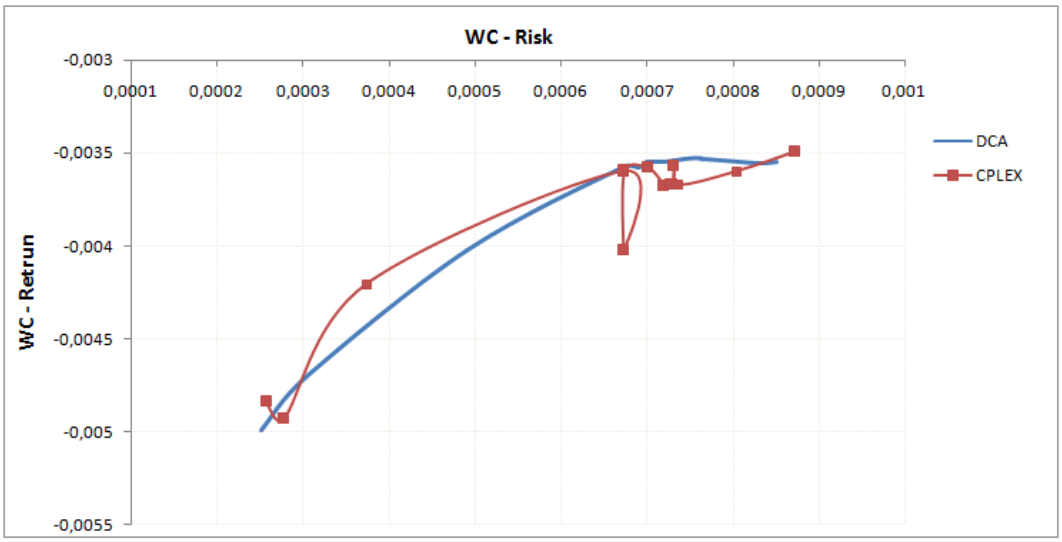


FIG. 3.3 – La frontière efficiente ($Card = 20$)

♢ Gap_{DCA} est calculé par $Gap_{DCA} = |Val_{DCA} - LB|$.
♢ Gap_{CPLEX} est calculé par $Gap_{CPLEX} = |Val_{CPLEX} - LB|$.

Commentaires sur les résultats :

A partir des tableaux de résultat, nous constatons que :

- DCA trouve toujour une ϵ-solution optimale avec $\epsilon \leq 6.7.10^{-4}$ (voir la valeur de GAP_{DCA}).
- DCA est très rapide : le temps d'exécution de DCA est moins de 6 seconds pour tous les jeux de données lors que celui de CPLEX est beaucoup plus grand, il varie entre 22.7 seconds et 3642.4 seconds. La proportion entre le temps d'exécution de CPLEX et celui de DCA varie de 4.97 à 935 fois.
- DCA est plus robuste : DCA trouve une ϵ-solution optimale pour toutes les instances alors que CPLEX a échoué à résoudre quelques problèmes. En effet, dans le cas où $\alpha = 0.5$, il y a 8 instances ($Card = 17, 18, 20, 22, 23, 24, 25$) pour lesquelles CPLEX doit s'arrêter avant le temps d'exécution limite à cause de la faillite de résolution d'un sous-problème généré dans le schéma de "Branch and Cut". De plus, bien que le temps d'exécution de CPLEX soit beaucoup plus grand que celui de DCA, la solution fournie par DCA est meilleure que celle fournie par CPLEX (sauf le cas où $Card = 17$). D'ailleurs, la différence est très importante. Les Figures 3.1, 3.2 nous permettent de voir plus clairement la qualité de la solution de DCA par rapport au celle du CPLEX.
- La Figure 3.3 montre aussi la robustesse et l'efficacité de DCA, dans cette figure, on trace la frontière efficiente du portefeuille ($Card = 20$). On peut voir que la frontière efficiente qui correspond aux résultats de DCA est une courbe croissante. Cela reflète un fait en gestion de portefeuille que si le rendement est plus grand, le risque est également plus grand. Contrairement à DCA, les résultats fournis par CPLEX ne peuvent pas montrer

cette propriété.

3.5 Conclusion

Dans ce chapitre, nous avons présenté une approche basée sur la programmation DC et DCA pour résoudre le problème min-max continu en présence des contraintes de cardinalité en gestion de portefeuille. La contrainte de cardinalité est importante mais sa présence rend le problème plus difficile en raison des variables binaires. Le problème proposé est en premier temps reformulé sous la forme de minimisation d'une fonction convexe sous les contraintes linéaire, de complémentarité et en variables mixtes 0-1. Nous montrons ensuites que les contraintes de complémentarité peuvent être ignorées. Ce problème est transformé à un programme DC en appliquant une technique de pénalité exacte. La méthode DCA est utilisée pour résoudre le résultant problème. Les résultats numériques ont montré l'efficacité de DCA par rapport au logiciel CPLEX. Plus précieusement, nous constatons que DCA est très rapide et plus robuste.

Chapitre 4

Résolution d'une classe des problèmes d'optimisation à deux niveaux et une application en gestion de portefeuille

Résumé *Le problème d'optimisation à deux niveaux joue un rôle important en programmation mathématique. En effet, il a beaucoup d'applications en économie, en finance et en théorie de jeux. Dans ce chapitre, nous traitons une classe des problèmes d'optimisation à deux niveaux où la fonction du premier niveau est quadratique convexe alors que celle du deuxième niveaux et toutes les contraintes sont linéaires. Le problème considéré est reformulé sous la forme d'un problème DC pour lequel la méthode DCA est développée. Le schéma de DCA est assez simple et il converge vers une solution locale après un nombre fini d'itérations.*

Afin de trouver une solution globale, nous combinons DCA avec l'algorithme SE (SE-DCA). Les résultats numériques ont montré l'efficacité des algorithmes proposés par rapport à l'algorithme SE sans DCA. Enfin, nous présentons une application intéressant et importante en gestion de portefeuille. Les données utilisées pour le test sont les prix des actifs sur les marchés de la France, Luxembourg, Angleterre, Etats-Unis. Les résultats obtenus ont montré la rapidité et la globalité du DCA.

4.1 Introduction

La programmation à deux niveaux (ou biniveaux) est une classe des problèmes difficile ayant des applications importantes en programmation mathématique. Plusieurs problèmes en économie, en finance, en théorie des jeux sont modélisés comme une programmation à deux niveaux. La difficulté vient de la non-convexité même quand les fonctions objectifs dans tous les deux niveaux et toutes les contraintes sont linéaires (la programmation linéaire à deux niveaux). Ce problème a été prouvé NP-difficile dans nombreux travaux (cf. [141],[142],[143]).

En général, un problème d'optimisation à deux niveaux se présente comme suit :

$$\begin{cases} \min & F(x,y) \\ s.c & (x,y) \in D_1 \subset \mathbb{R}^n \times \mathbb{R}^m, \\ & y \in argmin\{f(x,y) : (x,y) \in D_2 \subset \mathbb{R}^n \times \mathbb{R}^m\}. \end{cases}$$

Les variables de ce problème sont classées en deux classes, appelées les variables du premier niveau, $x \in \mathbb{R}^n$, et les variables du second niveau, $y \in \mathbb{R}^m$. De même, la fonction $F(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ (resp. $f(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$) est appelée la fonction objectif du premier niveau (resp. la fonction objectif du second niveau).

Dans ce chapitre, nous considérons le problème d'optimisation à deux niveaux suivant :

$$\begin{cases} \min\{F(x,y) := x^T Q_{xx} x + 2x^T Q_{xy} y + y^T Q_{yy} y + a^T x + b^T y\} \\ s.c \quad x \in S \subset \mathbb{R}^n,\ y \in argmin\{f(x,y) = c^T x + d^T y : Ax + By \leq q\}, \end{cases} \tag{4.1}$$

où $a, c \in \mathbb{R}^n, b, d \in \mathbb{R}^m, q \in \mathbb{R}^k$, S est un simplexe dans $\mathbb{R}^n$ et $Q_{xx}, Q_{xy}, Q_{yy}, A, B$ sont les matrices de dimensions $(n \times n)$, $(n \times m)$, $(m \times m)$, $(k \times n)$, $(k \times m)$. La matrice $Q = \begin{bmatrix} Q_{xx} & Q_{xy} \\ Q_{xy}^T & Q_{yy} \end{bmatrix}$ est symétrique semi-définie positive. Dans [139], les auteurs ont proposé un algorithme SE pour résoudre ce problème. Cette méthode est très compliquée et les résultats numériques sont effectués sur seulement 2 exemples avec les données de dimension petite (n=2, m=3).

Dans ce travail, nous traitons le problème de façon plus efficace par rapport aux approches existantes par un algorithme plus simple. Notre travail est basé sur la programmation DC et DCA. Nos contributions portent sur les points suivantes :

- Nous reformulons le problème sous la forme d'une programmation DC polyédrale. Le schéma de DCA pour le problème résultant est très simple : il consiste à résoudre une programmation quadratique convexe à chaque itération. En outre, il a des propriétés intéressantes, surtout la convergence finie. Il est possible que la solution obtenue par DCA ne soit pas réalisable car DCA travaille sur le domaine relaxé. Une procédure qui permet de calculer une solution réalisable à partir de celle de DCA est proposée.
- Dans le but de profiter l'efficacité de DCA, nous combinons DCA avec l'algorithme SE. Nous appliquons DCA pour calculer la borne supérieure alors que la borne inférieure est obtenue en résolvant un problème relaxé convexe.
- Pour évaluer l'efficacité des algorithmes proposés, nous les avons comparés avec l'algorithme SE sans DCA. La comparaison est effectuée sur les jeux de données de grande dimension et générés aléatoirement. Plus précisément, nous avons considéré trois cas : (n=50, m=50), (n=100, m=150) et (n=150, m=150).
- Finalement, nous présentons un problème en gestion de portefeuille qui est modélisé sous la forme du problème (4.1). Deux algorithmes DCA et SE sont appliqués pour le résoudre. Les données utilisées sont les prix des actifs sur les marchés de France, Luxembourg, Angleterre, Etats-Unis.

Le reste de ce chapitre est organisé comme suit. La section suivante est consacrée à la discussion de la méthode de résolution alors que ses résultats numériques sont rapportés dans la section 4.3. Dans la section 4.4, nous présentons une application intéressante en gestion de portefeuille. La conclusion est donnée dans la dernière section.

4.2 Résolution du problème (4.1)

4.2.1 Méthode de résolution par DCA

Tout d'abord, nous reformulons le problème (4.1) sous la forme d'une programmation DC.

Posons

$$\mu(x) = \min_y\{c^Tx + d^Ty : Ax + By \leq q\},$$

$$p(x, y) = c^Tx + d^Ty - \mu(x),$$

$$K = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m : x \in S,\ Ax + By \leq q\}.$$

Nous avons la proposition suivante :

Proposition 4.1 *Soit K un ensemble non vide borné. Alors $\mu(x) = \min_y\{c^Tx + d^Ty : Ax + By \leq q\}$ est une fonction convexe polyédrale dont le sous-différentiel en x^* est calculé par*

$$\partial\mu(x^*) = Conv\{c + A^T\zeta(x^*)\}, \tag{4.2}$$

où Conv désigne l'enveloppe convexe d'un ensemble et $\zeta(x^*)$ est une solution du problème suivant :

$$\max\{(Ax^* - q)^T\zeta : B^T\zeta = -d, \zeta \geq 0\}.$$

Preuve : Grâce à la dualité lagrangienne en programmation linéaire, nous avons

$$\begin{aligned}\mu(x) &= \min_y\{c^Tx + d^Ty : Ax + By \leq q\}\\ &= c^Tx + \min\{d^Ty : -By \geq Ax - q\}\\ &= c^Tx + \max\{(Ax - q)^T\zeta : B^T\zeta = -d, \zeta \geq 0\}\\ &= c^Tx + \max\{(Ax - q)^T\zeta : \zeta \in V\},\end{aligned}$$

où $V := \{\zeta^i, i = 1, 2, ..., I\}$ est l'ensemble des sommets du polyèdre borné K_1 qui est défini par

$$K_1 = \{\zeta \in \mathbb{R}^m : B^T\zeta = -d, \zeta \geq 0\}.$$

Par conséquence, $\mu(x)$ est une fonction convexe polyédrale et le sous-différentiel de $\mu(x)$ est donné par (4.2) □

Le problème (4.1) est réécrit comme suit :

$$(P) \quad \begin{cases} \min\{F(x,y) = x^T Q_{xx} x + 2x^T Q_{xy} y + y^T Q_{yy} y + a^T x + b^T y\} \\ t.q \quad p(x,y) = c^T x + d^T y - \mu(x) \leq 0, \\ \qquad (x,y) \in K. \end{cases}$$

Il est claire que $p(x,y)$ est finie, concave et non négative sur K. En appliquant le Théorème 1.7, nous obtenons, pour un nombre suffisamment grand η, le problème suivant qui est équivalent à (4.1).

$$\begin{cases} \min\{F_\eta(x,y) = F(x,y) + \eta(c^T x + d^T y - \mu(x))\} \\ t.q \quad (x,y) \in K. \end{cases}$$

Ce problème peut être écrit comme une programmation DC sous la forme suivante :

$$\min\{F_\eta(x,y) := G(x,y) - H(x,y) : (x,y) \in \mathbb{R}^n \times \mathbb{R}^m\} \tag{4.3}$$

où

$$G(x,y) = \chi_K(x,y) + x^T Q_{xx} x + 2x^T Q_{xy} y + y^T Q_{yy} y + (a + \eta.c)^T x + (b + \eta.d)^T y$$

et

$$H(x,y) = \eta\mu(x).$$

A partir de la définition de $H(x,y)$, le sous-différentiel $\partial H(x,y)$ est calculé par

$$\partial H(x,y) = \eta\partial\mu(x) \times \{0\}, \tag{4.4}$$

où $\partial\mu(x)$ est donné par (4.2).

Selon le schéma général, l'algorithme DCA pour résoudre le problème DC (4.3) consiste à déterminer, à chaque itération ℓ, deux suites $\{(u^\ell, v^\ell)\}$ et $\{(x^\ell, y^\ell)\}$ telles que $(u^\ell, v^\ell) \in \partial H(x^\ell, y^\ell)$ et $(x^{\ell+1}, y^{\ell+1})$ est une solution de la programmation quadratique convexe suivant :

$$\min\{G(x,y) - \langle u^\ell, x\rangle - \langle v^\ell, y\rangle : (x,y) \in \mathbb{R}^n \times \mathbb{R}^m\},$$

ou encore

$$\min\{x^T Q_{xx} x + 2x^T Q_{xy} y + y^T Q_{yy} y + (a + \eta.c - u^\ell)^T x + (b + \eta.d - v^\ell)^T y \ : (x,y) \in K\}. \tag{4.5}$$

Algorithme 4.1 *(DCA appliqué à (4.1))*

Initialization
– *Choisir la tolérance ϵ positive suffisamment petite.*
– *Choisir un point $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$ et $\ell = 0$.*
Répéter

– *Calculer* $(u^\ell, v^\ell) \in \partial H(x^\ell, y^\ell)$ *via (4.4).*
– *Résoudre la programmation quadratique convexe (4.5) pour obtenir* $(x^{\ell+1}, y^{\ell+1})$.
– $\ell \longleftarrow \ell + 1$.
Jusqu'à $\|(x^{\ell+1}, y^{\ell+1}) - (x^\ell, y^\ell)\| \leq \epsilon$ *ou* $|F_\eta(x^{\ell+1}, y^{\ell+1}) - F_\eta(x^\ell, y^\ell)| \leq \epsilon$.

Puisque H est une fonction convexe polyédrale, (4.3) est un problème DC polyédral. Alors, l'algorithme 4.1 a des propriétés de convergence intéressantes suivantes :

Théorème 4.1 *(Propriété de convergence de l'algorithme 4.1)*

i) L'Algorithme 4.1 génère une suite $\{(x^\ell, y^\ell)\}$ *telle que la suite* $\{F_\eta(x^\ell, y^\ell)\}$ *est décroissante.*

ii) La suite $\{(x^\ell, y^\ell)\}$ *converge vers un point critique* (x^*, y^*) *après un nombre fini d'itérations.*

iii) Si le problème $\max\{(Ax^* - q)^T\zeta : B^T\zeta = -d, \zeta \geq 0\}$ *a une solution unique alors* (x^*, y^*) *est une solution locale de (4.3).*

Preuve : Les propriétés (i) et (ii) sont les conséquences directes des propriétés de convergence de DCA pour une programmation DC polyédrale.

iii) Il est clair que $H(x, y)$ est une fonction convexe polyédrale. Plus précisément ,

$$\begin{array}{rl} H(x^*, y^*) = \eta\mu(x^*) & = \eta c^T x^* + \eta \max\{(Ax^* - q)^T\zeta : B^T\zeta = -d, \zeta \geq 0\} \\ & = \eta c^T x^* + \eta \max\limits_i\{(Ax^* - q)^T\zeta^i : \zeta^i \in V\} \end{array},$$

où $V := \{\zeta^i, i = 1, 2, ..., I\}$ est l'ensemble des sommets du K_1 qui est défini par

$$K_1 = \{\zeta \in \mathbb{R}^m : B^T\zeta = -d, \zeta \geq 0\}.$$

Puisque le problème $\max\{(Ax^* - q)^T\zeta : B^T\zeta = -d, \zeta \geq 0\}$ a une solution unique, $H(x, y)$ est différentiable en (x^*, y^*). Par ailleurs, selon la propriété convergente de DCA, (x^*, y^*) est un point critique, i.e. $\partial H(x^*, y^*) \cap \partial G(x^*, y^*) \neq \emptyset$. Par conséquence, $\partial H(x^*, y^*) \subset \partial G(x^*, y^*)$ et grâce à Corollaire 1.3, (x^*, y^*) est une solution locale du problème DC polyhérdal (4.3). □

Il est possible que la solution (x^*, y^*) fournie par DCA ne soit pas réalisable car DCA travaille sur le domaine relaxé. Dans un tel cas, nous utilisons la procédure suivante pour calculer une solution réalisable (x^*, y_{fea}) à partir de (x^*, y^*).

Procédure **FP**(*Calculer une solution réalisable*)

– Étape 1 : Calculer $\mu(x^*) = \min\limits_y\{c^T x^* + d^T y : By \leq q - Ax^*\}$.
– Étape 2 : Résoudre la programmation quadratique convexe

$$\left\{ \begin{array}{ll} \min & \{y^T Q_{yy} y + 2x^{*T} Q_{xy} y + d^T y\} \\ st & By \leq q - Ax^*, \\ & c^T x^* + d^T y - \mu(x^*) \leq 0, \end{array} \right.$$

pour obtenir y_{fea}.

4.2.2 Algorithme combiné de DCA et SE (SE-DCA)

Afin d'obtenir une résolution globale du problème 4.1, nous combinons DCA avec la méthode SE. Sa différence par rapport à la méthode SE donnée dans [139] est que l'algorithme 4.1 est utilisé pour améliorer la borne supérieure. La borne inférieure est obtenue en résolvant un problème relaxé convexe où nous linéarisons la contrainte concave $p(x, y) \leq 0$.

Soit $V(S) := \{s^1, s^2, ..., s^{n+1}\}$ l'ensemble des sommets du simplexe S, un point $x \in S$ peut être exprimé comme suit :

$$x = \sum_{i=1}^{n+1} \lambda_i s^i$$

où

$$\sum_{i=1}^{n+1} \lambda_i = 1,\ \lambda_i \geq 0\ \forall i = 1, 2, ..., n+1.$$

Notons M la matrice $[s^1, s^2, ..., s^{n+1}]$, la contrainte $c^T x + d^T y - \mu(x) \leq 0$ est relaxée par

$$c^T M\lambda + d^T y - \sum_{i=1}^{n+1} \mu(s^i)\lambda_i \leq 0.$$

Par conséquence, le problème suivant est un problème relaxé de (4.1)

$$(P_{relax}) \quad \begin{cases} \beta = & \min\{\lambda^T(M^T Q_{xx} M)\lambda + 2\lambda^T(M^T Q_{xy})y + y^T Q_{yy} y + (c^T M)\lambda + d^T y\} \\ s.t & (AM)\lambda + By \leq q, \\ & (c^T M)\lambda + d^T y - \sum\limits_{i=1}^{n+1} \mu(s^i)\lambda_i \leq 0, \\ & \sum\limits_{i=1}^{n+1} \lambda_i = 1,\ \lambda \geq 0. \end{cases}$$

Alors, β fournit une borne inférieure de la valeur optimale du problème (4.1).

Dans la description de l'algorithme combiné SE-DCA, nous utilisons les notations suivantes :

- (x^S, y^S) : la solution optimale du problème relaxé convexe sur le simplexe S,
- (x^S_{DCA}, y^S_{DCA}) : la solution donnée par DCA sur le simplexe S,
- (x_{opt}, y_{opt}) : la meilleure solution,
- α : la meilleure borne supérieure,
- β : la meilleure borne inférieure,
- $\Re$: l'ensemble des simplexes,
- $S_0 := S$ et ϵ est un nombre positif suffisamment petit.

L'algorithme combiné de DCA et SE (SE-DCA) peut être décrit comme suit :

Algorithme 4.2 *(L'algorithme combiné SE-DCA)*

Initialisation :

– *Choisir le premier rectangle* $S_0 := S$.
– *Choisir un nombre positif suffisamment petit* ϵ.

Étape 1 :

– *Résoudre le problème relaxé convexe* (P_{relax}) *pour obtenir* (x^{S_0}, y^{S_0}).
– *Calculer une solution réalisable* (x^{S_0}, y_{fea}) *via le procédure* ***FP***.
– *La première borne inférieure est donnée par* $\beta = \beta(S_0) = F(x^{S_0}, y^{S_0})$ *et la première borne supérieure est* $\alpha = F(x^{S_0}, y_{fea})$, *la première solution* $(x_{opt}, y_{opt}) = (x^{S_0}, y_{fea})$.
– ***Si*** $\alpha - \beta \leq \epsilon|\alpha|$ ***Alors*** *S'ARRÊTER*
Sinon
Résoudre le problème (4.1) par DCA pour obtenir $(x^{S_0}_{DCA}, y^{S_0}_{DCA})$.
Calculer une solution réalisable $(x^{S_0}_{DCA}, y_{fea})$ *via le procedure* ***FP***.
Mettre à jour la meilleure borne supérieure α *et la meilleure solution.*
FinSi
– ***Si*** $\alpha - \beta \leq \epsilon|\alpha|$ ***Alors*** *S'ARRÊTER*
Sinon
$\Re \longleftarrow \{S_0\}$.
FinSi

Étape 2 :

– *Sélectionner un simplexe* $S_\ell \in \Re$ *tel que* $\beta(S_\ell)$ *est le plus petit.*
– *Supposons que* $[s^i, s^j]$ *est l'arête la plus longue du simplexe* S_ℓ *et* $w = (s^i + s^j)/2$. *Diviser* S_ℓ *en deux sous-simplexes* S_{ℓ_1} *et* S_{ℓ_2} *via le point médian* w.
– *Résoudre le problème convexe* (P_{relax}) *sur le simplexe* S_{ℓ_i} *pour obtenir la solution* $(x^{S_{\ell_i}}, y^{S_{\ell_i}})$.
– *Calculer une solution réalisable* $(x^{S_{\ell_i}}, y_{fea})$ *et mettre à jour la meilleure solution et la meilleure borne supérieure*
– ***Si*** $\beta(S_{\ell_i}) < (1-\epsilon)\alpha$ ***Alors***
Lancer l'algorithme 4.1 à partir du point initial $(x^{S_{\ell_i}}, y^{S_{\ell_i}})$ *pour obtenir* $(x^{S_{\ell_i}}_{DCA}, y^{S_{\ell_i}}_{DCA})$.
Calculer une solution réalisable $(x^{S_{\ell_i}}_{DCA}, y_{fea})$ *et mettre à jour* α *et* (x_{opt}, y_{opt}).
FinSi
– $\Re = \{S \in \Re : \beta(S) < \alpha\} \cup \{S_{\ell_i} : \beta(S_{\ell_i}) < \alpha, i = 1, 2\} \setminus \{S_\ell\}$.
– *Mettre à jour* $\beta = \min\{\beta(S) : S \in \Re\}$.

Étape 3 :

– ***Si*** $(\alpha - \beta) < \epsilon|\alpha|$ ***Alors*** *S'ARRÊTER*
Sinon
Retouner à l'étape 2.
FinSi

4.3 Résultats numériques

Dans le but d'évaluer l'efficacité de l'algorithme 4.1 (DCA) et l'algorithme 4.2 (l'algorithme combiné SE-DCA), nous avons comparé leurs résultats avec ceux de l'algorithme SE sans DCA. Tous les trois algorithmes ont été programmés en MATLAB 2009b. Le logiciel com-

mercial CPLEX 12.1 est utilisé pour résoudre des programmations quadratiques convexes et des problèmes linéaires. Le temps d'exécution est fixé 3600 secondes pour tous les trois algorithmes.

Au niveau des données, nous considérons trois cas (cas 1 : n=50, m=50, cas 2 : n=100, m=150, cas 3 : n=150, m=150). Pour chaque cas, nous avons testé les algorithmes sur 10 jeux de données qui sont générés aléatoirement et de manière suivante :

- Pour générer une matrice symétrique et semi-définie positive Q, nous générons aléatoirement une matrice $Q_1 = (q_{ij})$, $\forall i, j = 1, 2, ..., m + n$ avec les composants $q_{i,j} \in [-1.0, 1.0]$ et puis la matrice Q est déterminée par $Q = Q_1^T Q_1$.
- Les matrices $A = [a_{ij}]_{k\times n}$ et $B = [b_{ij}]_{k\times m}$ sont aléatoirement générées dans $[-1.0, 1.0]$. Le paramètre k est choisi 100 dans les cas 1, 2 et 200 dans le cas 3. En outre, toutes les variables y sont bornées dans l'intervalle $[-10.0, 10.0]$.
- Les composants q_i du vecteur q sont générés aléatoirement dans $[0, n + m]$.
- Les composants a_i, b_i, c_i, d_i sont aléatoirement générés dans $[-2.0, 2.0]$.
- Le simplexe S est défini par $S = \{x \in \mathbb{R}^n_+ : \sum_{i=1}^{n} x_i \leq s\}$ avec $s = 10$ dans le cas 1 et $s = 20$ dans les cas 2 et 3.

Les résultats fournis par DCA, SE-DCA et SE sont rapportés dans les mêmes tableaux où nous utilisons les notations ci-dessous :

– Pro : Problème,
– Av : Moyenne ("average" en anglais),
– OptVal : la valeur optimale de la fonction objectif donnée par chaque algorithme,
– Time : le temps d'exécution en seconde de chaque algorithme,
– *Gap* est calculé par $Gap = \frac{OptVal - LB}{|OptVal|}$ où LB est la meilleure borne inférieure.

Les tableaux 4.1-4.3 donnent les résultats obtenus dans les cas $i = 1, 2, 3$ respectivement. Les Figure 4.1-4.3 présentent le *Gap* de tous les trois algorithmes dans les cas ($i = 1, 2, 3$) respectivement. Rappelons que $Gap \leq 5\%$ est un critère d'arrêt pour les algorithmes SE-DCA et SE.

Commentaires sur les résultats

Nous constatons, à partir des résultats, que :

- DCA est très rapide : pour la plupart des jeux de données, le temps d'exécution de DCA est inférieur à 5 secondes.
- DCA fournit une bonne solution optimale approchée. La moyenne de Gap de DCA est environ 8 %.
- La solution donnée par DCA est meilleure que celle fournie par l'algorithme SE sans DCA malgré que le temps d'exécution de SE soit beaucoup plus long. Les Figures 4.1-4.3 montrent que la courbe de Gap donnée par DCA est plus basse que celle de SE(BB).
- Grâce à l'efficacité de DCA, l'algorithme combiné SE-DCA est meilleur que l'algorithme SE sans DCA. La moyenne de Gap de SE-DCA est plus petite que celle de SE. Avec la limite de temps d'exécution de 3600 secondes, les moyennes de Gap de SE-DCA dans les

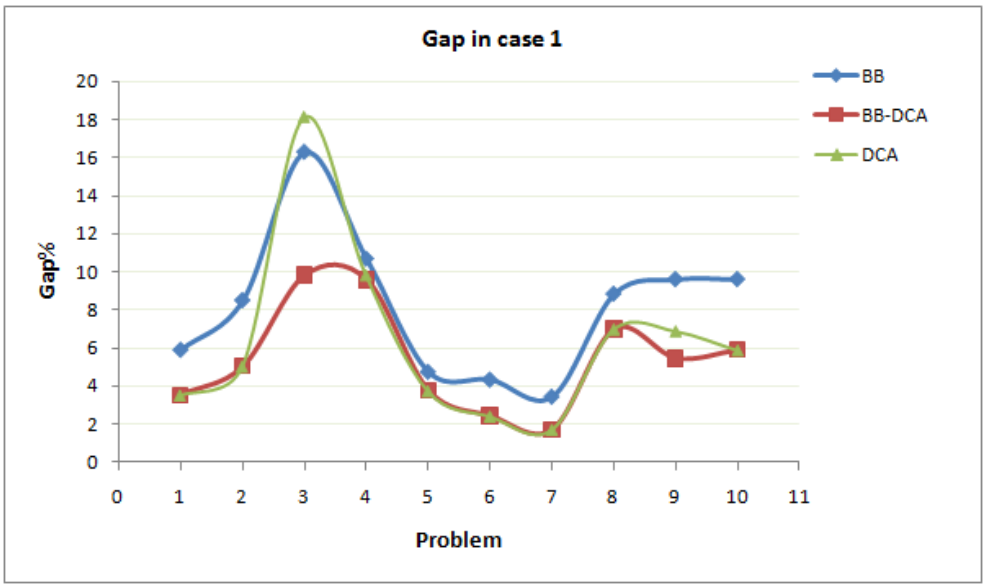


FIG. 4.1 – Gap de tous le trois algorithmes dans le cas 1

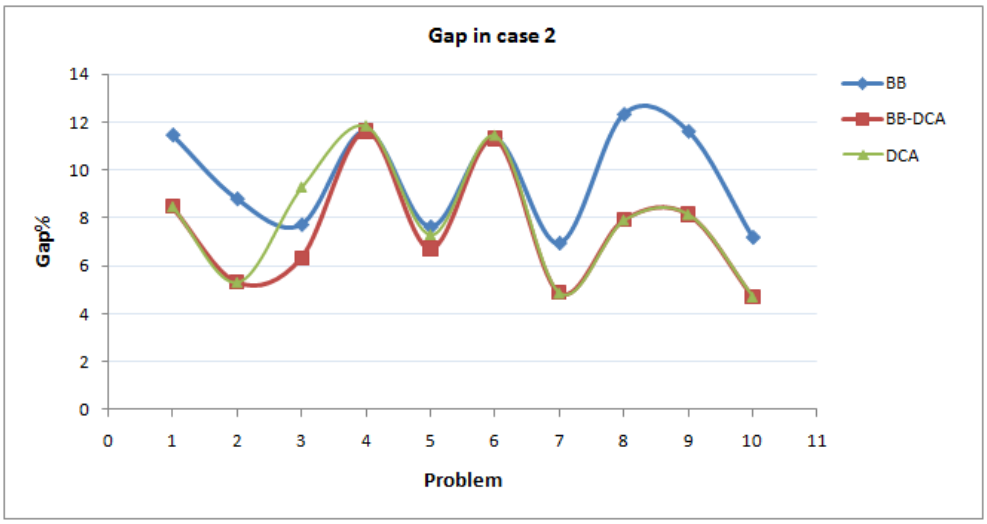


FIG. 4.2 – Gap de tous le trois algorithmes dans le cas 2

cas 1, 2, 3 sont respectivement 5.42, 7.55, 9.43 alors que celles de SE sont respectivement 8.18, 9.67, 11.38. Dans le cas 1, SE-DCA trouve une ϵ-solution globale ($Gap < 5\%$) pour 4 jeux de données tandis que SE ne trouve qu'une solution globale pour 3 jeux de données. Dans le cas 2, pour aucun jeu de données SE donne une solution globale mais SE-DCA a réussi pour 2 jeux de données.

- DCA est le meilleur : en comparant avec SE-DCA, DCA est beaucoup plus rapide et la différence entre leurs solutions est négligeable. En résumé, DCA trouve une bonne solution

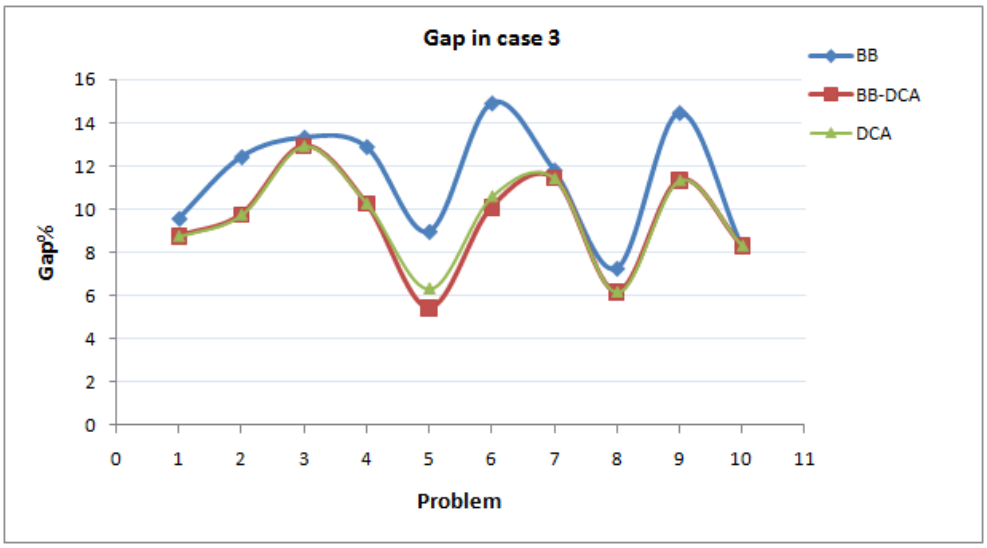


FIG. 4.3 – Gap de tous le trois algorithmes dans le cas 3

réalisable en temps d'exécution petit. Il réalise très bien le compromis entre l'efficacité et l'exactitude.

4.4 Application en gestion de portefeuille

Dans cette section, nous présentons une application intéressante en gestion de portefeuille qui est modélisé sous la forme du problème d'optimisation à deux niveaux (4.1). Les algorithmes proposés sont employés pour le résoudre. Les résultats numériques et la comparaison sont également présentés.

Considérons m actifs risqués dont les rendements historiques sont $r_{i,t}, i = 1, 2, .., m;\ t = 1, 2, .., T$. Le rendement moyen et la matrice covariance calculés à partir des rendements historiques sont $r = (r_1, r_2, ..., r_m)^T$ et $Q = (q_{ij})$ respectivement. Plus précisément, les composants r_i et q_{ij} sont calculés comme suit :

$$r_i = \frac{1}{T}\sum_{t=1}^{T} r_{it}\ \forall i = 1, 2, ..., m,$$

$$q_{ij} = \frac{1}{T}\sum_{t=1}^{T}(r_{it} - r_i)(r_{jt} - r_j)\ \forall i, j = 1, 2, ..., m.$$

En outre, il y a un actif sans risque dont le rendement est ρ. En pratique, on considère souvent les emprunts d'État à court terme comme un actif sans risque. Cet actif n'est pas corrélé avec les autres actifs et son rendement ρ est connu à l'avance.

Le problème de sélection de portefeuille considéré est décrit comme ci-dessous :

Un siège social veut choisir un compte de caisse x pour investir dans l'actif sans risque avec le rendement connu à l'avance ρ. Basant sur la décision du siège social la sous-compagnie sélectionne un portefeuille y parmi m actifs tel que le rendement espéré $\rho.x+r^Ty$ est maximal et la perte maximale historique ("maximal drawdown" en anglais) qui est définie par

$$Maxdrawdown = \max_{t=1,2,...,T}\{-\rho.x - \sum_{i=1}^{m} r_{i,t}y_i\}$$

ne dépasse pas un constant MAX (refer [140]). Le siège social veut minimiser le compromis entre le rendement $\rho.x + r^Ty$ et le risque y^TQy (cf. [81],[138]).

Par conséquence, nous avons le problème d'optimisation à deux niveaux suivant :

$$\begin{cases} \min & \delta.y^TQy - (1-\delta)(\rho.x + r^Ty) \\ s.t & y \in argmax\{\rho.x + r^Ty : (x,y) \in K\} \end{cases} \tag{4.6}$$

où δ est un constant dans $[0,1]$ et K est l'ensemble des point (x,y) qui satisfont les contraintes :

$$\begin{cases} x + \sum_{i=1}^{m} y_i = 1, \\ -\rho.x - \sum_{i=1}^{m} r_{i,t}y_i \leq MAX, \ \forall t = 1,2,...,T, \\ x, y \geq 0. \end{cases}$$

Les algorithmes SE et DCA sont employés pour résoudre le problème (4.6). Les données utilisées sont les prix d'actifs sur le marché de France, Luxembourg, Angleterre et Etats-Unis durant le période 2000-2007. Plus précisément, nous utilisons les rendements historiques semestriels pour le test. Le critère d'arrêt de l'algorithme SE est $Gap < 2\%$. Pour chacune des données, nous faisons varier le paramètre δ entre 0.1 et 0.9. Le rendement de l'actif sans risque est fixé 0.5% et le paramètre MAX est fixé 4.0%. Les résultats donnés par tous les deux algorithmes SE et DCA sont rapportés dans les Tableaux 4.4, 4.5, 4.6, 4.7 où nous utilisons les notations qui sont similaires à celles dans la section précédente.

Commentaires sur les résultats

- Dans tous les jeux de données, DCA fournit une ϵ-solution globale ($Gap < 2\%$). De plus, la solution donnée par DCA est meilleure que celle de SE. Particulièrement, les résultats sur les données des Etats-Unis et de Luxembourg sont excellents : tous les Gap de DCA sont égaux à zéro. Cela montre que DCA trouve une solution globale exacte pour ces jeux de données bien que DCA soit une méthode locale.
- DCA est meilleure que SE : les moyennes de temps d'exécution ainsi que Gap de DCA sont plus petites que celles de SE.

4.5 Conclusion

Dans ce chapitre, nous avons étudié l'approche basée sur la programmation DC et DCA pour une classe des problèmes d'optimisation à deux niveaux où la fonction objectif du premier niveau est quadratique convexe et la fonction objectif du deuxième niveau est linéaire. Une technique de pénalité a été utilisée pour le reformuler sous la forme d'un programme DC équivalent. Le schéma DCA pour le problème résultant est assez simple : à chaque itération nous résolvons seulement un problème quadratique convexe. De plus, DCA converge vers une solution locale du programme DC après un nombre fini d'itérations. Basant sur DCA, l'algorithme combiné SE-DCA a été développé. L'efficacité des algorithmes proposés est comparée avec celle de l'algorithme SE sans DCA. Les résultats sur les données aléatoires de grande dimension ont montré que DCA fournit une bonne solution approchée optimale et l'algorithme combiné SE-DCA est aussi efficace. Finalement, une application intéressante en gestion de portefeuille a été présentée. Les résultats numériques sur les données de France, Etats-Unis, Angleterre, Luxembourg ont montré l'efficacité de DCA : DCA fournit toujours une solution meilleure que SE et il est plus rapide.

	B&B			BB-DCA			DCA		
Pro	OptVal	Time	Gap%	OptVal	time	GAp%	OptVal	time	Gap%
1	108241.11	3601.64	5.88	105603.22	5.17	3.53	105603.24	1.94	3.53
2	127537.31	3603.05	8.49	122886.91	3606.11	5.03	122886.92	2.06	5.03
3	113415.54	3600.13	16.30	105258.18	3609.95	9.82	115937.09	3.11	18.12
4	104125.93	3603.14	10.68	102863.00	3608.13	9.58	103151.00	1.83	9.83
5	127139.80	216.34	4.72	125865.35	6.47	3.75	125865.37	1.98	3.75
6	117239.45	1.95	4.31	114972.39	6.45	2.42	114972.39	1.91	2.42
7	110196.41	2.11	3.41	108300.82	6.50	1.72	108300.83	2.03	1.72
8	81980.09	3602.63	8.83	80358.67	3606.23	6.99	80358.67	1.89	6.99
9	114894.67	3604.19	9.60	109858.66	3602.55	5.45	111555.13	3.13	6.89
10	117023.63	3603.63	9.60	112415.11	3610.73	5.89	112415.09	3.05	5.89
Av	112179.39	2543.88	8.18	108838.23	2166.83	5.42	110104.57	2.29	6.42

TAB. 4.1 – Résultat dans le cas où n=50, m=100 (cas 1)

	B&B			BB-DCA			DCA		
Pro	OptVal	Time	Gap%	OptVal	time	GAp%	OptVal	time	Gap%
1	255377.88	3601.70	11.47	247024.34	3602.89	8.48	247026.88	2.09	8.48
2	310629.43	3600.27	8.80	299230.17	3600.20	5.33	299229.96	2.03	5.33
3	296707.23	3601.19	7.72	292339.25	3601.25	6.34	301871.28	2.11	9.30
4	273116.06	3601.78	11.64	273116.06	3602.44	11.64	273772.31	1.36	11.85
5	254936.64	3601.30	7.63	252494.22	3603.36	6.73	254103.99	3.36	7.32
6	243884.54	3600.45	11.31	243884.54	3607.91	11.31	244262.39	4.11	11.45
7	242453.84	3601.19	6.95	237157.01	1230.66	4.88	237157.01	2.03	4.88
8	304820.21	3600.91	12.33	290275.10	3607.28	7.93	290275.22	3.48	7.93
9	271907.17	3602.30	11.62	261672.50	3602.66	8.16	261672.49	1.45	8.16
10	281731.56	3602.02	7.19	274434.95	3.48	4.72	274434.95	1.97	4.72
Av	273556.46	3601.31	9.67	267162.81	3006.21	7.55	268380.65	2.40	7.94

TAB. 4.2 – Résultats dans le cas où n=100, m=150 (cas 2)

	B&B			BB-DCA			DCA		
Pro	OptVal	Time	Gap%	OptVal	time	GAp%	OptVal	time	Gap%
1	319956.27	3601.53	9.57	317050.05	3604.89	8.74	317050.05	3.97	8.74
2	377627.97	3606.06	12.41	366467.40	3620.25	9.74	366467.50	3.78	9.74
3	302689.66	3601.05	13.31	301305.80	3620.33	12.91	301305.97	3.47	12.91
4	298013.49	3601.53	12.88	289307.93	3612.84	10.25	289309.09	2.72	10.26
5	334314.02	3605.19	8.97	321773.00	3600.33	5.42	324856.04	4.05	6.32
6	331434.73	3600.53	14.89	313644.35	3611.80	10.07	315469.82	5.17	10.59
7	291370.33	3604.47	11.80	290102.86	3613.38	11.42	290102.65	4.14	11.42
8	324404.97	3603.97	7.27	320561.21	3602.08	6.16	320561.24	3.80	6.16
9	334784.49	3606.50	14.44	323059.81	3627.02	11.33	323059.80	12.25	11.33
10	300928.03	3605.88	8.29	300928.03	3606.39	8.29	300929.90	2.58	8.30
Av	321552.40	3603.67	11.38	314420.04	3611.93	9.43	314911.21	4.59	9.58

TAB. 4.3 – Résultats dans le cas où n=150, m=150 (cas 3)

δ	B&B				DCA		
	OptVal	LB	Time	Gap%	OptVal	Time	Gap%
0,1	-0.769447	-0.770844	24.01	0.18	-0.769479	2.92	0.18
0.2	-0.526214	-0.526214	9.08	0.00	-0.526214	2.89	0.00
0.3	-0.414422	-0.414422	9.03	0.00	-0.414422	2.89	0.00
0.4	-0.335498	-0.335498	8.94	0.00	-0.335498	2.89	0.00
0.5	-0.269721	-0.269721	9.00	0.00	-0.269721	2.85	0.00
0.6	-0.208751	-0.212080	4.74	1.59	-0.210518	2.96	0.74
0.7	-0.154219	-0.155824	4.82	1.04	-0.155071	2.85	0.49
0.8	-0.101641	-0.102265	4.79	0.61	-0.101972	2.87	0.29
0.9	-0.049650	-0.050625	1.36	1.96	-0.050438	2.87	0.37
Av	-0.314396	-0.315277	8.42	0.60	-0.314815	2.89	0.23

TAB. 4.4 – Résultats avec les données de France

δ	B&B				DCA		
	OptVal	LB	Time	Gap%	OptVal	Time	Gap%
0.1	-0.605104	-0.605104	12.11	0.00	-0.605104	2.84	0.00
0.2	-0.461276	-0.461276	5.41	0.00	-0.461276	2.85	0.00
0.3	-0.379330	-0.385586	1.31	1.65	-0.381276	2.89	1.13
0.4	-0.316161	-0.319609	1.33	1.09	-0.317234	2.87	0.75
0.5	-0.258979	-0.260894	1.34	0.74	-0.259574	2.84	0.51
0.6	-0.204789	-0.205810	1.34	0.50	-0.205106	2.84	0.34
0.7	-0.152309	-0.152801	1.36	0.32	-0.152462	2.85	0.22
0.8	-0.100898	-0.101089	1.34	0.19	-0.100957	2.89	0.13
0.9	-0.050200	-0.050242	1.34	0.08	-0.050213	2.85	0.06
Av	-0.281005	-0.282490	2.99	0.51	-0.281467	2.86	0.35

TAB. 4.5 – Résultats avec les données d'Angleterre

	B&B				DCA		
δ	OptVal	LB	Time	Gap%	OptVal	Time	Gap%
0.1	-0.664712	-0.664712	5.91	0	-0.664712	4.17	0
0.2	-0.484824	-0.484824	5.90	0	-0.484824	2.96	0
0.3	-0.393296	-0.393296	5.94	0	-0.393296	3.00	0
0.4	-0.323857	-0.323857	5.77	0	-0.323857	2.93	0
0.5	-0.263254	-0.263254	6.02	0	-0.263254	2.93	0
0.6	-0.207069	-0.207069	6.01	0	-0.207069	2.89	0
0.7	-0.153408	-0.153408	6.16	0	-0.153408	3.09	0
0.8	-0.101325	-0.101325	5.96	0	-0.101325	2.92	0
0.9	-0.050295	-0.050295	6.46	0	-0.050295	3.01	0
Av	-0.293560	-0.293560	6.01	0	-0.293560	3.10	0

TAB. 4.6 – Résultats avec les données des Etats-Unis

	B&B				DCA		
δ	OptVal	LB	Time	Gap%	OptVal	Time	Gap%
0.1	-0.529011	-0.529011	12.50	0	-0.529011	3.09	0
0.2	-0.431214	-0.431214	7.72	0	-0.431214	3.07	0
0.3	-0.365932	-0.365932	7.63	0	-0.365932	3.06	0
0.4	-0.308779	-0.308779	8.08	0	-0.308779	3.00	0
0.5	-0.254877	-0.254877	7.99	0	-0.254877	3.12	0
0.6	-0.202601	-0.202601	7.96	0	-0.202601	2.95	0
0.7	-0.151254	-0.151254	7.57	0	-0.151254	3.00	0
0.8	-0.100488	-0.100488	7.68	0	-0.100488	2.96	0
0.9	-0.050108	-0.050108	7.69	0	-0.050108	2.92	0
Av	-0.266029	-0.266029	8.31	0.0	-0.266029	3.02	0.0

TAB. 4.7 – Résultats avec les données de Luxembourg

Troisième partie

Optimisation en Gestion de Production

Introduction

Les entreprises doivent gérer leurs productions pour imposer leur efficacité. Ainsi, le rôle de la gestion de production est aussi ancien que l'entreprise. Les premières réelles expériences en gestion de production ont été obtenues au moment de la réalisation des pyramides égyptiennes[144]. Ces grands chantiers ont suscité les premières réflexions dans le domaine d'approvisionnements, des ressources humaines et de la standardisation des tâches. Aujourd'hui, en raison de la compétitivité économique, des entreprises qui veulent exister et développer doivent optimiser leur performance. En outre, il peut apparaître des surcoûts, des erreurs de fabrication ou encore des problèmes de sécurité au moment de la production. Une bonne gestion de production réduira les coûts en tenant compte des conditions de travail des salariés et du respect de l'environnement. Ainsi, la gestion de production devient un élément fondamental de la gestion d'une entreprise et se place au coeur de la stratégie de l'entreprise.

Pour maintenir l'avantage concurrentiel, l'entreprise cherche à assurer la satisfaction des clients et à diminuer le prix de vente. Par ailleurs, l'entreprise nécessite aussi une marge bénéficiaire pour ses investissements futurs, son développement. Donc, la gestion de production a pour plusieurs objectifs parmi lesquels on peut citer :

- La diminution du coût de revient des produits : le coût de revient est la somme des dépenses que l'entreprise doit fournir avant de vendre une unité de sa production. Nous avons la formule :

  $$\text{Marge bénéficiaire} = \text{Prix de vente - Coût de revient.}$$

  En économie de marché, on sait que le prix de vente n'est pas fixé par l'entreprise. Il est déterminé par la loi du marché. Par conséqence, la formule ci-dessus devient

  $$\text{Marge bénéficiaire} = \text{Prix du Marché - Coût de revient.}$$

  Ainsi, l'entreprise n'a pas d'autre solution que de chercher à diminuer les coûts de revient pour soit augmenter sa marge bénéficiaire soit faire varier le prix de vente sur le marché pour un avantage concurrentiel
- L'augmentation de la productivité : ça ne veut dire pas uniquement produire plus vite à un poste de travail mais il consiste à diminuer globalement le cycle de fabrication des produits.
- Minimisation des risques : les risques de l'entreprise se traduisent par plusieurs facteurs, notamment la panne de machines, l'accident de travail, le dommage de matériels et de corps....

De ce qui précède, on peut dire que la gestion de la production comprend en général toutes les activités liées à la conception, à la planification des ressources (matérielles, financières, ou humaines), à l'ordonnancement, à l'enregistrement des activités de production, au contrôle des activités de production de l'entreprise. Ces activités doit être réalisées dans le respect des procédures établies (implicitement ou explicitement) par l'entreprise et tenir compte à

la fois de la qualité de ses produits ou services, mais aussi de la sécurité de ses salariés ou de son environnement. Plus précisément, les activités de gestion de production consistent en :

- L'amélioration de la chaîne logistique qui part des fournisseurs pour aller jusqu'à la livraison aux clients.
- La diminution des stocks et en-cours.
- L'enchaînement des opérations par une meilleure implantation des moyens de production et un meilleur ordonnancement-lancement-suivi de production...
- La diminution des tailles de lots de fabrication et des temps de changement de séries.

Pour ces objectifs, des outils informatiques adaptés ont été développés afin d'assister la gestion de production. C'est ce que l'on appelle la GPAO ou la Gestion de la Production Assistée Par Ordinateur. Cependant, la mise en oeuvre d'un outil informatique de gestion de production dans l'entreprise n'est pas banale. Il faut passer par trois phases [146] :

- Phase initiale : l'introduction d'un système GPAO doit être précédée d'une phase d'analyse très sérieuse, se déroulant obligatoirement dans laps de temps raisonnable, afin de ne pas décourager les services qui attendent l'introduction du système. Cette phase d'analyse est préalable aux choix de l'application informatique et débouche sur une réponse à la question : doit-on acheter un logiciel du machine ? si oui, le quel, ou bien doit on effectuer un développement spécifié ?
- Phase d'implantation : cette phase se démarre dès que le choix de solution informatique à faire. On développe les systèmes qui peuvent être utilisés dans l'industrie.
- Phase d'exploitation : dans cette phase, il faut insister sur ce que le système a améliorés notamment au niveau de mise à jour des informations pour la quelles l'homme-gestion occupe une position clé.

Nos travaux dans cette partie concernent la deuxième phase. Tout d'abord, il faut donner des modèles mathématiques sous la forme d'un problème d'optimisation. Les problèmes d'optimisation en gestion de production sont souvent de grande taille et en variables mixtes à cause des variables décisionnelles. Des outils/ méthodes en optimisation sont ensuite utilisés pour résoudre ces problèmes. De nombreux chercheurs ont développé les méthodes heuristiques comme la cherche Tabou, l'algorithme génétique, l'algorithme recuit simulé pour résoudre leurs problèmes. Dans nos travaux, nous développons les techniques d'optimisation basées sur la programmation DC et DCA pour trois problèmes suivants :

- Le premier est le problème de minimisation du coût de maintenance comprenant le temps de séjour et la pénalité de retard : le principe est de trouver une façon efficace de distribution des tâches de maintenance aux réparateurs. La fonction objectif est mesurée par le total du temps de séjours et la pénalité du retard. Ce travail a pour but d'assurer la fiabilité et la qualité des machines et diminuer les risques surtout la panne de machines, le dommage de matériel et de corps, la cessation de productivité. En général, il contribue à diminuer la dépense et à maximiser la performance de l'entreprise.
- Le deuxième problème concerne la minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement. Il consiste à déterminer la taille de lot pour diminuer le coût de stockage et le coût de transport.

– Le dernier problème considéré a pour l'objectif de trouver la politique de production, de transport, de stockage et les prix de transfert entre deux sous entreprises pour assurer que le bénéfice total soit maximal et que la distribution de profit entre deux entreprise soit équilibrée.

Du point de vu mathématique, tous ces trois problèmes sont ceux d'optimisation non convexes très difficiles. Nous avons développé les modèles et les nouvelles méthodes pour les résoudre. Les résultats numériques ont montré l'efficacité des approches proposées.

Chapitre 5

Minimisation du coût de maintenance comprenant le temps de séjour et la pénalité du retard

Résumé *Ce chapitre étudie le problème de minimisation de coût de maintenance impliquant le temps de séjour ("flow-time" en anglais) et la pénalité de retard. Nous proposons son premier modèle déterministe qui est une programmation linéaire en variables mixtes 0-1. La méthode DCA est ensuite développée pour le résoudre. Les résultats numériques montrent que notre approche est efficace.*

5.1 Introduction

Ce chapitre est consacré à l'étude du problème d'ordonnancement pour un ensemble de tâches génératives ou "pseudo périodique" dont les dates d'exécution ("release date" en anglais) sont inégales. En plus, il y a des contraintes sur la disponibilité des ressources de maintenance. Le type de ressource que nous considérons est humain(technicien/réparateur...). Une tâche est réalisée seulement si une ressource est disponible. De plus, les tâches sont groupées en familles indépendantes et nous adoptons les hypothèses suivantes. Premièrement, les tâches dans les familles différentes peuvent être exécutées parallèlement par les ressources disponibles. Deuxièmement, dans une famille, sauf la première tâche, la date d'exécution d'une tâche est calculée à partir de la date de terminaison de la tâche qui la précède. Enfin, nous considérons le problème dans un horizon de temps fixé.

De façon générale, le problème a pour but de maximiser le nombre des tâches qui sont effectuées ou de minimiser le temps de séjour et le retard des tâches dans l'horizon de temps considéré. Le problème abordé ici est celui d'ordonnancement dynamique car le nombre des tâches est en fonction de la date décisionnelle. Il faut rappeler que le problème non dynamique a été prouvé NP-difficile [106]. Dans la littérature, nombreuses méthodes heuristiques ont

été développées pour ce type de problème. Dans [98], les auteurs ont proposé un algorithme dont la complexité est O(n) pour résoudre le problème de prise de décision en temps réel ("real time decision making problem" en anglais). [102] a introduit un algorithme basé sur la recherche Tabou pour le problème de maintenance des tâches classées par ordre de priorité et sous les contraintes de ressource. Dans [107], les auteurs ont étudié un algorithme de colonies de fourmis afin de résoudre le problème d'ordonnancement de multiprocesseurs sous les contraintes de ressource et la contrainte de précédence. L'efficacité de cette approche a été également montrée.

Nous nous intéressons au problème d'optimisation de coût de maintenance et d'ordonnancement de tâches. Cela vient du fait que la minimisation du coût lié aux activités de maintenance est un but primordial de l'entreprise. Ce coût comprend : coût de perte de qualité du produit, coût de perte de productivité due à l'indisponibilité d'équipement (entité/machine), coût de travail, coût des pièces de rechange et leur stockage, coût d'expertise et diagnostic, coût de dommage de matériels et de corps en cas de l'accident catastrophique, coût de dommages de l'environnement, ect... Par ailleurs, tous les retards par rapport au plan préventif ou la qualité moins élevée de machine augmentent des risques, notamment la panne de machine et l'accident de travail. Alors, les activités de maintenance ont pour objectif de maintenir la fiabilité et la qualité de machine au niveau le plus élevé possible.

En pratique, la machine se dégrade en fonctionnant. S'il manque des opérations de maintenance préventive alors il entraine des opérations curatives qui sont souvent beaucoup plus chers. Il faut donc une dépense pour les opérations de maintenance. Cependant, on peut minimiser le coût de maintenance et le temps d'arrêt en attribuant en manière optimale les tâches de maintenance aux réparateurs. Par cette façon, on peut conserver à la fois la fiabilité et le coût le moins cher.

L'approche introduite dans ce chapitre a pour objectif de minimiser le temps de séjour et la pénalité de retard d'un ensemble de tâches pseudo périodiques avec les dates d'exécution inégales. Par ailleurs, le problème contient les contraintes d'antériorité et la contrainte de limite de ressource. Le principe est de trouver une solution optimale pour la gestion de maintenance d'un système qui comprend N entités (machines). On suppose que le coût d'exécution soit constant sur l'horizon de temps considéré. Alors, le coût considéré est le total de la pénalité du retard et le temps de séjour. Dans [93], les auteurs ont considéré ce problème avec l'horizon de temps continu. Une méthode heuristique a été proposée. Dans ce travail, nous discrétisons l'horizon de temps et formulons le problème sous la forme d'une programmation linéaire en variables mixtes. Puis, nous la transformons à une programmation DC en appliquant une technique de pénalité exacte. Finalement, la méthode DCA est utilisée pour résoudre le problème résultant. L'algorithme proposé a des propriétés intéressantes, par exemple, il converge après un nombre fini d'itérations et à chaque itération nous ne résolvons qu'un problème linéaire. De plus, les résultats numériques ont montré que DCA fournit toujours une solution entière. C'est rarement rencontré pour les approches continues. L'efficacité de DCA est comparée avec celle de l'algorithme basé sur la règle de temps de séjour ("flow time rule"en anglais) (FTR) donné dans [93]. Les résultats obtenus ont montré que la solution fournie par DCA est meilleure.

Le reste de ce chapitre est organisé comme suit. La section 5.2 est consacrée à la description du problème. La formulation mathématique est exprimée dans la section 5.3. La section 5.4 s'adresse à la méthode de résolution basée sur programmation DC et DCA. Les résultats numériques sont rapportés dans la section 5.5 alors que la section 5.6 donne la conclusion.

5.2 Description du problème

Comme c'est indiqué dans la section précédente, le coût des activités de maintenance préventive dépend du temps de séjour et des retards des maintenances par rapport au plan prévu. Les retards peuvent être dus à l'indisponibilité de ressource (réparateurs/ techniciens) au moment où il est nécessaire qu'une machine soit maintenue. Ils accroissent le risque abordé dans la section précédente et alors contribue à augmenter le coût de maintenance. Alors, un problème important est de chercher à minimiser le temps de séjour et le retard.

Pour la résolution du problème abordé ci-dessus, nombreuses méthodes ont été récemment proposées. Nous citons, par example, les méthodes basées sur le modèles de queue et de Markov ("queuing and Markov models" en anglais) proposées dans [97], [101],[100], [99], [104] et [110]. En outre, dans [105], [103], [109] et [113] les auteurs ont étudié le problème d'ordonnancement et d'allocation des réparateurs pour les activités de maintenance en considérant la planification des productivités et des tâches de maintenance. Les approches ci-dessus ont proposé un processus de décision statique. Ça signifie que le nombre de tâches et celui de réparateurs sont fixés. Cependant, il est très difficile de connaître le nombre de tâches à faire durant un horizon de temps car le plan de maintenance est souvent perturbé par les pannes aléatoires de machines. De plus, le risque de panne de machines est augmenté par le retard des opérations. L'origine de ce fait est l'insuffisance de ressource. Dans [93] et [94], les auteurs ont proposé une méthode heuristique pour le cas où le nombre de tâches n'est pas fixé.

Dans ce travail, nous considérons le problème abordé dans [93],[94]. La discrétisation de l'horizon de temps est appliquée. Nous partitionnons l'horizon de temps en H intervalles (périodes), chaque intervalle peut corespondre à une semaine, un jour, une heure, ...

Considérons un système qui est composé de N entités (machines). Nous supposons que les entités de ce système fonctionnent de façon indépendante. C'est-à-dire qu'une panne sur une entité quelconque n'influence pas le fonctionnement des autres. Les activités de maintenance sont assurées par q réparateurs. Nous adoptons l'hypothèse que les réparateurs ont une performance identique. En pratique, le nombre des réparateurs est souvent beaucoup plus petit que celui des entités $q << N$. En outre, nous supposons que le temps d'installation pour une tâche soit égal à zéro et que l'horizon de temps considéré soit [0,H].

Sauf la première tâche, la date d'exécution r_{i,k_i} ("release date" en anglais) de la tâche k_i sur l'entité i est calculée par $r_{i,k_i} = c_{i,k_i-1} + \rho_i$ où c_{i,k_i-1} est la date de terminaison de la tâche précédente (tâche $k_i - 1$) et ρ_i est une valeur donnée qui désigne la durée souhaitée entre deux maintenances consécutives. Par ailleurs, cette tâche devrait se terminer avant la date limite d_{i,k_i} où $d_{i,k_i} = c_{i,k_i-1} + \delta_i$. Evidemment, δ_i est une valeur donnée et $\delta_i < \rho_i$. Si la date

de terminaison de la tâche k_i, c_{i,k_i}, dépasse d_{i,k_i} alors on doit supporter une pénalité qui est proportionnelle à la durée du retard.

L'objectif du problème considéré est de minimiser le coût total, C_H, qui est exprimé par

$$C_H = \sum_{i=1}^{N} \Big(\sum_{k_i / r_{i,k_i} \in H} \big(W_0(c_{i,k_i} - r_{i,k_i}) + W_1 \max(0, c_{i,k_i} - d_{i,k_i}) \big) \Big)$$

où W_1 et W_2 représentent les coûts (par période) du temps de séjour et du retard.

5.3 Formulation mathématique

Pour la formulation, nous introduisons un autre paramètre p_i qui désigne la durée d'une tâche sur l'entité i, et les variables binaires $x_{i,t}$ qui sont définies par

$$x_{i,t} = \begin{cases} 1 \text{ si l'entité } i \text{ est maintenu à la période } t, \\ 0 \text{ si non.} \end{cases}$$

Nous avons au total q réparateurs. Ainsi, il y a au plus q entités qui sont maintenues à la période t, i.e.

$$\sum_{i=1}^{N} x_{i,t} \leq q \quad \forall t = 1, 2, .., H.$$

La durée entre deux maintenances consécutives sur l'entité i est plus grand que ρ_i. Alors nous avons :

$$x_{i,t} + x_{i,t+1} + ... + x_{i,t+\rho_i+p_i-1} \leq p_i \quad \forall\, i = 1, 2, .., N, t = 1, 2, ..., H - \rho_i - p_i + 1. \tag{5.1}$$

Au début, toutes les entités sont considérées toutes neuves. Ce fait entraine que

$$x_{i,t} = 0 \quad \forall i,\ \forall t < \rho_i.$$

La présence des contraintes ci-dessous assure que la durée d'une tâche sur l'entité i est égale à p_i, i.e.

$$x_{i,H} + \sum_{t=1}^{H-1} |x_{i,t+1} - x_{i,t}| = \frac{2}{p_i} \sum_{t=1}^{H} x_{i,t}. \tag{5.2}$$

En effet, s'il y a une tâche sur l'entité i dont la durée n'est pas égale à p_i, à partir de (5.1), il ét facile de vérifier que cette durée est strictement plus petite que p_i. Par conséquent, nous avons

$$\frac{2}{p_i} \sum_{t=1}^{H} x_{i,t} < 2T_i,$$

où T_i désigne le nombre des tâches de maintenance sur l'entité i qui sont réalisées pendant l'horizon $[0, H]$.

Par ailleurs, il est évident à constater que

$$x_{i,H} + \sum_{t=1}^{H-1} |x_{i,t+1} - x_{i,t}| = 2T_i$$

ce qui est une contradiction à l'égalité (5.2).

La contrainte (5.2) n'est pas linéaire. Cependant, elle peut être remplacée par l'ensemble des contraintes linéaires ci-dessous en ajoutant les variables auxiliares $z_{i,t}$:

$$\begin{aligned}
0 \leq z_{i,t} \leq 1 \quad \forall i,\ \forall t = 1, 2, .., H; \\
\sum_{t=1}^{H} z_{i,t} = \frac{2}{p_i} \sum_{t=1}^{H} x_{i,t} \quad \forall i = 1, 2, ..., N; \\
x_{i,t+1} - x_{i,t} \leq z_{i,t} \quad \forall i = 1, 2, .., N\ \forall t = 1, 2, .., H-1; \\
x_{i,t} - x_{i,t+1} \leq z_{i,t} \quad \forall i = 1, 2, .., N\ \forall t = 1, 2, .., H-1; \\
z_{i,H} = x_{i,H} \quad \forall i = 1, 2, .., N.
\end{aligned}$$

Le reste de la modélisation consiste à exprimer la fonction objectif. En ce qui concerne le temps de séjour, nous constatons que $\frac{1}{p_i} \sum_{i=1}^{H} x_{ik}^t$ est égal au nombre de maintenances sur l'entité i pendant l'horizon $[0, H]$. Le temps de séjour de l'entité i est donc calculé de manière suivante :

$$\sum_{k_i / r_{i,k_i} \in H} (c_{i,k_i} - r_{i,k_i}) = H - (1 + \frac{1}{p_i} \sum_{i=1}^{H} x_{i,t}).\rho_i.$$

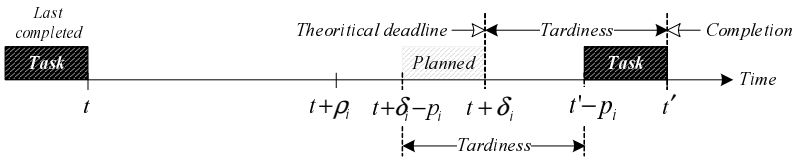


FIG. 5.1 – Le schéma de la relation entre la date d'exécution, la date limite, la date de terminaison

Pour calculer le retard des tâches, nous considérons deux tâches consécutives k_i et $k_i - 1$ (voir Figure 5.1). Dans le Figure refMaintenance :Fig01 nous utilisons les notations suivantes :
– t représente la date de terminaison de la tâche précédente (tâche $k_i - 1$).

– $t + \delta_i$ et t', $t' - p_i$ sont la date limite, la date de terminaison, la date d'exécution de la tâche k_i respectivement.

Nous introduisons les variables $y_{i,t}$ qui sont définies par

$$y_{i,t} = x_{i,t} + x_{i,t+1} + ... + x_{i,t+\delta_i - p_i - 1}.$$

Nous constatons que si $t' > t + \delta_i$, alors le retard est $t' - t - \delta_i$, cette valeur est égale au nombre des variables $y_{i,t}$ qui sont égales à zéro. Si $t' \leq t + \delta_i$ alors il n'y a aucune variable $y_{i,t} = 0$. Par conséquent, le coût du retard est calculé par

$$\sum_{k_i / r_{i,k_i} \in H} W_1 \max(0, c_{i,k_i} - d_{i,k_i}) = W_1. \sum_{t=1}^{H-\delta_i+p_i+1} \big[1 - min(y_{i,t}, 1)\big].$$

Ajoutons les variables $w_{i,t}$ qui vérifient

$$0 \leq w_{i,t} \leq 1 \text{ et } 1 - y_{i,t} \leq w_{i,t}.$$

Il est facile de vérifier que la minimisation de $W_1. \sum_{t=1}^{H-\delta_i+p_i+1} \big[1 - \min(y_{i,t}, 1)\big]$ est équivalente à la minimisation de $W_1. \sum_{t=1}^{H-\delta_i+p_i+1} w_{i,t}$. Donc, le coût sur l'entité i est exprimé comme suit :

$$W_0.H - W_0(1 + \frac{1}{p_i} \sum_{i=1}^{H} x_{i,t}).\rho_i + W_1. \sum_{t=1}^{H-\delta_i+p_i+1} w_{i,t}.$$

Finalement, nous obtenons le problème d'optimisation ci-dessous :

$$(P_1) \qquad \min \sum_{i=1}^{N} \Big\{ W_0.(H - \rho_i) - W_0.\frac{\rho_i}{p_i} \sum_{t=1}^{H} x_{i,t} + W_1. \sum_{t=1}^{H-\delta_i+p_i+1} w_{i,t} \Big\},$$

sous les contraintes

$$\sum_{i=1}^{N} x_{i,t} \leq q \quad \forall t = 1, 2, .., H. \tag{5.3}$$

$$x_{i,t} + x_{i,t+1} + ... + x_{i,t+\rho_i+p_i-1} \leq p_i \quad \forall\, i = 1, 2, .., N, t = 1, 2, ..., H - \rho_i - p_i + 1. \tag{5.4}$$

$$x_{i,t} = 0 \quad \forall i,\ \forall t < \rho_i. \tag{5.5}$$

$$0 \leq z_{i,t} \leq 1 \quad \forall i,\ \forall t = 1, 2, .., H; \tag{5.6}$$

$$\sum_{t=1}^{H} z_{i,t} = \frac{2}{p_i} \sum_{t=1}^{H} x_{i,t} \quad \forall i = 1, 2, ..., N; \tag{5.7}$$

$$x_{i,t+1} - x_{i,t} \leq z_{i,t} \quad \forall i = 1, 2, .., N\ \forall t = 1, 2, .., H - 1; \tag{5.8}$$

$$x_{i,t} - x_{i,t+1} \leq z_{i,t} \quad \forall i = 1, 2, .., N \; \forall t = 1, 2, .., H-1; \tag{5.9}$$

$$z_{i,H} = x_{i,H} \quad \forall i = 1, 2, .., N; \tag{5.10}$$

$$x_{i,t} + x_{i,t+1} + ... + x_{i,t+\delta_i - p_i - 1} + w_{i,t} \geq 1 \quad \forall i, \; \forall t = 1, 2, .., H - \delta_i + p_i + 1; \tag{5.11}$$

$$0 \leq w_{i,t} \leq 1 \quad \forall i, \; \forall t = 1, 2, ..., H - \delta_i + p_i + 1; \tag{5.12}$$

$$x_{i,t} \in \{0, 1\} \quad \forall i, \; \forall t = 1, 2, .., H. \tag{5.13}$$

Dans le problème (P_1) le nombre des réparateurs q est un paramètre. Par contre, nous pouvons considérer q comme une variable. Dans ce cas, nous devons introduire un autre paramètre, q_{max}, qui désigne le nombre des réparateurs maximal. La contrainte (5.14) est ajoutée au problème (P_1). Nous avons un autre problème d'optimisation :

$$(P_2) \qquad \min \sum_{i=1}^{N} \Big\{ W_0.(H - \rho_i) - W_0.\frac{\rho_i}{p_i} \sum_{t=1}^{H} x_{i,t} + W_1. \sum_{t=1}^{H-\delta_i+p_i+1} w_{i,t} \Big\},$$

sous les contraintes (5.3-5.13) et

$$q \leq q_{max} \tag{5.14}$$

où q_{max} est le nombre des réparateurs maximal.

Les problèmes (P_1) et (P_2) sont les programmes linéaires en variables mixtes. Il convient de souligner que nous sommes les premiers qui proposent une formulation mathématique déterministe pour ce problème. Dans la section suivante, nous allons présenter une approche basée sur la programmation DC et DCA pour les résoudre.

5.4 La résolution basée sur la programmation DC et DCA

5.4.1 DCA appliqué au problème (P_1)

En premier temps, nous reformulons le problème (P_1) sous la forme d'un programme DC. Pour ce travail, nous appliquons le théorème de pénalité qui a été introduit par Le Thi (voir Section 1.4).

Soit L_1 le nombre des variables du problème (P_1), il est calculé par

$$L_1 = 2NH + \sum_{i=1}^{N} (H - \delta_i + p_i + 1).$$

Soit S_1 l'ensemble des solutions réalisables du problème (P_1). Le domaine relaxé de S_1 est noté K_1, c'est à dire que

$$K_1 := \{(x, z, w) \in \mathbb{R}^{L_1} \text{ tels que } (5.3 - 5.12) \text{ et } x_{i,t} \in [0, 1]\}.$$

Considérons la fonction $p : \mathbb{R}^{L_1} \to \mathbb{R}$ qui est définie par

$$p(x, z, w) = \sum_{i=1}^{N} \sum_{t=1}^{H} x_{i,t}(1 - x_{i,t})$$

Il est facile de vérifier que :
- $p(x, z, w)$ est concave et finie sur K_1.
- $p(x, z, w)$ est non negative sur K_1.

Alors, S_1 peut être écrit comme suit

$$S_1 = \{(x, z, w) \in \mathbb{R}^{L_1} \text{ tels que } (x, z, w) \in K_1 \text{ et } p(x, z, w) \leq 0\}.$$

Par conséquent, le problème (P_1) peut être réécrit comme ci-dessous :

$$\begin{cases} \min \sum_{i=1}^{N} \Big\{ W_0(H - \rho_i) - W_0 \frac{\rho_i}{p_i} \sum_{t=1}^{H} x_{ik}^t + W_1 \sum_{t=1}^{H-\delta_i+p_i+1} w_{i,t} \Big\} \\ \text{tel que} \quad (x, z, w) \in K_1 \text{ et } p(x, z, w) \leq 0. \end{cases} \tag{5.15}$$

En appliquant le Théorème 1.8, nous obtenons, pour un nombre suffisamment grand η, le problème de équivalent au problème (5.15) de la forme

$$\min \Big\{ \sum_{i=1}^{N} \Big\{ W_0(H - \rho_i) - W_0 \frac{\rho_i}{p_i} \sum_{t=1}^{H} x_{i,t} + W_1. \sum_{t=1}^{H-\delta_i+p_i+1} w_{i,t} \Big\} - \eta.p(x, z, w) : (x, z, w) \in K_1 \Big\}. \tag{5.16}$$

Le problème (5.16) est une programmation DC dont l'ensemble des solutions réalisables est K_1 et la fonction objectif est

$$f_\eta(x, z, w) = g(x, z, w) - h(x, z, w) \tag{5.17}$$

où

$$g(x, z, w) := \chi_{K_1}(x, z, w) = \begin{cases} 0 \text{ si } (x, z, w) \in K_1, \\ +\infty, \text{ si non.} \end{cases}$$

$$h(x, z, w) := -\sum_{i=1}^{N} \Big\{ W_0(H - \rho_i) - W_0. \frac{\rho_i}{p_i} \sum_{t=1}^{H} x_{i,t} + W_1. \sum_{t=1}^{H-\delta_i+p_i+1} w_{i,t} \Big\} + \eta. \sum_{i=1}^{N} \sum_{t=1}^{H} x_{i,t}(x_{i,t} - 1).$$

Finalement, l'algorithme DCA appliqué au problème (P_1) consiste en la détermination de deux suites $\{u^l\}$ et $\{\varsigma^l = (x^l, z^l, w^l)\}$ telles que $u^l = \nabla h(x^l, z^l, w^l)$ et $(x^{l+1}, z^{l+1}, w^{l+1})$ est une solution du problème linéaire suivant :

$$\min\{-\langle (x, z, w), u^l \rangle : (x, z, w) \in K_1\}. \tag{5.18}$$

Par sa définition, h est différentiable et $u^l = \nabla h(x^l, z^l, w^l)$ est calculé par

$$\begin{cases} u^l_{H.(i-1)+t} = 2.\eta.x^l_{i,t} + W_0.\frac{\rho_i}{p_i} \quad \forall i = 1, 2, .., N, \forall t = 1, 2, ..., H, \\ u^l_j = 0 \quad \forall j = NH, NH + 1, ..., 2NH. \\ u^l_j = -W_1 \quad \text{si non.} \end{cases} \tag{5.19}$$

Algorithme 5.1 *Schéma de DCA appliqué à* (P_1).

Initialisation
– *Choisir la tolérance* ϵ *positive suffisamment petite.*
– *Choisir un point* $(x^0, z^0, w^0) \in \mathbb{R}^{L_1}$ *et* $l = 0$.
Répéter
– *Calculer* $u^l = \nabla h(x^l, z^l, w^l)$ *via (5.19).*
– *Résoudre le problème (5.18) afin d'obtenir* $(x^{l+1}, z^{l+1}, w^{l+1})$.
– $l \leftarrow l + 1$.
Jusqu'à $\|(x^{l+1}, z^{l+1}, w^{l+1}) - (x^l, z^l, w^l)\| \leq \epsilon$ *ou* $|f_\eta(x^{l+1}, z^{l+1}, w^{l+1}) - f_\eta(x^l, z^l, w^l)| \leq \epsilon$.

Grâce aux théorèmes de convergence de DCA pour la programmation polyédrale (cf. [35, 40, 24]), nous avons les propriétés suivantes :

Théorème 5.1 *(Propriété de convergence de l'Algorithme 5.1)*

(i) L'algorithme 5.1 génère une suite (x^l, z^l, w^l) *qui appartient à* $V(K)$*, l'ensemble des sommets de* K *telle que la suite* $\{f_\eta(x^l, z^l, w^l)\}$ *est décroissante.*

ii) La suite (x^l, z^l, w^l) *converge vers une solution* $(x^*, z^*, w^*) \in V(K)$ *après un nombre fini d'itération.*

(iii) Le point (x^*, z^*, w^*) *est un point KKT du problème (5.17).*

(iv) Pour un nombre suffisamment grand η*, si* $x^r \in \{0,1\}^{NH}$ *alors* $x^l \in \{0,1\}^{NH} \quad \forall l \geq r$.

Preuve : Voir, par exemple, Le Thi *et al.* [24]. □

5.4.2 DCA appliqué au problème (P_2)

Soit L_2 le nombre des variables du problème (P_2) $(L_2 = L_1 + 1)$. Soient S_2 l'ensemble des solutions réalisables du problème (P_2) et K_2 le domaine relaxé de S_2, c'est-à-dire

$$K_2 = \Big\{(x, z, w, q) \in \mathbb{R}^{L_2} \text{ tels que } (5.2 - 5.12), (5.14) \text{ et } x_{i,t} \in [0,1]\Big\}.$$

En utilisant la même façon de reformulation dans 5.4.1 nous obtenons le programme DC équivalent au problème (P_2) :

$$\min\Big\{f_{\eta_2}(x, z, w, q) = g(x, z, w, q) - h(x, z, w, q) : (x, z, w, q) \in K_2\Big\} \tag{5.20}$$

où

$$g(x, z, w, q) := \chi_{K_2}(x, z, w, q) = \begin{cases} 0 \text{ si } (x, z, w, q) \in K_2, \\ +\infty, \text{ si non.} \end{cases}$$

$$h(x,z,w,q) := -\sum_{i=1}^{N}\Big\{W_0(H-\rho_i)-W_0.\frac{\rho_i}{p_i}\sum_{t=1}^{H}x_{i,t}+W_1.\sum_{t=1}^{H-\delta_i+p_i+1}w_{i,t}\Big\}+\eta_2.\sum_{i=1}^{N}\sum_{t=1}^{H}x_{i,t}(x_{i,t}-1).$$

Selon le schéma général de DCA, nous devons calculer deux suites $\{\varsigma^l = (x^l, z^l, w^l, q^l)\}$ et $\{u^l\}$ telles que $u^l = \nabla h(x^l, z^l, w^l, q^l)$ et $(x^{l+1}, z^{l+1}, w^{l+1}, q^{l+1})$ est une solution du problème linéaire suivant :

$$\min\{-\langle(x,z,w,q), u^l\rangle : (x,z,w,q) \in K_2\}. \tag{5.21}$$

Par sa définition, h est différentiable et $u^l = \nabla h(x^l, z^l, w^l, q^l)$ est déterminé par :

$$\begin{cases} u^l_{H.(i-1)+t} = 2\eta_2 x^l_{i,t} + W_0.\frac{\rho_i}{p_i} \quad \forall i = 1,2,..,N, \forall t = 1,2,...,H, \\ u^l_j = 0 \quad \forall j = NH, NH+1, ..., 2NH \text{ et } j = L_2 \\ u^l_j = -W_1 \quad \text{si non.} \end{cases} \tag{5.22}$$

Algorithme 5.2 *Schéma de DCA appliqué à* (P_2).

Initialisation
– *Choisir la tolérance* ϵ *positive suffisamment petite.*
– *Choisir un point* $\varsigma^0 = (x^0, z^0, w^0, q^0) \in \mathbb{R}^{L_2}$ *et* $l = 0$.
Répéter
– *Calculer* $u^l = \nabla h(x^l, z^l, w^l, q^l)$ *via (5.22).*
– *Résoudre le problème linéaire (5.21) afin d'obtenir* $\varsigma^{l+1} = (x^{l+1}, z^{l+1}, w^{l+1}, q^{l+1})$.
– $l \leftarrow l + 1$.
Jusqu'à $\|(x^{l+1}, z^{l+1}, w^{l+1}, q^{l+1}) - (x^l, z^l, w^l, q^l)\| \leq \epsilon$ *ou* $|f_{\eta_2}(\varsigma^{l+1}) - f_{\eta_2}(\varsigma^l)| \leq \epsilon$.

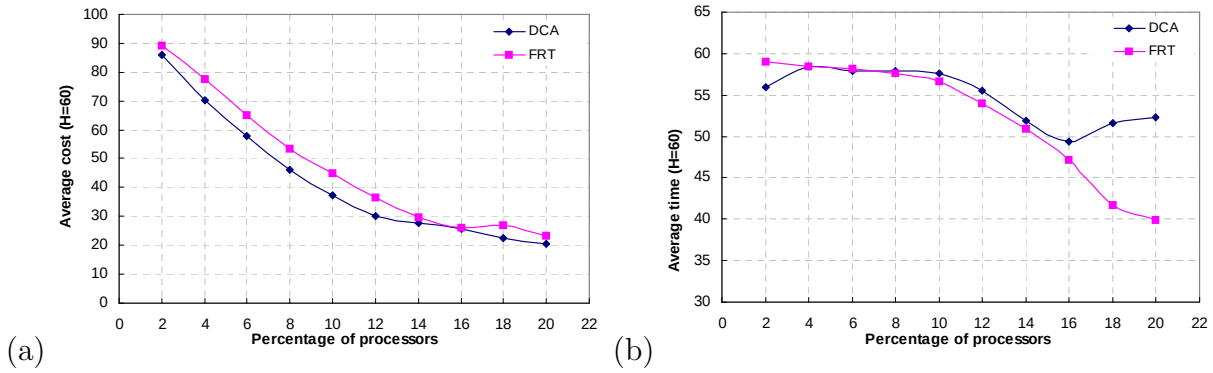
Remarque 5.1 *Les conclusions du Théorème 5.1 (les propriétés de convergence) sont encore validées pour l'Algorithme 5.2.*

5.5 Les résultats numériques

Les Algorithmes 5.1, 5.2 sont programmés en Visual Studio C++2005. Nous utilisons le logiciel CPLEX version 11.2 pour résoudre le problème linéaire à chaque itération. L'efficacité de l'Algorithme 5.1 (DCA appliqué à (P_1)) est comparée avec celle de l'algorithme basé sur FTR qui est donné dans [93]. Tous les algorithmes sont exécutés sur un ordinateur Core 2duo 3.0 GHz et 2.0 GB RAM.

Au niveau des données, nous considérons un système de 100 entités (N=100). Les paramètres ρ_i, δ_i, p_i sont générés aléatoirement et les paramètres W_0, W_1 sont fixés à 1.0 (les données sont construites par la même façon dans [93]). Nous considérons deux horizons de temps $H = 60$ jours (2 mois) et $H = 90$ jours (3 mois). Le nombre des réparateurs est varié entre 2 et 20.

Les résultats sont présentés dans les Figures 5.2 et 5.3 où nous utilisons les notations ci-dessus :



(a) (b)

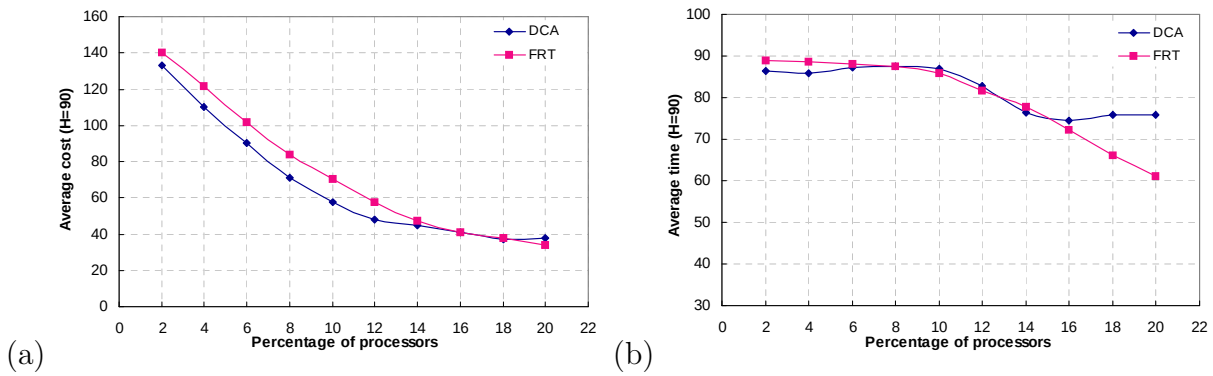
FIG. 5.2 – Résultats dans le cas où l'horizon H=60

◇ Le coût moyen ("the average cost" ou "the mean cost" en anglais) est calculé par $\frac{BestVal}{N}$, où $BestVal$ est la valeur de la fonction objectif (coût total) et N est le nombre des entités.

◇ Le coût moyen relatif ("the mean relative cost" en anglais) est le total du coût moyen et la quantité $\frac{q.H}{N}$ où N est le nombre des entités, q est le nombre des réparateurs, H est l'horizon de temps.

◇ Le temps d'utilisation moyen ("the average processing time" ou "the mean time" en anglais) est calculé par $\frac{T}{q}$, où T est le temps total d'utilisation de réparateurs.

◇ La proportion d'utilisation d'entité ("the utilization ratio of machines" en anglais) est calculée par $\frac{N.H-T}{N.H}$.



(a) (b)

FIG. 5.3 – Résultats dans le cas où l'horizon H=90

Dans le Figure 5.2 (resp. Figure 5.3), nous donnons les résultats obtenus dans le cas où $H = 60$ (resp. $H = 90$). Le coût moyen est exprimé dans le Figure 5.2(a) et le Figure 5.3(a) alors que le Firfure 5.2(b) et le Figure 5.3(b) rapportent le temps d'utilisation moyen.

A partir des résultats obtenus, nous constatons que :
◇ Le coût moyen est décroissant quand le nombre des réparateurs est croissant.

◇ Dans la plupart des cas, le coût moyen fourni par DCA est plus petit que celui fourni par l'algorithme FTR. Particulièrement, les différences sont très importantes dans les cas où $q = 4, 6, 8, 10, 12$. Il n'y a qu'une instance ($q = 20, H = 90$) où le coût moyen fourni par DCA est plus grand que celui fourni par l'algorithme FTR. Mais la différence est très faible.

◇ Quant au temps d'utilisation des réparateurs, celui de DCA est plus grand que celui de FTR. Alors, les résultats de DCA nous permettent d'utiliser les réparateurs de manière plus efficace par rapport aux résultats de FTR.

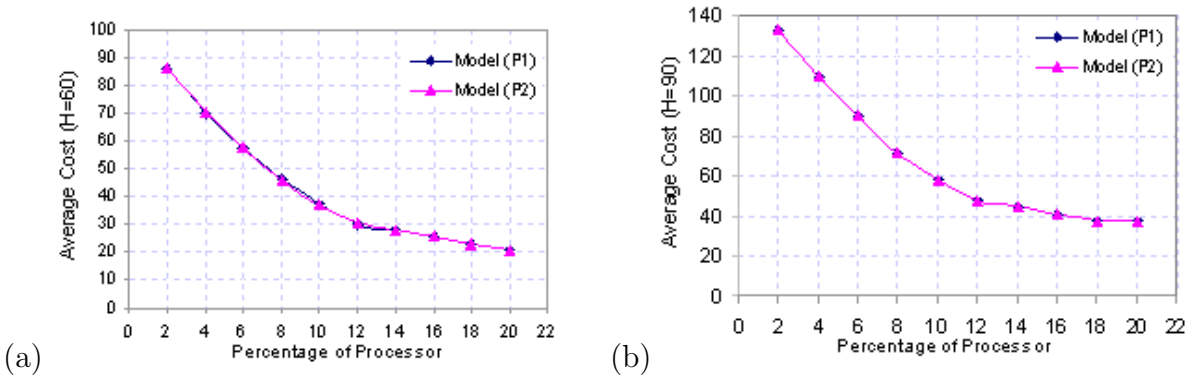


FIG. 5.4 – La comparaison entre (P_1) et (P_2)

Pour comparer les résultats donnés par le modèle (P_1) avec ceux fournis par le modèle (P_2) nous faisons varier le nombre des réparateurs, le paramètre q dans le modèle (P_1) et le paramètre q_{max} dans le modèle (P_2), entre 2 et 20. Les coûts moyens donnés par tous les deux modèles sont présentés dans le même figure. Le Figure 5.4(a) rapporte les résultats obtenus dans le cas où $H = 60$. Alors que les résultats obtenus dans le cas où $H = 90$ sont présentés dans le Figure 5.4(b). Nous constatons que les résultats donnés par deux modèles sont presque pareils. Les différences sont négligeables. De plus, dans les résultats du modèle (P_2) la valeur optimale q_{opt} est toujours égale à q_{max}. Par ailleurs, mathématiquement, le modèle (P_2) est plus général que (P_1). C'est pourquoi nous utilisons les résultats donnés par (P_2) pour l'analyse dans le reste de cette section.

Afin d'étudier l'influence des paramètres ρ_i, δ_i sur les résultats, nous réduisons ρ_i, δ_i en les multipliant avec un constant σ(sigma). Nous considérons trois cas qui correspondent respectivement à $\sigma = 0.8,\ 0.6$ et 0.5. Le paramètre q_{max} est varié entre 2 et 30, l'horizon de temps $H = 90$ (3 mois) est également choisi.

Le Figure 5.5 présente les coûts moyens dans tous les trois cas, tandis que le Figure 5.6 exprime les coûts moyens relatifs. Les proportions d'utilisation d'entités sont données dans le Figure 5.7 .

Nous constatons, à partir des résultats, que :

◇ Les coûts moyens et les coûts moyens relatifs sont croissants quand les paramètres ρ_i et δ_i sont décroissants.

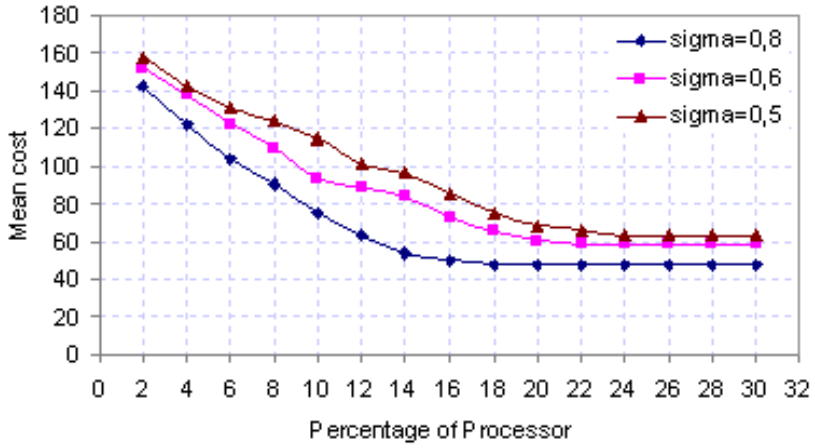


FIG. 5.5 – Les coûts moyens dans tous les trois cas

◇ Dans le cas où $\sigma = 0.8$, le coût moyen relatif est le plus petit quand le nombre des réparateurs est égal à 18. Alors, on peut considérer q=18 comme la meilleure valeur du nombre des réparateurs (q^*). Dans les cas $\sigma = 0.6$ et $\sigma = 0.5$, q^* sont 22 et 24 respectivement.

◇ Le proportion d'utilisation d'entité décroît linéairement de $q_{max} = 2$ à $q_{max} = q^*$. Après, cette proportion est constante. Ça signifie que l'augmentation du nombre des réparateurs n'entraine pas un coût meilleur si le nombre des réparateurs est déjà plus grand que q^*.

5.6 Conclusion

Dans ce chapitre, nous avons présenté, pour la première fois dans la litérature, un modèle d'optimisation déterministe pour le problème de minimisation du coût de maintenance qui est composé du temps de séjour et de la pénalité du retard. Premièrement, nous avons traité le problème dans le cas où le nombre des réparateurs est fixé. Deuxièmement, nous avons proposé le modèle dans le cas où le nombre des réparateurs est une variable au lieu d'un paramètre. Il s'agit de la programmation linéaire en variables mixtes 0-1. La méthode DCA a été développée pour les résoudre. Les algorithmes proposés ont des propriétés intéressantes, notamment il converge vers une solution locale après un nombre fini d'itération et à chaque itération nous ne résolvons qu'un problème linéaire. De plus, DCA donne toujours une solution entière bien qu'il soit une approche continue et que le nombre des variables binaires est très grand (6000 variables binaires dans le cas où H=60, 9000 variables binaires dans le cas où H=90). L'efficacité de DCA est comparée avec celle de FTR, un algorithme heuristique récent. Les résultats numériques montrent que DCA est plus efficace que FTR. Dans le futur, nous allons étudier les modèles alternatifs qui permettent de trouver un nombre des réparateurs optimal afin de minimiser le coût relatif.

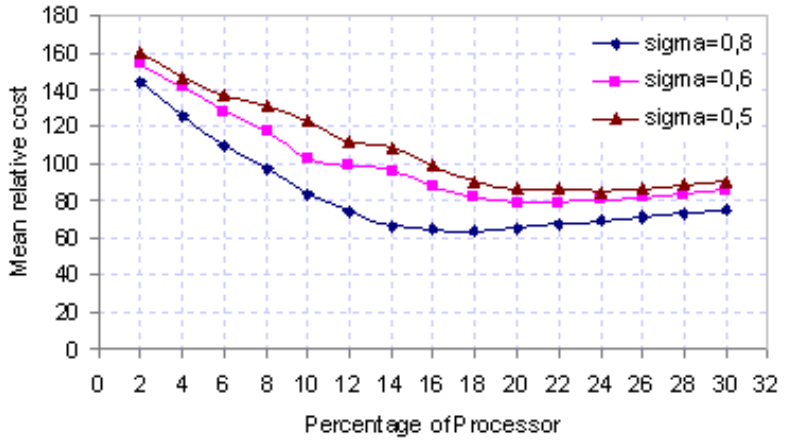


FIG. 5.6 – les couts moyens relatifs dans tous les trois cas

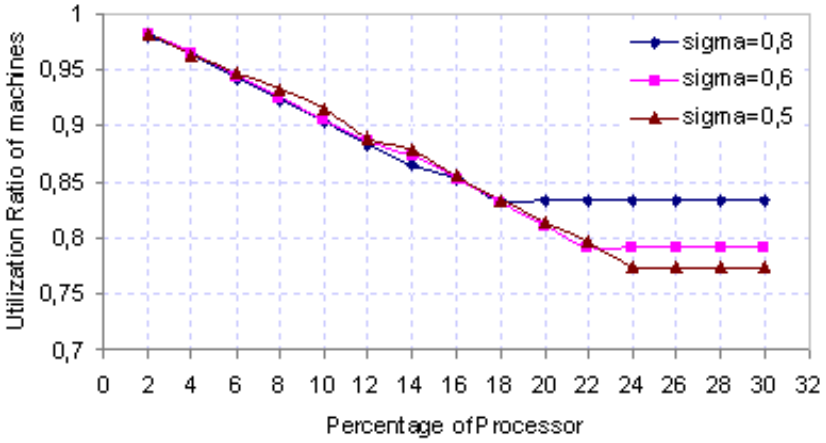


FIG. 5.7 – Les proportions d'utilisation d'entité dans tous les trois cas

Chapitre 6

Minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement

Résumé *La minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement est modélisée dans les travaux précédents sous la forme d'une programmation non linéaire en variables mixtes qui est NP-difficile. Dans ce chapitre, nous proposons deux algorithmes efficaces basés sur la programmation DC et DCA : un algorithme DCA explicite et un algorithme combiné SE-DCA. Les résultats numériques ont montré que DCA donne une solution globale dans la plupart de cas et l'algorithme combiné SE-DCA est efficient.*

6.1 Introduction

Dans un système de production multi-étapes, un produit est fabriqué par une séquence des opérations séparées. L'objectif du producteur est de déterminer la taille des lots ("lot sizing" en anglais) pour minimiser le coût comprenant celui d'installation, de transportation et de stockage. Ce problème a été étudié par nombreux chercheurs. Dans [114], l'auteur a proposé un modèle où un lot Q est partitionné en b groupes ("batches" en anglais). Evidemment, le nombre des groupes satisfait ($1 \leq b \leq Q$). L'effet de ce partitionnement est de réduire le cycle de production et le processus de stockage. Dans [115], un modèle pour le problème de production de 2 étapes a été proposé. L'idée principale est de diviser un lot en plusieurs groupes dont les tailles suivent une série géométrique. L'auteur a également démontré que son approche entraine un coût moins cher que celui obtenu en utilisant les groupes de taille égale [115, 118, 120]. En 2001, Bogaschewsky et al. ont présenté une extension du modèle introduit dans [115]. Ils ont considéré le problème de production/stockage multi-étapes. Pour la formulation, les auteurs ont adopté de différentes hypothèses. Premièrement, tous les lots ont la même taille. Deuxièmement, le producteur doit payer un coût d'installation par lot qui

dépend de l'étape. Troisièmement, les groupes peuvent être transportés à l'étape suivante. Finalement, à deux étapes différents, le nombre des groupes peut être différent. En outre, les auteurs supposent que les capacités de production de toutes les étapes soient plus grandes que la demande des clients.

Dans [117], Hsiao et al. ont proposé une extension du modèle donné par Bogaschewsky. Ils ont considéré le problème en présence de goulot d'étrangement ("bottleneck" en anglais), i.e. il existe une étape où la capacité de production est plus petite que la demande des clients. Pour la modélisation, les auteurs ont utilisé deux tailles de lot. Une taille correspond au lot aux étapes en amont ("upstream stages" en anglais). L'autre correspond au lot aux étapes en avals ("downstream stages" en anglais). Le problème proposé se modélise sous la forme d'une programmation non linéaire en variables mixtes. Pour la résolution, les auteurs ont tout d'abord divisé le modèle original en deux sous problèmes qui sont appelés le problème tiré ("pull problem" en anglais) et le problème poussé ("push problem " en anglais). Bien évidemment, ces deux sous problèmes sont aussi non linéaires et en variables mixtes. L'avantage de cette division est de réduire la dimension du problème à résoudre. Par conséquence, le temps d'exécution est diminué. Enfin, les auteurs ont donné un algorithme heuristique basé sur la méthode de division du lot ("lot size division" en anglais) et de serrage récursif ("recursive tightening" en anglais) pour résoudre ces deux sous problèmes [117, 119].

Clairement, le modèle proposé par Hsiao est plus général et plus souvent rencontré en pratique. Cependant, il est plus difficile à résoudre. La difficulté est due à la non convexité de la fonction objectif et de la présence des variables entières. D'une part, les méthodes heuristiques existantes ne donnent qu'une solution réalisable et il n'y a pas encore les outils pour évaluer la qualité des solutions obtenues. D'autre part, au point de vue d'optimisation déterministe, la résolution globale est normalement très coûteuse.

Quant à nos travaux dans ce chapitre nous étudions une nouvelle approche basée sur la programmation DC et DCA. En premier temps, nous reformulons le problème sous la forme d'un programme DC en appliquant une technique de pénalité. Le schéma de DCA est ensuite développé pour le problème résultant. Fort intéressant, à chaque itération, nous n'effectuons qu'une projection sur un rectangle. L'algorithme est donc explicite et il converge rapidement vers une solution locale. Enfin, grâce aux techniques de relaxation DC, nous considérons un problème relaxé convexe basé sur lequel nous développons un algorithme SE et celui combiné SE-DCA. Les résultats numériques ont montré l'efficacité de DCA et de l'algorithme combiné SE-DCA par rapport à celle de l'algorithme SE sans DCA.

Le reste de chapitre est organisé de façon suivante. La section 2 concerne la description du modèle mathématique. Dans la section 3, nous présentons la résolution par la programmation DC et DCA. Les résultats numériques et les commentaires sont rapportés dans la section 4 tandis que la section 5 est réservée à la conclusion.

6.2 Modèle mathématique

Dans cette section, nous présentons brièvement le modèle qui a été introduit par Hsiao [117]. Considérons un système de production/stockage où un produit est fabriqué par une séquence des opérations séparées. Chacune des opérations correspond à une étape et dans une étape le nombre des articles produits sans interruption est appelé lot. De plus, un lot est partitionné en plusieurs groupes et un groupe peut être transporté d'une étape à celle suivante. Après avoir terminé la production, les produits sont livrés aux clients. L'objectif est de déterminer la taille de lot et le nombre de groupes à chaque étape pour minimiser le coût total qui comprend tous les coûts d'installation, de stockage, de transport.

Nous supposons que le système considéré satisfasse les hypothèses suivantes :

- Le système comprend n étapes de production. L'étape $n+1$ représente la demande des clients.
- Le rythme de production ("the production rate" en anglais) de toutes les étapes et le rythme de demande ("the demand rate" en anglais) sont donnés.
- Le rythme de demande est considéré comme celui de production à l'étape $n+1$.
- Il existe un goulot d'étrangement dans le système. Cela veut dire qu'il y a au moins une étape où le rythme de production est plus petit que celui de demande.
- Les unités de production sont infiniment divisibles.
- A toutes les étapes, l'arriéré ("backlog" en anglais) n'est pas permi.
- Le producteur doit payer un coût d'installation par lot, ce coût dépend de l'étape de production.
- Un lot est divisé en plusieurs groupes dont les tailles suivent une série géométrique.
- Après avoir complété, un groupe est transporté à l'étape suivante. Le coût de transport par groupe entre deux étapes consécutives est fixé.
- Nous utilisons une seule taille de lot durant les étapes en amont à partir du goulot d'étrangement.
- Une autre taille de lot est utilisée durant les étapes en aval à partir du goulot d'étrangement.
- Toutes les deux tailles des lots en amont et en aval sont plus petites que le rythme de production au goulot d'étrangement.

Pour donner le modèle mathématique, nous utilisons les notations dans le Tableau 6.2 et les variables dans le Tableau 6.2. Nous supposons que le rythme de production à l'étape k soit le plus petit de tous les étapes i.e, $P_i \geq P_k, \forall i \neq k$. Le coût total qui comprend le coût d'installation, le coût de stockage et le coût de transport est calculé par $C_{total} = C_u + C_d$ (voir [116], [117] pour les informations plus détaillées) où

$$C_u = P_k \sum_{i=1}^{k-1} \Big\{ \frac{h_i(R_i-1)Q_u}{(R_i^{m_i}-1)[P_i]^+} + \frac{h_i}{2}\Big[\frac{1}{[P_i]^-} - \frac{1}{[P_i]^+}\Big]Q_u + \frac{S_i}{Q_u} + \frac{F_i m_i}{Q_u}\Big\},$$

$$C_d = P_k \sum_{i=k}^{n} \Big\{ \frac{h_i(R_i-1)Q_d}{(R_i^{m_i}-1)[P_i]^+} + \frac{h_i}{2}\Big[\frac{1}{[P_i]^-} - \frac{1}{[P_i]^+}\Big]Q_d + \frac{S_i}{Q_d} + \frac{F_i m_i}{Q_d}\Big\}.$$

D	le rythme de demande (unité par unité de temps) qui est constant,
P_i	le rythme de production à l'étape i (unité par unité de temps) qui est constant,
h_i	le coût de stockage au stage i (par unité par unité de temps),
S_i	le coût d'installation à l'étape i,
F_i	le coût de transport par lot entre l'étape i et l'étape $i+1$,
$[P_i]^+$	la valeur la plus grande parmi deux rythmes de production aux étapes i et $i+1$ i.e. $[P_i]^+ = \max(P_i, P_{i+1})$,
$[P_i]^-$	la valeur la plus petite parmi deux rythmes de production aux étapes i et $i+1$, i.e. $[P_i]^- = \min(P_i, P_{i+1})$,
R_i	la proportion entre deux rythmes de production aux étapes i and $i+1$ i.e. $R_i = \frac{[P_i]^+}{[P_i]^-}$.

TAB. 6.1 – Les notations utilisées

Q_u	la taille des lots en amont,
Q_d	la taille des lots en avals,
m_i	le nombre des groupes à l'étape i,

TAB. 6.2 – Les variables utilisées

Alors, nous avons un problème d'optimisation non linéaire en variables mixtes

$$\begin{cases} \min\{C_{total} = C_u + C_d\} \\ \quad t.q. \qquad 0 < Q_u, Q_d \leq P_k, \\ \qquad\qquad m_i \geq 1 \ \forall i = 1, 2, ..., n, \\ \qquad\qquad m_i \in \mathbb{N} \ \forall i = 1, 2, ..., n. \end{cases}$$

Ce problème peut être séparé en deux sous problème qui sont appelés le problème en amont et le problème en aval. Plus précisément, celui en amont s'écrit comme suit :

$$(P_u) \begin{cases} \min\left\{C_u = P_k \sum\limits_{i=1}^{k-1} \left\{ \frac{h_i(R_i-1)Q_u}{(R_i^{m_i}-1)[P_i]^+} + \frac{h_i}{2}\left[\frac{1}{[P_i]^-} - \frac{1}{[P_i]^+}\right]Q_u + \frac{S_i}{Q_u} + \frac{F_i m_i}{Q_u} \right\}\right\} \\ \quad t.q. \qquad 0 < Q_u \leq P_k, \\ \qquad\qquad m_i \geq 1 \ \forall i = 1, 2, ..., k-1, \\ \qquad\qquad m_i \in \mathbb{N} \ \forall i = 1, 2, ..., k-1. \end{cases}$$

Alors que le problème en aval est exprimé par

$$(P_d) \begin{cases} \min\left\{C_d = P_k \sum\limits_{i=k}^{n} \left\{ \frac{h_i(R_i-1)Q_d}{(R_i^{m_i}-1)[P_i]^+} + \frac{h_i}{2}\left[\frac{1}{[P_i]^-} - \frac{1}{[P_i]^+}\right]Q_d + \frac{S_i}{Q_d} + \frac{F_i m_i}{Q_d} \right\}\right\} \\ \quad t.q. \qquad 0 < Q_u \leq P_k, \\ \qquad\qquad m_i \geq 1 \ \forall i = k, k+1, ..., n, \\ \qquad\qquad m_i \in \mathbb{N} \ \forall i = k, k+1, ..., n. \end{cases}$$

Il est facile de vérifier que tous les deux problèmes (P_u) et (P_d) sont non convexes. Ainsi, nous ne présentons que la méthode de résolution du problème (P_u). Celle du problème (P_d) est similaire. Afin de faciliter la présentation nous réécrivons le problème (P_u) comme suit :

$$(P_u) \quad \begin{cases} \min \left\{ f(Q,m) = \sum\limits_{i=1}^{N} \left\{ \frac{a_i Q}{R_i^{m_i}-1} + b_i Q + \frac{c_i}{Q} + d_i \frac{m_i}{Q} \right\} \right\} \\ t.q. \qquad 0 < Q \leq P_k, \\ \qquad\quad m_i \geq 1 \ \forall i = 1,2,...,N, \\ \qquad\quad m_i \in \mathbb{N} \ \forall i = 1,2,...,N, \end{cases}$$

où $N = k-1$ et $a_i = \frac{P_k h_i (R_i - 1)}{[P_i]^+}$, $b_i = \frac{P_k h_i}{2}\left[\frac{1}{[P_i]^-} - \frac{1}{[P_i]^+}\right]$, $c_i = P_k S_i$, $d_i = P_k F_i$.

Les problèmes ci-dessus sont les problèmes d'optimisation non linéaires en variables mixtes. Ils sont très difficiles à résoudre. La difficulté vient d'un part des variables entières et d'autre part de la fonction objectif non convexe. De plus, on constate qu'ils sont de même forme.

6.3 Résolution basée sur la programmation DC et DCA

6.3.1 Résolution de (P_u) par DCA

En premier lieu, nous donnons une sous-estimation de la variable Q et une surestimation des variables m_i. Soit $(Q^0, m^0) \in \mathbb{R}_+ \times \mathbb{N}^N$ un point qui satisfait $Q^0 \leq P_k$ et $m_i^0 \geq 1 \ \forall i$ (par exemple $Q^0 = P_k$ et $m_i^0 = 1 \ \forall i$). Supposons que (Q^{opt}, m^{opt}) soit une solution optimale du problème (P_u). Par conséquence, nous avons

$$f(Q^{opt}, m^{opt}) = \sum_{i=1}^{N} \left\{ \frac{a_i Q^{opt}}{R_i^{m_i^{opt}} - 1} + b_i Q^{opt} + \frac{c_i}{Q^{opt}} + d_i \frac{m_i^{opt}}{Q^{opt}} \right\} \leq f(Q^0, m^0).$$

Alors,

$$Q^{opt} \geq \frac{1}{f(Q^0, m^0)} \sum_{i=1}^{N} (c_i + d_i) := Q^{min}.$$

Par ailleurs, $f(Q,m) = \sum\limits_{i=1}^{N} f_i(Q, m_i)$ où

$$f_i(Q, m_i) := \frac{a_i Q}{R_i^{m_i} - 1} + b_i Q + \frac{c_i}{Q} + d_i \frac{m_i}{Q}.$$

Il est clair que

$$m_i^*(Q) = \log_{R_i} \left[\left(2 + \alpha(Q) + \sqrt{\alpha^2(Q) + 4\alpha(Q)} \right) / 2 \right]$$

est un minimum de la fonction $f_i(Q, m_i)$ sur le domaine $m_i \in [1, +\infty)$ où $\alpha(Q) = a_i Q^2 \log(R_i)/d_i$. Alors,

$$\max\{1, [m_i^*(Q^{min})]\} := m_i^{min} \leq m_i^{opt} \leq m_i^{max} := [m_i^*(P_k)] + 1.$$

Ici, $[x]$ désigne la partie entière de x, i.e. le nombre entier le plus grand parmi tous les nombres entiers plus petits que x.

Soient $m^{min} = (m_1^{min}, m_2^{min}, ..., m_N^{min})^T$ et $m^{max} = (m_1^{max}, m_2^{max}, ..., m_N^{max})^T$. L'ensemble des solutions réalisables du problème (P_u) peut être écrit comme suit :

$$D = \{(Q, m) \in \mathbb{R}^{N+1} : Q^{min} \leq Q \leq Q^{max} := P_k,\ m^{min} \leq m \leq m^{max},\ m \in \mathbb{N}^N\}.$$

Conssidérons la fonction $p(m) :\ \mathbb{R}^N \longrightarrow \mathbb{R}$ définie par

$$p(m) = \sum_{i=1}^{N} \sin^2 m_i \pi.$$

Il est clair que $p(m)$ est non négative et $p(m) = 0$ si et seulement si $m \in \mathbb{N}^N$. Alors, D peut s'écrire comme

$$D = \{(Q, m) \in B,\ p(m) \leq 0\},$$

où B est le domaine relaxé de D

$$B = \{(Q, m) \in \mathbb{R}^{N+1} : Q^{min} \leq Q \leq Q^{max},\ m^{min} \leq m \leq m^{max}\}.$$

Le problème (P_u) est réécrit comme suit :

$$(P_u) \quad \min\{f(Q, m) : (Q, m) \in B,\ p(m) \leq 0\}.$$

En utilisant la technique de pénalité avec le paramètre de pénalité δ (un nombre positif) nous obtenons le problème ci-dessous (voir Section 1.4) :

$$(P) \quad \min\Big\{F(Q, m) = f(Q, m) + \delta p(m) : (Q, m) \in B\Big\}.$$

Afin d'appliquer DCA pour résoudre le problème (P) nous allons trouver une décomposition DC de la fonction $F(Q, m)$. Il est clair que $F(Q, m)$ est séparable, plus précisément

$$F(Q, m) = \sum_{i=1}^{N} F_i(Q, m_i) \text{ où } F_i(Q, m_i) = \frac{a_i Q}{R_i^{m_i} - 1} + b_i Q + \frac{c_i}{Q} + d_i \frac{m_i}{Q} + \delta \sin^2 m_i \pi.$$

Alors, la recherche d'une décomposition de $F(Q, m)$ revient à la recherche d'une décomposition de $F_i(Q, m_i)$. Nous avons

$$\frac{\partial^2 F_i}{\partial Q^2} = \frac{2c_i}{Q^3} + \frac{2d_i m_i}{Q^3}, \quad \frac{\partial^2 F_i}{\partial Q \partial m_i} = -a_i \log(R_i).[(R_i^{m_i} - 1)^{-1} + (R_i^{m_i} - 1)^{-2}] - \frac{d_i}{Q^2},$$

$$\frac{\partial^2 F_i}{\partial m_i^2} = a_i Q \log^2(R_i).[(R_i^{m_i}-1)^{-1} + 3(R_i^{m_i}-1)^{-2} + 2(R_i^{m_i}-1)^{-3}] + 2\delta\pi^2\cos(2m_i\pi).$$

Soit λ_i le nombre défini par

$$\lambda_i = \max\Big\{a_i\log(R_i)\Big[\tfrac{R_i}{(R_i-1)^2} + P_k\log(R_i).\tfrac{R_i(R_i+1)}{(R_i-1)^3}\Big] + \tfrac{d_i}{Q_{min}^2} + 2\delta\pi^2, \tfrac{R_i a_i\log(R_i)}{(R_i-1)^2} + \tfrac{d_i}{Q_{min}^2} + \tfrac{2(c_i+d_i.m_i^{max})}{Q_{min}^3}\Big\}. \quad (6.1)$$

On constate que

$$\lambda_i \geq \|\nabla^2 F(Q,m_i)\|_\infty.$$

Avec λ_i défini par (6.1), la fonction $H_i(Q,m_i) = \lambda_i Q^2 + \lambda_i m_i^2 - F_i(Q,m_i)$ est convexe. De plus, la fonction $G_i(Q,m_i) = \lambda_i(Q^2+m_i^2)$ est clairement convexe. Par conséquence, une décomposition DC de $F_i(Q,m_i)$ peut être écrite comme suit :

$$F_i(Q,m_i) = \lambda_i(Q^2+m_i^2) - (\lambda_i Q^2 + \lambda_i m_i^2 - F_i(Q,m_i)).$$

Alors, nous obtenons une décomposition DC de F : $F(Q,m) = G(Q,m) - H(Q,m)$ où

$$G(Q,m) = (\sum_{i=1}^N \lambda_i)Q^2 + \sum_{i=1}^N \lambda_i m_i^2,$$

$$H(Q,m) = (\sum_{i=1}^N \lambda_i)Q^2 + \sum_{i=1}^N \lambda_i m_i^2 - \sum_{i=1}^N \Big[\frac{a_i Q}{R_i^{m_i}-1} + b_i Q + \frac{c_i}{Q} + d_i\frac{m_i}{Q} + \delta\sin^2 m_i\pi\Big].$$

Par sa définition, $H(Q,m)$ est différentiable et $(u,v) = \nabla H(Q,m)$ est calculé par :

$$\begin{cases} u = \frac{\partial H}{\partial Q} = 2Q\sum\limits_{i=1}^N \lambda_i - \sum\limits_{i=1}^N [a_i(R^{m_i}-1)^{-1} + b_i] + Q^{-2}\sum\limits_{i=1}^N (c_i + d_i m_i), \\ v_i = \frac{\partial H}{\partial m_i} = 2\lambda_i.m_i + \frac{a_i Q R_i^{m_i}\log(R_i)}{(R_i^{m_i}-1)^2} - \frac{d_i}{Q} - \delta\pi\sin(2m_i\pi)\ \forall i = 1,2,...,N. \end{cases} \quad (6.2)$$

Selon le schéma général de DCA, l'algorithme DCA appliqué au programme DC (P) consiste en la détermination, à chaque itération ℓ, de deux suites $\{(Q^\ell, m^\ell)\}$ et $\{(u^\ell, v^\ell)\}$ telles que $(u^\ell, v^\ell) = \nabla H(Q^\ell, m^\ell)$ et $(Q^{\ell+1}, m^{\ell+1})$ est une solution optimale du problème d'optimisation convexe suivant :

$$\min\Big\{(\sum_{i=1}^N \lambda_i)Q^2 + \sum_{i=1}^N \lambda_i.m_i^2 - u^\ell Q - \sum_{i=1}^N v_i^\ell.m_i :\ (Q,m)\in B\Big\}. \quad (6.3)$$

Il est clair qu'une solution optimale de la programmation convexe (6.3) est donnée par la projection du point $(u^\ell/(2\sum\limits_{i=1}^N \lambda_i), \frac{v_1^\ell}{2\lambda_1}, ..., \frac{v_N^\ell}{2\lambda_N})$ sur le rectangle $B = [Q^{min}, Q^{max}] \times [m^{min}, m^{max}]$

(voir [30]). Plus précisément, $(Q^{\ell+1}, m^{\ell+1})$ est défini par

$$Q^{\ell+1} = \begin{cases} Q^{min} \text{ si } u^\ell/(2\sum_{i=1}^{N}\lambda_i) < Q^{min}, \\ Q^{max} \text{ si } u^\ell/(2\sum_{i=1}^{N}\lambda_i) > Q^{max}, \\ u^\ell/(2\sum_{i=1}^{N}\lambda_i), \text{ si non.} \end{cases} \quad m_i^{\ell+1} = \begin{cases} m_i^{min} \text{ si } \frac{v_i^\ell}{2\lambda_i} < m_i^{min}, \\ m_i^{max} \text{ si } \frac{v_i^\ell}{2\lambda_i} > m_i^{max}, \\ \frac{v_i^\ell}{2\lambda_i}, \text{ si non.} \end{cases} \tag{6.4}$$

Finalement, le schéma de DCA appliqué au problème (P_u) peut être décrit comme suit :

Algorithme 6.1 *Schéma de DCA appliqué à (P_u) (DCA-PRO).*

Initialization
- *Choisir la tolérance ϵ positive et suffisamment petite.*
- *Choisir $(Q^0, m^0) \in \mathbb{R}^{N+1}$ et $\ell = 0$.*

Répéter
- *Calculer $(u^\ell, v^\ell) = \nabla H(Q^\ell, m^\ell)$ via (6.2).*
- *Calculer $(Q^{\ell+1}, m^{\ell+1})$ via (6.4).*
- $\ell \longleftarrow \ell + 1$

Jusqu'à $\|(Q^{\ell+1}, m^{\ell+1}) - (Q^\ell, m^\ell)\| \leq \epsilon$ *ou* $\|F(Q^{\ell+1}, m^{\ell+1}) - F(Q^\ell, m^\ell)\| \leq \epsilon$.

Grâce aux théorèmes sur la convergence de DCA [18, 62], nous avons les propriétés suivantes :

Théorème 6.1 *(Propriété de convergence de l'Algorithme 6.1)*

i) L'Algorithme 6.1 génère une suite $\{(Q^\ell, m^\ell)\}$ telle que la suite $\{F(Q^\ell, m^\ell)\}$ est décroissante.

ii) La suite $\{(Q^\ell, m^\ell)\}$ converge vers une solution KKT (Q^*, m^*) qui satisfait la condition nésessaire d'optimalité locale

Preuve : Ces propriétés sont les conséquences directes des propriétés convergentes de la programmation DC et du fait que H(Q,m) est différentiable (voir par exemple [62]). □

6.3.2 Algorithme combiné de DCA et SE

Afin de résoudre globalement le problème (P_u), nous proposons un algorithme combiné de DCA et SE où l'algorithme DCA est utilisé pour calculer la borne supérieure. Au niveau de la borne inférieure, nous proposons un problème relaxé convexe de (P_u). Plus précisément, nous relaxons l'ensemble réalisable et remplaçons la fonction objectif de (P_u) par son minorant convexe. Pour ce travail, nous appliquons le résultat suivant [5] :

Théorème 6.2 *[5] Soient $p \geq 0, r \geq 0$ et $u(x), v(x)$ deux fonctions convexes finies sur $\mathbb{R}^n$. Alors* $\max\{ru(x) + pv(x) - pr, su(x) + qv(x) - qs\}$ *est un minorant convexe de $u(x)v(x)$ sur le domaine $\{x \in \mathbb{R}^n | p \leq u(x) \leq q,\ r \leq v(x) \leq s\}$.*

Preuve : Une démonstration est donnée dans [5]. □

Considérons $\mu(m) = \sum_{i=1}^{N} \frac{a_i}{R_i^{m_i}-1}$, $t(m) = \sum_{i=1}^{N} d_i m_i$. Il est clair que $\mu(m)$ est convexe, $t(m)$ est linéaire et

$$f(Q,m) = \mu(m)Q + (\sum_{i=1}^{N} b_i)Q + (\sum_{i=1}^{N} c_i)\frac{1}{Q} + t(m)\frac{1}{Q}.$$

En appliquant le Théorème 6.2, nous obtenons un minorant convexe de la fonction $f(Q,m)$ sur le rectangle B :

$$conv(f) = \max\Big\{\mu(m^{min})Q + Q^{max}\mu(m) - Q^{max}\mu(m^{min}), \mu(m^{max})Q + Q^{min}\mu(m) - Q^{min}\mu(m^{max})\Big\} +$$
$$\max\{t(m^{min})\tfrac{1}{Q} + \tfrac{1}{Q^{max}}t(m) - \tfrac{t(m^{min})}{Q^{max}}, t(m^{max})\tfrac{1}{Q} + \tfrac{1}{Q^{min}}t(m) - \tfrac{t(m^{max})}{Q^{min}}\} + (\sum_{i=1}^{N} b_i)Q + (\sum_{i=1}^{N} c_i)\tfrac{1}{Q}.$$

Par conséquence,

$$\eta = \min\{conv(f) : (Q,m) \in B\} \tag{6.5}$$

fournit une borne inférieure de la valeur optimale du problème (P_u). Ailleurs, le problème (6.5) est équivalent à

$$\min\Big\{x + y + (\sum_{i=1}^{N} b_i)Q + (\sum_{i=1}^{N} c_i)\frac{1}{Q}\Big\}$$

t.q.

$$\mu(m^{min})Q + Q^{max}\mu(m) - Q^{max}\mu(m^{min}) \leq x, \tag{6.6}$$
$$\mu(m^{max})Q + Q^{min}\mu(m) - Q^{min}\mu(m^{max}) \leq x, \tag{6.7}$$
$$t(m^{min})\frac{1}{Q} + \frac{1}{Q^{max}}t(m) - \frac{t(m^{min})}{Q^{max}} \leq y, \tag{6.8}$$
$$t(m^{max})\frac{1}{Q} + \frac{1}{Q^{min}}t(m) - \frac{t(m^{max})}{Q^{min}} \leq y. \tag{6.9}$$

Nous remplaçons $\mu(m)$ dans les contraintes (6.6),(6.7) par la fonction linéaire

$$\mu_1(m) = \langle \nabla\mu(m^0), m - m^0\rangle + \mu(m^0) \leq \mu(m).$$

Alors, nous avons deux nouvelles contraintes au lieu de (6.6),(6.7) :

$$\mu(m^{min})Q + Q^{max}\langle \nabla\mu(m^0), m\rangle - Q^{max}[\mu(m^{min}) + \langle \nabla\mu(m^0), m^0\rangle - \mu(m^0)] \leq x, \tag{6.10}$$

$$\mu(m^{max})Q + Q^{min}\langle \nabla\mu(m^0), m\rangle - Q^{min}[\mu(m^{max}) + \langle \nabla\mu(m^0), m^0\rangle - \mu(m^0)] \leq x. \quad (6.11)$$

Nous obtenons maintenant un problème relaxé convexe de (P_u)

$$(P_{rel}) \quad \begin{cases} \min \left\{ f_{rel}(Q, m, x, y) = x + y + (\sum_{i=1}^{N} b_i)Q + (\sum_{i=1}^{N} c_i)\frac{1}{Q} \right\} \\ s.t. \quad (6.8 - 6.11) \text{ and } (Q, m) \in B. \end{cases}$$

Pour la description de l'algorithme SE-DCA, nous utilisons les notations ci-dessous :
– $\overline{m}$ est le nombre arrondi de m,
– (Q^{opt}, m^{opt}) est la meilleure solution,
– $(Q^{R_k}, m^{R_k}, x^{R_k}, y^{R_k})$ est la solution optimale du problème (P_{rel}) sur la rectangle R_k,
– $\gamma(R_k) = f_{rel}(Q^{R_k}, m^{R_k}, x^{R_k}, y^{R_k})$ est la borne inférieure sur la rectangle R_k.
– γ, α sont respectivement la meilleure borne inférieure et la meilleure borne supérieure.
L'algorithme combiné de DCA et SE (SE-DCA) peut être décrit comme suit :

Algorithme 6.2 *(l'algorithme combiné SE-DCA)*

Initialisation :
– *Choisir* $R_0 := [lb, ub] \subset \mathbb{R}^N$ *où* $lb = m^{min}$, $ub = m^{max}$
– *Choisir un nombre positif suffisamment petit* ϵ.
Étape 1 :
– *Résoudre le problème relaxé convexe* (P_{rel}) *pour obtenir la solution optimale* (Q^{R_0}, m^{R_0}).
– *La première solution, la première borne inférieure et la première borne supérieure sont données par* $(Q^{opt} = Q^{R_0},\ m^{opt} = \overline{m}^{R_0})$ *et* $\gamma := \gamma(R_0),\ \alpha = f(Q^{opt}, m^{opt})$.
– **Si** $(\alpha - \gamma) < \epsilon(1 + \alpha)$ **Alors** *ARRÊTER*
Sinon
Résoudre le problème (P_u) *par DCA-PRO avec le point initial* (Q^{R_0}, m^{R_0}) *pour obtenir la solution* $(Q^{R_0}_{DCA}, m^{R_0}_{DCA})$.
Mettre à jour la borne supérieure $\alpha = \min\{\alpha, f(Q_{DCA}, \overline{m}_{DCA})\}$ *et mettre à jour la meilleure solution.*
FinSi
– **Si** $(\alpha - \gamma) \leq \epsilon(1 + \alpha)$ **Alors** *ARRÊTER*
Sinon
$\Re \longleftarrow \{R_0\}$
FinSi
Étape 2 :
– *Sélectionner le rectangle* $R_\ell \in \Re$ *dont* $\gamma(R_\ell)$ *est le plus petit.*
– *Sélectionner un index* $i^* \in \{1, 2, ..., N\}$ *qui satisfait* $m^{min}_{i^*} < m^{max}_{i^*}$.
– *Diviser le rectangle* R_ℓ *en deux sous-rectangles* R_{ℓ_0}, R_{ℓ_1} *via* i^* *et* $m^{R_\ell}_{i^*}$ *comme suit :*

$$lb_i^{R_{\ell_j}} = lb_i^{R_\ell},\ ub_i^{R_{\ell_j}} = ub_i^{R_\ell}\ \forall i \neq i^*,\ j = 0, 1,$$
$$lb_{i^*}^{R_{\ell_0}} = lb_{i^*}^{R_\ell},\ ub_{i^*}^{R_{\ell_0}} = [m_{i^*}^{R_\ell}],\ lb_{i^*}^{R_{\ell_1}} = [m_{i^*}^{R_\ell}] + 1,\ ub_{i^*}^{R_{\ell_1}} = ub_{i^*}^{R_\ell}.$$

– *Résoudre le problème relaxé convexe*

$$(P_{rel})\ \min\{f_{rel}(Q,m,x,y) : (Q,m,x,y) \in K, m \in R_{\ell_j}\}$$

pour obtenir la solution $(Q^{R_{\ell_j}}, m^{R_{\ell_j}})$.
– *Mettre à jour* $\alpha = \min\{\alpha, f(Q^{R_{\ell_j}}, \overline{m}^{R_{\ell_j}})\}$ *et mettre à jour la meilleure solution.*

– **Si** $\gamma(R_{\ell_j}) < (1-\epsilon)\alpha$ **Alors**
Résoudre le problème (P_u) *by DCA-PRO pour obtenir la solution* $(Q_{DCA}^{R_{\ell_j}}, m_{DCA}^{R_{\ell_j}})$.
Mettre à jour $\alpha = \min\{\alpha, f(Q_{DCA}^{R_{\ell_j}}, \overline{m}_{DCA}^{R_{\ell_j}})\}$ *mettre à jour la meilleure solution.*
FinSi
– $\Re = \{R \in \Re : \gamma(R) < \alpha\} \cup \{R_{\ell_j} : \gamma(R_{\ell_j}) < \alpha, j = 0,1\} \setminus \{R_\ell\}$.

– *Mettre à jour* $\gamma = \min\{\gamma(R) : R \in \Re\}$.
Étape 3 :
– **Si** $(\alpha - \gamma) < \epsilon(1+\alpha)$ **Alors** *ARRÊTER*
Sinon
Retourner à l'étape 2.
FinSi

6.4 Résultats numériques

Pour résoudre le problème original nous appliquons les algorithmes proposés et utilisons le logiciel COUENNE [124] afin de résoudre deux sous-problèmes (celui en amont (P_u) et celui en aval (P_d)). La solution du problème relaxé convexe est choisie comme le point initial de l'algorithme DCA. Les résultats fournis par DCA et l'algorithme combiné SE-DCA sont comparés avec ceux fournis par l'algorithme SE sans DCA. Tous les trois algorithmes sont programmés en MATLAB (R2007a) et exécutés sur un ordinateur Core2Duo 2.8GHz, RAM 2.0 GB. De plus, la limite de 3 heures a été fixée sur le temps d'exécution dont $10800(k-1)/n$ secondes pour le problème en amont et le reste du temps pour le problème en aval. Le logiciel CVX [123] est utilisé pour résoudre le problème relaxé convexe.

Nous avons testé la performance des algorithmes proposés sur les données générées aléatoirement et de différentes tailles (le nombre d'étapes $n = 5, 10, 20, 50, 100$). Dans tous les jeux de données, les rythmes de production P_i, les rythmes de demande D, sont aléatoirement générés dans $[100, 500]$. Les coûts d'installation S_i, les coûs de stockage h_i, les coûts de transport F_i sont générés dans $[10, 50]$, $[0.1, 5.0]$ et $[1.0, 10.0]$ respectivement. Pour chaque taille, nous avons testé les algorithmes sur 10 jeux de données. Les Tableaux 6.3-6.7 présentent les résultats donnés par DCA, SE, COUENNE avec même temps d'exécution. Les résultats fournis par DCA, SE-DCA, SE et COUENNE après 3 heures sont présentés dans les Tableaux 6.4-6.4. Les Tableaux 6.13-6.15 rapportent les résultats donnés par SE-DCA, SE et COUENNE au moment où SE-DCA s'arrête. Nous utilisons les notations suivantes :

- Pr : Problème,
- Av : la moyenne ("average" en anglais),
- No : le nombre des variables entières,
- $Time$: le temps d'exécution en seconde de chaque algorithme,
- Val : la valeur de la fonction objectif fournie par chaque algorithme,
- LB : la meilleure borne inférieure qui est donnée soit par SE soit par SE-DCA,
- Gap est calculé par $Gap = \frac{Val-LB}{1+Val}.100\%$ où Val et LB sont respectivement la valeur de la fonction objectiif et la borne inférieure données par chaque algorithme.
- $Gap_{DCA} = \frac{Val_{DCA}-FirstLB}{1+Val_{DCA}}.100\%$ où Val_{DCA} est la valeur de la fonction objectif donnée par **DCA** et $FirstLB$ est la première borne inférieure qui est obtenue en résolvant le problème relaxé.
- Le symbole NA dans les colonnes Gap et Val de **COUENNE** signifie que **COUENNE** ne trouve aucune solution réalisable au moment où DCA trouve une solution approximée.

Pr	No	**DCA**			**SE**		**COUENNE**	
		Val	Gap_{DCA}	Time	Val	Gap	Val	Gap
1	5	604,17	12,95	0,0780	610,17	13,80	NA	NA
2	5	671,54	11.50	0.1092	741.23	19.81	681.22	7.18
3	5	411.82	7.74	0.0312	466.62	18.55	NA	NA
4	5	589.22	13.09	0.0468	601.39	14.84	NA	NA
5	5	509.33	10.24	0.0468	533.47	14.29	NA	NA
6	5	479.14	6.13	0.0312	531.70	15.39	NA	NA
7	5	349.69	4.80	0.0312	364.88	8.75	NA	NA
8	5	526.31	10.90	0.0312	565.89	17.12	NA	NA
9	5	414.89	7.00	0.0156	448.87	14.02	NA	NA
10	5	457.55	9.75	0.0624	524.66	21.27	NA	NA
Av	5	501.37	9.41	0.0484	538.89	15.78	//	//

TAB. 6.3 – Les résultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 5$

Commentaires sur les résultats

A partir des résultats obtenus, nous constatons que :
- DCA est très rapide : le temps d'exécution de DCA varie entre 0.0010 et 0.2340 secondes. Pour la plupart des jeux de données, il est plus petit que 0.1 secondes. En outre, pour résoudre un problème de 100 variables entières, le temps moyen d'exécution n'est que 0.1607 secondes.
- Les solutions fournies par DCA sont très bonnes : la moyenne de Gap (par rapport à la première borne inférieure) est environ 8.0%.
- Pour 43/50 jeux de données, COUENNE ne peut pas trouver une solution réalisable au moment où DCA fournit une bonne solution approximée.
- L'algorithme combiné SE-DCA est beaucoup meilleur que l'algorithme SE sans DCA. Nous constatons que le temps d'exécution moyen de SE-DCA est plus petit que celui de SE tandis que la solution de SE-DCA est meilleure que celle donnée par SE. Ce fait montre

Pr	No	**DCA**			**SE**		**COUENNE**	
		Val	Gap_{DCA}	Time	Val	*Gap*	Val	Gap
1	10	681.58	5.15	0.0780	759.64	14.88	NA	NA
2	10	945.29	9.26	0.0780	1106.29	22.46	NA	NA
3	10	831.21	8.33	0.0312	905.65	15.85	NA	NA
4	10	1019.72	12.39	0.0624	1063.45	15.99	NA	NA
5	10	951.51	6.87	0.0624	1075.19	17.57	NA	NA
6	10	750.00	5.21	0.0312	866.00	17.89	NA	NA
7	10	831.32	6.79	0.0010	861.99	10.11	NA	NA
8	10	906.79	9.03	0.0468	981.89	15.98	NA	NA
9	10	1237.83	10.66	0.1560	1403.80	21.22	1233.76	12.84
10	10	1029.16	11.20	0.0780	1088.87	16.06	NA	NA
Av	10	918.44	8.49	0.0624	1011.28	16.80	//	//

TAB. 6.4 – Les résultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 10$

Pr	No	**DCA**			**SE**		**COUENNE**	
		Val	Gap_{DCA}	Time	Val	*Gap*	Val	Gap
1	20	1611,04	5.00	0.0468	1704.00	10.18	NA	NA
2	20	1960.44	8.55	0.0624	2094.15	14.38	NA	NA
3	20	1622.98	6.20	0.0468	1780.58	14.50	NA	NA
4	20	1782.90	6.46	0.0312	1960.91	14.95	NA	NA
5	20	2217.54	10.56	0.0936	2362.95	16.06	2225.23	21.48
6	20	2098.54	10.20	0.1404	2185.93	13.79	2106.61	19.18
7	20	2121.07	8.74	0.0624	2339.25	17.24	NA	NA
8	20	1998.31	8.79	0.0312	2148.38	15.16	NA	NA
9	20	1689.47	6.09	0.0936	1824.57	13.04	NA	NA
10	20	1898.80	7.98	0.0780	2038.76	14.30	NA	NA
Av.	20	1900.11	7.86	0.0686	2043.95	14.36	//	//

TAB. 6.5 – Les résultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 20$

l'efficacité de DCA : grâce à DCA, l'algorithme combiné SE-DCA trouve une bonne borne supérieure et réduit ainsi rapidement le *Gap*.

- Pour les problème de petite taille ($n = 5, 10$), COUENNE est meilleure que SE-DCA et SE. Il trouve une solution globale dans un temps d'exécution plus court.
- Dans les cas $n = 20, 50, 100$, au moment où SE-DCA s'arrête, le Gap de SE est meilleur que celui de COUENNE alors que la valeur de la fonction objectif donnée par COUENNE est meilleure. Cela montre la supériorité de la borne inférieure proposée à l'égard de celle utilisée dans COUENNE.
- Pour les problèmes de grande taille ($n = 50, 100$), le Gap fourni par SE-DCA est le meilleure parmi les trois algorithmes globale SE-DCA, SE, COUENNE et celui de

Pr	No	**DCA**			**SE**		**COUENNE**	
		Val	Gap_{DCA}	Time	Val	*Gap*	Val	Gap
1	50	4131.02	6.11	0.1404	4553.39	14.81	4403.42	21.77
2	50	4401.03	8.35	0.1404	4842.00	16.70	NA	NA
3	50	4094.90	8.74	0.0468	4445.40	15.93	NA	NA
4	50	4983.90	9.06	0.1404	5444.97	16.76	NA	NA
5	50	4507.86	7.82	0.1872	5028.27	17.35	4503.89	22.67
6	50	4872.33	11.06	0.1560	5314.96	18.46	NA	NA
7	50	4899.07	7.59	0.0936	5371.93	15.72	NA	NA
8	50	4836.84	9.88	0.1560	5350.19	18.52	4824.30	22.19
9	50	4155.28	5.62	0.0780	4627.04	15.25	NA	NA
10	50	4571.79	6.99	0.0936	4978.41	14.59	NA	NA
Av.	50	4545.40	8.12	0.1232	4995.65	16.41	//	//

TAB. 6.6 – Les réesultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 50$

Pr	No	**DCA**			**SE**		**COUENNE**	
		Val	Gap_{DCA}	Time	Val	*Gap*	Val	Gap
1	100	8668.41	6.96	0.2340	9557.82	15.62	NA	NA
2	100	7877.99	6.16	0.1716	8546.41	13.50	NA	NA
3	100	8846.81	7.45	0.2028	9724.32	15.80	NA	NA
4	100	7864.49	5.71	0.1560	8617.34	13.95	NA	NA
5	100	7695.70	5.37	0.0936	8604.77	15.36	NA	NA
6	100	8879.26	7.01	0.2028	9860.67	16.27	NA	NA
7	100	8550.25	7.26	0.1092	9436.02	15.96	NA	NA
8	100	9006.53	7.90	0.1560	9840.82	15.70	NA	NA
9	100	8986.65	7.10	0.1248	9775.30	14.59	NA	NA
10	100	8873.96	8.31	0.1560	9823.99	17.18	NA	NA
Av.	100	8525.01	6.92	0.1607	9378.75	15.39	//	//

TAB. 6.7 – Les résultats de DCA, SE et COUENNE avec le même temps d'exécution dans le cas où $n = 100$

COUENNE est le pire (voir les Figure 6.1-6.2). Avec la limite de temps de 3 heures, la valeur de la fonction objectif fournie par SE-DCA est la meilleure et celle de SE est la pire.

– Dans le cas où $n = 100$, SE-DCA et DCA sont beaucoup meilleurs que BB et COUENNE. La différence entre la solution de DCA et celle de SE-DCA est négligeable alors que le temps d'exécution de SE-DCA est beaucoup plus grand. Il est donc intéressant d'appliquer SE-DCA afin de résoudre les problèmes de taille moyenne, et d'utiliser COUENNE pour les problèmes de petite taille. Pour les problèmes de grande taille, nous conseillons d'utiliser l'algorithme DCA.

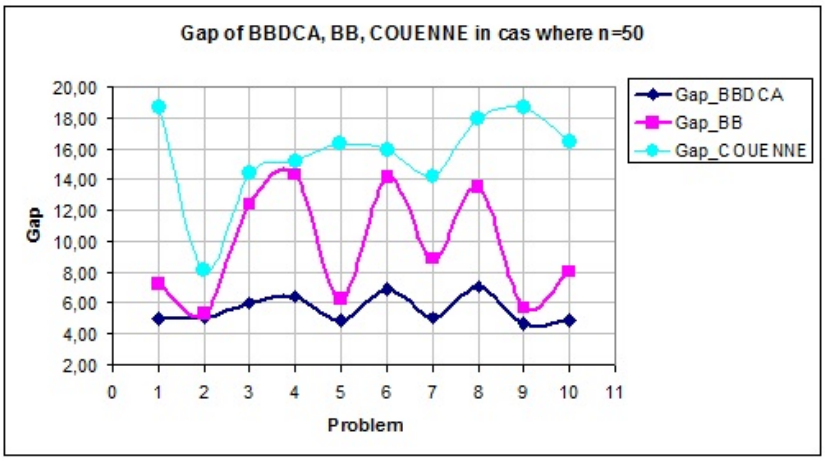


FIG. 6.1 – Comparaison entre SEDCA, SE et COUENNE dans le cas où $n = 50$.

6.5 Conclusion

Dans ce chapitre, nous avons présenté deux algorithmes efficaces pour le problème de minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement. Ce problème se modélise sous la forme d'une programmation non linéaire en variables mixtes. Premièrement, il est reformulé comme une programmation DC. L'algorithme DCA est ensuite développé pour résoudre le problème résultant. Finalement, nous combinons DCA et l'algorithme SE (SE-DCA) dans le but de trouver une résolution globale. Les résultats numériques comparés avec COUENNE ont montré que DCA réalise très bien le compromis entre la rapidité et la qualité de solution : il trouve une bonne solution approximée dans un temps d'exécution petit. Grâce à l'efficacité de DCA l'algorithme combiné SE-DCA est beaucoup meilleur que l'algorithme SE sans DCA : SE-DCA donne une solution meilleure que celle de SE malgré que son temps d'exécution soit beaucoup plus petit. Dans le futur, nous allons développer l'algorithme DCA et SE-DCA pour d'autres problèmes dans le système de production et dans la chaîne d'approvisionnement.

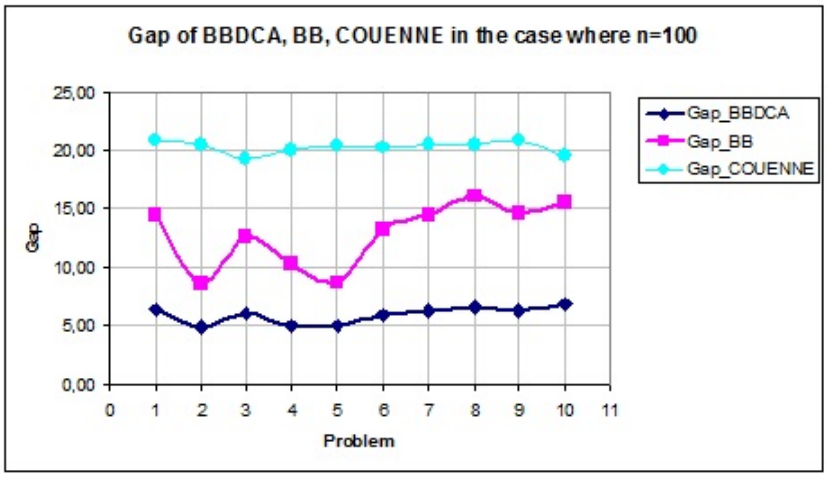


FIG. 6.2 – Comparaison entre SEDCA, SE et COUENNE dans le cas où $n = 100$.

Pr	No	**DCA**			**SEDCA**			**SE**			**COUENNE**		
		Val	Time	Gap_{DCA}	Val	Time	Gap	Val	Time	Gap	Val	Time	Gap
1	5	604.17	0.0780	12.95	604.17	8.53	0.81	604.17	11.93	0.81	604.17	0.39	0.00
2	5	671.54	0.1092	11.50	668.30	4.91	2.79	668.30	6.21	2.79	667.47	1.048	0.00
3	5	411.82	0.0312	7.74	406.05	1.81	2.80	406.91	2.92	3.00	404.07	0.407	0.00
4	5	589.22	0.0468	13.09	587.53	8.60	2.80	587.53	10.80	2.80	584.3	0.6	0.00
5	5	509.33	0.0468	10.24	506.00	5.34	3.08	506.00	6.93	3.08	504.3	0.348	0.00
6	5	479.14	0.0312	6.13	479.14	2.96	3.47	485.48	5.85	4.73	476.64	0.243	0.00
7	5	349.69	0.0312	4.80	349.70	1.25	3.59	351.88	1.50	4.19	347.01	0.299	0.00
8	5	526.31	0.0312	10.94	519.81	5.66	2.57	521.77	6.40	2.93	519.81	0.369	0.00
9	5	414.89	0.0156	7.00	412.87	1.83	2.95	412.87	2.23	2.95	410.11	0.315	0.00
10	5	457.55	0.0624	9.75	452.50	6.65	4.60	452.50	7.80	4.60	447.34	0.951	0.00
Av.	5	501.37	0.0484	9.41	498.61	4.75	2.95	499.74	6.26	3.19	496.52	0.4970	0.00

TAB. 6.8 – Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 5$

Pr	No	**DCA**			**SEDCA**			**SE**			**COUENNE**		
		Val	Time	Gap_{DCA}	Val	Time	Gap	Val	Time	Gap	Val	Time	Gap
1	10	681.58	0.0780	5.15	681.58	4.34	3.64	685.59	5.65	4.21	677.27	1.709	0.00
2	10	945.29	0.0780	9.26	943.93	13.88	3.12	941.39	17.97	2.86	935.98	1.318	0.00
3	10	831.21	0.0312	8.33	829.66	21.76	4.59	828.37	35.02	4.45	821.77	16.12	0.00
4	10	1019.72	0.0624	12.29	1012.47	21.28	4.26	1012.47	26.35	4.30	1011.72	2.985	0.00
5	10	951.51	0.0624	6.87	949.53	5.27	4.00	948.90	10.76	3.95	946.37	1.084	0.00
6	10	750.00	0.0312	5.21	750.01	1.79	2.65	760.06	19.25	3.93	744.48	2.173	0.00
7	10	831.32	0.0010	6.79	831.32	2.84	3.60	831.90	4.40	3.66	827.38	0.978	0.00
8	10	906.79	0.0468	9.03	906.79	6.40	4.23	913.31	10.65	4.91	900.31	1.203	0.00
9	10	1237.83	0.1560	10.66	1237.83	25.16	3.91	1241.12	44.10	4.17	1231.69	2.835	0.00
10	10	1029.16	0.0780	12.20	1022.41	36.88	3.33	1022.41	45.41	3.35	1018.21	1.337	0.00
Av	10	918.44	0.0624	8.49	916.55	13.96	3.73	918.55	21.96	3.98	911.52	3.17	0.00

TAB. 6.9 – Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 10$

Pr	No	**DCA**			**SEDCA**			**SE**			**COUENNE**		
		Val	Time	Gap_{DCA}	Val	Time	Gap	Val	Time	Gap	Val	Time	Gap
1	20	1611.04	0.0468	5.00	1611.04	4.51	4.76	1617.48	22.39	4.44	1615.85	4.64	3h
2	20	1960.44	0.0624	8.55	1955.56	48.03	4.95	1953.33	83.48	4.67	1943.09	1.63	3h
3	20	1622.98	0.0468	6.20	1618.87	25.55	4.63	1620.49	32.39	4.38	1609.56	3.72	3h
4	20	1782.90	0.0312	6.46	1782.90	10.55	4.60	1796.52	39.56	4.91	1779.46	0.00	8247.89
5	20	2217.54	0.0936	10.56	2217.54	120.50	4.26	2219.68	157.70	4.35	2214.83	4.16	3h
6	20	2098.54	0.1404	10.20	2098.54	659.21	4.87	2108.47	1336.51	4.83	2098.18	6.21	3h
7	20	2121.07	0.0624	8.74	2108.95	44.91	4.46	2120.44	93.01	4.75	2097.84	3.72	3h
8	20	1998.31	0.0312	8.79	1993.39	506.24	4.91	1994.65	776.93	4.85	1991.97	0.00	0.24
9	20	1689.47	0.0936	6.09	1689.47	7.04	4.44	1708.53	38.86	4.49	1689.94	6.60	3h
10	20	1898.80	0.0780	7.98	1898.80	49.14	3.95	1911.52	100.01	4.31	1889.95	0.00	234.25
Av	20	1900.11	0.0686	7.86	1897.51	147.57	4.58	1905.11	268.08	4.60	1893.07	3.07	//

TAB. 6.10 – Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 20$

Pr	No	**DCA**			**SEDCA**			**SE**			**COUENNE**		
		Val	Time	Gap_{DCA}	Val	Time	Gap	Val	Time	Gap	Val	Time	Gap
1	50	4131.02	0.1404	6.11	4124.72	1499.98	4.96	4249.88	3h	7.24	4133.60	18.70	3h
2	50	4401.03	0.1404	8.35	4387.09	2230.38	5.00	4440.08	3h	5.27	4403.41	8.09	3h
3	50	4094.90	0.0468	8.74	4076.33	3h	6.03	4372.74	3h	12.39	4077.19	14.43	3h
4	50	4983.90	0.1404	9.06	4979.29	3h	6.39	5442.13	3h	14.36	4965.90	15.20	3h
5	50	4507.86	0.1872	7.82	4472.71	5684.30	4.89	4546.49	3h	6.25	4493.97	16.35	3h
6	50	4872.33	0.1560	11.06	4838.57	3h	6.89	5250.76	3h	14.20	4830.35	15.91	3h
7	50	4899.07	0.0936	7.59	4874.49	3h	5.05	5087.91	3h	8.87	4872.79	14.22	3h
8	50	4836.84	0.1560	9.88	4822.66	3h	7.03	5200.02	3h	13.55	4770.30	17.92	3h
9	50	4155.28	0.0780	5.62	4145.17	5.89	4.63	4227.85	3h	5.67	4172.28	18.67	3h
10	50	4571.79	0.0936	6.99	4571.79	5224.63	4.87	4739.69	3h	8.03	4576.45	16.49	3h
Av	50	4545.40	0.1232	8.12	4529.28	//	5.57	4755.75	//	9.58	4529.62	15.60	//

TAB. 6.11 – Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 50$

Pr	No	**DCA**			**SEDCA**			**SE**			**COUENNE**		
		Val	Time	Gap_{DCA}	Val	Time	Gap	Val	Time	Gap	Val	Time	Gap
1	100	8668.41	0.2340	6.96	8668.41	3h	6.36	9508.06	3h	14.51	8662.07	20.83	3h
2	100	7877.99	0.1716	6.16	7870.55	7616.69	4.92	8201.86	3h	8.56	7923.19	20.47	3h
3	100	8846.81	0.2028	7.45	8827.13	3h	5.99	9493.60	3h	12.58	8851.04	19.23	3h
4	100	7864.49	0.1560	5.71	7854.80	1657.36	4.93	8336.03	3h	10.25	7874.65	19.97	3h
5	100	7695.70	0.0936	5.73	7695.81	1113.31	4.98	8043.88	3h	8.65	7709.41	20.40	3h
6	100	8879.26	0.2028	7.01	8874.62	3h	5.94	9634.11	3h	13.24	8876.21	20.22	3h
7	100	8550.25	0.1092	7.26	8549.44	3h	6.26	9378.66	3h	14.45	8561.76	20.47	3h
8	100	9006.53	0.1560	7.90	8999.05	3h	6.64	10012.33	3h	16.07	8999.55	20.44	3h
9	100	8986.65	0.1248	7.10	8976.33	3h	6.22	9862.75	3h	14.59	9009.62	20.78	3h
10	100	8873.96	0.1560	8.31	8871.19	3h	6.84	9788.45	3h	15.54	8843.70	19.55	3h
Av.	100	8525.01	0.1607	6.92	8518.73	//	5.91	9225.97	//	12.84	8531.12	20.24	//

TAB. 6.12 – Les résultats de DCA, SEDCA, SE et COUENNE après 3 heures dans le cas où $n = 100$

Pr	No	**SEDCA**		**SE**		**COUENNE**	
		Val	*Gap*	Val	*Gap*	Val	*Gap*
1	20	1611.04	4.76	1676.75	8.39	1619.23	11.33
2	20	1955.56	4.95	1991.64	6.83	1943.09	9.64
3	20	1618.87	4.63	1620.49	4.36	1614.82	7.55
4	20	1782.90	4.60	1878.12	9.70	1787.51	6.69
5	20	2217.54	4.26	2219.68	4.88	2218.40	6.80
6	20	2098.54	4.87	2108.96	5.26	2100.74	8.02
7	20	2108.95	4.46	2176.97	7.91	2101.62	9.13
8	20	1993.39	4.91	2018.74	6.08	1991.97	0.00
9	20	1689.47	4.44	1736.91	7.49	1698.10	11.33
10	20	1898.80	3.95	1956.83	7.74	1891.40	2.57
Av	20	1897.51	4.58	1938.51	6.86	1896.69	7.30

TAB. 6.13 – Les résultats de SEDCA, SE et COUENNE au moment où SEDCA s'arrête dans le cas $n = 20$

Pr	No	**SEDCA**		**SE**		**COUENNE**	
		Val	*Gap*	Val	*Gap*	Val	*Gap*
1	50	4124.72	4.96	4313.92	8.90	4135.83	19.25
2	50	4387.09	5.00	4571.08	8.78	4403.41	8.34
3	50	4076.33	6.03	4372.74	12.39	4077.19	14.43
4	50	4979.29	6.39	5442.13	14.36	4965.90	15.20
5	50	4472.71	4.89	4645.13	8.42	4493.97	17.07
6	50	4838.57	6.89	5250.76	14.20	4830.35	15.91
7	50	4874.49	5.05	5087.91	8.87	4872.79	14.22
8	50	4822.66	7.03	5200.02	13.55	4770.30	17.92
9	50	4145.17	4.63	4533.73	13.29	4172.28	21.19
10	50	4571.79	4.87	4781.64	8.98	4576.45	16.80
Av	50	4529.28	5.57	4819.91	11.17	4529.85	16.03

TAB. 6.14 – Les résultats de SEDCA, SE et COUENNE au moment où SEDCA s'arrête dans le cas $n = 50$

Pr	No	**SEDCA**		**SE**		**COUENNE**	
		Val	*Gap*	Val	*Gap*	Val	*Gap*
1	100	8669.41	6.36	9508.06	14.51	8662.07	20.83
2	100	7870.55	4.92	8201.86	8.58	7923.19	20.57
3	100	8827.13	5.99	9493.60	12.58	8851.04	19.23
4	100	7854.80	4.93	8388.66	10.87	7876.39	20.53
5	100	7695.81	4.98	8168.15	10.32	7709.41	20.73
6	100	8874.62	5.94	9634.11	13.24	8876.21	20.22
7	100	8549.44	6.26	9378.66	14.45	8561.76	20.47
8	100	8999.05	6.64	10012.33	16.07	8999.55	20.44
9	100	8976.33	6.22	9862.75	14.59	9009.62	20.78
10	100	8871.19	6.84	9788.45	15.54	8843.70	19.55
Av	100	8518.73	5.91	9243.66	13.08	8531.29	20.34

TAB. 6.15 – Les résultats de SEDCA, SE et COUENNE au moment où SEDCA s'arrête dans le cas $n = 100$

Chapitre 7

Détermination des prix de transfert et des politiques de stockage pour une chaîne d'approvisionnement de deux entreprises

Résumé *Ce chapitre aborde le problème de détermination des prix de transfert et des politiques de stockage pour une chaîne d'approvisionnement de deux entreprises. En 2002, Jonatan a formulé ce problème sous la forme d'une programmation non linéaire en variables mixtes (PNLM) et proposé un algorithme pour le résoudre [128]. L'algorithme de Jonatan est très coûteux car il exige de résoudre une suite de problème linéaire en variable mixtes. Dans ce travail, nous proposons une approche continue basée sur la programmation DC et DCA. En premier temps, une méthode DCA multi-redémarré est donnée dans le but de trouver une bonne solution réalisable. Nous proposons ensuite un problème relaxé basé sur lequel un algorithme SE et un algorithme combiné SE-DCA sont développés. Les résultats numériques montrent que l'algorithme combiné SE-DCA est efficient et DCA est aussi très efficace.*

7.1 Introduction

Aujourd'hui, la concurrence sur le marché est de plus en plus intense. Les clients s'attendent à profiter des meilleurs services avec les prix les moins chers. Alors, d'une part les entreprises doivent assurer les meilleurs services aux clients. D'autre part, elles essaient d'obtenir un bénéfice le plus haut possible. Dans ce contexte, une gestion efficace de leur chaîne d'approvisionnement est devenue un facteur crucial de la réussite. En général, la gestion des chaînes d'approvisionnement est un ensemble de méthode utilisées pour intégrer efficacement les fournisseurs, les manufacturiers, les entrepôts, les distributeurs, les détaillants et les clients de manière à produire et à distribuer les bonnes quantités de produits, aux bons endroits et au bon moment pour réduire les coûts inhérents à l'ensemble du système, tout

en satisfaisant les niveaux de services désirés des clients. Plus précisément, la gestion de la chaîne d'approvisionnement comprend des façons d'améliorer l'efficience dans les activités suivantes :

- prévision et planification de l'équilibre entre l'offre et la demande ;
- localisation des sources d'approvisionnement en matières premières ;
- fabrication du produit ;
- stockage du produit ;
- livraison et distribution du produit ;
- retour éventuel du produit par le client ;

Le but primordial d'une chaîne d'approvisionnement multi-entreprises est de déterminer une politique de production/stockage/distribution afin de maximiser la performance de la chaîne d'approvisionnement. Normalement, on veut maximiser le bénéfice total en assurant l'égalité de bénéfice entre les sous entreprises. Cela a été étudié et montré par nombreux travaux [125, 126, 134]. Dans [135], Chau Chan et Proth ont proposé un modèle de partage des risques et des bénéfices dans une chaîne d'approvisionnement. Ils ont considéré une chaîne d'approvisionnement qui comprend une grossiste et un détaillant. La demande du client est supposé dépendante au prix. L'objectif est de maximiser le bénéfice total en sorte que la grossiste et le détaillant aient un bénéfice proportionnel à leur investissement [137]. Cependant, les auteurs ont considéré un seul produit et single période alors que les firmes préfèrent souvent un plan optimal de production dans un horizon de temps. Ainsi, nous avons besoin d'étudier des modèles multi périodes pour le problème considéré. Par ailleurs, le prix de transfert de produit entre deux sous entreprises est largement utilisé en pratique (voir [127, 130, 131, 133, 136] pour avoir plus d'informations sur le prix de transfert). Particulièrement, les recherches récentes ont montré que le prix de transfert joue un rôle important pour distribuer plus justement le bénéfice entre les sous entreprises. Cela explique la nécessité de l'utilisation du prix de transfert dans les modèles de chaîne d'approvisionnement multi entreprises. En outre, en raison des problèmes fiscaux le siège social donne souvent des niveaux de prix de transfert pour chaque produit aux sous entreprises. De plus, la présence des niveaux des prix de transfert réduit le temps de négociation entre deux sous entreprises.

En ce qui concerne nos travaux, nous considérons une chaîne d'approvisionnement de deux entreprises. Une entreprise qui fabrique les produits et l'autre les distribue aux clients. L'objectif des gestionnaires est de déterminer le prix de transfert de chaque produit, le plan de production/stockage/transport dans un horizon de temps pour maximiser le bénéfice total en assurant l'équilibre de distribution des bénéfices entre deux sous entreprises. Dans la littérature, les modèles mathématiques de ce problème ont été présentés par Jonatan et al. [128] en 2002. Concernant les prix de transfert, les auteurs ont introduit plusieurs niveaux pour chaque produit et y choisi le meilleur. Par rapport aux modèles, ils ont proposé trois approches suivantes :

- La première est l'approche single qui consiste à maximiser le bénéfice total de deux sous entreprises. Cette approche se ramène à un problème d'optimisation linéaire en variables mixtes 0-1(PLM). Pour le résoudre, il existe maintenant des logiciels, par exemple Cplex,

Lingo,...Pourtant, les résultats obtenus ont montré que la distribution de bénéfice de deux sous entreprises est inéquilibrée. Ceci n'est pas surprenant car dans ce modèle on s'intéresse seulement au bénéfice total.

– La deuxième approche est proposée pour éviter l'inéquilibre de bénéfice entre deux sous entreprises. L'idée principale est de baser sur la fonction de négociation [129] dans la théorie des jeux qui a été introduit par Nash. Cette approche est appelée l'approche Nash. La fonction objectif dans ce cas est le produit $(\Pi_A - \Pi_A^L)(\Pi_B - \Pi_B^L)$ où Π_A^L, Π_B^L sont les bénéfices minimums exigés par les entreprises A et B respectivement, et Π_A, Π_B sont les bénéfices de A et B. Cette fonction objectif est non linéaire, non convexe. Il s'agit d'un problème d'optimisation non linéaire en variables mixtes (PNLM).
– La troisième approche est celle qui mélange l'approche single et l'approche Nash. Plus précisément, on maximise le bénéfice total sous la contrainte que le bénéfice de chaque entreprise doit être supérieur au bénéfice donné par l'approche Nash.

Pour résoudre le problème d'optimisation non linéaire en variables mixtes dans la deuxième approche, Jonatan et al. ont proposé un algorithme qui consiste à résoudre une suite des programmations linéaires en variables mixtes 0-1. Les résultats numériques ont montré que l'approche Nash fournit un bénéfice total très proche de celui fourni par l'approche single [128]. De plus, la distribution de bénéfice donné par l'approche Nash est beaucoup plus équilibrée. Ainsi, l'approche Nash est une approche cruciale pour le problème considéré. Néamoins, l'algorithme existant proposé par Jonatan est très compliqué et très coûteux. C'est pourquoi nous nous intéressons à l'approche Nash et développons les algorithmes plus efficaces.

Notre idée est de profiter des avantages de la programmation DC pour traiter directement le problème PNLM. En premier lieu, nous le reformulons comme une programmation DC en appliquant une technique de pénalité. La méthode DCA est ensuite développée pour résoudre le problème résultant. A chaque itération, nous ne résolvons qu'une programmation quadratique convexe au lieu d'une programmation linéaire en variables mixtes comme dans la méthode de Jonatan. Le temps d'exécution de DCA est très court. En outre, les résultats numériques nous montrent que la solution de DCA est assez bonne. Dans l'objectif de donner une résolution globale, nous proposons une relaxation convexe et un algorithme SE. Finalement, dans le but de profiter de l'efficacité de DCA nous combinons DCA et SE. Les résultats numériques de tous les trois algorithmes DCA, SE, SE-DCA sont également étudiés avec soin. La comparaison a montré que l'algorithme combiné SE-DCA est le meilleur parmi les trois et DCA est aussi très efficace.

Le reste de ce chapitre est organisé comme suit. La section suivante est consacrée à la description du modèle mathématique. La section 7.3 discute la méthode de résolution alors que la section 7.4 rapporte les résultats numériques. La conclusion est donnée dans la dernière section.

7.2 Formulation mathématique

Dans cette section, nous représentons le modèle mathématique qui a été introduit par Jonatan [128]. Des contraintes, des variables inutiles sont éliminés pour réduire le nombre de contraintes et le nombre de variables. Pour faciliter la compréhension et pour éclairer la présentation nous rappelons le problème.

Nous considérons une chaîne d'approvisionnement comprenant deux entreprises A et B. L'entreprise A fabrique et livre les produits à l'entreprise B. Après, L'entreprise B les vend aux clients avec les demandes déterministes durant l'horizon de temps $[0, T]$. Nous supposons que l'entreprise A manufacture I produits à partir de R matériels ("resources" en anglais). Le prix de chaque produit que A vend à B est choisi parmi K niveaux de prix de transfert. En outre, les entreprises doivent stocker un nombre de produits qui est supérieur à un seuil donné qui est appelé le seuil minimum de stockage. Naturellement, l'entreprise A doit payer le coût de fabrication (y compris le coût de matériel), le coût d'administration, le coût de stockage et le coût de transport. L'entreprise B doit payer le coût d'administration, le coût de transport, le coût de stockage et le coût des produits achetés de A. Concernant le revenu, celui de A est déterminé par les coûts de produits vendus à B tandis que celui de B est déterminé par les prix de produits sur le marché. L'objectif des gestionnaires de la chaîne d'approvisionnement est de trouver un plan de production, de stockage, de transport et les niveaux de prix de transfert tels que toutes les demandes de client sont satisfaites et la distribution de bénéfice de A et B est équilibrée.

Pour la formulation, nous utilisons les notations suivantes :

- $UR_{i,r}$: la quantité du matériel r pour la fabrication d'un produit i.
- $R_{r,t}$: la capacité du matériel r durant période t.
- MFC_i : le coût de fabrication du produit i.
- TC_i^A : le coût de transport du produit i que l'entreprise A doit payer.
- TC_i^B : le coût de transport du produit i que l'entreprise B doit payer.
- H_i^A : le coût d'administration du produit i que l'entreprise A doit payer.
- H_i^B : le coût d'administration du produit i que l'entreprise B doit payer.
- $\hat{P}_{i,k}$: le niveau k ième du prix de transfert du produit i.
- $D_{i,t}$: la demande du client pour le produit i durant période t.
- σ_i^A, σ_i^B : les seuils minimaux de stockage du produit i en A et B respectivement.
- ρ_i : le prix du produit i sur le marché.
- h_c : le constant basé sur lequel le coût de stockage est calculé.
- Π_A^L : le bénéfice minimum que l'entreprise A demande.
- Π_B^L : le bénéfice minimum que l'entreprise B demande.

Au niveau des variables, nous introduisons les variables décisionnelles $X_{i,k}$ qui sont définies par

$$X_{i,k} = \begin{cases} 1 \text{ si le niveau } k \text{ ième de prix de transfer est choisi pour le produit } i, \\ 0, \text{ si non.} \end{cases} \tag{7.1}$$

et les variables continues sont définies comme suit :
– Π_A, Π_B : le bénéfice de A et B respectivement,
– P_i : le prix de transfert du produit i (entre A et B),
– M_{it} : la quantité du produit i fabriqué durant période t,
– $I^A_{i,t}, I^B_{i,t}$: la quantité du produit i stocké en A et B durant période t,
– $\overline{XI}_{i,k,t}$: les variables ajoutées pour linéariser les termes non linéaires $I_{i,t}X_{i,k}$,
– $F_{i,t}$: le nombre du produit i transporté de A à B durant période t,
– $\overline{FX}_{i,k,t}$ les variables ajoutées pour linéariser les termes non linéaires $F_{i,t}X_{i,k}$.
La quantité du matériel utilisé à chaque période doit être inférieure à sa capacité totale. Mathématiquement, cette contrainte est modélisée comme suit :

$$\sum_i UR_{i,r}M_{i,t} \leq R_{r,t} \quad \forall r, t. \tag{7.2}$$

La relation entre la quantité du produit stocké, la quantité de produit fabriqué, le nombre du produit transporté de A à B et la demande du client est exprimée par

$$I^A_{i,t} = I^A_{i,t-1} + M_{i,t} - F_{i,t} \quad \forall i, \forall t \geq 2, \tag{7.3}$$

$$I^A_{i,1} = M_{i,1} - F_{i,1} \quad \forall i, \tag{7.4}$$

$$I^B_{i,t} = I^B_{i,t-1} + F_{i,t} - D_{i,t} \quad \forall i, \forall t \geq 2, \tag{7.5}$$

$$I^B_{i,1} = F_{i,1} - D_{i,1} \quad \forall i. \tag{7.6}$$

La quantité de produits stockés en A et B doit être supérieure au seuil minimal. Alors, nous avons les contraintes suivantes :

$$I^A_{i,t} \geq \sigma^A_i \quad \forall i, t, \tag{7.7}$$

$$I^B_{i,t} \geq \sigma^B_i \quad \forall i, t. \tag{7.8}$$

Nous considérons, pour chaque produit i, K niveaux différents de prix de transfert. Un seul prix est choisi pour le produit i quand l'entreprise A le vend à l'entreprise B. Alors les contraintes sur les prix de transfert sont présentées comme suit :

$$P_i = \sum_k \hat{P}_{i,k}X_{i,k} \quad \forall i, \tag{7.9}$$

$$\sum_k X_{i,k} = 1 \quad \forall i. \tag{7.10}$$

Le bénéfice de A est la différence entre le revenu de A et sa dépense. La dépense de A est le total du coût de fabrication, coût d'administration, coût de stockage, coût de transport. Le revenu de A est déterminé par la somme de $P_iF_{i,t}$ $\forall i, t$. Notons que le coût de stockage dépend au prix du produit et au constant h_c. Le bénéfice de l'entreprise B est calculé de manière similaire.

$$\Pi_A = \sum_i \sum_t \Big[P_i F_{i,t} - MFC_i M_{i,t} - h_c.P_i I^A_{i,t} - H^A_i(M_{i,t} + F_{i,t}) - TC^A_i F_{i,t} \Big],$$

$$\Pi_B = \sum_i \sum_t \Big[D_{i,t}\rho_i - P_i F_{i,t} - h_c.\rho_i I^A_{i,t} - H^B_i(D_{i,t} + F_{i,t}) - TC^B_i D_{i,t} \Big].$$

Dans les équations ci-dessus, nous voyons que les termes $P_i F_{i,t}$ et $P_i I^A_{i,t}$ sont non linéaires. Nous les linéarisons comme suit [126] :

$$\overline{FX}_{i,k,t} \leq X_{i,k}.MAX \quad \forall i,k,t,$$

$$F_{i,t} = \sum_k \overline{FX}_{i,k,t} \quad \forall i,t,$$

$$P_i F_{i,t} = \sum_k \hat{P}_{i,k} \overline{FX}_{i,k,t} \; \forall i,t,$$

$$\overline{XI}_{i,k,t} \leq X_{i,k}.MAX \quad \forall i,k,t,$$

$$I^A_{i,t} = \sum_k \overline{XI}_{i,k,t} \quad \forall i,t,$$

$$P_i I^A_{i,t} = \sum_k \hat{P}_{i,k} \overline{XI}_{i,k,t} \; \forall i,t,$$

où MAX est un nombre suffisamment grand. Par conséquence, les contraintes sur les bénéfices de A et B sont exprimées comme ci-dessous :

$$\begin{aligned} \Pi_A = \sum_i \sum_t \Big[& \sum_k \hat{P}_{i,k} \overline{FX}_{i,k,t} - MFC_i M_{i,t} - h_c. \sum_k \hat{P}_{i,k} \overline{XI}_{i,k,t} \\ & - H^A_i(M_{i,t} + F_{i,t}) - TC^A_i F_{i,t} \Big], \end{aligned} \tag{7.11}$$

$$\Pi_B = \sum_i \sum_t \Big[D_{i,t}\rho_i - P_i F_{i,t} - h_c.\rho_i I^A_{i,t} - H^B_i(D_{i,t} + F_{i,t}) - TC^B_i D_{i,t} \Big], \tag{7.12}$$

$$\overline{FX}_{i,k,t} \leq X_{i,k}.MAX \quad \forall i,k,t, \tag{7.13}$$

$$F_{i,t} = \sum_k \overline{FX}_{i,k,t} \quad \forall i,t, \tag{7.14}$$

$$\overline{XI}_{i,k,t} \leq X_{i,k}.MAX \quad \forall i,k,t, \tag{7.15}$$

$$I^A_{i,t} = \sum_k \overline{XI}_{i,k,t} \quad \forall i,t, \tag{7.16}$$

Les contraintes de bénéfice minimum de A et B sont données par

$$\Pi_A \geq \Pi^L_A, \tag{7.17}$$

$$\Pi_B \geq \Pi_B^L. \tag{7.18}$$

Finalement, la contrainte de non négativité est appliquée pour toutes les variables.

$$\overline{FX}_{i,k,t}, \overline{XI}_{i,k,t}, M_{i,t}, F_{i,t}, I_{i,t}^A, I_{i,t}^B, P_i \geq 0 \quad \forall i,k,t. \tag{7.19}$$

Selon l'approche Nash, on cherche à maximiser la fonction de négociation Nash $(\Pi_A - \Pi_A^L)(\Pi_B - \Pi_B^L)$. Ainsi, le modèle mathématique du problème considéré est donné comme suit :

$$\text{(P)} \quad \max\{f(\zeta) = (\Pi_A - \Pi_A^L)(\Pi_B - \Pi_B^L) : \ \zeta \in S\},$$

où S est l'ensemble de solutions réalisable qui contient tous les points ζ satisfaisant les contraintes (7.1)-(7.19).

Le problème (P) est une programmation non linéaire en variables mixtes 0-1(PNLM). Dans la section suivante, nous allons présenter une résolution basée sur la programmation DC et DCA.

7.3 Méthode de résolution basée sur la programmation DC et DCA

7.3.1 Reformulation DC et l'algorithme DCA

Nous allons reformuler le problème (P) sous la forme d'une programmation DC et utiliser DCA pour la résoudre. L'idée est d'appliquer le résultat dans le Théorème 1.7 (théorème de pénalité).

Le nombre des variables du problème (P) est noté L. Il est facile de voir que

$$L = 2 + 2IKT + 4IT + I + IK.$$

Nous désignons $\zeta = (\Pi_A, \Pi_B, X, \overline{XI}, \overline{FX}, M, F, I^A, I^B, P) \in \mathbb{R}^L$.

Soit Ω le domaine relaxé de S, c'est à dire

$$\Omega := \{\zeta \in \mathbb{R}^L \text{ satisfait } (7.2 - 7.19) \text{ et } X_{i,k} \in [0,1]\}.$$

Considérons la fonction $p : \mathbb{R}^L \to \mathbb{R}$ définie par

$$p(\zeta) = \sum_{i=1}^{I} \sum_{k=1}^{K} X_{i,k}(1 - X_{i,k}).$$

Il est clair que $p(\zeta)$ est finie, concave et non négative sur Ω. En appliquant le Théorème 1.7 nous obtenons, pour un nombre η suffisamment grand, le problème d'optimisation équivalent à (P)

$$\min\Big\{f_\eta(\zeta) = -(\Pi_A - \Pi_A^L)(\Pi_B - \Pi_B^L) + \eta p(\zeta) : \zeta \in \Omega\Big\}.$$

C'est une programmation DC de la forme

$$\min\Big\{f_\eta(\zeta) = g(\zeta) - h(\zeta) :\ \zeta \in \Omega\Big\} \tag{7.20}$$

où

$$\begin{aligned} g(\zeta) &:= \chi_\Omega(\zeta) + 0.25\Pi_A^2 - 0.5\Pi_A\Pi_B + 0.25\Pi_B^2, \\ h(\zeta) &:= \ 0.25\Pi_A^2 + 0.5\Pi_A\Pi_B + 0.25\Pi_B^2 - \Pi_B^L\Pi_A - \Pi_A^L\Pi_B \\ &\quad +\eta\sum_{i=1}^{I}\sum_{k=1}^{K} X_{i,k}(X_{i,k}-1) + \Pi_A^L\Pi_B^L. \end{aligned}$$

Il est clair que g et h sont convexes.

Par sa définition, h est différentiable et $u = \nabla h(\zeta)$ est calculé par

$$\begin{cases} u_1 = 0.5(\zeta_1+\zeta_2) - \Pi_B^L, \\ u_2 = 0.5(\zeta_1+\zeta_2) - \Pi_A^L, \\ u_{2+(i-1)I+k} = 2\eta X_{i,k} - \eta, \\ u_j = 0, \quad \text{si non.} \end{cases} \tag{7.21}$$

Selon la description de DCA, l'algorithme DCA pour le problème (7.20) consiste à déterminer, à chaque itération ℓ, deux suites $\{u^\ell\}$ et $\{\zeta^\ell\}$ telles que $u^\ell = \nabla h(\zeta^\ell)$ et $\zeta^{\ell+1}$ est une solution optimale de la programmation quadratique convexe suivante :

$$\min\{g(\zeta) - \langle \zeta, u^\ell\rangle :\ \zeta \in \Omega\}. \tag{7.22}$$

DCA appliqué à (7.20) se décrit comme suit :

Algorithme 7.1 *(***DCA-SC***)*

Initialisation
- *Choisir la tolérance ϵ positive suffisamment petite.*
- *Choisir un point initial $\zeta^0 \in \mathbb{R}^L$ et $\ell = 0$.*

Répéter
- *Calculer $u^\ell = \nabla h(\zeta^\ell)$ via (7.21).*
- *Résoudre le problème d'optimisation quadratique convexe (7.22) pour obtenir une solution $\zeta^{\ell+1}$.*
- $\ell \longleftarrow \ell + 1$.

Jusqu'à $\|\zeta^{\ell+1} - \zeta^\ell\| \le \epsilon$ *ou* $\|f_\eta(\zeta^{\ell+1}) - f_\eta(\zeta^\ell)\| \le \epsilon$.

Théorème 7.1 *(Propriété de convergence de l'Algorithme 7.1)*

i) L'Algorithme 7.1 génère une suite $\{\zeta^\ell\}$ telle que la suite $\{f_\eta(\zeta^\ell)\}$ est décroissante.

ii) La suite $\{\zeta^\ell\}$ converge vers une solution ζ^* qui satisfait la condition nésessaire d'optimalité locale

Preuve : Ces propriétés sont les conséquences directes des propriétés convergentes de la programmation DC (voir par exemple [62]) □

Puisque DCA travaille sur le domaine relaxé, la solution obtenue ζ^* n'est pas forcement entières. Dans un tel cas, nous modifions la solution obtenue pour avoir un nouveau point initial et après redémarrons l'algorithme **DCA-SC** jusqu'à obtenir une solution entière. Le nouveau point initial ζ^0 pour **DCA-SC** est calculé à partir de la solution ζ^* comme suit :

Procédure IP(calculer le point initial ζ^0)
- Choisir le premier index $i \in \{3, 4, ..., 2 + IK\}$ tel que $\zeta_i^* \notin \{0, 1\}$.
- Définir ζ^0 de façons suivante

$$\zeta_i^0 = \begin{cases} 1 \text{ if } \zeta_i^* \geq 0.5, \\ 0, \text{ si non.} \end{cases} \quad \text{et } \zeta_j^0 = \zeta_j^* \quad \forall j \neq i.$$

Algorithme 7.2 ***(DCA-SC multi redémarré)***

Initialization
– *Résoudre le problème (P) par* **DCA-SC** *pour obtenir une solution* ζ^*.
– *Si* ζ^* *est réalisable Alors S'ARRÊTER.*
Répéter
– *Calculer le point initial* ζ^0 *à partir de* ζ^**, la solution fournie par* **DCA-SC***, via la procédure IP.*
– *Lancer* **DCA-SC** *avec le point initial* ζ^0 *pour obtenir la solution* ζ^*.
Jusqu'à ζ^* *est réalisable.*

7.3.2 Algorithme combiné DCA et SE pour le problème (P)

L'algorithme présenté dans cette section a pour objectif de trouver une solution globale du problème (P). Ici, la borne suppérieure de la valeur de fonction objectif est calculé via un problème relaxé (rappelons que (P) est un problème de maximisation). De plus, sa solution est choisie pour le point initial de **DCA-SC**. Le problème relaxé considéré est

$$(P') \quad \max\{f(\zeta) = (\Pi_A - \Pi_A^L)(\Pi_B - \Pi_B^L) \ : \zeta \in \Omega\}.$$

Bien que le problème (P') soit non convexe, nous pouvons le résoudre via un problème convexe équivalent. Nous allons présenter sa résolution après la description de l'algorithme combiné SE-DCA où nous utilisons les notations suivantes :

– $\zeta = (\Pi_A, \Pi_B, X, \overline{XI}, \overline{FX}, M, F, I^A, I^B, P)$ désigne la variable du problème.
– ζ^R : la solution du problème relaxé sur le rectangle R,
– ζ_{DCA}^R : la solution donnée par **DCA-SC** sur le rectangle R,
– ζ^* : la meilleure solution,
– γ : la meilleure borne supérieure,

– α : la meilleure borne inférieure.

L'algorithme combiné de DCA et SE (SE-DCA) peut être décrit comme suit :

Algorithme 7.3 *(L'algorithme combiné SE-DCA)*

Initialisation :
– *Choisir le premier rectangle* $R_0 := [0,1]^{IK}$
– *Choisir un nombre positif suffisamment petit* ϵ.
Étape 1 :
– *Résoudre le problème relaxé* (P') *pour obtenir une solution* ζ^{R_0}.
– **Si** ζ^{R_0} *est réalisable* **Alors**
  ζ^{R_0} *est une solution optimale globale, S'ARRÊTER*
  Sinon
    La première borne suppérieure est donnée par $\gamma = f(\zeta^{R_0})$.
    Résoudre le problème (P) *par* **DCA-SC** *avec le point initial* ζ^{R_0} *pour obtenir* $\zeta^{R_0}_{DCA}$.
    Si $\zeta^{R_0}_{DCA}$ *est réalisable* **Alors**
      Déterminer $\alpha = \alpha(R_0) = f(\zeta^{R_0}_{DCA})$.
    Sinon $\alpha = -\infty$.
    FinSi
  FinSi
– **Si** $(\gamma - \alpha) \leq \epsilon\gamma$ **Alors** *S'ARRÊTER*
  Sinon
    $\Re \longleftarrow \{R_0\}$.
  FinSi
Étape 2 :

– *Sélectionner un rectangle* $R_\ell \in \Re$ *et* i^*, k^* *tels que* $X^{R_\ell}_{i^*,k^*} \notin \{0,1\}$.

– *Diviser le rectangle* R_ℓ *en deux sous rectangles* R_{ℓ_0}, R_{ℓ_1} *via* i^*, k^*

$$R_{\ell_j} = \{X \in R_\ell,\ X_{i^*,k^*} = j,\ j = 0,1\}.$$

– *Résoudre les problèmes* $\max\{f(\zeta)\ : \zeta \in \Omega, X \in R_{\ell_j}\}$ *pour obtenir les solutions* $\zeta^{R_{\ell_j}}$.
– **Si** $\zeta^{R_{\ell_j}}$ *est réalisable* **Alors**
  Mettre à jour la meilleure borne inférieure $\alpha = \max\{\alpha, f(\zeta^{R_{\ell_j}})\}$.
  Metre à jour la meilleure solution ζ^*.
  Sinon
  Appliquer **DCA-SC** *pour résoudre le problème*

$$\min\{f_\eta(\zeta) : \zeta \in \Omega, X \in R_{\ell_i}\}$$

  pour obtenir la solution $\zeta^{R_{\ell_j}}_{DCA}$.
  Si $\zeta^{R_{\ell_j}}_{DCA}$ *est réalisable* **Alors**
    Mettre à jour la meilleure borne inférieure $\alpha = \max\{\alpha, f(\zeta^{R_{\ell_j}}_{DCA})\}$.

Mettre à jour a meilleure solution ζ^*.
FinSi
FinSi
– $\Re = \{R \in \Re : \gamma(R) > \alpha\} \cup \{R_{\ell_j} : \gamma(R_{\ell_j}) > \alpha, j = 0, 1\} \setminus \{R_\ell\}$.
– *Mettre à jour* $\gamma = \max\{\gamma(R) : R \in \Re\}$.

Étape 3 :
– **Si** $(\gamma - \alpha) < \epsilon\gamma$ **Alors** *S'ARRÊTER*
Sinon
Retourner à l'étape 2.
Finsi

Résolution du problème (P')

S'il existe un point $\zeta^0 \in \Omega$ tel que $f(\zeta^0) > 0$ alors la solution optimale ζ du problème (P') satisfait

$$f(\zeta) = (\Pi_A - \Pi_A^L)\Pi_B - \Pi_B^L) \geq f(\zeta^0).$$

De plus, il en constate que Π_A et Π_B sont plus petits que $C = \sum_i \sum_t (\rho_i - MFC_i.M_{i,t})Di, t.$

Par conséquent, Π_A et Π_B satisfont

$$\Pi_A - \Pi_A^L \geq \xi_A =: \frac{f(\zeta^0)}{C - \Pi_B^L}, \tag{7.23}$$

$$\Pi_B - \Pi_B^L \geq \xi_B =: \frac{f(\zeta^0)}{C - \Pi_A^L}. \tag{7.24}$$

Par ailleurs, la maximisation de $f(\zeta) = (\Pi_A - \Pi_A^L)(\Pi_B - \Pi_B^L)$ est équivalente à la minimisation de $f_1(\zeta) = -\log(\Pi_A - \Pi_A^L) - \log(\Pi_B - \Pi_B^L)$. Alors, une solution optimale du problème (P') peut être obtenue en résolvant le problème d'optimisation convexe suivant :

$$(P_1) \quad \min\Big\{f_1(\zeta) = -\log(\Pi_A - \Pi_A^L) - \log(\Pi_B - \Pi_B^L) \ : \ \zeta \in \Omega, (7.23), (7.24)\Big\}.$$

Pour résoudre le problème (P_1), on peut utiliser un algorithme pour la programmation convexe. Dans nos travaux, nous utilisons DCA pour trouver une solution optimale de (P_1). Rappelons que DCA donne, pour un problème convexe, une solution optimale globale. Nous utilisons la décomposition DC ci-dessous :

$$f_1(\zeta) = g_1(\zeta) - h_1(\zeta)$$

où

$$g_1(\zeta) = \frac{\lambda}{2}\Pi_A^2 + \frac{\lambda}{2}\Pi_B^2,$$

$$h_1(\zeta) = \frac{\lambda}{2}\Pi_A^2 + \frac{\lambda}{2}\Pi_B^2 + \log(\Pi_A - \Pi_A^L) + \log(\Pi_B - \Pi_B^L).$$

Ici, λ prend une valeur plus grande que $\max\{\frac{1}{\xi_A^2}, \frac{1}{\xi_B^2}\}$ pour assurer que $h_1(\zeta)$ soit convexe.

7.4 Résultats numériques

Pour évaluer l'efficacité de **DCA-SC** et **SE-DCA**, nous faisons une comparaison entre leurs résultats et ceux fournis par l'algorithme SE sans DCA (**SE**). Tous les trois algorithmes ont été programmés en C++ et exécutés sur un ordinateur Core2Duo 2.8GHz, RAM 2.0 GB. Le logiciel commercial CPLEX 11.2 a été utilisé pour résoudre le problème d'optimisation quadratique convexe. La limite de temps d'exécution de 3600 secondes est fixée pour tous les trois algorithmes. Nous avons testé la performance des algorithmes proposés sur 125 jeux de données qui sont aléatoirement générés et ont des tailles différentes. Dans tous les jeux de données, les coûts de fabrication MFC_i sont aléatoirement générés dans $[1.0, 10.0]$. Les prix de produits sur le marché ρ_i sont aléatoirement générés dans $[1.2MFC_i, 1.4MFC_i]$. Les coûts d'administration, les coûts de transport sont respectivement déterminés par 1% et 2% des coûts de fabrication. En outre, pour chaque produit, nous fixons 10 niveaux de prix de transfert qui varient entre 1.0 et 10.0. Le nombre des produits stockés est fixé à 10 pour tous les produits tandis que le constant h_c est fixé 0.05.

Nous considérons 5 cas ci-dessous :
- *Cas 1* (50 variables binaires) : $I = 5$, $K = 10$, $T = 12$, il n'y a pas la limite sur les capacités de matériel. La demande du client $D_{i,t}$ sont aléatoirement générée dans $[100, 300]$. Tous les deux entreprises A et B exigent un bénéfice minimum de 100.
- *Cas 2* (100 variables binaires) : $I = 10$, $K = 10$, $T = 12$, il n'y a pas la limite sur les capacités de matériel. La demande du client $D_{i,t}$ sont aléatoirement générée dans $[300, 500]$. L'entreprise A exige un bénéfice minimum de 1500 lorsque celui de B est de 1000.
- *Cas 3* (150 variables binaires) : $I = 15$, $K = 10$, $T = 12$, $R = 6$, la demande du client $D_{i,t}$ et la demande de matériel $UR_{i,r}$ sont générées dans $[300, 500]$ et $[0, 1.0]$ respectivement alors que la capacité de matériel $R_{r,t}$ est aléatoirement générée dans $[3000, 5000]$. L'entreprise A exige un bénéfice minimum de 15000 et celui de B est de 5000.
- *Cas 4* (200 variables binaires) : $I = 20$, $K = 10$, $T = 12$, $R = 6$, la capacité de matériel $R_{r,t}$ est aléatoirement générée dans $[4000, 6000]$. Le bénéfice minimum exigé par A est de 20000 tandis que l'entreprise B exige un bénéfice minimum de 15000. Les autres paramètres sont générés de manière similaire dans le cas 3.
- *Cas 5* (300 variables binaires) : $I = 30$, $K = 10$, $T = 12$, $R = 6$, la capacité de matériel $R_{r,t}$ est aléatoirement générée dans $[6000, 8000]$. Le bénéfice minimum de A est fixé 15000 et celui de B est de 30000. Les autres paramètres sont générés de manière similaire dans le cas 3.

Les résultats donnés par tous les trois algorithmes **DCA-SC**, **SE-DCA** et **SE** sont présentés dans les mêmes tableaux où nous utilisons les notations suivantes :
- ◇ Prob : problème,
- ◇ OptVal : la valeur de la fonction objectif fournie par chaque algorithme,
- ◇ Time : le temps d'exécution en seconde de chaque algorithme,
- ◇ *Gap* est calculé par $Gap = \frac{UB - OptVal}{UB}.100\%$ où UB est la meilleure borne supérieure qui est donnée souvent par **SE**.
- ◇ Le symbole NA dans les colonnes *Gap* et $OptVal$ de SE signifie que l'algorithme **SE** ne

trouve aucune solution réalisable après une heure.
Les Tableaux 7.1-7.5 rapportent les résultats obtenus dans les cas i, $i = 1, 2, 3, 4, 5$ respectivement. La dernière ligne désigne la moyenne de Gap, la moyenne de temps d'exécution, la moyenne de la valeur de fonction objectif. En outre, les Figures 7.1-7.5 présentent la moyenne de temps d'exécution et la moyenne de Gap dans les cas $i = 1, 2, ..., 5$ respectivement. Ces figures nous permettent de voir plus clairement l'efficacité de chaque algorithme

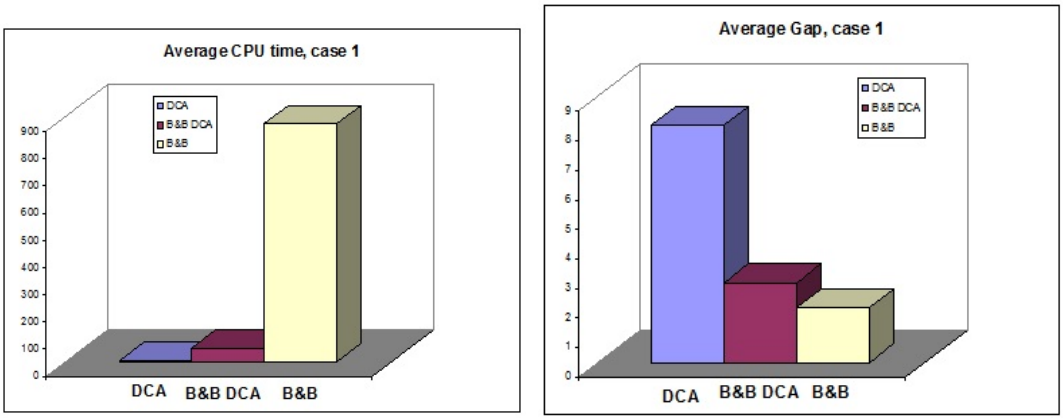


FIG. 7.1 – Cas 1 : I=5, K=10, T=12

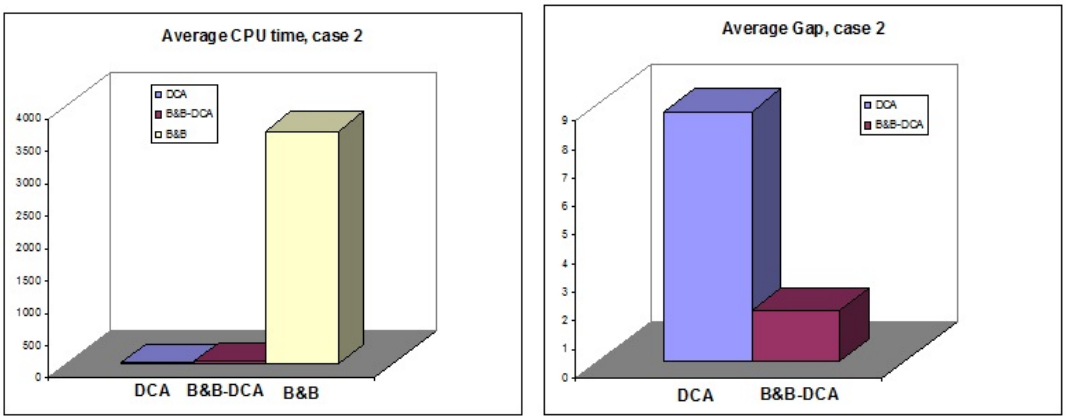


FIG. 7.2 – Cas 2 : I=10, K=10, T=12

Commentaires sur les résultats

A partir des résultats ci-dessus, nous constatons que :

- **DCA-SC** est très rapide : dans le cas 1 (50 variables binaires), **DCA-SC** trouve une solution avec la moyenne de temps d'exécution de 3.16 secondes alors que celles de **SE**, **SE-DCA** sont 875.64 secondes et 52.45 secondes respectivement. Dans les autres cas, **SE** ne trouve aucune solution réalisable après une heure. Mais **DCA-SC** fournit une solution locale avec la moyenne de temps d'exécution plus petite que 46.85 secondes.

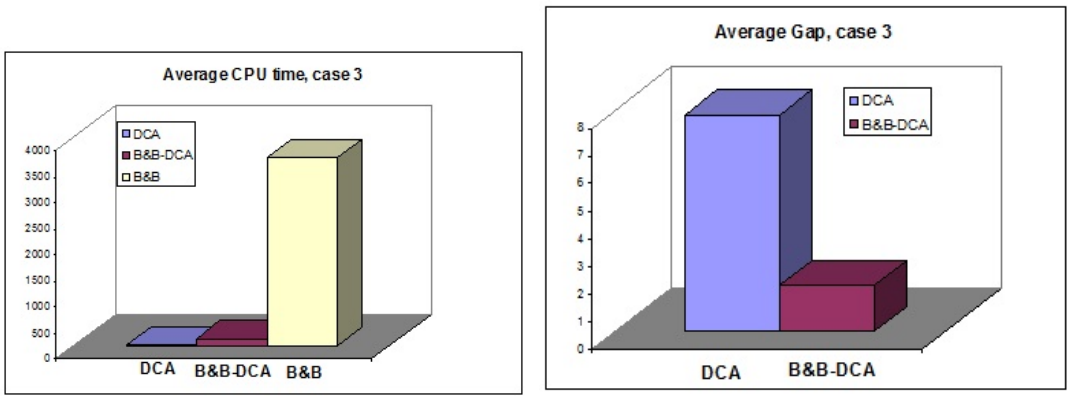


FIG. 7.3 – Cas 3 : I=15, K=10, R=6, T=12

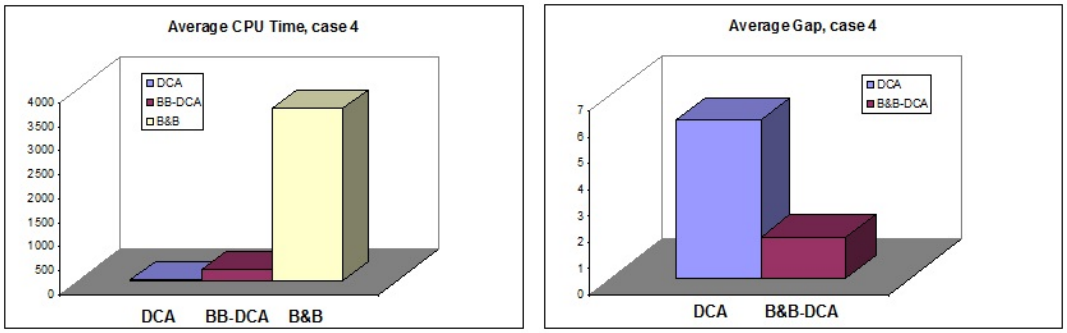


FIG. 7.4 – Cas 4 : I=20, K=10, R=6, T=12

- **DCA-SC** donne une solution très proche d'une solution globale : **DCA-SC** est non seulement très rapides mais il fournit également une solution assez bonne. Sa moyenne de Gap varie entre 6.03% et 8.72 %. Il y a 61/125 jeux de données où **DCA-SC** donne une solution dont $Gap < 5\%$.
- L'algorithme combiné **SE-DCA** est très efficace : grâce à l'efficacité de **DCA-SC**, le *Gap* de **SE-DCA** est réduit rapidement et il trouve une ϵ-solution optimale avec le temps de calcul plus petit que 707.09 secondes. Contrairement à SE-DCA, **SE** ne fonctionne pas pour les problèmes de grande taille (quand le nombre des variables binaires est plus grand que 100). **SE** ne fonctionne que dans le cas 1. Dans les autres cas, il ne trouve aucune solution réalisable après 3600 secondes.

7.5 Conclusion

Dans ce chapitre, nous avons présenté deux algorithmes efficaces basés sur la programmation DC et DCA pour le problème de détermination des prix de transfert et des politiques de stockage dans une chaîne d'approvisionnement de deux entreprises. Nous éliminons en

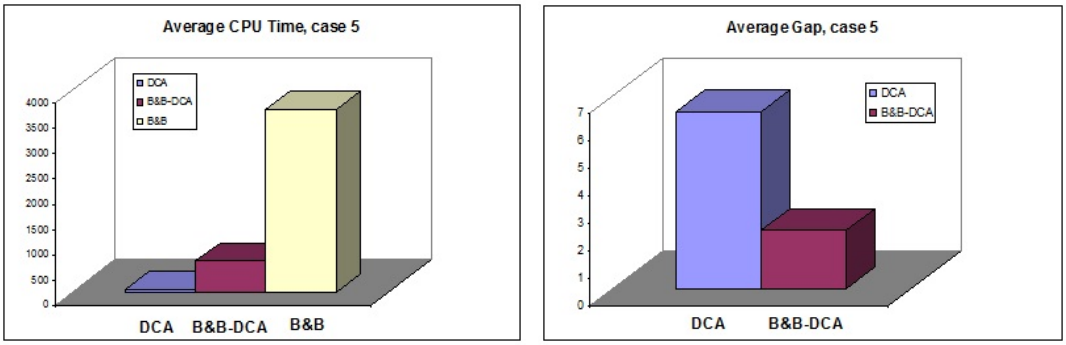


FIG. 7.5 – Cas 5 : I=30, K=10, R=6, T=12

premier temps des contraintes et des variables inutiles dans le modèle original qui est une programmation non linéaire en variables mixtes. Ensuite, le problème est reformulé sous la forme d'une programmation DC afin d'appliquer DCA. Un algorithme DCA de multi-redémarrages est développé pour trouver une solution réalisable du problème considéré. Enfin, nous proposons un problème relaxé convexe basant sur lequel un algorithme combiné SE-DCA est présenté. Nous avons effectué une comparaison sur les résultats fournis par DCA, l'algorithme combiné SE-DCA et l'algorithme SE sans DCA. Les résultats numériques ont montré que SE-DCA est le meilleur parmi les 3 et DCA est aussi très efficace. Par conséquence, il est très intéressant d'étudier les approches basées sur la programmation DC et DCA pour les autres problèmes d'optimisation non convexes en gestion de chaîne d'approvisionnement.

	DCA			B&B-DCA			B&B		
Prob	OptVal	Gap%	Time	OptVal	Gap%	Time	OptVal	Gap%	Time
0	9534192,47	10,50	1,05	10279457,25	3,50	17,03	10246469,62	3,81	417,14
1	8930692,34	10,31	1,062	9743395,65	2,148	9,45	9780315,91	1,77	431,69
2	19372803,06	2,09	1,05	19246065,47	2,73	74,03	19503634,66	1,43	969,00
3	31443450,20	18,23	1,02	37876555,30	1,49	14,23	37873161,31	1,50	1428,61
4	20941785,31	12,04	1,30	23006798,03	3,37	119,97	23267425,61	2,27	1284,39
5	13210669,14	1,81	1,06	12972003,20	3,56	69,27	12807515,30	4,81	136,98
6	14825076,57	3,25	1,078	15107840,81	1,40	22,56	14959886,49	2,37	1403,83
7	33930715,50	1,05	15,344	33734192,65	1,62	23,80	33701086,62	1,72	1617,50
8	12602531,43	3,67	1,062	12728989,99	2,71	8,47	12891349,26	1,47	751,31
9	14421353,12	10,19	1,047	15561276,9	3,09	523,47	15577620,94	2,99	1391,31
10	17743225,80	12,80	1,33	19599901,62	3,68	7,23	20059626,20	1,42	633,09
11	14557541,44	16,50	1,08	16927544,45	2,91	31,60	17211646,26	1,28	1470,31
12	47883789,31	0,93	1,03	47475436,00	1,77	9,84	47941406,81	0,81	2081,99
13	29554179,65	2,59	1,11	29239426,01	3,62	31,125	29732876,16	1,99	864,29
14	12058412,93	3,02	1,141	11993016,78	3,55	8,422	12242677,29	1,54	544,17
15	17646623,55	13,18	8,06	20032074,65	1,45	20,74	20088341,23	1,17	244,91
16	23732915,34	8,14	8,27	25549633,98	1,11	19,719	25329466,23	1,96	1542,36
17	31931723,51	1,35	12,36	31852274,90	1,59	18,95	32061133,61	0,95	1343,84
18	16603357,10	19,94	1,34	19833209,96	1,98	14,88	19975358,06	1,27	1069,17
19	19497219,22	2,00	1,33	19320831,85	2,89	10,11	19578776,47	1,59	387,27
20	17262091,91	10,57	1,14	18758071,77	2,82	7,97	19066746,65	1,22	432,66
21	15373054,92	5,25	1,13	15576025,38	3,99	9,94	15951415,70	1,68	248,91
22	23114826,53	18,52	1,45	27250982,72	3,94	194,30	28008580,20	1,27	700,03
23	12385437,14	2,40	12,00	12342336,40	2,74	25,22	12499000,80	1,51	286,89
24	17123967,39	12,19	1,28	18792370,05	3,63	19,03	19006013,97	2,54	209,25
Aver.	19827265,40	8,02	3,16	20991988,47	2,69	52,45	21174461,25	1,85	875,64

TAB. 7.1 – Résultats dans le cas où $I = 5, K = 10, T = 12$.

	DCA			B&B-DCA			B&B		
Prob	OptVal	Gap%	Time	OptVal	Gap%	Time	OptVal	Gap%	Time
0	434911516,14	9,03	3,25	472026342,02	1,27	32,36	NA	NA	3600,67
1	488714023,32	13,50	3,27	554551329,79	1,85	37,25	NA	NA	3602,19
2	436443982,16	17,33	3,33	520976862,41	1,32	30,91	NA	NA	3600,45
3	604895797,62	8,63	14,06	647780187,50	2,15	29,33	NA	NA	3600,41
4	320239006,95	1,30	3,55	319596820,43	1,50	35,78	NA	NA	3600,80
5	465006980,00	18,67	3,30	562636842,12	1,60	33,86	NA	NA	3601,67
6	551205360,84	1,06	3,22	547060690,90	1,81	36,30	NA	NA	3600,41
7	486383350,44	1,86	3,28	489652859,26	1,20	33,00	NA	NA	3600,61
8	448761674,67	1,52	3,31	441263320,64	3,16	30,28	NA	NA	3600,91
9	189815027,93	10,46	3,72	209792214,20	1,04	28,45	NA	NA	3600,52
10	511363469,93	7,65	3,24	547268110,26	1,16	28,49	NA	NA	3602,24
11	341577228,45	9,15	3,27	371053577,59	1,30	39,67	NA	NA	3601,17
12	273554943,28	3,14	3,22	276835531,95	1,97	55,39	NA	NA	3600,39
13	237415117,51	15,28	3,34	272854774,58	2,63	47,53	NA	NA	3600,55
14	451199619,25	4,92	3,33	469225478,43	1,12	32,47	NA	NA	3600,69
15	271662743,92	20,14	3,41	324236551,60	4,69	28,50	NA	NA	3602,38
16	440000804,14	19,85	3,66	539094140,25	1,81	35,58	NA	NA	3600,97
17	384021504,64	4,76	3,48	394256155,07	2,22	33,16	NA	NA	3600,50
18	359961828,44	1,49	3,53	361322523,74	1,12	27,14	NA	NA	3601,56
19	546568954,91	6,50	3,53	573805150,49	1,84	38,48	NA	NA	3601,91
20	486484969,80	2,52	3,67	492669714,27	1,28	36,45	NA	NA	3602,03
21	376322586,21	1,32	4,11	373114923,15	2,16	37,17	NA	NA	3600,02
22	392858277,90	9,46	3,31	426250869,08	1,77	26,19	NA	NA	3601,38
23	396168745,75	12,87	3,25	449135899,46	1,22	28,70	NA	NA	3600,25
24	392338513,71	15,70	3,34	459939096,13	1,18	30,84	NA	NA	3601,95
Aver.	411515041,12	8,72	3,84	443855998,61	1,77	34,13	NA	NA	3601,06

TAB. 7.2 – Résultats dans le cas où $I = 10, K = 10, T = 12$.

	DCA			B&B-DCA			B&B		
Prob	OptVal	Gap%	Time	OptVal	Gap%	Time	OptVal	Gap%	Time
0	698062807,24	15,35	20,38	778744477,49	3,98	132,93	NA	NA	3605,53
1	798045590,20	4,21	42,41	828418886,53	0,56	133,50	NA	NA	3605,30
2	668711817,47	10,45	20,52	745167801,38	0,22	135,20	NA	NA	3605,38
3	486122134,58	0,75	20,59	487221450,79	0,53	215,58	NA	NA	3604,25
4	514377300,56	16,28	19,55	610028031,10	0,72	131,23	NA	NA	3602,61
5	525091725,52	16,96	19,06	631387518,86	0,15	143,01	NA	NA	3604,09
6	793358609,84	7,79	19,67	859037807,70	0,17	132,34	NA	NA	3601,22
7	1093214050,00	0,512	19,70	1068895324,20	2,72	122,70	NA	NA	3601,95
8	902138290,28	17,87	18,61	1096259758,55	0,20	121,92	NA	NA	3607,31
9	596042995,94	5,41	19,83	599900311,62	4,79	132,17	NA	NA	3604,58
10	1038183640,32	1,81	18,91	1019634499,01	3,56	116,16	NA	NA	3603,28
11	700399087,20	1,35	19,73	667224329,00	6,02	164,97	NA	NA	3603,94
12	1088032245,08	13,07	19,75	1249576606,75	0,16	136,41	NA	NA	3603,69
13	936065494,25	9,12	18,98	1027450674,64	0,25	127,28	NA	NA	3602,74
14	897837687,96	0,64	24,00	901003377,12	0,29	145,22	NA	NA	3603,17
15	1037427052,39	0,27	20,52	1002201212,67	3,65	122,24	NA	NA	3606,80
16	556152651,75	17,96	19,77	648876575,52	4,28	118,67	NA	NA	3600,03
17	1065884515,60	3,60	18,77	1103382490,95	0,21	107,36	NA	NA	3601,88
18	710983259,09	0,93	19,453	681825543,08	4,99	130,47	NA	NA	3607,69
19	1407099271,85	3,55	20,45	1446145845,09	0,88	117,63	NA	NA	3602,86
20	270046597,30	9,32	20,25	297133821,28	0,22	140,09	NA	NA	3602,45
21	791136967,01	16,16	20,67	940634334,22	0,32	139,33	NA	NA	3602,27
22	840006433,19	9,20	20,47	907639476,18	1,89	104,74	NA	NA	3603,72
23	705667136,22	0,14	18,89	705615175,23	0,14	120,98	NA	NA	3601,39
24	710810146,14	14,58	20,41	822996154,85	1,10	149,42	NA	NA	3604,83
Aver.	793235900,20	7,84	20,85	845056059,19	1,68	133,66	NA	NA	3603,72

TAB. 7.3 – Résultats dans le cas où $I = 15, K = 10, R = 6, T = 12$.

	DCA			B&B-DCA			B&B		
Prob	OptVal	Gap%	Time	OptVal	Gap%	Time	OptVal	Gap%	Time
0	1013427362,85	5,93	18,30	1076033768,41	0,12	221,44	NA	NA	3606,64
1	2030357826,71	0,15	16,78	1933452971,12	4,92	227,49	NA	NA	3601,66
2	1181174871,09	13,84	18,17	1315336884,20	4,05	245,38	NA	NA	3604,56
3	1033679348,07	14,11	17,73	1156522786,65	3,91	220,61	NA	NA	3605,94
4	1201662957,87	2,47	16,98	1228358457,45	0,31	227,63	NA	NA	3606,83
5	845841793,31	2,89	18,05	833347115,34	4,33	246,61	NA	NA	3604,31
6	728556229,27	4,31	16,17	753628164,22	1,02	226,19	NA	NA	3600,33
7	729072339,11	10,97	18,61	816689624,15	0,27	260,92	NA	NA	3607,56
8	907525727,79	3,01	17,23	934317219,43	0,15	233,27	NA	NA	3604,25
9	1056402972,44	19,43	18,05	1308928333,16	0,18	235,78	NA	NA	3603,09
10	1225961892,12	0,20	17,80	1193729837,78	2,83	223,25	NA	NA	3601,00
11	498206711,43	3,28	15,59	511064602,03	0,78	268,17	NA	NA	3603,14
12	1816016238,25	2,50	17,89	1833613124,86	1,56	207,84	NA	NA	3603,30
13	1274623437,62	16,07	17,64	1516292015,23	0,15	246,02	NA	NA	3601,03
14	1956825477,10	5,31	18,59	2046628532,15	0,96	210,42	NA	NA	3606,03
15	1325846380,33	2,54	17,88	1355900323,53	0,33	222,74	NA	NA	3606,95
16	777916571,15	1,73	16,50	787286249,75	0,55	243,89	NA	NA	3604,00
17	854509493,47	4,93	18,09	891777056,70	0,79	242,13	NA	NA	3604,52
18	1491036863,18	6,74	17,69	1587668456,75	0,69	209,99	NA	NA	3602,63
19	1282562320,01	0,20	17,06	1219989561,68	5,07	243,17	NA	NA	3610,84
20	1405574646,99	15,90	18,39	1655397381,24	0,95	243,25	NA	NA	3603,98
21	1868703438,06	5,41	15,80	1943667517,34	1,61	209,08	NA	NA	3603,66
22	1515382016,17	4,77	16,44	1577024605,91	0,89	202,19	NA	NA	3602,55
23	474442822,96	0,57	16,14	468753042,59	1,76	254,88	NA	NA	3609,95
24	1564316628,68	3,55	16,63	1605416633,12	1,02	202,55	NA	NA	3606,33
Aver.	1202385054,64	6,03	17,37	1262032970,59	1,57	230,99	NA	NA	3604,60

TAB. 7.4 – Résultats dans le cas où $I = 20, K = 10, R = 6, T = 12$.

	DCA			B&B-DCA			B&B		
Prob	OptVal	Gap%	Time	OptVal	Gap%	Time	OptVal	Gap%	Time
0	3537618094,69	0,11	34,14	3453912928,65	2,47	565,20	NA	NA	3610,23
1	2366148755,64	1,25	38,61	2323409986,58	3,04	619,63	NA	NA	3602,95
2	2951462205,56	16,37	35,14	3523281680,51	0,16	643,30	NA	NA	3603,27
3	2393346548,72	5,70	54,83	2534952599,72	0,12	628,81	NA	NA	3611,14
4	2864313788,55	9,50	51,88	3128958202,73	1,14	596,47	NA	NA	3604,64
5	1941901921,85	16,70	55,45	2282657348,64	2,08	563,59	NA	NA	3601,13
6	2461483985,52	18,23	52,41	2881086710,53	4,29	608,59	NA	NA	3607,89
7	2443178900,02	3,78	51,30	2535191445,78	0,16	689,13	NA	NA	3616,11
8	2356784830,28	4,88	44,20	2413627863,77	2,59	612,47	NA	NA	3607,84
9	3039213187,59	0,23	43,95	2988108719,32	1,91	658,22	NA	NA	3602,33
10	3467788768,60	6,56	52,64	3707623993,90	0,10	835,81	NA	NA	3619,81
11	2218264111,90	3,35	52,80	2207552537,00	3,82	625,11	NA	NA	3601,59
12	2655981970,50	0,49	42,00	2541177531,43	4,79	554,06	NA	NA	3600,94
13	2786864547,67	7,57	46,97	3011235428,00	0,12	186,14	NA	NA	3609,44
14	2313637716,21	8,11	48,08	2450245531,85	2,69	620,38	NA	NA	3605,27
15	2538223027,16	0,09	48,09	2432596729,00	4,25	707,09	NA	NA	3603,84
16	4702605160,57	19,73	51,84	5591151756,38	4,56	557,97	NA	NA	3600,84
17	3499595620,93	1,59	51,89	3543569667,10	0,36	645,39	NA	NA	3603,17
18	2077285973,71	0,89	38,03	2014736703,55	3,87	586,58	NA	NA	3607,16
19	2581408040,54	9,96	51,42	2800250268,70	2,33	610,31	NA	NA	3602,63
20	1701035319,05	0,16	39,39	1637092878,62	3,91	680,94	NA	NA	3613,38
21	3214326642,13	0,27	43,16	3219533957,72	0,11	570,36	NA	NA	3604,39
22	3164500474,59	7,63	43,23	3323431388,09	2,99	634,91	NA	NA	3609,34
23	2566229377,20	11,66	47,48	2900182619,13	0,16	620,50	NA	NA	3600,50
24	2251024815,81	6,54	52,38	2370363511,92	1,58	623,49	NA	NA	3611,83
Aver.	2723768951,40	6,45	46,85	2872637279,54	2,14	609,78	NA	NA	3606,47

TAB. 7.5 – Résultats dans le cas où $I = 30, K = 10, R = 6, T = 12$.

Conclusion et Perspectives

Nous avons étudié dans cette thèse certains problèmes d'optimisation non convexes difficiles qui jouent un rôle important dans deux domaines : gestion de portefeuille et gestion de production. Les approches présentées ici se basent sur la méthode par Séparation et Evaluation (SE), la programmation DC et DCA parmi lesquelles DCA occupe le rôle crucial. Cette démarche est justifiée par la complexité et la taille très grande des problèmes considérés d'une part et l'efficacité de DCA dans les travaux précédents pour résoudre des problèmes complexes d'autre part. Notre ambition est de proposer les nouveaux algorithmes combinés efficaces basés sur DCA étant capables de traiter des problèmes de dimension réelle.

"L'art" de modélisation et de reformulation occupent 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 considérés. Nous avons exploité la structure spéciale de chaque problème pour trouver les algorithmes bien adaptés, robustes et peu coûteux. Les résultats numériques ont montré que notre objectif est atteint pour cinq problèmes suivants :

Pour le problème min max continu en présence des contraintes de cardinalité en gestion de portefeuille, nous l'avons reformulé comme un problème d'optimisation sous les contraintes de complémentarité et en variables mixtes 0-1. Nous avons ensuite prouvé que les contraintes de complémentarité peuvent être ignorées. Le problème résultant peut s'écrire comme un programme quadratique convexe en variables mixtes 0-1. L'algorithme DCA a après été développé pour ce problème. Les résultats de DCA sont comparés avec ceux de CPLEX.

Concernant le problème d'optimisation à deux niveaux, nous l'avons reformulé sous la forme d'un programme DC polyédral. L'algorithme DCA pour ce problème a donc les propriétés de convergence intéressantes. L'algorithme combiné SE-DCA est largement supérieure à l'algorithme SE. En outre, nous avons présenté un problème important en gestion de portefeuille qui est modélisé comme une programmation à deux niveaux. Les résultats de DCA avec des données réelles qui sont les prix d'actif sur le marché de la France sont comparés avec ceux de SE. La comparaison a montré la globalité des solutions données par DCA.

Quant au problème de minimisation du coût de maintenant comprenant le temps de séjour et la pénalité du retard, nous avons donné son premier modèle déterministe qui est un programme linéaire en variables mixtes 0-1 de très grande dimension. La méthode DCA a réussi à résoudre un problème avec 9000 variables binaires et environs 18000 variables continues alors que CPLEX 11.2 ne fonctionne pas dans ce cas puisque le nombre des variables

binaires est trop grand. Bien que DCA travaille sur un domaine continu, il fournit une solution entière. Les résultats de DCA est meilleurs que ceux de FTR - une méthode heuristique récente.

En ce qui concerne le problème de minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement, nous l'avons reformulé comme la minimisation d'une fonction DC sur un rectangle. Par conséquent, nous avons pu donner un algorithme DCA explicit. Basant sur la technique de relaxation DC, nous avons proposé un problème relaxé convexe et ensuite proposé un algorithme SE et un algorithme combiné SE-DCA. Les résultats numériques de SE-DCA et DCA ont été comparés avec ceux de la méthode SE et COUENNE, un logiciel pour la programmation non convexe.

De même pour le problème de détermination des prix de transfert et les politiques de stockage dans une chaîne d'approvisionnement de deux entreprises qui est celui de programmation non linéaire en variable mixtes 0-1, nous l'avons structuré et avons donné un algorithme DCA multi-démarré et un algorithme combiné SE-DCA. Leurs résultats sont comparés avec ceux de l'algorithme SE sans DCA.

Finalement, grâce aux expériences numériques, nous avons montré la supériorité des algorithmes proposés par rapport aux algorithmes standards.

Perspectives

Plusieurs perspectives sont ouvertes à la suite de nos travaux.

Premièrement, nous avons choisi, dans nos travaux, la solution du problème relaxé comme le point initial de DCA. Vu que le choix de point initial influence considérablement à la qualité des solutions obtenues, il nous semble que les résultats de DCA pour les problèmes dans le Chapitre 5 et le Chapitre 7 puissent être améliorés. Alors, des stratégies de choix de point initial pour ces problèmes nécessitent une recherche plus approfondie.

Deuxièmement, pour le problème d'optimisation à deux niveaux, nous avons considéré le cas où la fonction objectif du premier niveau est quadratique convexe et la fonction objectif du second niveau est linéaire. Il est intéressant d'étudier ce problème dans d'autres cas, par exemple la fonction objectif du premier niveau est DC ou/et la fonction objectif du second niveau est non linéaire. Par ailleurs, comment améliorer la borne inférieure reste une question ouverte.

Enfin, pour reformuler le problème d'optimisation non convexe en variables entières dans le Chapitre 6 comme un programme DC, nous avons pénalisé les constraintes entières mais ce n'est pas une technique de pénalité exacte. La question se pose alors : pouvons - nous reformuler ce problème par une technique de pénalité exacte.

Toutes ces questions feront l'objet de nos futurs travaux.

Bibliographie

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

[2] Hoang Tuy, G*lobal Minimization of a Difference of Two Convex Functions. Mathematics Programming Study*, Vol. 30 pp. 150-182, 1987. Acta Mathematica Vietnamica, Vol. 8 , pp. 3-34, 1983

[3] Hoang Tuy *Global Optimization : Deterministic Approaches* 2nd revised edition, Springer-Verlag, Berlin, 1993.

[4] Hoang Tuy *DC Optimisation : Theory, Methods and Algorithms, Handbook of Global Optimisation* Horst and Pardalos eds, Klaver Academic Publishers, pp. 149–216, 1995.

[5] Hoang Tuy *Convex Analysis and Global Optimization*, Kluwer Academic Publishers, 1998.

[6] Hoang Tuy, T.V. Thieu and Ng.Q. Thai, *A conical algorithm for globally minimizing a concave function over a convex set*, Math. Operations Research, Vol. **10**, pp. 498–514, 1985.

[7] Horst, R., *Deterministic global optimization with partition sets whose feasiblity is not known. Application to concave minimization, reverse convex constraints, DC programming and Lipschitzian optimization,* Journal of Optimization Theory and Application, Vol. **58**, pp. 11–37, 1988.

[8] Horst, R., *A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization*, Journal of Optimization Theory and Application, Vol. **51** (2), pp. 271–291, 1986.

[9] Horst, R., Nguyen Van T. and Benson, H., *Concave minimization via conical partitions and polyhedral outer approximation*, Mathematical Programming, Vol. **50**, pp. 259–274, 1991.

[10] Horst, R., T. Nguyen Van, De Vries J., *On finding new vertices and redundant constraints in cutting plane algorithms for global optimization*, Operations Research Letters, Vol. **7**(2), pp. 85–90, 1988.

[11] R. Horst, Nguyen. V. T, and Hoang Tuy, *On a outer approximation concept in global optimisation*, Optimization, Vol. 20 pp. 255-264. 1989

[12] Horst, R. and Hoang Tuy, *Global optimization (Deterministic Approaches)*, Third edition, Springer, Berlin 1996.

[13] Horst R. and Hoang Tuy, *On the convergence of global methods in multiextremal optimization*, Journal of Optimization Theory and Application, **54** (2), pp 253–271, 1987.

[14] Horst R., P.M. Pardalos,and Nguyen V.T *Introduction to global optimization*, Kluwer Academic, 1995.

[15] P.J Laurent *Approximation et optimisation* Paris : Hermann, 1972.

[16] Le Dung Muu, Thai Quynh Phong and Pham Dinh Tao, *Decomposition methods for solving a class of nonconvex programming problems dealing with bilinear and quadratic functions*, Computational Optimization and Application, Vol. **4**, pp. 203–216, 1995.

[17] H.A Le Thi *Analyse numérique des algorithmes de l'optimisation DC. Approches locale et globale. Codes et simulations numériques en grande dimension. Applications.* Thèse de Doctorat de l'Université de Rouen, 1994.

[18] H.A Le Thi *Contribution à l'optimisation non convexe et l'optimisation globale : Théorie, Algorithmes et Applications* Habilitation à Diriger des Recherches, Université de Rouen, 1997.

[19] H.A Le Thi *An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints* Mathematical Programming, Ser. A, Vol. **87**(3), pp. 401–426, 2000.

[20] H.A Le Thi *Solving large scale Molecular distance geometry problem by a smoothing technique via the Gaussian transform an DC programming* Journal of Global Optimization, Vol. **27**, pp. 375–397, 2003.

[21] H.A Le Thi, T. Belghiti and T. Pham Dinh *A new efficient algorithm based on DC programming and DCA for Clustering* In Press, Available July 2006, Journal of Global Optimization.

[22] H.A Le Thi, M. Le Hoai and T. Pham Dinh *Optimization based DC programming and DCA for Hierarchical Clustering* In Press, Available Online June 2006, European Journal of Operational Research.

[23] H.A Le Thi, M. Le Hoai and T. Pham Dinh *Une nouvelle approche basée sur la programmation DC et DCA pour la classification floue* EGC 2007, RNTI, pp. 703–714, 2007.

[24] H.A. Le Thi, T.P. Nguyen, T. Pham Dinh *A Continuous DC Programming Approach to the Strategic Supply Chain Design Problem from Qualified Partner Set.* European Journal of Operational Research. Vol. **183**(3), pp. 1000-1012, 2007.

[25] Le Thi H. A., Ndiaye B. M., Pham Dinh T., *Solving a multimodal transport problem by DC*, Proceedings of RIVF'08, IEEE International conference on Research, Innovation and Vision for the future in Computing and Communications Technologies, Ho Chi Minh city, July 13-17, pp. 49-56 2008

[26] Le Thi H. A., Ndiaye B. M., Pham Dinh T., *Models and Simulations for Container Allocation on trains in a rapid transhipment shunting yard by DC Programming and*

DCA, Third International Conference on Computational Management Science, special invited session, 2006

[27] Le Thi H. A., Ndiaye B. M., Pham Dinh T., *Programmation DC et DCA pour Le Dispatching des Véhicules Guidés Automatisés dans un Terminal à Conteneurs Automatisé*, Session spéciale invitée, Conférence conjointe FRANCORO V / ROADEF pp. 241-342, Grenoble 20-23 février 2007

[28] H. A. Le Thi, Nguyen.V. V., T. Pham Dinh, *A Combined DCA and New Cutting Plane Techniques for Globally Solving Mixed Linear Programming*, SIAM Conference on Optimization, Stockholm (Sweden) special invited session, 2005

[29] H.A Le Thi and T. Pham Dinh *Solving a class of linearly constrained indefinite quadratic problems by DC algorithms* Journal of Global Optimization, Vol. **11**, pp. 253–285, 1997.

[30] H.A Le Thi and T. Pham Dinh *A Branch-and-Bound method via DC Optimization Algorithm and Ellipsoidal techniques for Box Constrained Nonconvex Quadratic Programming Problems* Journal of Global Optimization, Vol. **13**, pp. 171–206, 1998.

[31] H.A Le Thi and T. Pham Dinh *DC programming approach for large-scale molecular optimization via the general distance geometry problem* Nonconvex Optimization and Its Applications 40, in Optimization in Computational Chemistry and Molecular Biology : Local and Global Approaches, Kluwer Academic Publishers, pp. 301–339, 2000.

[32] H.A Le Thi and T. Pham Dinh *Large Scale Molecular Conformation via the Exact Distance Geometry Problem* In Optimization, Lecture Notes in Economics and Mathematical Systems, Vol. **481**, Heidelberg, Springer-Verlag, pp. 260-277, 2000.

[33] H.A Le Thi and T. Pham Dinh *DC programming approach and solution algorithm to the multidimensional scaling problem* Nonconvex Optimization and Its Applications 53 : In From Local to Global Optimization, Kluwer Academic Publishers, pp. 231–276, 2001.

[34] H.A Le Thi and T. Pham Dinh *DC optimization approaches via Markov models for restoration of signals (1-D) and (2-D)* Nonconvex Optimization and Its Applications 54 : In Advances in Convex Analysis and Global Optimization, Kluwer Academic Publishers, pp. 300–317, 2001.

[35] H.A. Le Thi, T. Pham Dinh *A Continuous Approach for Globally Solving Linearly Constrained Quadratic Zero - One Programming Problems.* Optimization, Vol. **50**, pp. 93–120, 2001.

[36] H.A Le Thi and T. Pham Dinh *DC Programming Approach for Multicommodity Network Optimization Problems with Step Increasing Cost Functions* Special Issue of Journal of Global Optimization (dedicated to Professor R. Horst on the occasion of his 60 th birthday), Vol. **22**, pp. 204–233, 2002.

[37] H.A Le Thi and T. Pham Dinh *Dc Programming. Theory, Algorithms, Applications : The State of the Art* First InternationalWorkshop on Global Constrained Optimization and Constraint Satisfaction, October 2-4, 2002, Valbonne-Sophia Antipolis, France,

Research Report, Laboratory of Modeling, Optimization & Operations Research, Insa-Rouen, France, 2002.

[38] H.A Le Thi and T. Pham Dinh *Large scale molecular optimization from distances matrices by a DC optimization approach* SIAM Journal of Optimization, Vol. **14**, N°.1, pp. 77–116, 2003.

[39] H.A Le Thi and T. Pham Dinh *A new algorithm for solving large scale molecular distance geometry problems, Applied Optimization : in Hight Performance Algorithms and Software for Nonlinear Optimization* Kluwer Academic Publishers, pp. 276–296, 2003.

[40] H.A. Le Thi, T. Pham Dinh *The DC (Difference of Convex functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems.* Annals of Operations Research, Vol. **133**, pp. 23–46, 2005.

[41] H.A Le Thi and T. Pham Dinh *A continuous approach for the concave cost supply problem via DC Programming and DCA* Discrete Applied Mathematics, Vol. **156**, pp. 325–338, 2008.

[42] H.A Le Thi, T. Pham Dinh and H. Dinh Nho *Towards Tikhonov regularization of nonlinear ill-posed problems : a DC programming approach* C.R. Acad. Sci. Paris, Ser. I, Vol. **335**, pp. 1073–1078, 2002.

[43] H.A Le Thi, T. Pham Dinh and H. Dinh Nho *Solving inverse problems for an elliptic equations by DC (Difference of Convex functions) programming* Journal of Global Optimization, Vol. **25**, pp. 407–423, 2003.

[44] H.A Le Thi, T. Pham Dinh and H. Dinh Nho *On the ill-posedness of the trust region Subproblem* Journal of Inverse and Ill-posed Problems, Vol. **11**, pp. 545–577, 2003.

[45] H.A Le Thi, T. Pham Dinh and N. Huynh Van *Exact Penalty and Error Bounds in DC Programming*, to appear in Journal of Global Optimization, 27 pages.

[46] H.A Le Thi, T. Pham Dinh and M. Le Dung *Numerical solution for optimization over the efficient set by DC optimization algorithm* Operations Research Letters, Vol. **19**, pp.117–128, 1996.

[47] H.A Le Thi, T. Pham Dinh and M. Le Dung *A combined DC Optimization : Ellipsoidal Branch-and-Bound Algorithm for Solving Nonconvex Quadratic Programming Problems* Journal of Combinatorial Optimization, Vol. **2**, N°.1, pp. 9–28, 1998.

[48] H.A Le Thi, T. Pham Dinh, M. Le Dung, *Exact penalty in DC Programming.* Vietnam Journal of Mathematics Vol. **27** (2), pp. 169–178, 1999.

[49] H.A Le Thi, T. Pham Dinh and M. Le Dung *Simplicially Constrained DC Optimization over the Efficient Set and Weakly Efficient Sets* Journal of Optimization, Theory and Applications, Vol. **117**, N°.3, pp. 503-531, 2003.

[50] H.A Le Thi, T. Pham Dinh and T. Nguyen Van *Combination between Local and Global Methods for Solving an Optimization Problem over the Efficient Set* European Journal of Operational Research, Vol. **142**, pp ; 257–270, 2002.

[51] B. M. Ndiaye, T. Pham Dinh, H. A. Le Thi, *DC programming and DCA for SS-CRP*, in Modelling, Computation and Optimization in Information Systems and Management Sciences, Communications in Computer and Information Science CCIS Volume 14, Springer, pp. 21-30, 2008

[52] G.L. Nemhauser, L.A. Wolsey, *A recursive procedure to generate all cuts for 0-1 mixed integer programs*, Mathematical Programming, vol. 46, pp. 379-390, 1990.

[53] Nguyen Van T. and Hoang Tuy, *Convergent algorithms for minimizing a concave function*, Mathematic Operations Research, Vol. **5**, pp. 556–566, 1980.

[54] V.V. Nguyen, *Méthodes exactes pour l'optimisation DC polyédrale en variables mixtes 0-1 basées sur DCA et nouvelles coupes*. Thèse de Doctoral, INSA de Rouen, 2006.

[55] T. Pham Dinh *Elements homoduaux relatifs à un couple de normes (φ, ψ). Applications au calcul de $S_{\varphi\psi}(A)$* Technical Report, Grenoble, 1975.

[56] T. Pham Dinh *Calcul du maximum d'une forme quadratique définie positive sur la boule unité de la norme du max* Technical Report, Grenoble, 1976.

[57] T. Pham Dinh *Algorithms for solving a class of non convex optimization problems. Methods of subgradients* Fermat days 85. Mathematics for Optimization, Elsevier Science Publishers B.V. North-Holland, 1986.

[58] T. Pham Dinh *Duality in DC (difference of convex functions) optimization. Subgradient methods* Trends in Mathematical Optimization, International Series of Numer Math., Vol. **84**, pp. 277–293, 1988.

[59] T. Pham Dinh and H.A Le Thi *Stabilité de la dualité lagrangienne en optimisation DC (différence de deux fonctions convexes)* C.R. Acad. Paris, P.318, Série I, pp. 379–384, 1994.

[60] T. Pham Dinh and H.A Le Thi *Lagrangian stability and global optimality in nonconvex quadratic minimization over Euclidean balls and spheres*. Journal of Convex Analysis, Vol. **2**, pp. 263–276, 1995.

[61] T. Pham Dinh and H.A Le Thi *DC optimization algorithms for globally minimizing nonconvex quadratic forms on Euclideans balls and spheres* Operations Research Letters, Vol. **19**, pp. 207–216, 1996.

[62] T. Pham Dinh and H.A Le Thi *Convex analysis approach to d.c. programming : Theory, Algorithms and Applications* Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, Vol. **22**, N°.1, pp. 289–355, 1997.

[63] T. Pham Dinh and H.A Le Thi *DC optimization algorithms for solving the trust region subproblem* SIAM Journal of Optimization, Vol. **8**, N°.2, pp.476–505, 1998.

[64] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, 1970.

[65] Rosen, J.B. and Pardalos, P.M., *Global minimization of large scale constrained quadratic problem by separable programming*, Mathematical Programming, Vol. **34** (2), pp. 163-174, 1986.

[66] Kalantari, B. and Rosen, J.B., *Algorithm for global minimization of linearly constrained concave quadratic functions*, Operations Research, Vol. **12**, pp. 554-561, 1987.

[67] A.T. Phillips and J.B. Rosen, *A parallel algorithm for constrained concave quadratic global minimization*, Mathematical Programming Vol. **42**, pp. 412-448, 1988.

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

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

[70] J.B.H. Urruty *Generalized differentiability, duality and optimization for problem dealing with differences of convex functions*Lecture Notes in Economics and Mathematical Systems, Vol. **256**, Heidelberg, Springer-Verlag, pp. 260–277, 1985.

[71] J.B.H. Urruty *Conditions nécessaires et suffisantes d'optimalité globale en optimisation de différences de deux fonctions convexes* CRAS, Vol. 309, Série I, pp. 459–462, 1989.

[72] J.B.H. Urruty, C.,Lemaréchal, *Convex Analysis and Minimization Algorithms,* Springer Verlag, Berlin, Heidelberg, 1993.

[73] Q.P. Thai, *Analyse Numérique des Méthodes d'Optimisation Globale. Codes et Simulations numériques. Applications,* Thèse de doctorat, Université de Rouen, 1994.

[74] Q. P. Thai, Le Thi H. A. and Pham D. T., *On the global solution of linearly constrained indefinite quadratic minimization problems by decomposition branch and bound method*. RAIRO, Recherche Opérationnelle, Vol. **30**(1), pp. 31–49, 1996.

[75] J.F Toland *Duality in nonconvex optimization* Journal of Mathematical Analysis and Applications, Vol.66, pp. 399-415, 1978.

[76] H.A. Le Thi, T. Pham Dinh, N. Nguyen Canh and T. Nguyen Van, *DC programming techniques for solving a class of nonlinear bilevel programs,* Journal of Global Optimization **44**, pp.313-337, 2009.

[77] Pham Dinh Tao, Nguyen Canh Nam and Le Thi Hoai An, *DC Programming and DCA for Globally Solving the Value-At-Risk,* Computational Management Science, Vol. 6, Issue 4, pp. 477-501, 2009.

[78] Le Thi Hoai An, Madhi Moeini, Pham Dinh Tao, *Portfolio Selection under Downside Risk Measures and Cardinality Constraints based on DC Programming and DCA,* Computational Management Science, Vol. 6, issue 4, pp. 459-475, 2009.

[79] Le Thi Hoai An, Madhi Moeini, Pham Dinh Tao, *DC Programming Approach for Portfolio Optimization under Step Increasing Transaction Costs,* Optimization, Vol.58 (3), pp. 267-289, 2009.

[80] Le Thi Hoai An, Nguyen Quang Thuan, Nguyen Huynh Tuong, Pham Dinh Tao, *Solving the earliness tardiness scheduling problem by DC programming and DCA, Mathematica Balkanica,* Vol. 23, Fasc. 3-4, pp. 271-288, 2009.

[81] H. Markowitz, *Portfolio selection,* Journal of Finance, Vol. **7**, pp.77-99, 1952.

[82] N. Gulpinar and B. Rustem, *Continuous min max approach for Single period protfolio selection problem,* Numerical Methods in Finance, Springer, pp.241-258, 2005.

[83] T.J. Chang, N. Meade, J.B. Beasley and Y.M. Sharaiha, *Heuristics for Cardinality Constrained Portfolio Optimization,* Computers and Operations Resarch, Vol. **27**, pp.1271-1302, 2000.

[84] B. Rustem, R. Becker, W. Marty, *Robust Min Max Portfolio strategies for Rival Forecast and Risk Scenarios,* Journal of Economic Dynamics and Control, Vol. **24**, pp. 1591-1623, 2000.

[85] NAG Library, *http ://nag.co.uk,* The Numerical Algorithms Group, OXford.

[86] N. Gulpinar, B. Rustem, *Worst-case Robust Decisions for Multi-period Portfolio Optimization,* European Journal of Operational Research, Vol. **183(3)**, pp.981-100, 2007.

[87] N. Gulpinar, H.A. Le Thi, M.Moeini, *Robust Investment Strategies with Discrete Asset Choice Constraints Using DC Programming and DCA,* Optimization, Vol. **59**(1), pp45-62, 2010.

[88] A. Fernandez and S. Gomez, *Portfolio selection using neural networks,* Computers & Operation Research (2007) **34**, pp1177-1191.

[89] N. Jobst, M. Horniman, C. Lucas and G. Mitra, *Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints,* Quantitative Finance, Vol. **1**, pp.1-13, 2001.

[90] H. Konno and R. Yamamoto, *Integer programming approaches in mean-risk models,* Computational Management Science, Vol. **2**, pp.339-351, 2005.

[91] J. Hamon, *Bourse et Gestion de Prtefeuille,* 2^e édition Economica, Paris (2005).

[92] A. Aftalion, *La Nouvelle Finance et La Gestion des Prtefeuilles,* édition Economica, Paris (2003).

[93] Adjallah K.H., Adzakpa K.P. *Minimizing maintenance cost involving flow-time and tardiness penalty with unequal release dates.* In Press of Journal of Risk and Reliability, Part O, Vol. 221, no 1, pp. 57-66, April 2007.

[94] Adzakpa K.P. *Maintenance of distributed systems : methods for real-time decision-making.* PhD Thesis, University of Technology of Troyes, France, October 2004.

[95] Berg M., *The marginal cost analysis and its application to repair and replacement policies,* European Journal of Operational Research, Vol. 82, no 2, pp. 214–224, 1995.

[96] Burke E.K., Smith A.J. *Hybrid evolutionary techniques for the maintenance scheduling problem.* Ieee2000 Trans. on Power Systems,Vol.15, no1, p.122-128, 2000.

[97] Derman C., Lieberman G.J., Ross S.M., *On the optimal assignment of servers and a repairman,* Journal of Applied Probability, Vol. 17, no. 2, p. 577-581, 1980.

[98] Duron C., Ould Louly M.A., Proth J.-M. *The one machine scheduling problem : Insertion of a job under the real-time constraint.* European Journal of Operational Research, Vol. 199, p.695-701, 2009.

[99] Frostig E., *Jointly optimal allocation of a repairman and optimal control of service rate for machine repairman problem,* European Journal of Operational Research, Vol. 116, pp. 274-280, 1999.

[100] Frostig E., *Optimal allocation of machines to distinguishable repairmen in order to maximize some reward functions,* Probability in the Engineering and Informational Sciences, Vol. 9, pp. 633-645, 1995.

[101] Frostig E., *Optimal policies for machine repairmen problems,*Journal of Applied Probability, Vol. 30, no 3, pp. 703-715, 1993.

[102] Gopalakrishnan M., Mohan S., He Z. *A tabu search heuristic for preventive maintenance scheduling.* Computers & Industrial Engineering, Vol. 40, no 1-2, pp.149-160, June 2001.

[103] Graves G., Lee C. *Scheduling maintenance and semiresumable jobs on a single machine,* Naval Research Logistics, Vol. 46, no 7, pp. 845-863, 1999.

[104] Koole G., *Optimal repairman assignment in two symmetric maintenance models,* European Journal of Operational Research, Vol. 82, pp. 295-301, 1995.

[105] Lee C., Chen Z. *Scheduling jobs and maintenance activities on parallel machines,* Naval Research Logistics, Vol. 47, no 2, p.145-165, 2000.

[106] Lenstra J., Kan A. R., Brucker P., *Complexity of machine scheduling problems,* Annals of Discrete Mathematics, Volum 1, p.343-362, 1977.

[107] Lo S-T., Chen R-M., Huang Y-M., Wu C-L., *Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system.* Expert Systems with Applications, Vol. 34, p.2071-2081, 2008.

[108] Park K., *Optimal number of minimal repairs before replacement,* IEEE Transactions on Reliability, Vol. R-28, no. 2, pp. 137 - 140, 1979.

[109] Qi X., Chen T., Tu F., *Scheduling the maintenance on a single machine,* Journal of the Operation Research Society, Vol. 50, no 10, pp. 1071-1078, 1999.

[110] Seshadri S., *Determination of aggregate preventive maintenance programs using production schedules,* Computers and Industrial Engineering, Vol. 14, pp. 193-200, 1988.

[111] Stadje W., Zuckerman D., *Optimal repair policies with general degree of repair in two maintenance models,*Operations Research Letters, Vol. 11, no 2, pp. 77-80, 1992.

[112] Tajima H., Sugimoto J., Funabashi T., Yokoyama R., *Auction Price-Based Thermal Unit Maintenance Scheduling by Reactive Tabu Search.* Proc. 41st IEEE Int. UPEC'06 (Universities Power Engineering Conference), 2006.

[113] Weinstein L., Chung C., *Integrating maintenance and production decisions in a hierarchical production planning environment,* Computer and Operations Research, Vol. 26, no 10-11, pp. 1059-1074, 1999.

[114] A.Z. Szendrovits, *Manufacturing cycle time determination for a multi-stage economic production quantity model,* Mgmt Sci, Vol. 22, pp. 298-308, 1975.

[115] S.K. Goyal, *Determination of optimum production quantity for two-stage production system,* Operational Research Quartery, Vol. 28(4), pp. 865-870, 1977.

[116] R.W. Bogaschewsky, U.D. Buscher and G. Lindner, *Optimizing multi-stage production with constant lot size and varying number of unequal sized bactches,* Omega, Vol. 29, pp. 183-191, 2001.

[117] Y.C. Hsiao, Y. Lin and Y.K. Huang, *Optimal multi-stage logistic and inventory policies with production bottleneck in a serial supply chain,* Intenational Journal Production Economics, Vol. 124, pp. 408-413, 2010.

[118] S.K. Goyal, A.Z. Szendrovits, *A constant lot size model with equal and unequal sized batch shipments between production stages,* Engineering Costs and Production Economics, Vol. 10, pp. 203-210, 1986.

[119] Y.C. Hsiao, *Optimal single-cycle polices for the one-warehouse multi-retailer inventory/distribution system,* International Journal of Production Economics, Vol. 114, pp. 219-229, 2008.

[120] A.Z. Szendrovits, Z. Drezner, *Optimizing multi-stage production with constant lot size and varying numbers of batches,* Omega, Vol. 8, pp. 623-629, 1980.

[121] E. Goldratt, *Compute rized shop floor scheduling,* International Journal of Production Research, Vol. 26, pp. 443-455, 1988.

[122] C. Vercellis, *Multi-plant production planning incapacitated self-configuring two-stage serial systems,* European Journal of Operational Research, Vol. 119, pp. 451-460, 1999.

[123] M. Grant and S. Boyd, *CVX : Matlab Software for Disciplined Convex Programming, version 1.21*, http ://cvxr.com/cvx, october 2010.

[124] *Counne 3.2 : http ://www.coin-or.org/download/binary/Couenne/.*

[125] C.J. Vidal, M. Goetschalckx, *Strategic Production-distribution models : A critical review with emphasis on global supply chain models* European Journal of Operational Research, Vol.**98**, pp. 1-18, 1997.

[126] F. Glover, *Improved linear integer programming formulation of non linear integer problem,* Management Science, Vol. **22(4)**, pp. 455-460, 1975.

[127] I. Waysman, *A model of negotiated transfer pricing,* Journal of Accounting and Economics, Vol. **25**, pp. 349-384, 1998.

[128] J. Gjerdrum, N. Shah and L.G. Papageogiou, *Fair transfer price and inventory holding policies in two-enterprise supply chain,* European Journal of Operational Research, Vol. **143**, pp. 582-599, 2002.

[129] J. Nash, *The bargaining problem,* Econometrica, Vol. **18**, pp. 155-160, 1950.

[130] M. Shunko, S. Gavirneni, *Role of transfer price in global supply chain with random demands,* Journal of Industrial and Management Optimization, Vol. **3(1)**, pp. 99-117, 2007.

[131] P. Sylvain, H. Pierre, L.D. Sebastien, M. Nenad *Exact and heuristic solutions of the global supply chain problem with transfer pricing,* European Journal of Operational Research, Vol. **202(3)**, pp. 864-879, 2010.

[132] P.H. Hansen, B. Hegedahl, S. Hjorkjaer, B. Obel, *A heuristic solution to the warehouse location-routing problem* European Journal of Operational Research, Vol. **76**, pp. 111-127, 1993.

[133] T. Pfeiffer, *Transfer pricing and decentralized lot sizing in multistage, multiproduct production processes,* European Journal of Operational Research, Vol. **116**, 319-330, 1999.

[134] T.W. Chien, *Determining profit-maximizing production/shipping, policies in a one-to-one direct shipping stochastic demand environment,* European Journal of Operational Research, Vol.**64**, pp. 83-102, 1993.

[135] S.S. Chauhan and J.M. Proth, *Analysis of a supply chain partnership with revenue sharing,* International Jornal of Production Economics, Vol. **97**, pp. 44-51, 2005.

[136] W.G. Howe, J.F. Cox, *Enterprise implications of transfer pricing,* Production and Inventory Management Journal, Vol.**35(2)**, pp. 35-38, 1994.

[137] S.S. ChauChan, *Chaîne d'appovisionnement : approches stratégique et tactique,* Thèse doctorat 2003, Université de Metz.

[138] Marc C. Steinbach, *Markowitz Revisited Mean Variance Model in Financial Protfolio Analysis,* SIAM review, Vol. **43(1)**, pp. 31-85, 2001.

[139] D.M. Le, V.T. Nguyen, *A Global Optimization Method for Solving Convex Quadratic Bilevel Programming Problems,* Journal of Global Optimization, Vol. **26**, pp. 199-219, 2003.

[140] G.J. Alexander, A.M. Baptista, *A Portfolio selection with a drawdown constraint,* Journal of Banking & Finance Vol. **30**, pp. 3171-3189, 2006.

[141] J.F. Bard, *Some properties of the bileval programming problem,* Journal of Optimization Theory and Applications,Vol. **68(2)**, pp. 371-378, 1991.

[142] P. Hansen, B. Jaumard, G. Savard, *New branch-and-bound rules for for linear bilevel programming,* SIAM Journal on Scientific and Statistical Computing, Vol. **13(5)** , pp. 1194-1217, 1992.

[143] B. Colson, P. Marcotte and G. Savard, *Bilevel programming : a survey,* A Quarterly Journal of Operations Research, Vol. **3(2)**, pp. 87-107, 2005.

[144] A. Courtois, M. Pillet and C. Martin-Bonnefous, *Gestion de Production,* 4e édition, Organisation Groupe Eyrolles, Paris, 2006.

[145] Georges Javel, *Organisation et Gestion de Production,* 4e édition, Bunod, Paris, 2004.

[146] *http ://www.itbooster.net*

[147] David G. Luenberger, *Linear and Nonlinear Programming,* 2e édition, Springer, 2004.

Résumé

Cette thèse porte sur la recherche des techniques d'optimisation pour la résolution de certains problèmes importants en deux domaines : gestion de portefeuille et gestion de production. Il s'agit des problèmes d'optimisation non convexe de grande dimension. Notre travail est basé sur la programmation DC (Différence de fonctions convexes), DCA (DC Algorithmes), la méthode par Séparation et Evaluation (SE). Cette démarche est motivée par la robustesse et la performance de la programmation DC et DCA comparée aux autres méthodes.

La thèse comprend trois parties :

Dans la première partie, nous présentons les outils fondamentaux et les techniques essentielles en programmation DC, DCA ainsi que les méthodes par Séparation et Evaluation. La deuxième partie concerne les problèmes d'optimisation non convexes en gestion de portefeuille. Nous avons étudié deux problèmes :
– Problème min max continu en gestion de portefeuille en présence des contraintes de cardinalité.
– Une classe des problèmes d'optimisation à deux niveaux et son application en sélection de portefeuille.
La troisième partie est consacrée à la recherche sur les problèmes d'optimisation non convexe en gestion de production. Trois problèmes ont été étudiés :

– Minimisation du coût de maintenance comprenant le temps de séjour et la pénalité du retard.
– Minimisation du coût d'un système de production/stockage multi-étapes en présence de goulot d'étrangement.
– Détermination des prix de transfert et les politiques de stockage pour une chaîne d'approvisionnement de deux entreprises.

Mots clés : Optimisation non convexe, Programmation DC, DCA, Séparation et Evaluation (SE), Relaxation DC, Programmation en variables mixtes entières, Programmation à deux niveaux, Gestion de portefeuille, Gestion de Production.

Abstract

This thesis deals with optimization techniques for solving some optimization problems in two domains : portfolio selection and production management. They are large scale non convex optimization problems due to integer variables and/or the non convexity of the objective function. Our approach is based on DC programming and DCA, DC relaxation techniques and the algorithm Branch and Bound. This work is motivated by the robustness and the performance of the DC programming and DCA compared to other methods.

The thesis includes three parts :

In the first part, we present the fundamental tools and the essential techniques in DC programming, DCA as well as the method Branch and Bound. The second one concerns some non convex optimization problem in portfolio selection. Two following problems are considered :
– Min max continuous problem with the cardinality constraints in portfolio selection.
– A class of bilevel programming problems and its application in portfolio selection.
The third part contains some non convex optimization problems in production management. We study three problems :
– Minimization of the maintenance cost involving the flow time and the tardiness penalty.
– Minimization of the cost of multi-stages production/inventory systems with bottleneck.
– Determination of transfer prices and inventory policy in supply chain of two enterprises.
Keywords : Non convex optimization, DC programming, DCA, Branch and Bound, DC relaxation, Bilevel Programming, Mixed integer programming, Portfolio selection, Production management.