THÈSE

en vue de l'obtention du titre de

DOCTEUR DE L'UNIVERSITÉ DE PAUL VERLAINE-METZ

(arrêté ministériel du 30 October 1993)

Spécialité INFORMATIQUE

présentée par

LE HOAI MINH

Titre de la thèse :

MODÉLISATION ET OPTIMISATION NON CONVEXE BASÉES SUR LA PROGRAMMATION DC ET DCA POUR LA RÉSOLUTION DE CERTAINES CLASSES DES PROBLÈMES EN FOUILLE DE DONNÉES ET CRYPTOLOGIE.

Date de soutenance : 24 Octobre 2007

Composition du Jury :

Président	Tao PHAM DINH	*Professeur, INSA-Rouen*
Rapporteurs	Jean Pierre CROUZEIX	*Professeur, Université Blaise Pascal*
	Abdel LISSER	*Professeur, Université de Paris-Sud*
Examinateurs	Noëlle CARBONELL	*Professeur, Université Henri Poincaré- Nancy 1*
	Hans-Herman BOCK	*Professeur, University of Aachen Wuellnerstr 3*
	Pascal BOUVRY	*Professeur, Université du Luxembourg*
	Franciszek SEREDYNSKI	*Professeur, Institute of Computer Science Polish Academy of Sciences*
Directeur de thèse	Hoai An LE THI	*Professeur, Université Paul Verlaine-Metz*

THÈSE PRÉPARÉE AU SEIN DU
LABORATOIRE D'INFORMATIQUE THÉORIQUE ET APPLIQUÉE (LITA)
UNIVERSITÉ DE PAUL VERLAINE-METZ



AVERTISSEMENT

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

- Contact SCD Metz : theses.scd@univ-metz.fr

CPI articles L 122.4
CPI articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Je dédie cette thèse à ma famille
qui m'a soutenu tout au long de ce parcours.

Remerciements

Je souhaite ici témoigner mon affection à mes grand-parents et ma mère pour les sacrifices consentis, pour les encouragements qu'ils m'ont donnés tout au long de ces 3 années.

La préparation de cette thèse, sous la direction du Madame le Professeur Hoai An Le Thi, a été réalisée au sein du laboratoire LITA de l'université de Paul-Verlaine Metz que je tiens vivement à remercier tous ceux m'ont accordé cette ambiance de travail.

Je voudrais exprimer mon endettement à Madame le Professeur Hoai An Le Thi qui a accepté d'être ma directrice de thèse pour son aide inestimable, les conseils pertinents et formateurs, les encouragements qu'elle m'a donnés.

Je tiens à remercier plus particulièrement Monsieur le Professeur Tao Pham Dinh, directeur de l'équipe Modélisation et Optimisation Appliquée de L'INSA de Rouen pour ses conseils, son suivi de mes recherches, sa sympathie et les discussions très intéressantes qu'il a menées pour me suggérer les voies de recherche.

J'adresse toute ma gratitude à Monsieur le Professeur Pascal Bouvry qui m'a offert l'opportunité de travailler dans son équipe dans le cadre du projet "Techniques for Secure Grid and Ad-hoc Networks (TeSeGrAd)" financé par le Fonds National de la Recherche du gouvernement Luxembourgeois. Grâce à lui je découvre des problèmes d'actualité en Cryptologie étudiés dans cette thèse ainsi que les aspects intéressants des algorithmes génétiques. Ses expériences dans ces domaines m'ont été très bénéfiques. Merci également à son équipe pour m'avoir accueilli dans une ambiance agréable.

Je souhaite également exprimer ma gratitude à Monsieur Abdel Lisser, Professeur à l'Université Paris-Sud, Monsieur Jean Pierre Crouzeix Professeur à Université Blaise Pascal Clermont-Ferrand de m'avoir fait l'honneur d'accepter la charge de rapporteur de ma thèse.

Je tiens à remercier Madame le Professeur Noëlle Carbonell à Université Henri Poincaré-Nancy 1, Monsieur Professeur Hans-Herman Bock à Institute for Statistics Technical University of Aachen Wuellnerstr 3, Monsieur Pascal Bouvry Professeur à l'université Luxembourg, Monsieur Professeur Franciszek Seredynski à Institute of Computer Science Polish Academy of Sciences d'avoir participé à juger mon travail.

Je remercie Huyen, qui m'a donné tant d'amour et de tendresse. Je tiens à montrer tout ma gratitude enver tous mes collègues et en particulièrement Phuc, Vinh pour les conseils, le partage dans le travail et dans la vie.

Enfin à tous ceux qui m'ont soutenu de près ou de loin et à tous ceux qui m'ont incité même involontairement à faire mieux, veuillez trouver ici le témoignage de ma profonde gratitude.

Table des matières

I Méthodologie **17**

1 Introduction à la programmation DC et DCA **19**

1.1 Eléments de base de l'analyse DC 20

1.1.1 Notations et propriétés 20

1.1.2 Fonctions convexes polyédrales 22

1.1.3 Fonction DC 23

1.2 Optimisation DC 24

1.2.1 Dualité DC 25

1.2.2 Optimalité globale en optimisation DC 26

1.2.3 Optimalité locale en optimisation DC 27

1.3 DCA 29

1.3.1 Principe de DCA 29

1.3.2 Existence des suites générées 30

1.3.3 Calcul des sous-gradients 32

1.3.4 Optimisation DC polyédrale 32

1.3.5 Interprétations de DCA 33

2 Introduction aux algorithmes génétiques **37**

2.1 Fonction d'adaptation 38

2.2 Codage 39

2.2.1 Codage binaire 39

2.2.2 Codage Gray 41

2.2.3 Codage réel 41

2.2.4 Codage à caractères multiples (codage multiparamétré) 42
2.2.5 Codage sous forme d'arbre 42
2.3 Population initiale 42
2.4 Opérateurs 43
2.4.1 Sélection 43
2.4.2 Croisement 45
2.4.3 Mutation 47
2.5 Principales caractéristiques de l'algorithme génétique 48

II Modélisation DC et DCA pour le clustering flou et le clustering hiérarchique 49

3 Introduction à la classification 51
3.1 Fouille de données et Classification 51
3.2 Notions préliminaires 52
3.2.1 Objets et Classes 52
3.2.2 Classification supervisée et classification non supervisée 53
3.2.3 Processus de classification 53
3.2.4 Variables et sélection de variables 53
3.2.5 Distance et similarité 54
3.3 Clustering (Classification non supervisée) 54
3.3.1 Clustering dur et clustering flou 55
3.3.2 Clustering hiérarchique et clustering par partitionnement 55
3.4 Principales approches classiques de clustering 55
3.4.1 Approches statistiques : clustering via le modèle de mélange 56
3.4.2 Approches d'optimisation pour le clustering 56
3.4.3 Méthodes de clustering hiérarchique 59
3.4.4 Chois de l'algorithme de clustering 60
3.5 La programmation DC et DCA pour clustering et nos contributions 60

4 Le clustering flou via la programmation DC et DCA 61

4.1 Introduction 61
4.2 Résolution du problème (4.3) par la programmation DC et DCA 64
4.2.1 La première approche : une décomposition DC naturelle 65
4.2.2 La deuxième approche : une intéressante décomposition DC pour le cas $m \geq 2$ 67
4.2.3 La troisième approche : la meilleure décomposition DC 69
4.2.4 Algorithmes 71
4.3 La recherche d'un bon point initial pour DCA par une procédure alternative FCM-DCA 73
4.4 Expériences numériques 74
4.5 Segmentation d'image par le clustering flou 78
4.5.1 Introduction 78
4.5.2 Modèle de FCM avec l'information spatiale 80
4.5.3 Expériences numériques 80

5 Clustering via la programmation DC pour la détermination d'arbre hiérarchique de multidiffusion 89

5.1 Introduction 89
5.2 Formulation du problème 91
5.2.1 Première approche : un centre total artificiel 92
5.2.2 Deuxième approche : le centre total réel 93
5.2.3 La troisième approche : chercher simultanément tous les centres . . . 93
5.3 Résolution de ($P1_\tau$), ($P2_\tau$) et ($P3_\tau$) par DCA 94
5.3.1 Résoudre ($P1_\tau$) par DCA 94
5.3.2 Résoudre ($P2_\tau$) par DCA 98
5.3.3 Résoudre ($P3_\tau$) par DCA 100
5.3.4 Retrouver les centres réels 104
5.3.5 Chercher un point initial pour DCA 104
5.4 Expériences numériques 105

III Modélisation et Optimisation Globale basée sur DCA, GA

et une méthode de coupes en Cryptologie 109

6 Introduction à la cryptologie 111

6.1 Objectifs de la cryptographie 111

6.2 Les méthodes historiques de la cryptographie 113

6.2.1 La cryptographie par substitution mono-alphabétique 114

6.2.2 La cryptographie par substitution polyalphabétique 115

6.2.3 Le chiffre de substitution homophonique 116

6.2.4 La substitution par polygrammes 117

6.3 Les méthodes modernes de la cryptographie 117

6.3.1 Cryptographie symétrique 117

6.3.2 Cryptographie asymétrique 121

6.3.3 Le RSA 121

6.3.4 Comparaison entre cryptographie asymétrique et symétrique 122

6.4 Une méthode du future : Cryptographie quantique 122

6.4.1 Photons polarisés 123

6.4.2 Protocole de transmission de la clé 124

6.5 Cryptanalyse : quelques attaques connues 125

6.6 Optimisation en Cryptologie et nos contributions 126

6.6.1 Cryptanalyses des chiffrements classiques par les techniques d'optimisation méta-heuristique 126

6.6.2 Schémas de cryptanalyses basés sur les problèmes standard difficiles 126

6.6.3 Optimisation en cryptographie du $20^{\text{ième}}$ siècle : stratégies de concevoir des systèmes cryptographiques 127

6.6.4 Optimisation dans les systèmes cryptographiques du $21^{\text{ième}}$ siècle : améliorations de conception des systèmes 127

6.6.5 Nos contributions 128

7 Contributions aux techniques cryptographiques modernes via la construction des fonctions booléennes équilibrées de haut degré de non-linéarité 129

7.1 Introduction 129

7.2 Préliminaire 131

7.3 Formulation mathématique et reformulation du problème 133
7.4 Résolution du problème (Q) par la programmation DC et DCA 135
7.5 Combinaison de DCA et Algorithmes Génétiques pour la résolution de (Q_{dc}) 137
7.6 Expériences numériques 141

8 Contribution à la cryptanalyse via la résolution des problèmes "Perceptron" et "Permuted Perceptron" 143

8.1 Introduction 143
8.2 La résolution du "Perceptron Problem" (PP) 146
8.2.1 Modélisation 146
8.2.2 Résolution de (8.2) par DCA 147
8.2.3 Expériences numériques 149
8.3 La résolution du "Permuted Perceptron Problem" (PPP) 149
8.3.1 Modélisation 149
8.3.2 Résolution de (8.15) par DCA 152
8.3.3 Chercher un point initial 154
8.3.4 DCA et méthode de coupes 157
8.3.5 Expériences numériques 160

Table des figures

2.1 Schéma d'un algorithme génétique simple 38
2.2 De l'espace réel au codage 39
2.3 Exemple de la sélection par roulette 44
2.4 Exemple de la sélection par roulette 45
2.5 *Slicing crossover classique*(gauche) et *2-point slicing crossover*(droite) 46
2.6 La mutation 47

3.1 Exemple de dendogramme 59

4.1 Le clustering dur (gauche) et le clustering flou (droite) 62
4.2 Résultat sur données "Yeast" : nombre d'itérations (gauche) et temps de calcul (droite) 77
4.3 Fuzzy-DCA-3 avec et sans la procédure FCM-DCA, le coût de cluster (gauche) et nombre d'itérations (droite) 77
4.4 Le pixel et ses 4 voisinages 80
4.5 L'image originale et les résultats de segmentation (c=3). 82
4.6 L'image originale avec le bruit et les résultats de segmentation (c=3). 82
4.7 L'image médical originale et les résultats de segmentation (c=2). 83
4.8 L'image médicale avec le bruit Gaussien et les résultats de segmentation (c=3). 83
4.9 L'image médical originale et les résultats de segmentation (c=3). 84
4.10 L'image médicale avec le bruit Gaussien et les résultats de segmentation (c=3). 84
4.11 L'image **Blume** et les résultats de segmentation (c=3). 85
4.12 L'image médical avec le bruit Gaussien et les résultats de segmentation (c=5). 85
4.13 Comparaison sur le temps de calcul de **Algorithme FCM** et **Algorithme 4.5, 4.4**. 86

5.1 L'arbre hiérarchique de 2 niveaux 90

6.1 Le schéma de base de la cryptographie 112
6.2 Les principales méthodes de cryptographie 113
6.3 Batôn de Plurtaque 113
6.4 Le carré de Vigenere 116
6.5 DES 119
6.6 Triple-DES 119
6.7 AES 120
6.8 Photon polarisé 123
6.9 Traversée d'un filtre rectiligne 123
6.10 Protocole de transmission de clé dans cryptographie quantique 124

7.1 Opérateur croisement 139
7.2 Opérateur mutation 139

Liste des tableaux

4.1 Résultat sur premier ensemble de jeux de données 75
4.2 Résultat sur jeu de données "Yeast" 76
4.3 Résultat sur jeu de données "Serum" 76
4.4 Résultat sur jeu de données "Human cancer" 76
4.5 Résultat sur jeu de données "Breast cancer" 76
4.6 Résultat sur jeu de données "Ovarian cancer" 77
4.7 Comparaison sur le temps de calcul de **Algorithme FCM** et **Algorithme 4.5, 4.4**. 86

5.1 Résultat numérique sur les données de 51 villes north américaines 107
5.2 Comparaison avec les autres méthodes 107
5.3 Résultats numériques sur les données aléatoires 108
5.4 Comparaison de **CHBN-DCA-3-IP** et **CHBN-DCA-3** 109

7.1 Résultat sur différentes valeurs de nbIterMax dans l'algorithme génétique ($n = 9$). 141
7.2 Non linearité $N_f(7.7)$ pour n de 8 à 15 142
7.3 Nombre d'iterations 142

8.1 Comparaison entre **PP-DCA-2**, **RC** sur 100 instances du problème PP . . . 150
8.2 Résultat de test de **PPP-DCA**, **DCA&CUT** sur 100 instances du problème PPP 161

Liste des Publications et Conférences

LE Hoai Minh

Article avec comité de lecture

- H.A Le Thi, M. Le Hoai and T. Pham Dinh, *Optimization based DC programming and DCA for Hierarchical Clustering*, European Journal of Operational Research, Vol. 183, p. 1067-1085, 2007.

- H.A Le Thi, M. Le Hoai and T. Pham Dinh, *Hierarchical Clustering based on Mathematical Optimization*, Advances on Knowledge Discovery and Data Mining, p.160-173, Lecture Notes in Artificial Intelligence (LNAI) 3918, 2006.

- H.A Le Thi, M. Le Hoai and T. Pham Dinh, *Fuzzy clustering based on nonconvex optimisation approaches using difference of convex (DC) functions algorithms*, Advances in Data Analysis and Classification, p.85-104, Vol. 1, 2007.

- H.A Le Thi, M. Le Hoai, T. Pham Dinh and P. Bouvry, *A combined DCA - GA for constructing highly nonlinear balanced Boolean functions in cryptography*, submitted in "Journal of Global Optimization".

- H.A Le Thi, M. Le Hoai, P. Nguyen Trong and T. Pham Dinh, *Noisy image segmentation by a robust Fuzzy C-Means clustering algorithm based DC programming and DCA,* Submitted to Optimization and Engineering.

- 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, p.703-714, 2007.

- H.A Le Thi, M. Le Hoai and T. Pham Dinh, *Clustering via la programmation DC pour la détermination d'arbre hiérarchique de multidiffusion*, p. 231-240, Proceedings de la 12èmes Rencontres de la Société Francophone de Classification, 2005.

- H.A Le Thi, M. Le Hoai and T. Pham Dinh, *Une approche en programmation DC pour la classification floue*, p. 140-144, Proceedings de la 13èmes Rencontres de la Société Francophone de Classification, 2006.

Communications aux colloques internationaux avec actes publiés

- M. Le Hoai, H.A Le Thi, T. Pham Dinh and P. Bouvry, *A Determinist Optimization Approach for generating highly nonlinear balanced Boolean Function in Cryptography*, High Performance Scientific Computing (HPSC), 6-10 March 2006 Hanoi, Vietnam.

- 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*, 7èmes Journées Francophones "Extraction et Gestion des Connaissance" EGC 2007,23-26 Février 2007 Namur, Belgique.

- M. Le Hoai, H.A Le Thi, T. Pham Dinh and P. Bouvry, *A combined DCA - GA for constructing highly nonlinear balanced Boolean functions in cryptography*, 4th International Conference on Computational Management Science, 20-22 Avril 2007, Genava.

Communications aux colloques nationaux avec actes publiés

- M. Le Hoai, H.A Le Thi and T. Pham Dinh, *Une approche en programmation DC pour le Fuzzy Clustering*, ROADEF 06, Février 2006 Lille, France.

- M. Le Hoai, H.A Le Thi, T. Pham Dinh and P. Bouvry, *Une combinaison de DCA et algorithme génétique pour la construction des fonctions booléennes dans la cryptographie*, Conférence Scientifique conjointe en Rechercher Opérationelle et Aide à la décision FRANCORO/ROADEF 07, 20-23 Février 2007, Grenoble, France.

- P. Nguyen Trong, H.A Le Thi, M. Le Hoai and T. Pham Dinh, *Segmentation des images RMI par la classification floue via DCA*, Conférence Scientifique conjointe en Rechercher Opérationelle et Aide à la décision FRANCORO/ROADEF 07, 20-23 Février 2007, Grenoble, France.

Introduction générale

Contexte général et problématique

L'optimisation offre un cadre algorithmique très riche pour tous les domaines de sciences appliquées. On peut distinguer deux branches de l'optimisation déterministe : la programmation convexe et la programmation non convexe. 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) 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), R. Horst, P. Pardalos et N. V. Thoai ([85], [86], [87], [88]) ... L'inconvénient majeur des méthodes globales est leur lourdeur (encombrement en places-mémoires) et leur coût trop important. Elles ne sont pas applicables aux problèmes d'optimisation non convexes réels qui sont souvent de très grande dimension.

ii) Approches locales et globales d'analyse convexe qui sont basées sur l'analyse et l'optimisation convexe. Ici la programmation DC (Différence de deux fonctions Convexes) et DCA (DC Algorithmes) jouent le rôle central car la plupart des problèmes d'optimisation non convexe sont formulés/reformulés sous la forme DC. Sur le plan algorithmique, l'essentiel repose sur les algorithmes de l'optimisation DC (DCA) introduits par T . Pham Dinh en 1985 à l'état préliminaire et développés intensivement à travers de nombreux travaux communs de H.A Le Thi et T. Pham Dinh depuis 1993 pour devenir maintenant classiques et de plus en plus utilisés par des chercheurs et praticiens de par le monde, dans différents domaines des sciences appliquées ([52] - [68]).

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

Un programme DC est de la forme

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

où $g, h \in \Gamma_0(\mathbb{R}^n)$, le cône convexe de toutes les fonctions convexes semi-continues inférieurement et propres sur $\mathbb{R}^n$. Une telle fonction f est appelée fonction DC et g et h des composantes DC de f. 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 l'optimisation non convexe basées sur *la programmation DC et DCA* pour certaines classes de problèmes issus de deux domaines importants, à savoir la *Fouille de Données* et la *Crytologie*.

L'un des défis pour les scientifiques à l'heure actuelle consiste en l'exploitation optimale des informations stockées/évoluées dans de très nombreuses ressources. Les méthodes d'apprentissage automatique et la fouille de données qui offrent aujourd'hui une technologie mature pour traiter les problèmes « classiques » doivent faire face à l'explosion des nouveaux besoins liée au développement du web, aux masses de données, aux nouveaux types et formats de données. Un premier enjeu consiste à dépasser le cadre actuel de l'apprentissage pour s'attaquer à cette nouvelle gamme de problèmes et pour répondre à ces nouveaux besoins. Il faut pour cela développer des liens avec d'autres approches algorithmiques du traitement de données comme l'analyse combinatoire, l'optimisation, les différentes branches du traitement du signal ... Durant cette dernière décennie de nombreuses méthodes d'optimisation se sont développées avec succès pour l'apprentissage comme témoigne le nombre sans cesse croissant des chercheurs en extraction et gestion de connaissance qui y ont recours. En particulier en Data Mining qui constitue *une mine des programmes DC* dont une résolution appropriée devrait recourir à la Programmation DC et DCA. En effet la liste indicative (non exhaustive et nécessitant des fréquentes mises à jour) des références suivantes témoigne de la vitalité et la puissance et la percée de l'approche " Programmation DC et DCA " dans la communauté Data Mining : [52], [53], [54], [56], [59], [60], [62], [63], [64], [65], [184]. De notre part, nous nous intéressons dans ce travail aux deux classes de problèmes en *Classification* (Clustering), un des domaines principaux de Data Mining : la classification floue (fuzzy clustering) et la classification hiérarchique à multi-niveaux (multilevel hierarchical clustering).

La classification automatique (ou classification non supervisée, ou encore clustering) est une des problématiques majeures en extraction des connaissances à partir de données. La classification implique la tâche de classer des points dans les classes homogènes de telle sorte que les points dans la même classe soient aussi semblables que possible et les points dans différentes classes soient aussi différentes que possible. De nombreux sous-problèmes ont été identifiés, comme par exemple la sélection des données ou des descripteurs, la variété des espaces de représentation, la nécessité de découvrir des concepts, d'obtenir une hiérarchie, etc. La popularité, la complexité et toutes ces variantes du problème de la classification ont donné naissance à une multitude de méthodes de résolution.

Parmi les techniques de classification automatique, on peut distinguer les méthodes de classification *dure* et *floue*. Dans le cas classique, classification dure, chaque point est classé dans une et une seule classe. Mais dans les applications réelles, on constate qu'il y a très souvent une frontière pointue entre les classes. Pour mieux adapter aux telles situations, la classification floue a été introduite. Dans cette approche, à chaque point est associée une probabilité d'appartenance à une classe.

Il existe plusieurs méthodes pour la classification floue (voir par exemple [188], [207], [134], [135], [136]) parmi lesquelles Fuzzy C-Means (FCM) est la plus utilisée. FCM a été introduite par J. Bezdek en 1981 ([188]) et a été beaucoup développée dans les années 90. C'est une approche heuristique basée sur la minimisation d'un critère d'optimisation qui est non convexe. Elle a été appliquée avec succès dans plusieurs problèmes tels que le diagnostic médical [135], la classification de textes [136], l'analyse d'image, etc.

Dans ce travail nous considérons le modèle d'optimisation de FCM et développons la programmation DC et DCA pour sa résolution. Plusieurs formulations DC correspondants aux différentes décompositions DC sont proposées. La question de recherche d'un bon point initial est soigneusement étudiée. De nombreux tests numériques sur les données réelles sont réalisés, en particulier sur les données biomédicales, celles de d'images médicales (pour la segmentation d'image).

Si la clasification automatique par partitionnement consiste à regrouper des données en des groupes de rôle équivalent, il n'en est pas de même en classification hiérarchique. La classification hiérarchique à multi-niveaux consiste au regoupement des objets de données dans une *hiérarchie* des clusters. Ce problème historique ([241], [243], [244]) a nombreuses applications dans différents domaines car dans la pluspart des cas les données ont une structure hiérarchique. Notre travail en classification hiérarchique est motivé par une de ses applications intéressantes et très importante, à savoir la communication multicast. Une communication multicast est une communication dans laquelle un même paquet de données peut être envoyé à un groupe de récepteurs, quelque soit leur localisation. Citons ici quelques intérêts de la communication multicast : Systèmes de Groupware pour faciliter la conception et l'évaluation de collaboration, Vidéo conférence, Diffusion en temps réel des événements internationaux, Peer-to-peer applications pour le partage des données ou traitement, etc. Un réseaux multicast est représenté comme un arbre hiérarchique, et un routage d'un réseau multicast peut se faire par un algorithme de classification hiérarchique.

La programmation mathématique (l'optimisation) est largement utilisée pour la classification par partitionnement, alors qu'elle n'est pas populaire en classification hiérarchique. En effet, à notre connaissance, notre travail est le deuxième (après [244]) qui est basé sur des modèles d'optimisation déterministe pour la résolution de ce problème. C'est un problème non convexe, non différentiable de très grande dimension pour lequel nous avons reformulé sous la forme des trois programmes DC différents et développé les DCA correspondants.

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

1. Etude approfondie des modèles d'optimisation non convexe et la modélisation DC de ces problèmes ; Formulations et Reformulations des programmes DC équivalents, choix des décompositions DC les mieux adaptées.
2. Mise en place des DCA résultants.
3. Etude des techniques de recherche d'un bon point initial pour DCA.
4. Implémentations et Simulations numériques pour leurs validations.
5. Mise en valeur ces algorithmes à travers leurs applications dans différents domaines : la bioinformatique, la segmentation d'image, le réseaux multidiffusion (multicast), ...

Il est à noter que les méthodes globales ne sont pas applicables à ces deux problèmes complexes de très grande dimension en pratique. On verra dans la suite que nos algorithmes basés sur DCA sont supérieurs aux méthodes existantes, aussi bien sur la qualité de solution que le temps de calcul, en particulier pour des problèmes de grande taille.

Le deuxième domaine d'application étudié dans cette thèse est la *Cryptologie*. Malgré son antiquité (plus de 3000 ans), la cryptologie est toujours à l'état embryonnaire. La cryptologie est composée de deux branches principales : la cryptographie et la cryptanalyse. La cryptographie permet l'échange confidentiel de messages. L'émetteur, A, envoie un message à B ; s'il craint que des espions n'interceptent le message, il doit s'assurer que ceux-ci ne pourront en extraire aucune information. La cryptographie intervient à ce niveau : A peut à l'aide de certaines techniques de chiffrement produire un cryptogramme, que seul B, muni de la clé de chiffrement, pourra déchiffrer. À l'opposé de ces procédés de chiffrement se trouve la cryptanalyse, qui consiste à endosser le rôle d'un espion, tente de déchiffrer le message sans avoir la connaissance complète des paramètres qui ont permis de le chiffrer.

Ces dernières années ont révolutionné la manière de percevoir les télécommunications. Que ce soit par l'apparition des grilles de calcul ou par la généralisation des réseaux sans fils/ad-hocs (GSM, Wi-Fi, Bluetooth, etc.). Il est ainsi primordial de sécuriser ces réseaux et de protéger les flux de données et les technologies classiques s'avèrent peu adaptées. Les besoins liés aux nouveaux types de réseaux et de matériels et aux nouvelles utilisations de réseaux ont aussi fait évoluer les besoins en matière de sécurité. Un exemple flagrant est l'algorithme à clé secrète DES (Data Encryption Standard) qui est considéré comme sûr en 1990 est aujourd'hui considéré comme désuet. La puissance de calcul des nouveaux ordinateurs tend à rendre les algorithmes existants désuets et cassables. De plus les besoins d'informatique embarquée les rends aussi trop coûteux. Dès lors le besoin émerge de nouvelles techniques plus robustes et moins coûteuses.

Notre travail s'inscrit dans le cadre de la participation au développement d'une nouvelle génération d'outils, plus légère et robuste que la génération existante, destinés à la cryptologie. Nous nous intéressons à la fois à la *cryptographie* et la *cryptanalyse*.

Le premier travail se situe dans le cadre des techniques S-box (Substitution boxes) qui jouent un rôle important dans la cryptographie moderne. Il s'agit de la construction des fonctions booléennes équilibrées de haut degré de non-linéarité - un des problèmes cruciaux

en Cryptographie. En fait, les fonctions booléennes sont des éléments de base pour construire un S-box, et la difficulté de cryptanalyse d'un S-box augmente selon le degré de non-linéarité des fonctions booléennes qui le constituent. Plusieurs algorithmes d'optimisation heuristique ont été développés pour ce problème. Cependant aucune méthode déterministe est connue avant notre travail. Nous avons réussi à concevoir un modèle d'optimisation déterministe (de très grande dimension) qui est la minimisation d'une fonction convexe polyédrale sur un polyèdre convexe en variables 0-1. Grâce à la pénalité exacte nous reformulons ce problème en programmation DC et utilisons DCA pour la résolution numérique. Plusieurs versions de combinaison de deux approches - DCA et les algorithmes génétiques (AG) sont étudiées dans le but d'exploiter simultanément l'efficacité de chaque approche. Cette combinaison est très efficace pour la recherche d'un bon point initial pour DCA.

Le deuxième travail concerne des techniques de cryptanalyse d'un schéma d'identification basé sur les deux problèmes "Perceptron" et "Perceptron Permuté" (en Anglais : "Perceptron Problem" (PP) et "Permuted Perceptron Problem" (PPP)). PPP est un schéma d'authentification (i.e., une procédure permettant à une personne de vérifier, lors d'une communication, qu'il échange des données avec le bon interlocuteur) proposé en 1995 par David Pointcheval ([256]) en se basant sur le problème $\mathcal{NP}$-complet PP qui doit son existence au fameux problème perceptron dans les réseaux de neurones. Plusieurs tentatives d'attaques ont été faites sur les problèmes PP et PPP pour évaluer la sécurité des protocoles d'identification basés sur PPP. Certaines attaques basées sur le recuit simulé ont été développées depuis 1999. Cependant aucun modèle d'optimisation déterministe n'a vu le jour pour ces deux problèmes. Nous avons formulé PP comme la minimisation d'une fonction DC polyédrale sur un polyèdre convexe et utilisé DCA pour sa résolution. Notre algorithme peut résoudre PP en moins de 100 seconds pour la taille de 1001×1017.

La modélisation de PPP est beaucoup plus sophistiquée, mais nous obtenons à la fin également une programmation DC polyédrale en variables zéro-un pour laquelle nous développons DCA et une méthode de coupes.

Il est à noter que tous les trois problèmes étudiés dans cette deuxième partie sont des problèmes combinatoires $\mathcal{NP}$-difficiles et de très grande dimension en pratique. Nos contributions propres portent à la fois sur la modélisation et les méthodes de résolution.

Organisation de la thèse

La thèse est divisée en trois parties et est composée de huit chapitres. Dans la première partie intitulée "Méthodologie" nous présentons des outils théoriques et algorithmiques servant des références aux autres. Le premier chapitre concerne la programmation DC et DCA tandis que le deuxième porte sur les algorithmes génétiques. Dans la deuxième partie nous développons la programmation DC et DCA pour la résolution de deux classes de problèmes en Data Mining. Nous commençons par une introduction à la classification et au clustering (chapitre trois). La classification floue est étudiée dans le chapitre quatre et la classification hiérarchique - chapitre cinq. La troisième partie porte sur la Cryptologie. Après une introduction à la cryptologie (chapitre six) nous présentons la génération des fonctions booléennes

équilibrées de haut degré de non-linéarité par la programmation DC et DCA et par l'hybridation DCA-GA dans le chapitre sept. Enfin, la résolution des deux problèmes PP et PPP en cryptanalyse par DCA et une méthode de coupes est développée dans le dernier chapitre.

Première partie

Méthodologie

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 relative aux opérations fréquemment utilisées en optimisation. Cependant, il faut attendre le milieu des années 80 pour que la classe des fonctions DC soit introduite en optimisation, élargissant ainsi la classification des problèmes d'optimisation avec l'apparition de la programmation DC. On distingue deux grandes approches DC :

1. L'approche combinatoire (cette terminologie est due au fait que les nouveaux outils introduits ont été inspirés par les concepts de l'optimisation combinatoire) en optimisation globale continue.
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, techniques de coupes, méthodes d'approximation fonctionnelle et ensembliste) ; ces algorithmes relativement sophistiqués sont plutôt lourds à mettre en oeuvre, ils doivent donc être réservés à des problèmes de dimensions raisonnables possédant des structures bien adaptées aux méthodes lorsqu'il est important d'isoler l'optimum global.

Le pionnier de cette approche est H. Tuy dont le premier travail remonte à 1964. Ses travaux sont abondants, citons les livres de Horst-Tuy([107, 108]) qui présentent la théorie, algo-

rithmes et applications de l'optimisation globale. Viennent ensuite les principales contributions de l'Ecole Américaine (P.M. Pardalos, J.B. Rosen,...), Allemande (R. Horst, ...), Française (H.A Le Thi, T. Pham Dinh,...) et l'Ecole Vietnamienne (T. Phan Thien, M. Le Dung , ...).

La seconde approche repose sur l'arsenal puissant d'analyse et d'optimisation convexes. Le premier travail dû à T. Pham Dinh (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) ([101]) sur la dualité et l'optimalité locale en optimisation DC généralise de manière élégante les résultats établis par Pham Dinh en maximisation convexe. La théorie de l'optimisation DC est ensuite développée notamment par T. Pham Dinh, J.B.H. Urruty, Jean - Paul Penot, T. Phan Thien, H.A Le Thi. Sur le plan algorithmique dans le cadre de la seconde approche, on dispose actuellement que des DCA (DC Algorithms) introduits par T. Pham Dinh (1986), qui sont basés sur les conditions d'optimalité et de dualité en optimisation DC. Mais il a fallu attendre les travaux communs de H.A Le Thi et T. Pham Dinh (voir [1]-[32] et [41]-[46]) pour qu'il s'impose définitivement en optimisation non convexe comme étant des algorithmes les plus simples et performants, capables de traiter des problèmes de grande taille.

Nous reportons dans ce chapitre les principaux résultats relatifs à la programmation DC et DCA qui nous seront les plus utiles pour nos travaux. Ces résultats sont extraits de ceux présentés dans H.A Le Thi 1994 ([1]), H.A Le Thi 1997 ([2]). Pour une étude détaillé nous nous référons à ces deux références (voir également [1]-[32] et [41]-[46]).

1.1 Eléments de base de l'analyse DC

1.1.1 Notations et propriétés

Ce paragraphe est consacré à un rapide rappel d'analyse convexe pour faciliter la lecture de certains passages. Pour plus de détails, on pourra se référer aux ouvrages de P.J Laurent ([89]), de R.T Rockafellar ([96]) et d'A. Auslender ([69]). 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 $-\infty - (+\infty) = +\infty$ ([96]). Etant donnée une fonction $f : S \longrightarrow \overline{\mathbb{R}}$ définie sur un ensemble S convexe de X, on appelle domaine effectif de f l'ensemble

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

et épigraphe de f

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.1.2 Fonctions convexes polyédrales

Une partie convexe C est dite convexe polyédrale si elle est de la forme

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

Une fonction est dite convexe polyédrale si elle est de la forme

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

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 *([96])*
- *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*

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

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

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

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

1.1.3 Fonction DC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.2 Optimisation DC

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

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

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

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

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

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

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

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

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

souligner que, comme on a vu en haut, la classe des fonctions DC est remarquablement stable par rapport aux opérations fréquemment utilisées en optimisation. En outre, on dispose d'une élégante théorie de la dualité ([35, 36, 101, 110, 1, 2, 5]) qui, comme en optimisation convexe, a de profondes répercussions pratiques sur les méthodes numériques.

Sur le plan algorithmique, les algorithmes de l'optimisation DC (DCA) dus à T. Pham Dinh ([40, 41]) 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 ([35, 40]). Cependant, il a fallu attendre les travaux communs de H.A Le Thi et T. Pham Dinh au cours de ces dix dernières années (voir [1]-[32] et [41]-[46]) pour que le DCA devienne maintenant classique et populaire ([52] - [68]).

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 ([96]). 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 T. Pham Dinh (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 H.A Le Thi (1997) (voir également [5]).

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

Soient $g(x)$ et $h(x)$ deux fonctions convexes propres sur X($g, h \in \Gamma_0(X)$), considérons le problème DC général (les contraintes convexes peuvent être incorporées à la fonction objectif à l'aide de la fonction indicatrice) :

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

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 ([101]).

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 ([101]).

□

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 ([111]) est formulée à l'aide des ϵ-sous-différentiels de g et h. Sa démonstration (basée sur l'étude du comportement du ϵ-sous-différentiel d'une fonction convexe en fonction du paramètre ϵ) est compliquée. La démonstration dans [2] est plus simple et convient bien au cadre de l'optimisation DC : elle exprime tout simplement que cette condition d'optimalité globale est une traduction géométrique de l'égalité des valeurs optimales des programmes DC primal et dual.

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

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

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

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

Du fait que $\partial_\epsilon h(x^0) = \partial h(x^0) = \{0\}, \quad \forall \epsilon > 0, \forall x \in X$, et la croissance du ϵ-sousdifférentiel en fonction de ϵ, la relation (1.7) est équivalente à (1.6).
(ii) D'une manière plus générale, considérons les décompositions DC de $f \in \Gamma_0(X)$ de la forme $f = g - h$ avec $g = f + h$ et $h \in \Gamma_0(X)$ finie partout sur X. Le problème DC correspondant est un "faux" problème DC car c'est un problème d'optimisation convexe. Dans ce cas, la relation (1.7) est équivalente à

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

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

1.2.3 Optimalité locale en optimisation DC

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

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

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

où V_x désigne un voisinage de x.

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

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

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

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

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

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

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

Remarquons que pour un certain nombre de problème DC et en particulier pour h polyédrale, la condition nécessaire (1.9) est également suffisante, comme nous le verrons un peu plus loin. On dit que $x^\bullet$ est un point critique de $g-h$ si $\partial h(x^\bullet) \cup \partial g(x^\bullet)$ est non vide ([101]). C'est une forme affaiblie de l'inclusion sous-différentielle. La recherche d'un tel point critique est à la base de DCA (forme simple) qui sera étudiée dans la section suivante. En général DCA converge vers une solution locale d'un problème d'optimisation DC. Cependant sur le plan théorique, il est important de formuler des conditions suffisantes pour l'optimalité locale.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.3 DCA

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

1.3.1 Principe de DCA

La construction des DCA, découverte par T. Pham Dinh (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 $\{y^k\}$, candidats respectifs aux solutions des problèmes primal et dual que l'on améliore à chaque itération. 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. $h^\star - g^\star$).

Schéma de DCA simplifié

L'idée principale de la mise en oeuvre de l'algorithme (forme simple) est de construire deux suites $\{x^k\}$ et $\{y^k\}$ qui convergent vers des solutions primale et duale x^* et y^*verifiant des

conditions d'optimalité locale et

$$x^* \in \partial g^*(y^*),\ y^* \in \partial h(x^*). \tag{1.14}$$

Cette relation (1.14) implique que x^* est une solution optimale du programme convexe

$$\inf\{f(x) + h(x) - [h(x^*) + \langle x - x^* \rangle] : x \in X\}. \tag{1.15}$$

Le schéma général de DCA prend la forme :

$$y^k \in \partial h(x^k); : x^{k+1} \in \partial g^*(y^k). \tag{1.16}$$

On construit ainsi :

Algorithme 1.1
Algorithme DCA

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

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

1.3.2 Existence des suites générées

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

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

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

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

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

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

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

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

(ii)

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

où $dx^k = x^{k+1} - x^k$

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

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

2.

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

3.

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

4.

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

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

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

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

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

1.3.3 Calcul des sous-gradients

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

$$\begin{array}{lcl} x^k & \leftarrow & y^k \in \partial h(x^k) \\ & \swarrow & \\ x^{k+1} \in \partial g^\star(y^k) & \leftarrow & y^{k+1} \in \partial h(x^{k+1}) \\ (y^k \in \partial g(x^{k+1})) & & (x^{k+1} \in \partial h^\star(y^{k+1})) \end{array} \tag{1.17}$$

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,

$$\partial g^\star(y^k) = argmin\{g(x) - \langle y^k, x\rangle : x \in X\}, \tag{1.18}$$

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

D'après (1.18), remarquons que le calcul de x^{k+1} revient à minimiser une fonction convexe déduite de la fonction DC $f = g - h$, en approximant la composante concave $-h$ par une de ses minorantes affines au point x^k, i.e.

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

Et similairement, par dualité

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

1.3.4 Optimisation DC polyédrale

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

Soit le programme DC

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

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

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

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

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

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

1.3.5 Interprétations de DCA

La *première interprétation* de DCA est simple :

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

$$\inf\{\overline{f_k} := g(x) - h_k(x) : x \in \mathbb{R}^n\} \tag{1.19}$$

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

$$\inf\{h^\star(y) - (g^\star)_k(y) : y \in \mathbb{R}^n\} \tag{1.20}$$

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

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

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

qui coïncide en x^k avec f,

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

donc en déterminant l'itéré x^{k+1} comme le minimum du programme convexe (1.19), la décroissance de la suite des itérés est assurée,

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

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

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

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

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

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

Avant de terminer ce chapitre, il est aussi crucial de garder en mémoire cette *deuxième interprétation* de DCA :

Soit x^* une solution optimale du programme DC primal (P) et $y^* \in \partial h(x^*)$. En vertu du Theorem 1.1 y^* est une solution optimale du programme DC dual (D). Soit h_* la minorante affine de h définie par

$$h_*(x) := h(x^*) + \langle x - x^*, y^*\rangle,$$

et considérons le programme convexe suivant :

$$\alpha_* := \inf\{g(x) - h_*(x) : \mathbb{R}^n\} = \inf\{f(x) + h(x) - h_*(x) : x \in \mathbb{R}^n\}. \tag{1.21}$$

Puisque la fonction $f_*(x) = f(x)+h(x)-h_*(x)$ est une majorante convexe de f , $\alpha_* \geq \alpha$. Or $f_*(x^*) = f(x^*) = \alpha$, d'où $\alpha_* = \alpha$. D'autre part l'ensemble des solutions optimales de (1.21) est $\partial g^*(y^*)$ qui est bien contenu dans l'ensemble des solutions optimales $\mathcal{P}$ de (P) d'après Théoreme 1.1. Tenant compte de (1.15) et la décroissance de $\{g(x^k) - h(x^k)\}$, on comprend mieux ainsi le rôle joué par les programmes linéarisés (1.19) au même titre que (1.21) et l'explication partielle du fait que DCA, avec un point initial assez proche d'une solution optimale de (P), converge vers une solution optimale de (P).

Pour une étude complète de la programmation DC et DCA, se reporter aux [1]-[32] et [41]-[46] et références incluses. Le traitement d'un programme non convexe par une approche DC et DCA devrait comporter donc deux tâches : la recherche d'une décomposition DC adéquate et celle d'un bon point initial. Pour un programme DC donné, la question de décomposition DC optimale reste ouverte, en pratique on cherche des décompositions DC bien adaptées à la structure spécifiques du programme DC étudié pour lesquelles les suites $\{x^k\}$ et $\{y^k\}$ sont faciles à calculer, si possible explicites pour que les DCA correspondants soient moins coûteux en temps et par conséquent capables de supporter de très grandes dimensions.

Chapitre 2

Introduction aux algorithmes génétiques

Résumé *Nous présentons brièvement dans ce chapitre les éléments de base et les principales caractéristiques des algorithmes génétiques.*

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle. Les premiers travaux dans ce domaine ont été commencés dans les années 50, lorsque les biologistes ont simulé des structures biologiques sur ordinateur. Puis, en 1975, J. Holland ([118]), sur la base des travaux précédents, a développé les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation. Malheureusement, la puissance des ordinateurs de cette époque n'était pas assez pour les applications réelles de grande dimension. Il a fallu attendre jusqu'aux années 90, quand le monde d'informatique se développait avec une vitesse spectaculaire. La parution en 1989 de l'ouvrage de référence écrit par D.E. Goldberg ([116]) a fortement participé à l'essor des algorithmes génétiques. Cet ouvrage qui décrit l'utilisation des algorithmes génétiques pour la résolution de problèmes concrets a permis à la communauté scientifique de mieux connaître algorithmes génétiques et a marqué le début d'un nouvel intérêt pour cette technique d'optimisation. De nos jours, algorithmes génétiques sont devenus très populaire et leurs domaines d'applications sont de plus en plus large.

Les algorithmes génétiques s'inspirent des théories de l'évolution proposées par C. Darwin ([114]) :

– Dans chaque environnement, seules les espèces les mieux adaptées survivent au cours du temps, les autres étant condamnées à disparaître.
– Au sein de chaque espèce, le renouvellement des populations est essentiellement dû aux meilleurs individus de l'espèce.

Le schéma d'un algorithme génétique est simple. On commence par générer une population initiale d'individus de façon aléatoire. A chaque étape, on crée une nouvelle population en utilisant les opérateurs génétiques, à savoir, sélection, croisement et mutation. Le schéma du

fonctionnement d'un algorithme génétique est présenté dans la figure suivante :

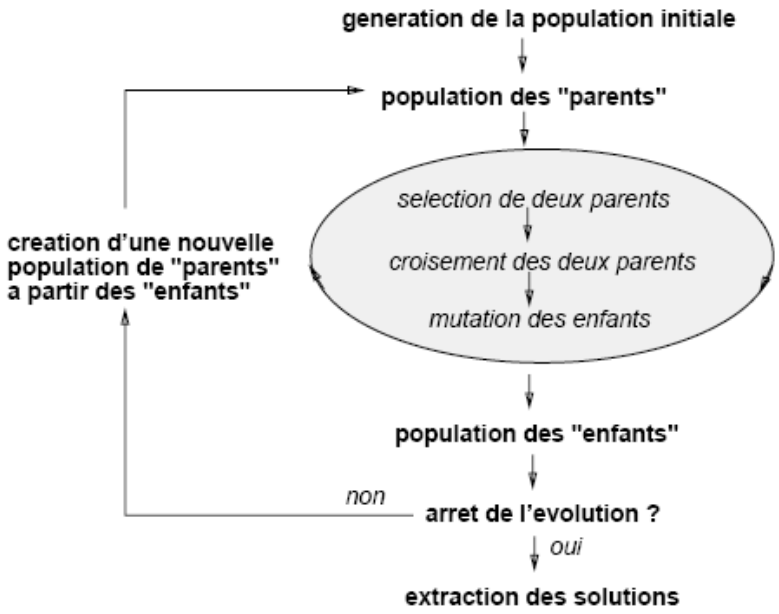

FIG. 2.1 – Schéma d'un algorithme génétique simple

La mise en œuvre d'un algorithme génétique nécessite les éléments suivants :

- **Une fonction d'adaptation (ou fonction fitness)** : la fonction que l'on utilise pour évaluer les individus dans l'espace de recherche.
- **Un codage** : comme nous avons mentionné, on associe chaque point de l'espace de recherche à une structure particulière, appelée génotype, qui caractérise chaque individu de la population.
- **Une population initiale** : la population à partir de laquelle l'algorithme évolue.
- **Un processus de sélection** : la sélection de deux parents selon un critère d'adaptation à l'environnement pour favoriser la reproduction des "bons" individus.
- **Opérateurs génétiques** : croisement et mutation.

Dans la suite, nous allons détailler ces éléments fondamentaux.

2.1 Fonction d'adaptation

Pour évaluer un point dans l'espace de recherche, on utilise une fonction d'évaluation. L'évaluation d'un individu ne dépend pas de celle des autres individus et le résultat fournit par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu pour la création de la nouvelle population à partir de la population courante.

Avec les algorithmes génétiques, aucune hypothèse de régularité n'est faite sur la fonction d'adaptation. Elle peut ne pas être dérivable et peut même être discontinue. Si le problème est sans contraintes, une façon simple de définir la fonction d'adaptation consiste à prendre la fonction objectif elle-même lorsqu'il s'agit d'un problème de maximisation ou son inverse pour un problème de minimisation.

Dans le cas qu'il s'agit des problèmes avec contraintes, on distingue deux cas :
- S'il est possible d'assurer que tous les individus respectent les contraintes, la fonction d'adaptation coïncide avec la fonction objectif ([119]). Logiquement, on fait en sorte que la population initiale et l'application des opérateurs d'évolution ne fournissent que des individus vérifiant les contraintes. Cette méthode est plus efficace car elle réduit directement l'espace de recherche.
- Dans le cas contraire, la fonction d'adaptation doit prendre en compte la pénalisation des contraintes des individus ne respectant pas tout ou partie des contraintes. Cette méthode a l'avantage de conserver, dans les premières générations, tout au moins, des individus situés dans le domaine non admissible. En effet, ces derniers peuvent malgré tout générer des individus de bonne qualité.

Dans le premier algorithme génétique de Holland, la fonction fitness est définit par : $\frac{f_i}{\overline{f}}$ où f_i est la fonction d'évaluation de l'individu i et $\overline{f}$ est la fonction moyenne des évaluations de toutes les chaînes de bits de la population. Cependant, en général, on ne fait pas la différence entre fonction fitness et fonction d'évaluation.

2.2 Codage

Le codage de chaque individu est une phase essentielle dans algorithmes génétiques. Il doit être fortement adapté au problème à résoudre. Dans cette étape, on associe à chacun des points de l'espace de recherche une structure de données qui synthétise toute l'information liée à ces derniers en passant par la phase de modélisation mathématique.



Fig. 2.2 – De l'espace réel au codage

2.2.1 Codage binaire

Historiquement, le codage utilisé par les algorithmes génétiques était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace de recherche. En effet, lors des premiers travaux de J. Holland, les théories ont

été élaborées en se basant sur ce type de codage. Le codage binaire permet de créer des opérateurs de croisement et de mutation simples.

Codage d'un entier :

Naturellement, on code un entier x par une chaîne $\mathcal{A} = \{a_1, a_2, ...a_l\}$ de taille l (l = le nombre de bits nécessaires pour coder ce nombre) telle que :

$$x = \sum_{i=1}^{l} a_i 2^{l-i}.$$

Comment déterminer la taille l pour coder des entiers avec borne fixe : x$\in [0, x_{\max}]$? C'est le plus petit entier l tel que $2^l > x_{\max}$.

Une chaîne binaire de taille l peut coder des entiers de 0 à $2^l - 1$. Pour coder des entiers x dans une intervalle $[x_{\min}, x_{\max}]$ par des chaînes binaires de taille l *fixé auparavant*, on va projeter x de $[x_{\min}, x_{\max}]$ vers $\left[0, 2^l - 1\right]$ puis utiliser un codage à caractères multiples.

Codage d'un réel :

La taille du chromosome codé d'un réel x dépend de la précision souhaitée. Connaissant le domaine des variables et fixant une précision, on peut calculer la taille du chromosome codé cette variable. Supposons que les variables x appartiennent à $D := [x_{\min}, x_{\max}]$. Notons $l_D := x_{\max} - x_{\min}$ la longueur de D et *prec* la précision (nombre de chiffres après la virgule). On doit diviser D en $n_i := l_D * 10^{prec}$ petites intervalles égales afin de respecter la précision. Donc la taille du chromosome sera l, le plus petit entier tel que $2^l > n_i$.

Exemple : $D = [-1, 2] \Longrightarrow l_D = 3$. Si $prec = 6$ alors $n_i = 3000000$, par suite $l = 22$ car $2^{21} = 2097152 < 3000000 < 2^{22} = 4194304$.

Pour coder un réel $x \in [x_{\min}, x_{\max}]$ on divise l'intervalle D :=$[x_{\min}, x_{\max}]$ en $n_i := l_D * 10^{prec}$ sous-intervalles qui représenteront chacun un chromosome. Chaque chromosome sera codé en l bits dont la valeur binaire correspondante $g \in [0, n_i]$ est

$$g = \frac{x - x_{\min}}{x_{\max} - x_{\min}} * n_i = (x - x_{\min}) * 10^{prec},$$

et le décodage (le nombre réel correspondant au chromosome) sera

$$x = x_{\min} + (x_{\max} - x_{\min}) * \frac{g}{n_i} = x_{\min} + g * 10^{prec}.$$

Exemple : pour coder des réels dans $[0, 4]$ avec une précision $prec = 3$ on doit diviser l'intervalle $[0, 4]$ en $n_i = 4 * 10^3 = 4000$ sous-intervalles. La taille de la chaîne binaire pour coder x est $l = 12$ car $2048 = 2^{11} < 4000 < 2^{12} = 4096$. Le codage de x est celui de l'entier g correspondant où $g = x * 10^3$.

Ainsi avec $x = 3.256$ on a $g = 3256$ et le codage de x est celui de g qui est $\{110010111000\}$.

Les inconvénients du codage binaire :

Il existe néanmoins des côtés négatifs à ce type de codage binaire qui font que d'autres existent.

- Ce codage est souvent *peu naturel* par rapport à un problème donné, par exemple, l'évolution des poids d'arcs dans un graphe est difficile à coder sous la forme d'une chaîne de bits.
- Deux individus voisins en terme de la distance de Hamming *ne codent pas nécessairement des éléments proches* dans l'espace de recherche, ce qui est une problématique si on utilise des opérateurs de mutation. Par exemple, les nombres binaires 100000 et 000000 sont proches en terme de distance de Hamming mais représentent des valeurs très éloignées (32 et 0).
- Pour des problèmes d'optimisation dans des espaces de grande dimension, particulièrement quand on code des réels, le codage binaire peut *rapidement devenir mauvais*. Par exemple, avec 100 variables appartenant au domaine [-500 ;500] et dont une précision de 6 chiffres après la virgule est requise, la taille du chromosome est 3000. Cela, en retour, génère un espace de recherche de taille 10^{1000}.
- D'autre part, l'ordre des variables a une importance dans la structure du chromosome binaire, alors qu'il n'en a pas forcément dans la structure du problème.

2.2.2 Codage Gray

Ce codage permet de changer un seul bit à la fois quand un nombre est augmenté une unité. Pour passer une ligne à la suivante on inverse le bit le plus possible à droite conduisant à un nombre nouveau.

Exemple :

	Codage Gray				Codage binaire			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1

2.2.3 Codage réel

Ce codage s'utilise naturellement pour les nombres réels. Chaque individu n'est alors plus qu'un chiffre à valeurs réelles dans l'espace des valeurs : 1 gène = 1 variable X_i, 1 allèle = 1 valeur du domaine de définition de X_i. La structure du problème est conservée dans le

codage.

Exemple : Chromosome A : 1.2324 5.3243 0.4556 2.3293 2.4545.

2.2.4 Codage à caractères multiples (codage multiparamétré)

Une autre manière de coder les chromosomes d'un algorithme génétique est donc le codage à l'aide de caractères multiples (par opposition aux bits). Pour construire un codage à caractères multiples on peut tout simplement concaténer autant de codage d'un seul paramètre qu'il est nécessaire.

Exemple : 1994 = 0001 1001 1001 0100

1*1000 + 9*100 + 9*10 + 4*1

2.2.5 Codage sous forme d'arbre

Ce codage utilise une structure arborescente avec une racine de laquelle peuvent être issus un ou plusieurs fils. Un de leurs avantages est qu'ils peuvent être utilisés dans le cas de problèmes où les solutions n'ont pas une taille finie. En principe, des arbres de taille quelconque peuvent être formés par le biais de crossing-over et de mutations.

Le problème de ce type de codage est que les arbres résultants sont souvent difficiles à analyser et que l'on peut se retrouver avec des arbres "solutions" dont la taille sera importante alors qu'il existe des solutions plus simples et plus structurées. De plus, ce type de codage ne fait que commencer et les performances de ce type de codage par rapport à des codages en chaînes n'ont pas encore été comparées ou très peu.

Exemple : pour le Problème du Voyageur de Commerce, un chromosome représente un élément de l'espace de recherche, c'est-à-dire un chemin possible pour le commis voyageur. Une population est donc un ensemble de chemin. Chaque ville est représentée par un point de l'espace physique et porte un numéro d'identité différent. Un chemin étant une suite de villes, un chromosome est une suite de nombres.

2.3 Population initiale

La population initiale joue un rôle important dans algorithmes génétiques dans la mesure où elle a une influence sur la vitesse de convergence de l'algorithme et la qualité de solution. Deux questions majeures sont la taille de la population et la méthode pour générer une population. Dans tous les cas, une "bonne" population dépend fortement de la nature du problème.

Si l'on n'a aucune connaissance de la position de l'optimum dans l'espace de recherche, on génère aléatoirement les individus, bien évidemment, en assurant que les individus respectent

les contraintes du problème. Par contre, si l'on dispose d'informations a priori sur le problème indiquant un sous-domaine où l'on est sûr de trouver l'optimum, il est donc naturel de générer les individus dans ce sous-domaine, afin d'accélérer la convergence.

Le choix de la taille de la population (N) est également un facteur pour la vitesse de convergence des algorithmes génétiques. Si l'on choisit une population avec la taille trop petite, algorithme évoluera probablement vers un optimum local peu intéressant. Par contre, si la taille est trop importante, la probabilité de convergence est grande mais le temps de convergence sera excessif dû au nombre d'évaluations. La taille de la population doit être choisie de façon à réaliser un bon compromis entre le temps de calcul et la qualité du résultat.

Théoriquement, l'on considère le critère suivant pour déterminer la taille d'une population initiale : *tous les points dans l'espace de recherche doit être retrouvable à partir de la population initiale par croisement seulement*. Pour un codage binaire dans une population générée par un échantillon aléatoire avec remplacement on peut démontrer que ce critère est vérifié avec ([113]) :

$$N \simeq \lceil 1 + \log(-l/\ln P_2^*)/\log 2 \rceil$$

où P_2^* est la probabilité pour qu'au moins un allèle se présente à chaque locus dans cette population.

En pratique, pour un codage binaire on prend $N = 30$.

2.4 Opérateurs

2.4.1 Sélection

La partie darwinienne de l'algorithme génétique comprend les deux étapes de sélection et de remplacement. On distingue deux catégories de procédures de sélection ou de remplacement (par abus de langage, nous appellerons sélection les deux types de procédures) : les procédures déterministes et les procédures stochastiques.

Sélections déterministes - élitisme

A la création d'une nouvelle population, il y a des grandes chances que les meilleurs chromosomes soient perdus après les opérations de croisement et de mutation. Pour éviter cela, pour créer la nouvelle génération, on garde les meilleurs individus (au sens de la fonction d'évaluation)de l'ancienne population. Les individus les moins performants sont totalement éliminés de la population, et le meilleur individu est toujours sélectionné - on parle alors d'*élitisme*. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleures solutions.

Sélections stochastiques

Il s'agit toujours de favoriser les meilleurs individus, mais ici de manière stochastique, ce qui laisse une chance aux individus moins performants. Comme dans la nature, il ne faut

pas confondre sélection et élitisme : ce n'est pas parce qu'un individu est bon qu'il survivra nécessairement et de même ce n'est pas parce qu'il est mauvais qu'il doit disparaître. En effet même les individus les moins bien adaptés peuvent, par croisement ou mutation, permettre de générer une descendance intéressante du point de vue de la recherche d'un optimum.

2.4.1.1 Sélection par roulette

Cette sélection consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa fonction d'évaluation. Par exemple, la probabilité l_i d'être choisi d'individu i est calculée par rapport sa valeur d'évaluation f_i :

$$l_i = \frac{|f_i|}{\sum |f|}.$$

On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on regarde quel est le segment sélectionné, et on reproduit l'individu correspondant. Avec cette technique, les bons individus seront plus souvent sélectionnés que les mauvais, mais néanmoins les mauvais individus peuvent avec cette méthode être sélectionnés plusieurs fois.

Néanmoins, sur des populations de petite taille, il est difficile d'obtenir exactement l'espérance mathématique de sélection à cause du faible nombre de tirages.

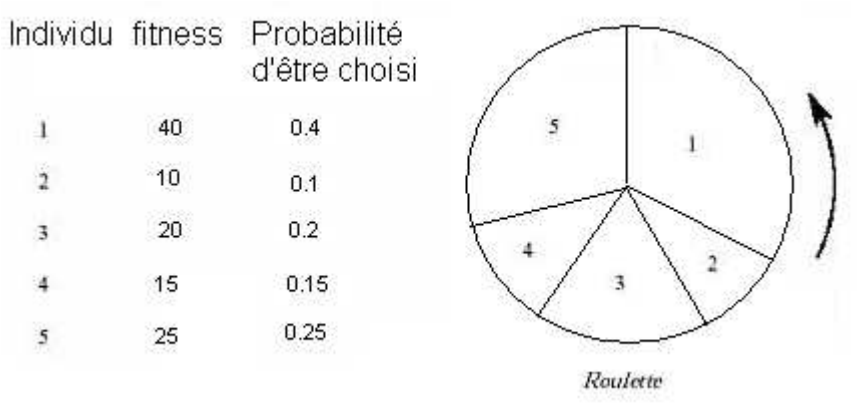

FIG. 2.3 – Exemple de la sélection par roulette

L'inconvénient majeur est qu'elle peut engendrer une perte de diversité par la domination d'un individu. Un autre inconvénient est sa faible performance vers la fin quand l'ensemble des individus se ressemblent.

2.4.1.2 Sélection par rang

La sélection par rang est la même que par roulette, mais les probabilités sont en relation avec le rang plutôt qu'avec la valeur d'évaluation. La sélection par rang trie d'abord la population

par la valeur d'évaluation. Ensuite, chaque individu est associé à un rang en fonction de sa position. Le plus mauvais chromosome a le rang 1, le meilleur individu a le rang N (N est le nombre d'individu de la population).

Avec cette méthode de sélection, tous les chromosomes ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution. Ceci est dû au fait que les meilleurs individus ne diffèrent pas énormément des mauvais individus.

Individu	1	2	3	4	5	6	Total
Rang	6	5	1	4	3	2	21
Probabilités d'être choisi	29%	24%	5%	19%	14%	9%	100%

FIG. 2.4 – Exemple de la sélection par roulette

2.4.1.3 Sélection par tournoi

La sélection par tournoi consiste à sélectionner n individus au hasard et à prendre le meilleur parmi ces n individus. On organise autant de tournois qu'il y a d'individus à choisir. Le nombre n permet de donner plus ou moins de chance aux individus peu adaptés. Avec un nombre élevé de participants, un individu faible sera presque toujours sûr de perdre, donc en général, on utilise $n = 2$.

2.4.1.4 Sélection uniforme

La sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité $\frac{1}{N}$ d'être sélectionné, (N est le nombre d'individu de la population).

2.4.2 Croisement

Le croisement a pour but de créer les nouveaux individus à partir du patrimoine génétique de plusieurs parents. Classiquement, le croisement est réalisé avec deux parents et génère deux enfants. Il consiste à échanger les gènes des parents afin de donner des enfants qui portent des propriétés combinées. Bien qu'il soit aléatoire, cet échange d'informations offre aux algorithmes génétiques une part de leur puissance : quelque fois, les "bons" gènes d'un parent viennent remplacer les "mauvais" gènes d'un autre et créent des fils mieux adaptés que ses parents.

On définit un taux de croisement en général entre 0.5 et 0.9. Plus le taux de croisement est élevé, plus il y aura de nouvelles structures qui apparaissent dans la population. Ce paramètre doit être délicatement réglé car s'il est :

– Trop élevé : les "bonnes" structures risquent d'être cassées trop vite par rapport à l'amélioration que peut apporter la sélection.
– Trop faible : la recherche risque de stagner à cause du faible taux d'exploration.

Il existe plusieurs manières d'effectuer un croisement dans la littérature, chacun correspond à un type de codage du problème. Il est difficile de dire quel type de croisement sera efficace pour un codage sans tester à priori.

Deux méthodes classiques, *croisement en un point* et *croisement en deux points* sont illustrés dans la figure ci-dessus :

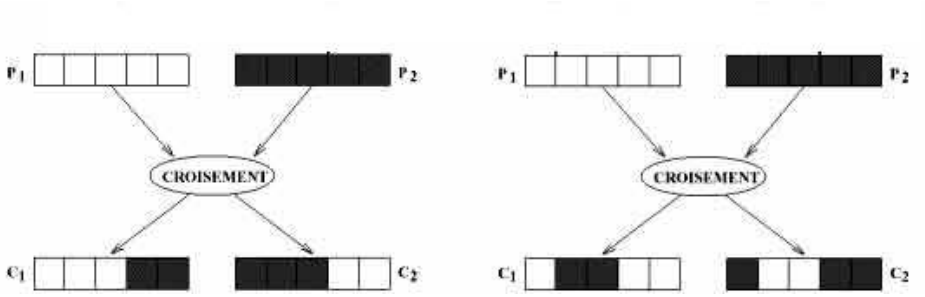

FIG. 2.5 – *Slicing crossover classique*(gauche) et *2-point slicing crossover*(droite)

Croisement dans le codage réel

– **Croisement uniforme** : L'opération de croisement simple tel que décrit dans le cas binaire ne peut s'effectuer ici dans le cas de recherche d'un point unique. Toutefois, pour une recherche de plus grande dimension, nous pouvons utiliser de façon analogique cet opérateur. Ainsi, soient Y = (y_1 ;y_2 ;y_3) et X = (x_1 ;x_2 ;x_3) deux membres (vecteur de dimension trois) de la population initiale. Nous recherchons donc trois points dans un espace de recherche de dimension trois. On génère un nombre aléatoire r à partir d'une distribution uniforme sur l'ensemble $\{1; 2; 3\}$. Les deux nouveaux individus, X et Y, sont créés selon la règle suivante :

$$\widetilde{x} = \begin{cases} x_i & \text{si } i < r \\ y_i & \text{sinon} \end{cases}, \quad \widetilde{y} = \begin{cases} y_i & \text{si } i < r \\ x_i & \text{sinon} \end{cases}.$$

– **Croisement arithmétique** : ce croisement est pour l'avantage d'être valable même pour une recherche de dimension un. On effectue une simple combinaison linéaire entre les parents. Soit, après avoir généré un chiffre aléatoire, α =U(0 ;1), les nouveaux individus sont :

$$X = \alpha X + (1-\alpha)Y,\ Y = (1-\alpha)X + \alpha Y.$$

– **Croisement heuristique** : cet opérateur effectue une extrapolation linéaire des deux individus. Un nouvel individu, X, est créé selon le processus suivant (sous l'hypothèse que $X > Y$ en terme de fitness, sinon nous inversons X et Y dans les équations) :

$$\widetilde{X} = X + r(X - Y), \widetilde{Y} = X.$$

2.4.3 Mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours de l'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace de recherche. Certaines implémentation n'utilisent que la mutation pour produire la nouvelle population ([115]).

En règle générale, on tire aléatoirement un ou plusieurs gènes de l'individu puis on les modifie de manière aléatoire en assurant tout de même que la résultante est une solution admissible au problème. La mutation est souvent délicate à régler en terme d'évaluation du nouvel individu :
– Insignifiante : on risque de s'enfermer dans la recherche d'un optimum local.
– Trop importante : on risque d'empêcher toute convergence de l'algorithme.
De la même manière que pour les croisements, on définit un taux de mutation qui est généralement compris entre 0.001 et 0.01.

Le schéma suivant montre un exemple simple de la mutation du codage binaire :

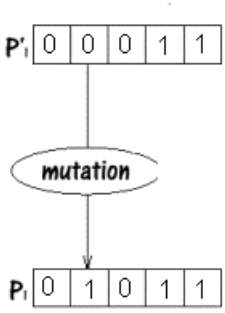

FIG. 2.6 – La mutation

Mutation dans le codage réel :
– La mutation uniforme est identique à celle du codage binaire. Chaque variable $x_i \in X$ est changée selon une certaine probabilité en un nombre aléatoire tiré dans une distribution uniforme sur l'intervalle $[b_1^i; b_2^i]$, avec $b_1^i; b_2^i$ les bornes inférieures et supérieures pour x_i.
– La mutation non uniforme revient à changer la variable x_i en un nombre tiré dans une distribution non uniforme. La nouvelle variable $\widetilde{x}_i$ est calculée par :

$$\widetilde{x}_i = \begin{cases} x_i + (b_2^i - x_i) f(G) & \text{si } \alpha < 0.5 \\ x_i - (x_i + b_1^i) f(G) & \text{si } \alpha \geq 0.5 \end{cases} ,$$

où $f(G) = \left(\widetilde{\alpha}(1 - \frac{G}{G\max})\right)^b$,
$\alpha, \widetilde{\alpha}$ = nombre aléatoire $\in (0, 1)$,
G = la génération courante,
G_{max}= le nombre maximum de générations (i.e. de création de nouvelle population),
b = un paramètre déterminant le degré de non uniformité.

2.5 Principales caractéristiques de l'algorithme génétique

Les algorithmes génétiques se diffèrent aux autres algorithmes d'optimisation en cinq points fondamentaux suivants :

- Les algorithmes génétiques ne sont pas déterministes, ils utilisent des règles de transition probabilistes.
- Dans algorithmes génétiques, on associe des éléments de l'espace de recherche à un codage particulier et ensuite travaille avec ce codage mais non pas avec les éléments eux même.
- Les algorithmes génétiques travaillent sur une population de points, au lieu d'un point unique.
- Un des très grand avantage des algorithmes génétiques est qu'ils n'imposent aucune régularité sur la fonction étudiée (continuité, dérivabilité, convexité,...) : ils n'utilisent que les valeurs de la fonction étudiée et non pas sa dérivée, ou une autre connaissance auxiliaire.

Néanmoins, il existe un certain nombre de limitations des algorithmes génétiques :

- Ils sont moins efficaces qu'un algorithme déterministe (lorsqu'il en existe un).
- Algorithmes génétiques nécessitent un réglage délicat de nombreux paramètres (probabilités de croisement et de mutation notamment), ainsi que le codage des chromosomes qui peut faire varier radicalement la vitesse de convergence.
- Afin de garantir la robustesse des algorithmes évolutifs, le calcul d'un très grand nombre de fitness (parfois de l'ordre de plusieurs centaines de milliers) est généralement nécessaire avant l'obtention d'une bonne solution. Lorsqu'on travaille en grande dimension sur des fonctions complexe, ce nombre de calcul important engendre forcement un coût de calcul important.
- Lorsqu'une population évolue, il se peut que certains individus qui à un instant occupent une place importante au sein de cette population deviennent majoritaires. À ce moment, la population converge vers cet individu et s'écarte ainsi d'individus plus intéressants mais trop éloignés. Pour éviter ce problème, il existe différentes méthodes comme l'ajout de quelques individus générés aléatoirement à chaque génération, des méthodes de sélection différentes de la méthode classique ...

La tendance actuelle est de combiner les algorithmes génétiques avec d'autres méthodes déterministes. Dans le cadre de cette thèse nous utilisons les algorithmes génétiques (AG) aux chapitres sept et huit pour la résolution de deux problèmes d'optimisation combinatoire extrêmement difficile (la construction des fonctions booléennes équilibrées de haut degré de non-linéarité en cryptographie). Nous y développons quelques variantes de hybridation DCA-AG.

Deuxième partie

Modélisation DC et DCA pour le clustering flou et le clustering hiérarchique

Chapitre 3

Introduction à la classification

3.1 Fouille de données et Classification

De manière générale, on peut définir la fouille de données comme l'ensemble des techniques et des méthodes pour l'extraction automatique d'informations ou de connaissances nouvelles ou cachées dans des entrepôts de données. Elle est composée de trois branches principales : sélection des attributs, classification et règles d'association. De nos jours, on estime que le volume de données double tous les six mois. Il est donc très important de savoir extraire les informations utiles afin de pouvoir les utiliser efficacement. Cela confirme le rôle primordial de la fouille de données dans le monde industriel ainsi que dans la recherche. Les applications de ce domaine incluent la prise de décision, l'analyse de marchés, l'analyse du panier de la ménagère, l'aide au diagnostique, et l'exploration scientifique ...

Au cours de ces dix dernières années la fouille de données s'est considérablement développée, tant du point de vue des méthodes que de la formalisation et des outils. La classification constitue une très bonne illustration. Ayant pour but l'obtention d'une représentation simplifiée des données initiales, elle est devenue un outil incontournable de la fouille de données. La classification est l'organisation d'un ensemble en classes homogènes ou classes naturelles. Il s'agit d'une démarche très courante qui permet de mieux comprendre l'ensemble analysé. Les données peuvent être des données statistiques collectées en vue d'une étude particulière, appartenir à des bases de données aujourd'hui de plus en plus volumineuses et donc riches en informations, ou encore provenir d'images digitalisées en vue d'une segmentation automatique. Historiquement, les premières recherches théoriques sur la classification étaient pour but de spécifier des classes ou des espèces animales, aux besoins des biologistes ([171]). De nos jours, les applications de la classification se sont multipliées. Elles s'appliquent à un grand nombre d'activités humaines et conviennent en particulier au problème de la prise de décision automatisée. Il s'agit, par exemple, d'établir un diagnostic médical, de donner une réponse à la demande de prêt bancaire d'un client, ... En économie, la classification peut aider les analystes à découvrir des groupes distincts dans leur base clientèle, et à caractériser ces groupes de clients, en se basant sur des habitudes de consommations. En biologie, on

peut l'utiliser pour dériver des taxinomies de plantes et d'animaux, pour catégoriser des gènes avec une ou plusieurs fonctionnalités similaires, pour mieux comprendre les structures propres aux populations. La classification peut tout aussi bien aider dans l'identification des zones de paysage similaire, utilisée dans l'observation de la terre, et dans l'identification de groupes de détenteurs de police d'assurance automobile ayant un coût moyen d'indemnisation élevé, ou bien dans la reconnaissance de groupes d'habitation dans une ville, selon le type, la valeur et la localisation géographique. Il est possible également de classifier des documents sur le Web, pour obtenir de l'information utile ... Aujourd'hui, avec les nouvelles technologies de l'information et de la communication, nous sommes amenés à traiter des données très complexes par leur nature, leur taille, leur provenance, leur diversité (texte, images, séquence vidéo,... et tout support combinant ces différents type de données). La fonction de classification dans un processus de fouille de données sera de découvrir la distribution des données, d'observer les caractéristiques de chaque groupe et de se focaliser pourquoi pas sur un ou plusieurs ensembles particuliers d'objets pour de prochaines analyses. Parallèlement, elle peut servir comme une étape de préparation de données, appelé aussi pre-processing, pour d'autres méthodes. Ainsi, ce type d'analyse est devenu un domaine hautement actif en fouille de données.

3.2 Notions préliminaires

3.2.1 Objets et Classes

Le but de la classification est d'organiser un ensemble d'objets en classes homogènes. Les premières questions se posent alors de savoir représenter des objets et définir ce qu'une classe.

Dans le cadre de *fouille de données numérique* auquel se situent nos travaux, un objet de l'ensemble E est modélisé par un vecteur x décrit lui même par p variables quantitatives qu'on appelle également les *attributs*.

Du point de vue mathématique, une classe est un sous-ensemble de l'ensemble E des objets à classer. Il existe deux approches différentes pour décrire une classe :
- L'approche générative : une classe est décrite par les propriétés caractéristiques des objets qui la composent ;
- L'approche discriminative : une classe est décrite par sa frontière avec ses voisines.

La structure la plus courante de classe est celle de la "partition".

Partition :

Définition 3.1 *L'ensemble $P = \{C_1, ..., C_k\}$ est une partition de l'ensemble $E = \{x_1, \ldots, x_n\}$ en c classes si et seulement si :*

$$C_i \neq \emptyset \text{ pour } i = 1, ..., c \tag{3.1}$$

$$\bigcup_{i=1}^{k} C_i = E \tag{3.2}$$

$$C_i \bigcap C_l = \emptyset \text{ pour tout } i \neq l. \tag{3.3}$$

Partition floue : en relaxant la contrainte (3.3), on permet à un objet d'appartenir à plusieur classes, on parle alors de la partition floue :

Définition 3.2 *Une partition floue de l'ensemble E, notée $P = \{C_1, ..., C_c\}$ est définie par la donnée de c fonctions*

$$u_i : E \to [0, 1], \quad i = 1, \dots c$$

$u_i(x_k)$ représente alors le degré d'appartenance de l'objet x_k à la classe C_i.

3.2.2 Classification supervisée et classification non supervisée

Etant pour but de classer chaque objet à une classe, la classification revient à définir une règle de décision qui associe un objet à une classe. En général, on construit la règle de décision θ à partir d'une phase d'apprentissage. On distingue deux types d'apprentissage :
- L'apprentissage supervisé : il s'agit de faire émerger d'un ensemble de données d'entraînement pré-classifées, les caractéristiques nécessaires et suffisantes pour permettre de classer correctement une nouvelle donnée. Les classes sont connues à l'avance, cette connaissance "supervise" la construction de la règle de classification.
- L'apprentissage non-supervisé : comme son nom indique, il consiste à apprendre sans superviseur. A partir d'une population, il s'agit d'extraire des classes ou groupes d'individus présentant des caractéristiques communes, la définition des classes n'étant pas donnés a priori. On utilise souvent le terme *regroupement* ou son équivalent anglais *clustering*, pour désigner la classification non-supervisée.

3.2.3 Processus de classification

Le processus de classification est divisé en trois étapes principales : préparation de données, application de l'algorithme de classification et exploitation des résultats. Les questions que l'on doit se poser lors d'une application d'un processus de classification sont :
- Quelles sont les variables et comment sélectionner les variables ?
- Comment transcrire la différence entre les objets ?
- Quel est l'algorithme le plus adapté au problème ?

3.2.4 Variables et sélection de variables

Un objets peut être présenté par des variables de différentes natures :
- Variables quantitatives : continues (e.g. le poids d'une personne), discrète (e.g. le nombre de personnes dans un groupe).

– Variables qualitatives : non-ordonnées (e.g. la couleur des voitures), ordonnées (e.g. la mention : "moyen", "bien", "excellent", . . .).
– Variables structurées : par exemple la forme d'un objet (carré, rond, rectangle,. . .)

La nature des variables influe forcement sur la définition de similarité des objets et ce choix est très important. En *fouille de données numériques* on travaille sur les variables *quantitatives*.

Dans les applications réelles, le nombre de variables représentés un objet est souvent grand. Ce nombre important de variable entraîne un grand coût de calcul (temps CPU, la taille de mémoire utilisée, ...). La question est donc de pouvoir choisir parmi les variables celles qui sont pertinentes et d'éliminer celles qui sont redondantes. Cette question de sélection de variables a été largement étudiée pour la classification supervisée mais reste encore ouverte pour la classification non-supervisée.

Pour la classification supervisée, on dispose deux types de méthodes pour la sélection de variables : les méthodes *wrapper* et les méthodes *filter*. Les méthodes *wrapper* consistent à utiliser une manière explicite le classifieur pour choisir un sous-ensemble de variables ([173]), alors que les méthodes *filter* tente d'enlever les variables non pertinentes avant la phase d'apprentissage.

Pour la classification non-supervisée, il est difficile d'évaluer la pertinence d'une variable en raison de l'absence des étiquettes de la classe. On contourne cette difficulté en utilisant un processus itératif : appliquer une première étape de classification pour obtenir les classes "pré-difinies" puis utiliser une procédure de sélection de variables en classification supervisée, et réitérer le processus ([174]). Récemment les approches utilisant des méthodes *filter* ont été également étudiées ([177],[172]).

3.2.5 Distance et similarité

La plupart des algorithmes de classification utilisent une mesure de proximité entre les objets. Cette notion de *proximité* est définie à l'aide d'une mesure de similarité, dissimilarité ou encore par une distance. Le choix de cette mesure est déterminant pour la suite du processus. Par exemple, pour les données spatiales, il est naturelle d'utiliser les distances traditionnelles comme distance Euclidienne, distance de Manhattan, ... Par contre, dans la classification de données textuelles, il est nécessaire de faire appel à des indices ou mesures de similarités basées sur des cooccurrences (coefficient de Dice [180], indice d'équivalence [179], etc.)

3.3 Clustering (Classification non supervisée)

Nous nous limitons dans ces travaux de thèse à la classification non supervisée et utilisons désormais le terme clustering pour la désigner.

3.3.1 Clustering dur et clustering flou

Comme nous avons vu dans la Définition (3.1) de partition, la contrainte (3.3) impose qu'il y a aucun recouvrement entre les classes. Nous parlons alors de *clustering dur* (hard clustering en anglais), dans lequel un objet est classé dans une et une seule classe. Le résultat prend alors la forme de partitions ou hiérarchiques strictes. L'avantage de clustering dur est de fournir une solution simple et sans ambiguïté et facilement exploitable.

En relaxant la contrainte (3.3), nous avons une partition floue et nous nous retrouvons dans le clustering flou (fuzzy clustering en anglais), une autre philosophie de la classification, celle qui permet à un objet d'appartenir à plus qu'une classe à la fois. Le clustering flou est le fruit de l'intersection de deux domaines de recherche : la classification est la théorie des ensembles floue. L'avantage du clustering flou est de tenir compte les incertitudes et ambiguïtés de données pouvant intervenir à la découverte de la règle de décision. Il est très efficace dans certains domaines tels que la classification de textes, la segmentation des images et en particulier la bio-informatique.

3.3.2 Clustering hiérarchique et clustering par partitionnement

On peut distinguer deux catégories de clustering : le clustering par partitionnement simple et le clustering hiérarchique. La première correspond à une partition des objets en un certain nombre de classes donné. Deux critères doivent être satisfaits :

- Chaque classe (ou cluster) doit contenir au moins un objet, et donc les classes vides ne sont pas tolérées.
- Chaque objet doit appartenir à au moins une classe (une seule classe dans le cas de clustering dur).

Le *clustering hiérarchique* consiste à créer une décomposition hiérarchique d'un tableau de données.

Le clustering hiérarchique est pour l'avantage de donner une visualisation claire de l'organisation des données. Les algorithmes de partitionnement, au contraire, proposent une partition de l'espace plutôt qu'uns structure organisationnelle, ils consistent alors de choisir une classification qui optimise un critère.

3.4 Principales approches classiques de clustering

Il existe plusieurs méthodes de clustering, chacun son domaine de prédilection, ses avantages, ses points faibles. Pour les données numériques il y a deux principales familles d'approches de clustering

– Approches statistiques basée sur des modèles probabilistes qui formalisent l'idée de classe.

– Approches d'optimisation (ou de la programmation mathématique) qui considère le clustering comme un problème d'optimisation.

L'optimisation joue un rôle fondamental en clustering (comme dans tous les domaines de l'apprentissage) : même dans les approches statistiques on passe souvent par une phase d'optimisation d'un critère et doit appeler ainsi un algorithme d'optimisation.

Par l'abus de langage nous utilisons dans la suite le terme *clustering* pour désigner le clustering par partitionnement.

3.4.1 Approches statistiques : clustering via le modèle de mélange

Dans cette famille d'approches statistiques, l'approche mélange basée sur un modèle de mélange probabiliste est incontestablement une contribution très utile à la classification. Elle est devenue ces dernières années une approche classique car elle offre une flexibilité considérable, et fournit des solutions au problème du nombre de classes et des données manquantes par exemple. L'idée de base de ces méthodes est de considérer que les observations (ou objets) $\{x_1, \ldots x_n\}$ ont été générées par un ensemble (ou mélange) de c distributions de probabilités. Le nombre c correspondant au nombre de clusters, il peut être connu a priori ou à définir en comparant les schémas issus de plusieurs valeurs de c. Des hypothèses sont faites sur la nature de ces distributions : l'hypothèse la plus courante est de considérer que les observations sont issues de lois normales, les densités de probabilités sont donc des gaussiennes. Il s'agit alors d'estimer les paramètres permettant de défnir chaque distribution afin de coller au mieux avec les observations. Pour estimer ces paramètres, R.C Rao ([181]) est le premier à utiliser la méthode du maximum de vraisemblance dans le cas de mélanges gaussiens. Dans cette approche pour la maximisation de la vraisemblance des données observées, l'algorithme populaire Expectation-Maximisation (EM) ([169]) est l'algorithme le plus utilisé, une itération de cet algorithme ne fait pas décroître la vraisemblance. Partant d'une solution initiale, l'algorithme EM alterne entre deux étapes baptisées respectivement E, pour Expectation, et M, pour Maximisation. A la convergence de EM, les probabilités a posteriori définies dans l'étape Estimation permettent de définir des classes floues ou dures par le principe du maximum a posteriori. Dans le contexte approche modèle de mélange, l'algorithme EM présente plusieurs avantages comparativement aux méthodes traditionnelles telles que Newton-Raphson ou gradient. Cependant, il présente deux inconvénients majeurs : l'étape d'initialisation qui conditionne fortement la qualité des estimations et le coût en temps de la convergence qui peut nécessiter de nombreuses itérations. Plusieurs versions modifiées de EM telles que CEM ([167]), SEM ([168]) ont été développées. Quelques travaux actuels s'attellent à accélérer les itérations, citons par exemple l'étude récente proposée dans [178].

3.4.2 Approches d'optimisation pour le clustering

Dans ces approches, le problème de clustering est formulé comme un problème d'optimisation dont la solution optimale permet d'obtenir explicitement les clusters. Cette solution optimale

correspond aux points représentant des clusters que nous appelons dans cette thèse des *centres* de clusters. Ces points peuvent être des *centroïds* ou des *médoïdes* (dans le cas où des variables descriptives sont de nature qualitative) ou encore des points particuliers parmi les objets à classer. Dans le cas des variables quantitatives (continues ou discrètes), le centroïde est défini, sur chaque composante, par la valeur moyenne des objets du cluster pour cette même composante. En ce sens, le centroïde correspond au centre de gravité du cluster. Le centroïde d'un cluster ne fait généralement pas partie des objets constituant ce cluster. Une fois que les centres de clusters sont déterminés, on affecte chaque point (objet) au cluster dont le centre est le plus proche.

La fonction objectif du problème d'optimisation représente un critère qui peut être la similarité intra-cluster ou la dissimilarité inter-clusters. Selon la distance choisie on obtient différents modèles d'optimisation. Citons, par exemple, les modèles qui utilisent la distance comme la norme L_1 et/ou le carré de la norme L_2. Si on représente chaque cluster par son centröid, le problème d'optimisation correspondant est sans contraintes. Dans le cas où chaque cluster est représenté par un point parmi ceux qui le constituent, on est en face un problème d'optimisation avec contraintes. Dans la plupart des cas le problème d'optimisation considéré est non convexe et non différentiable. Ci-après quelques formulations utilisant la distance comme les normes L_1 ou L_2.

- *La distance est la norme L_1 et chaque cluster est représenté par son centroïd* :
  En introduisant les variables artificielles $D^{i\ell}$, Mangasarian ([214]) a formulé ce problème (initialement sans contrainte) comme la minimisation d'une fonction concave sous les contraintes linéaires :

$$\begin{cases} \min \sum_{i=1}^{m} \min\limits_{\ell=1,\dots,k} e^T D^{i\ell} \\ \text{tel que :} \\ -D^{i\ell} \leq x^\ell - a^i \leq D^{i\ell} \\ D^{i\ell}, x^\ell \in \mathbb{R}^n,\ \ell = 1,\dots,k,\ i = 1,\dots,m \end{cases} \tag{3.4}$$

  où $e \in \mathbb{R}^n$ est le vecteur dont les composantes sont égales à 1.

- *La distance est le carrée de la norme L_2 et chaque cluster est représente par son centroïd* :

$$\min \quad \left\{ \sum_{i=1}^{m} \min_{\ell=1,\dots,k} \left\| x^\ell - a^i \right\|^2 : x^\ell \in \mathbb{R}^n, \ell = 1,\dots,k \right\}. \tag{3.5}$$

- *La distance est le carrée de la norme L_2 et chaque cluster est représenté par un point parmi ceux qui le constituent : le problème de k-median.*
  Une des formulations étudiées dans la littérature est la programmation en variables

zero-un suivante ([176]) :

$$(k-MP)\quad \begin{cases} \min \sum\limits_{x\neq y\in E} a(x,y)d(x,y) \\ \text{tel que :} \\ \sum\limits_{y\in E} a(x,y) \geq 1,\ \forall x \in E, \\ a(x,y) \leq m(y)\ \forall x,y \in N, \\ \sum\limits_{x\in E} m(x) \leq k, \\ m(x) \in \{0,1\}, \forall x \in E, \\ a(x,y) \in \{0,1\}, \forall x,y \in E, \end{cases} \tag{3.6}$$

où E désigne l'ensemble des objets données, $m(y)$ et $a(x,y)$ (pour $x,y \in E$) sont des variables zero-un définies par

$$m(y) = \begin{cases} 1 & \text{si } y \text{ est un centre,} \\ 0 & \text{sinon} \end{cases}$$

et

$$a(x,y) = \begin{cases} 1 & \text{si } x \text{ est un centre et y est affecté au cluster du centre x,} \\ 0 & \text{sinon.} \end{cases}$$

3.4.2.1 Clustering dur par l'algorithme k-moyennes

k-moyennes (k-means en anglais) ([175]) est la méthode heuristique la plus populaire en clustering dur. Le terme k-moyennes désigne le problème qui consiste à minimiser l'inertie intraclasse. Il est aussi utilisé pour désigner les algorithmes qui sont développés pour accomplir cette tâche.

Partant de c points choisis aléatoirement comme les centres de clusters, chaque itération de k-means est composée de deux étapes :
– Affecter chaque point au cluster dont le centre est le plus proche
– Une fois que tous les points sont partitionnés aux clusters, recalculer les centroïdes de clusters obtenus

On répète la procédure jusqu'à ce que les clusters ne changent plus ou un nombre d'itérations fixé auparavant est atteinte.

Cet algorithme est le plus utilisé dans divers domaines d'applications car il présente un rapport coût/efficacité avantageux. Néanmoins la qualité de la solution obtenue dépend beaucoup du point initial, et on peut tomber facilement à un minimum local du problème.

3.4.2.2 Clustering flou par l'algorithme c-moyennes floue

C-moyennes floue (Fuzzy C-Means anglais, ou plus brièvement FCM) est une version adaptée de k-means pour le clustering flou. Nous présenterons de manière plus détaillée cet algorithme dans le chapitre suivant.

3.4.3 Méthodes de clustering hiérarchique

Le principe des algorithmes hiérarchiques consiste à la construction d'un arbre de clusters (ou dendrogramme) tel que :
- La racine contient l'ensemble de tous les objets à classer.
- Chaque nœud de l'arbre représente un cluster et l'union des objets dans ses nœuds fils correspond aux objets dans le cluster.
- Une feuilles de l'arbre correspond à un objet considéré.

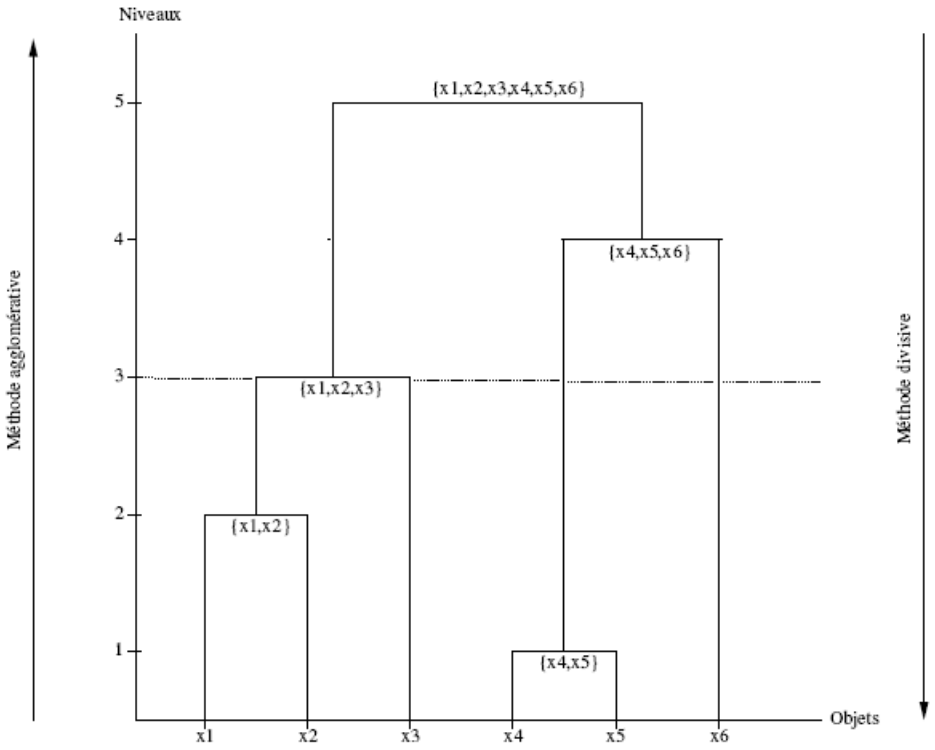

FIG. 3.1 – Exemple de dendogramme

A partir d'un dendogramme, il est facile d'obtenir une classification en donnant le niveau souhaité. Par exemple, sur le dendogramme dans la Figure 3.1, avec le niveau 3, la classifi-

cation est $\{x_1, x_2, x_3\}$, $\{x_4, x_5, \}$ et $\{x_6\}$.

Il existe dans la littérature très peu de modèles d'optimisation déterministe pour le clustering hiérarchique. Nous en parlerons plus tard dans le chapitre V.

3.4.4 Chois de l'algorithme de clustering

Face à un problème réel, le choix de l'algorithme est crucial. Le choix de l'algorithme doit tenir compte de plusieurs éléments, en particulier la nature des données et la taille de données.

La nature des données : la nature des variables influe forcement sur la définition de similarité des objets et comme nous avons vu précédemment, ce choix est très important.

La taille des données : le nombre d'objets à classer et le nombre de variables (attributs) qui représentent ces objets sont les premiers facteur de décision. Par exemple, face à un problème de petit taille de centaines objets voire quelque milliers d'objets, il est possible d'utiliser les méthodes complexes qui nécessite un temps de calcul important. Par contre, ces méthodes ne seront plus adaptées lorsque l'on a à traiter un problème de très grande taille (traitement d'images, classification de documents,...). La recherche des algorithmes performants étant capables de traiter des problèmes de très grande dimension est une question pertinente et d'actualité en fouille de données. L'utilisation de la programmation DC et DCA dans nos travaux est motivée par cette préoccupation.

3.5 La programmation DC et DCA pour clustering et nos contributions

Comme tous les domaines de l'apprentissage en général et ceux de fouille de données en particulier, le clustering constitue une mine de la programmation DC et DCA (voir, par exemple [18], [20], [53], [54], [56], [60], [65]). La difficulté majeure de clustering réside dans la non convexité du modèle d'optimisation associé d'une part, et la taille très grande de ce modèle d'autre part, vu la dimension et le volume de masse de données considérée. Avec les différentes mesures de distance, tous les modèles d'optimisation en clustering sont de la forme DC ou peuvent être transformés en une programmation DC par les techniques de reformulation. Dès lors DCA peut être développé pour la résolution de ces problèmes, en particulier pour les problèmes de très grande dimension. Récemment DCA a été appliqué avec succès au clustering dur via le modèle (3.5) ([18]) et (3.4) ([19]).

Dans le cardre de cette thèse, nous développons la programmation DC et DCA pour les deux problèmes : le clustering flou (chapitre 4) et le clustering hiérarchique à bi-niveaux (chapitre 5).

Chapitre 4

Le clustering flou via la programmation DC et DCA

Résumé

Nous présentons dans ce chapitre des approches robustes et efficaces de l'optimisation non convexe pour le clustering flou via le modèle de Fuzzy C-Means clustering (FCM). Nos approches sont basées sur la programmation DC et DCA qui ont été appliquées avec succès dans de nombreux domaines de sciences appliquées dont l'apprentissage. Le modèle de FCM est formulé sous la forme de trois équivalents DC programmes pour lelquels différents schémas de DCA sont proposés. Pour accélerer la convergence de DCA nous développons une procédure combinée de DCA et l'algorithme classique FCM. Les expériences numériques comparatives avec l'algorithme FCM sur les données réelles, en particulier en bioinformatique et segmentation d'images prouvent la robustesse, la performance et la supériorité de nos algorithmes, tant sur la qualité de solution que le temps de calcul.

4.1 Introduction

La notion de partition floue, qui s'est avérée très utile pour le développement des techniques floues de reconnaissance des formes, et notamment celui des c-moyennes floues, a été introduite par Ruspini ([225]). Bien que ce dernier auteur n'ait pas utilisé le terme de partition floue, il en a formalisé le concept avec, toutefois, une certaine ambiguïté due à l'amalgame entre degrés d'appartenance et probabilités. Le regroupement, selon Ruspini, revient à partager un élément quelconque entre les différentes classes qu'on cherche à identifier. La parenté de cet élément aux différentes classes est mesurée à l'aide de probabilités ou de degrés d'appartenance ou encore, pour simplifier, de pourcentages devant, naturellement, avoir une somme de 100%. Ruspini fut le premier à proposer, en 1969, une approche de classification combinant le concept de sous-ensemble flou et les techniques basées sur l'optimisation [225], [226]. Un peu plus tard, Dunn ([197]), et surtout Bezdek ([185], [186]), ont largement généralisé l'approche de Ruspini. Dunn s'est intéressé à la définition de critères pour détecter la présence de classes compactes et séparables au sein d'un ensemble d'objets,

et à la généralisation de l'algorithme des k-moyennes classiques. Bezdek ([188] a généralisé le critère proposé par Dunn à toute une famille d'algorithmes qu'il a appelée c-moyennes floues (Fuzzy C-means clustering ou brièvement FCM). Tout simplement, Bezdek a introduit au modèle de Dunn un paramètre, m, qui prend des valeurs strictement plus grandes que 1, et qui module le degré de flou de la partition obtenue à l'aide des c-moyennes floues. Le nombre de groupes est désormais désigné par c (et non plus k). C'est peut-être Dunn ([197]) qui a baptisé "k-partition floue", cette entité que Ruspini a définie mathématiquement sans la nommer. On ne trouve pas ce terme de partition floue dans d'autres travaux antérieurs à ceux de Dunn tels que [226] ou [201]. Le préfixe k vient, ici, de ce que la version classique des c-moyennes floues (fuzzy c-means) est connue sous le nom de "k-moyennes" (k-means). Nous ne connaissons pas la raison exacte pour laquelle c'est la lettre "c" qui a remplacé la lettre "k", notamment dans les travaux de Bezdek ([188](désirait-il d'évoquer le mot anglais "cluster" ?). Certains auteurs, comme Rousseeuw, parlent cependant, aujourd'hui encore, de k-moyennes floues ([224]).

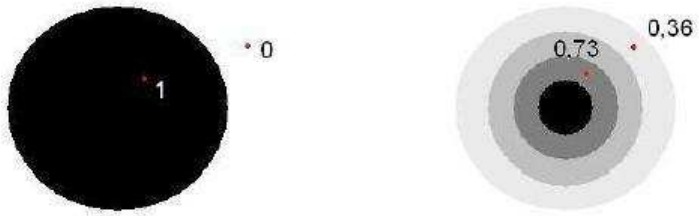

Fig. 4.1 – Le clustering dur (gauche) et le clustering flou (droite)

C'est, aussi, essentiellement, Bezdek qui s'est intéressé aux aspects mathématiques des c-moyennes floues et qui a étudié leur convergence [187], [190], [203], [203]. Les travaux de Ruspini, de Dunn et de Bezdek nous semblent les premiers travaux significatifs qui ont contribué à l'essor des méthodes de classification floue. De nombreux travaux ont été développés dans ce domaine depuis 1973 ([197], [185], [186], [187], [190], [203], [204], Davé [193], Krishnapuram et Keller [209][209],Pedrycz [216], ...) parmi lesquels c-moyennes floues (que nous appelons plus brièvement FCM) est incontestablement une méthode la plus utilisée. Le modèle de FCM est décrit comme suivant.

La formulation mathématique du modèle FCM ([188])

Soit $X := \{x_1, x_2, ..., x_n\}$ l'ensemble de n objets (points) à classer. Chaque point x_i est un vecteur dans l'espace $\mathbb{R}^p$. Nous avons à classer ces n points dans c ($2 \leq c \leq n$) classes différentes. Considérons une matrice de pourcentage U de taille $(c \times n)$ dont chaque élément $u_{i,k}$ définit le pourcentage d'appartenance du point x_k à la classe C_i. Il est clair que

$$u_{i,k} \in [0,1] \text{ pour } i = 1...c, k = 1...n \ ; \ \sum_{i=1}^{c} u_{i,k} = 1, \text{ pour } k = 1...n. \tag{4.1}$$

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

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

FCM consiste à la minimisation d'un critère J_m définie par :

$$J_m(U, V) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{i,k}^m ||x_k - v_i||^2, \tag{4.2}$$

où $\|.\|$ désigne la norme Euclidienne, V est une $(c \times p)$ - matrice dont chaque ligne v_i correspond au centre de la classe C_i, et $m \geq 1$ est un paramètre entier qui définit le degré de flou de la partition.

La tâche de chercher une partition revient donc de déterminer la matrice de pourcentage U et les centres v_i. Le modèle mathématique de FCM s'écrit ainsi :

$$\begin{cases} \min J_m(U, V) := \sum\limits_{k=1}^{n} \sum\limits_{i=1}^{c} u_{i,k}^m ||x_k - v_i||^2 \\ \text{tel que :} \\ \quad u_{i,k} \in [0, 1] \text{ for } i = 1, .., c \quad k = 1, .., n \\ \quad \sum\limits_{i=1}^{c} u_{i,k} = 1, \ k = 1, .., n \end{cases} . \tag{4.3}$$

L'algorithme FCM proposé par Bezdek se décrit comme suit :

Algorithme 4.1 *Schéma de FCM*

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

Répeter
- *Calculer les nouveaux centres v_i suivant l'équation :*

$$v_i = \sum_{k=1}^{n} u_{ik}^m x_k / \sum_{k=1}^{n} u_{ik}^m \quad \forall i = 1, 2, ..., c. \tag{4.4}$$

- *Calculer la nouvelle matrice de partition U par :*

$$u_{ik} = \left[\sum_{k=1}^{c} \frac{\|x_k - v_i\|^{2/(m-1)}}{\|x_k - v_j\|^{2/(m-1)}} \right]^{-1} \quad \forall i = 1, 2, ..., c \quad et \quad \forall k = 1, 2, ..., n. \tag{4.5}$$

Jusqu'à *[la modification de solution (U, V) entre les deux itérations successives est suffisament petite].*

Convergence de l'algorithme FCM : plusieurs études sur la convergence de l'algorithme FCM ont été faites depuis sa naissance ([187], [188], [189], [190], [219], [202], [204], Ismail et Selim [205][206], Sabin [222], [208], Yang et Yu [227], [228]) et finalement il s'en suit que cet algorithme heuristique ne converges pas vers un minimum local comme c'était annoncé par Bezdek au départ dans [187].

Du point de vue de l'optimisation, l'introduction de la matrice de pourcentage U rend le problème (4.3) beaucoupe plus difficile que le modèle k-means : la fonction objectif de (4.3) est plus compliquée, le nombre de variables est plus grand, la présence de contraintes. Puisqu'il s'agit d'un problème de très grande dimension en pratique (citons, par exemple, en analyse d'image, en bioinformatique où on utilise souvent ce modèle pour le clustering), les méthodes globales telles que Séparation et Evaluation (SE) ou Approximation de l'Extérieur (AE) ne sont pas envisageable pour sa résolution. Par souci constant de traiter efficacement les problèmes de grande taille nous développons dans ce travail l'approche de programmation DC et DCA pour ce modèle FCM. Dans l'optique de répondre aux deux questions cruciales lors de développement de DCA pour un problème concret, à savoir la recherche d'une *bonne* décomposition et celle d'un *bon* point initial, nous proposons trois équivalentes formulations DC de (4.3) qui donnent naissances les trois différents schémas de DCA, et une procédure combinée de DCA et l'algorithme standard FCM. Ces algorithmes sont testés sur nombreux jeux de données réelles, en particulier les données de biopuces. En plus, comme une application aux problèmes de très grande taille, nous utilisons cette approche pour la segmentation d'image et testé sur plusieurs images médicales par IRM (Imagerie par Résonance Magnétique). Les expériences numériques comparatives avec l'algorithme FCM prouvent la robustesse, la performance de nos algorithmes et leur supériorité par rapport à l'algorithme FCM, tant sur la qualité de solution que le temps de calcul.

Le reste du chapitre est organisé de la façon suivante. Dans la deuxième section, nous présentons nos approches, les trois décompositions DC et les schémas DCA correspondants, pour résoudre le problème (4.3). La troisième section concerne la technique de recherche d'un bon point initial en utilisant une procédure alternative FCM-DCA. Les résultats numériques sur les données réelles, en particulier les données de biopuces sont reportés dans la quatrième section. Enfin la segmentation d'image par le clustering flou via notre approche DCA est présentée dans la dernière section.

4.2 Résolution du problème (4.3) par la programmation DC et DCA

Dans toute la suite nous utilisons la présentation matricielle qui nous semble plus commode, sachant que l'on peut identifier une matrice par un vecteur (par ligne ou par colonne). Nous développons dans cette section les trois différentes décompositions DC pour le problème (4.3) et les schémas DCA correspondants.
Dans le problème (4.3), la variable U est a priori bornée dans $I\!R^{c\cdot n}$. On peut également

restreindre la variable V à un domaine borné. Soient $x_{k,j}$ le $j^{\text{ième}}(j = 1, ..., p)$ composant du vecteur x_k et

$$\alpha_j := \min_{k=1,...,n} x_{k,j},\ \beta_j := \max_{k=1,...,n} x_{k,j}.$$

On a : $v_i \in \mathcal{T}_i := \Pi_{j=1}^{p} [\alpha_j, \beta_j]$ pour $i = 1, ..., c$. Pour chaque $k \in \{1, ..., n\}$ on définit, Δ_k, un $(c-1)$-simplex dans $\mathbb{R}^c$:

$$\Delta_k := \left\{ U^k := (u_{i,k})_i \in \mathbb{R}_+^c : \sum_{i=1}^{c} u_{i,k} = 1 \right\}.$$

et $\Delta := \Pi_{k=1}^{n}\ \Delta_k, \mathcal{T} := \Pi_{i=1}^{c} \mathcal{T}_i$. Le problème (4.3) peut alors s'écrire sous la forme suivante :

$$\min \left\{ J_m(U, V) : U \in \Delta, V \in \mathcal{T} \right\}. \tag{4.6}$$

4.2.1 La première approche : une décomposition DC naturelle

En utilisant la formulation :

$$2f_1 f_2 = (f_1 + f_2)^2 - (f_1^2 + f_2^2)$$

on peut écrire la fonction objectif du problème (4.3) comme :

$$\begin{aligned} J_m(U, V) &= \sum_{k=1}^{n} \sum_{i=1}^{c} u_{i,k}^m \|x_k - v_i\|^2 \\ &= \tfrac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{c} \left(u_{i,k}^m + \|x_k - v_i\|^2 \right)^2 - \tfrac{1}{2} \left(u_{i,k}^{2m} + \|x_k - v_i\|^4 \right). \end{aligned}$$

La première décomposition suivante de $J_m(U, V)$ nous semble naturelle :

$$J_m(U, V) := G_1(U, V) - H_1(U, V) \tag{4.7}$$

où

$$G_1(U, V) = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{c} \left(u_{i,k}^m + \|x_k - v_i\|^2 \right)^2, \tag{4.8}$$

et

$$H_1(U, V) = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{c} \left(u_{i,k}^{2m} + \|x_k - v_i\|^4 \right). \tag{4.9}$$

En tenant compte du fait que la fonction $\theta(x) = f(x)^p$ est convexe pour $p \geq 1$ si f est convexe non négative, on démontre facilement que les fonctions $H_1(U, V)$ et $G_1(U, V)$ sont convexes, par suite J_m est une fonction DC. La première formulation DC du problème (4.3) est donc de la forme :

$$\min \left\{ G_1(U, V) - H_1(U, V) : (U, V) \in \Delta \times \mathcal{T} \right\}, \tag{4.10}$$

ou encore, sous la forme standard :

$$\min\left\{\chi_{\Delta\times\mathcal{T}} + G_1(U,V) - H_1(U,V) : (U,V) \in I\!R^{c\times n} \times I\!R^{c\times p}\right\}, \tag{4.11}$$

où χ_K désigne la fonction indicatrice de $K := \Delta \times \mathcal{T}$.

Résolution du problème (4.11) par DCA : selon la description de DCA dans le chapitre 1, la résolution de (4.10) par DCA consiste en la détermination de deux suites $\{(Y^l, Z^l)\}$ et $\{(U^l, V^l)\}$ telles que :

$$\begin{array}{rcl}(Y^l, Z^l) & \in & \partial H_1(U^l, V^l)\\ (U^{l+1}, V^{l+1}) & \in & \partial G_1^*(Y^l, Z^l).\end{array}$$

La fonction H_1 est différentiable et son gradient au point (V^l, U^l) est calculé de manière explicite :

$$\begin{aligned}\nabla H_1(U^l, V^l) &= (\nabla_U H_1(U^l, V^l)), \nabla_V H_1((U^l, V^l))\\ &= \left(\left(m\left(u_{i,k}^l\right)^{2m-1}\right)_{i=1,\dots c}^{k=1,\dots n}, 2\sum_{k=1}^{n}(\|x_k - V_i^l\|^2(V_i^l - x_k))_{i=1\dots c}\right).\end{aligned} \tag{4.12}$$

Le calcul de $(U^{l+1}, V^{l+1}) \in \partial G_1^*(Y^l, Z^l)$ se ramène à la résolution du problème convexe suivant :

$$\min\left\{G_1(U,V) - \left\langle (U,V), (Y^l, Z^l)\right\rangle : (U,V) \in \Delta \times \mathcal{T}\right\} \tag{4.13}$$

Puisque (4.13) est un problème convexe, nous pouvons appliquer n'importe quelle méthode de la programmation convexe pour le résoudre. Sachant que la projection d'un point sur un simplex ou sur un rectangle est explicite, nous utilisons la méthode Gradient Projeté ([95]) pour la résolution du sous-problème (4.13). Cette méthode se décrit comme suit :

Algorithme Gradient Projeté pour la résolution du problème (4.13)

Soit $\{\lambda_r\}$, une suite telle que $\lim_{r\to+\infty}\lambda_r = 0$ et $\sum_{r=1}^{+\infty}\lambda_r = +\infty$.
Soit une tolérance $\epsilon_{GP} > 0$

1. Pour $r = 1$, soit$(U^r, V^r) := (U^l, V^l)$, où (U^l, V^l) est la solution actuelle de DCA à l'itération l.
2. Calculer (U^{r+1}, V^{r+1}) de la manière suivante ($Proj_S$ désigne la projection de $I\!R^p$ sur S) :
   La $k^{\text{ième}}$ colonne $(U^{r+1})^k \in I\!R^c$ de la matrice U^{r+1} est

$$\left(U^{r+1}\right)^k := \mathrm{Proj}_{\Delta_k}\left((U^r)^k - \lambda_r\frac{\theta^r}{\|\theta^r\|}\right),$$

   où $\theta^r \in I\!R^c$ est la $k^{\text{ième}}$ colonne de $\nabla_U G_1(U^r, V^r) - Y^l$.
   La $i^{\text{ième}}$ ligne $(V^{r+1})_i \in I\!R^p$ de la matrice V^{r+1} est

$$\left(V^{r+1}\right)_i := \mathrm{Proj}_{\mathcal{T}_i}\left((V^r)_i - \lambda_r\frac{\left(\nabla_V G_1(U^r, V^r) - Z^l\right)_i}{\|\left(\nabla_V G_1(U^r, V^r) - Z^l\right)_i\|}\right).$$

3. Si $\|(U^{r+1}, V^{r+1}) - (U^r, V^r)\| \leq \epsilon_{GP}$ aller à l'étape 4, sinon poser $r = r + 1$ et aller à l'étape 2.
4. STOP, poser $(U^{l+1}, V^{l+1}) := (U^{r+1}, V^{r+1})$.

Remarque 4.1 *Malgré que le calcul de la projection d'un point sur un simplex/rectangle ne soit pas coûteux, l'algorithme Gradient Projeté est peu approprié pour les problèmes de grande taille. Cela vient du fait qu'il est un algorithme itératif et sa convergence est très sensible au choix de la suite $\{\lambda_r\}$. Sans doute qu'avec une méthode plus performante pour traiter les programmes convexes, nous pourrions améliorer notre DCA pour la résolution de FCM. La recherche d'autres décompositions DC pourrait, elle aussi, apporter de possibles améliorations.*

4.2.2 La deuxième approche : une intéressante décomposition DC pour le cas $m \geq 2$

Dans le but d'améliorer l'approche précédente en cherchant des méthodes efficaces pour la résolution des sous-problèmes convexes à chaque itération de DCA, nous étudions d'autres décompositions DC.

Considérons dans cette section le cas qui est souvent utilisé en pratique : $m \geq 2$. Pour avoir une autre décomposition DC, nous cherchons une boule de rayon r et du centre $0 \in \mathbb{R}^p$ qui contient nécessairement les centres optimaux v_i. La condition nécessaire d'optimalité du premier ordre en (U, V) implique que :

$$\nabla_V J_m(U,V) = 0,\ i.e.,\ \partial_{v_i} J_m(U,V) = \sum_{k=1}^{n} u_{i,k}^m 2(v_i - x_k),\ i = 1,...,c,\ k = 1,...,n$$

ou encore

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

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

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

Notons R_i $(i = 1,...,c)$, les boules Euclidiennes de rayon r dans $\mathbb{R}^p$ et $\mathcal{C} := \Pi_{i=1}^{c} R_i$. La seconde décomposition DC de J_m est basée sur le théorème suivant.

Théorème 4.1 *Il existe un scalaire $\rho > 0$ pour que la fonction*

$$H_2(U,V) := \frac{\rho}{2} \|(U,V)\|^2 - J_m(U,V) \tag{4.14}$$

soit convexe sur $\Delta \times \mathcal{C}$.

Preuve : définissons la fonction $h : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ par

$$h(x,y) \ = \frac{\rho}{2}x^2 + \frac{\rho}{2n}y^2 - x^m y^2 \ . \tag{4.15}$$

Il est clair que h est différentiable et son Hessien est calculé par :

$$J(x,y) = \begin{pmatrix} \rho - m(m-1)y^2x^{m-2} & -2mx^{m-1}y \\ -2mx^{m-1}y & \frac{\rho}{n} - 2x^m \end{pmatrix}. \tag{4.16}$$

Pour tout (x,y), $0 \leq x \leq 1, \mid y \mid \leq \alpha$, le déterminant de $J(x,y)$ est déterminé par :

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

Par suite, si

$$\rho \ \geq \frac{n}{2}\left[\frac{m}{n}(m-1)\alpha^2 + 2 + \sqrt{\left[\frac{m}{n}(m-1)\alpha^2 + 2\right]^2 + \frac{16}{n}m^2\alpha^2}\right] \tag{4.17}$$

alors $\mid J(x,y) \mid \geq 0$ pour tout $0 \leq x \leq 1, \mid y \mid \leq \alpha$.
Remarquons que $\rho > 0$ parce que $m > 1$. Avec ρ défini par (4.17), la fonction h est convexe sur $[0,1] \times [-\alpha, \alpha]$. Par conséquent, pour $x \to u_{i,k}$ et $y \to \|x_k - v_i\|^2$, les fonctions

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

sont convexes sur $\{0 \leq u_{i,k} \leq 1, \|v_i\| \leq r\}$ avec ρ donné dans (4.17) et

$$\alpha = r + \max_{1 \leq k \leq n} \|x_k\| \, . \tag{4.18}$$

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

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

est convexe sur $(u_{i,k}, v_i)$. Finalement, avec ρ défini dans (4.17) et $\alpha = r + \max\limits_{1 \leq k \leq n} \|x_k\|$, $H_2(U,V)$ est convexe sur $\Delta \times \mathcal{C}$ car

$$H_2(U,V) := \frac{\rho}{2}\,\|(U,V)\|^2 - J_m(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c} h_{i,k}(u_{i,k}, v_i).$$

■

Dans la suite, la fonction H_2 est définie avec la valeur de ρ satisfaisant (4.17). La deuxième décomposition DC de J_m est :

$$J_m(U,V) := G_2(U,V) - H_2(U,V) \tag{4.19}$$

avec $G_2(U,V) := \frac{\rho}{2}\|(U,V)\|^2$ étant convexe car $\rho > 0$. Par suite le problème (4.3) peut s'écrire comme :

$$\min\left\{\chi_{\Delta\times\mathcal{C}}(U,V) + \frac{\rho}{2}\|(U,V)\|^2 - H_2(U,V) : (U,V) \in (U,V) \in I\!R^{c\times n} \times I\!R^{c\times p}\right\}. \tag{4.20}$$

Résolution du problème (4.20) par DCA :

Pour déterminer le schéma DCA, nous devons calculer $(Y^l, Z^l) \in \partial H_2(U^l, V^l)$ et résoudre le problème convexe suivant :

$$\min\left\{\frac{\rho}{2}\|(U,V)\|^2 - \langle (U,V), (Y^l,Z^l)\rangle : (U,V) \in \Delta\times\mathcal{C}\right\}. \tag{4.21}$$

La fonction H_2 est différentiable et son gradient au point (U^l, V^l) est calculé par

$$\begin{aligned}(Y^l, Z^l) &= \nabla H_2(U^l, V^l)\\ &= \left(\left(\rho u_{i,k}^l - m(u_{i,k}^l)^{m-1}\|x_k - V_i^l\|^2\right)_{i=1,\dots c}^{k=1,\dots n}, \left(\rho V_i^l - 2\sum_{k=1}^{n}(V_i^l - x_k)(u_{i,k}^l)^m\right)_{i=1,\dots c}\right).\end{aligned} \tag{4.22}$$

Il est clair que la solution du problème auxiliaire (4.21) est déterminée via les projections d'un point sur un simplexe et/ou un rectangle :

$$\begin{aligned}\left(U^{l+1}\right)^k &= \mathrm{Proj}_{\Delta_k}\left(\left(Y^l\right)^k\right) \quad k = 1,\dots n,\\ V_i^{l+1} &= \mathrm{Pr\,oj}_{R_i}\left(\tfrac{1}{\rho}(Z^l)_i\right) \quad i = 1,\dots c.\end{aligned} \tag{4.23}$$

Remarque 4.2 *La seconde décomposition (4.19) est intéressante car le sous-problème convexe dans le schéma de DCA est résolue de manière explicite via les calculs de projection d'un point sur un simplexe et/ou un rectangle. DCA n'a donc plus besoin d'une méthode itérative à chaque itération comme dans la première approche.*

4.2.3 La troisième approche : la meilleure décomposition DC

Basé sur le théorème 4.1, nous développons dans cette partie une nouvelle décomposition DC qui nous semble la meilleure jusqu'à présent. Cette approche traite le cas général : $m \geq 1$. Considérons les nouvelles variables $t_{i,k}$ telles que $u_{i,k} = t_{i,k}^2$. La contrainte $\sum_{i=1}^{c} u_{i,k} = 1$ devient alors

$$\sum_{i=1}^{c} t_{i,k}^2 = 1 \text{ ou } \|t_k\|^2 = 1 \text{ avec } t_k = (t_{1,k}, t_{2,k}, \dots, t_{c,k}) \in I\!R^c. \tag{4.24}$$

Soient S_k la sphère de rayon 1 dans $\mathbb{R}^c$. Le problème (4.3) peut être reformulé comme :

$$\begin{cases} \min \quad J_{2m}(T,V) := \sum_{k=1}^{n}\sum_{i=1}^{c} t_{i,k}^{2m}||x_k - v_i||^2 \\ \text{tel que :} \\ \quad T = (t_{i,k})_{c\times n} \in \mathcal{S} := \Pi_{k=1}^{n}\ S_k, \\ \quad V = (v_1, v_2, ..., v_c) \in \mathcal{C} := \Pi_{i=1}^{c} R_i \end{cases} . \tag{4.25}$$

La fonction objectif de (4.24) peut s'écrire de la manière suivante, pour un scalaire quelconque ρ :

$$J_{2m}(T,V) = \ \frac{\rho}{2}(\|T\|^2 + \|V\|^2) - \left[\frac{\rho}{2}\|(T,V)\|^2 - J_{2m}(T,V)\right] . \tag{4.26}$$

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

$$J_{2m}(T,V) = \frac{\rho}{2}n + \frac{\rho}{2}\|V\|^2 - H_3(T,V), \tag{4.27}$$

où

$$H_3(T,V) \ := \frac{\rho}{2}\|(T,V)\|^2 - J_{2m}(T,V). \tag{4.28}$$

Dans le théorème suivant, nous donnerons les conditions pour que la fonction H_3 soit convexe.

Théorème 4.2 *Soit $\mathcal{B} := \Pi_{k=1}^{n}\ B_k$, où B_k est la boule de centre 0 et rayon 1 dans $\mathbb{R}^c$. La fonction $H_3(T,V)$ est convexe sur $\mathcal{B}\times\mathcal{C}$ pour toute valeur de ρ telle que*

$$\rho \ \geq \frac{m}{n}(2m-1)\alpha^2 + 1 + \sqrt{\left[\frac{m}{n}(2m-1)\alpha^2 + 1\right]^2 + \frac{16}{n}m^2\alpha^2}, \tag{4.29}$$

avec α défini dans (4.18).

Preuve. On remarque tout d'abord que $\rho > 0$ car $m \geq 1$. La preuve est similaire de celle du théorème 4.1 mais avec $u_{i,j}$ remplacé par $t_{i,k}$ et m remplacé par $2m$. ■

Dans la suite, nous utilisons la valeur de ρ et α vérifiant (4.29). Il est clair que, pour tout $T \in \mathcal{B}$ et avec un $V \in \mathcal{C}$ fixé, la fonction $J_{2m}(T,V)$ est concave en variable T (car $H(T,V)$ est convexe), par suite son minimum sur $\mathcal{B}$ est atteint sur la frontière $\mathcal{S}$ de $\mathcal{B}$, i.e.,

$$\min\left\{\frac{\rho}{2}\|V\|^2 - H_3(T,V) : (T,V) \in \mathcal{B}\times\mathcal{C}\right\}$$

$$= \min\left\{\frac{\rho}{2}\|V\|^2 - H_3(T,V) : (T,V) \in \mathcal{S}\times\mathcal{C}\right\}.$$

En ignorant le premier terme $\frac{\rho}{2}n$ dans (4.27) et définissant $G_3(T,V) = \frac{\rho}{2}\|V\|^2$ (qui est une fonction convexe), le problème (4.25) peut être écrit sous la forme d'une programmation DC :

$$\min\left\{G_3(T,V) - H_3(T,V) : \ (T,V) \in \mathcal{B}\times\mathcal{C}\right\},$$

ou encore :

$$\min\left\{\chi_{\mathcal{B}\times\mathcal{C}}(T,V) + G_3(T,V) - H_3(T,V) : (T,V) \in \mathbb{R}^{c\times n}\times\mathbb{R}^{c\times p}\right\}. \tag{4.30}$$

Résolution du problème (4.30) par DCA :

Le schéma général de DCA appliqué à (4.30) consiste en la construction de deux suites $(Y^l, Z^l) \in \partial H_3\ (T^l, V^l)$ et

$$(T^{l+1}, V^{l+1}) \in \arg\min \left\{ \frac{\rho}{2} \|(T,V)\|^2 - \langle (T,V), (Y^l, Z^l) \rangle : \mathcal{B} \times \mathcal{C} \right\}. \tag{4.31}$$

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

$$\begin{aligned} (Y^l, Z^l) &= \nabla H_3(U^l, V^l) \\ &= \left(\left(\rho t_{i,k}^l - 2m(t_{i,k}^l)^{2m-1} \|x_k - v_i^l\|^2\right)_{i=1,...c}^{k=1,...n}, \left(\rho V_i^l - 2\sum_{k=1}^{n} (V_i^l - x_k) t_{i,k}^{2m}\right)_{i=1,...c} \right). \end{aligned} \tag{4.32}$$

La solution de (4.31) est déterminée explicitement par

$$\begin{aligned} \left(T^{l+1}\right)^k &= \mathrm{Pr\,oj}_{B_k}\left((Y^l)^k\right)\ \ k = 1, ...n, \\ V_i^{l+1} &= \mathrm{Pr\,oj}_{R_i}\left(\tfrac{1}{\rho}(Z^l)_i\right)\ \ i = 1, ...c. \end{aligned} \tag{4.33}$$

Remarque 4.3 *Le changement de variables en $t_{i,k}$ nous amène à travailler sur $\mathcal{S}$, le domaine réalisable des variables $t_{i,k}$ qui est le produit des sphères et non pas le produit des simplexes comme dans le cas des variables initiales $u_{i,k}$. On pourrait penser alors que la non convexité de $\mathcal{S}$ rend le problème plus difficile que sa formulation initiale. Mais ce n'est pas le cas, car le problème (4.30) est reformulé sous une forme équivalente où $\mathcal{S}$ est remplacé par le produit des boules de rayon 1. Ce qui est intéressant, tant sur le plan algorithmique que numérique : la nouvelle formulation DC de (4.25) donne naissance à un schéma DCA extrêmement simple et qui ne nécessite que des calculs explicites et donc non coûteux. Il s'agit des calculs des projections d'un point sur une boule Euclidienne à chaque itération. La troisième décomposition DC est donc la meilleure.*

4.2.4 Algorithmes

Nous pouvons maintenant décrire en détail les trois schémas de DCA pour la résolution du problème (4.3).

Algorithme 4.2
Fuzzy-DCA-1 : DCA appliqué à la première formulation DC (4.11)

Initialisation : *Choisir la matrice de pourcentage U^0 et les centres V^0. Soit une tolérance $\epsilon_{DCA} > 0$, $0 \leftarrow l$.*

Répéter

– *Déterminer* $(Y^l, Z^l) := \left(\left(mu_{i,k}^{2m-1}\right)_{i=1,...c}^{k=1,...n}, 2\sum_{k=1}^{n}(\|x_k - v_i\|^2(v_i - x_k))_{i=1..c} \right)$.
– *Appliquer l'algorithme* ***GP*** *jusqu'à sa convergence pour obtenir la solution* (U^{l+1}, V^{l+1}).
– $l+1 \leftarrow l$
Jusqu'à $\|(U^{l+1}, V^{l+1}) - (U^l, V^l)\| \leq \epsilon_{DCA}$.

Algorithme 4.3
Fuzzy-DCA-2 : DCA appliqué à la deuxième formulation DC (4.20)

Initialisation ***:*** *Choisir la matrice de pourcentage* U^0 *et les centres* V^0. *Soit une tolérance* $\epsilon_{DCA} > 0$, $0 \leftarrow l$.

Répéter
– *Déterminer* $Y^l := \left(\rho u_{i,k}^l - m\left(u_{i,k}^l\right)^{m-1}\|x_k - V_i^l\|^2\right)_{i=1,...c}^{k=1,...n}$,
$Z^l := \left(\rho V_i^l - 2\sum_{k=1}^{n}(V_i^l - x_k)\left(u_{i,k}^l\right)^m\right)_{i=1,...c}$
– *Calculer* (U^{l+1}, V^{l+1}) *par :*

$$\left(U^{l+1}\right)^k = Proj_{\Delta_k}\left(\left(Y^l\right)^k\right) \text{ pour } k = 1, ...n,$$

$$V_i^{l+1} = \Pr oj_{R_i}\left(\frac{1}{\rho}(Z^l)_i\right) = \begin{cases} \frac{(Z^l)_{i,.}}{\rho} \text{ si } \|(Z^l)_{i,.}\| \leq \rho r \\ \frac{(Z^l)_{i,.}r}{\|(Z^l)_i\|} \text{ sinon} \end{cases}, (i = 1, .., c).$$

– $l+1 \leftarrow l$
Jusqu'à $\|(U^{l+1}, V^{l+1}) - (U^l, V^l)\| \leq \epsilon_{DCA}$.

Algorithme 4.4
Fuzzy-DCA-3 : DCA appliqué à la troisième formulation DC (4.30)

Initialisation ***:*** *Choisir la matrice de pourcentage* U^0 *et les centres* V^0. *Soit une tolérance* $\epsilon_{DCA} > 0$, $0 \leftarrow l$.

Répéter
– *Déterminer* $Y^l := \left(\rho t_{i,k}^l - 2mt_{i,k}^{2m-1}\|x_k - v_i\|^2\right)_{i=1,...c}^{k=1,...n}$,
$Z^l := \left(\rho V_i^l - 2\sum_{k=1}^{n}(V_i^l - x_k)t_{i,k}^{2m}\right)_{i=1,...c}$
– *Calculer* (T^{l+1}, V^{l+1}) *par :*

$$V_i^{l+1} = \begin{cases} \frac{(Z^l)_i}{\rho} \text{ si } \|(Z^l)_i\| \leq \rho r \\ \frac{(Z^l)_{i,.}r}{\|(Z^l)_i\|} \text{ sinon} \end{cases}, (i = 1, .., c), \tag{4.34}$$

$$\left(T^{l+1}\right)^k = \begin{cases} \left(Y^l\right)^k & si \parallel \left(Y^l\right)^k \parallel \leq 1 \\ \frac{\left(Y^l\right)^k}{\parallel \left(Y^l\right)^k \parallel} & sinon \end{cases}, (k = 1, .., n). \tag{4.35}$$

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

Sortie : $u^*_{i,k} = t^l_{i,k}.t^l_{i,k}$ *pour* $i = 1, .., c$ *et* $k = 1, .., n$.

4.3 La recherche d'un bon point initial pour DCA par une procédure alternative FCM-DCA

La recherche d'un bon point initial joue un rôle crucial dans la résolution d'une programmation DC par DCA. Elle dépend de la structure du problème considéré et elle peut être effectuée par, par exemple, une méthode heuristique bien adaptée au problème. D'une manière générale, un bon point initial pour le DCA ne doit pas être un minimum local, parce qu'à partir d'un tel point, le DCA est stationnaire. En plus, nous observons qu'à partir de n'importe quel point n'étant pas minimum local, la fonction objective diminue rapidement durant les premières itérations de DCA. Nous avons la même remarque pour l'algorithme standard FCM. C'est pourquoi, nous proposons une procédure alternative de FCM-DCA pour le problème (4.3).

Algorithme 4.5
Procédure FCM-DCA

Initialisation : Initialise la matrice de partition U^o *ainsi que les centres* V^o *aléatoirement. Soit maxiter un nombre entier positif.*

Répeter

- *Une itération de FCM : Calculer les centres* $V^l = (v^l_1, v^l_2, ..., v^l_c)$ *par la formule*

$$v^l_i = \sum_{k=1}^{n} u^m_{ik} x_k / \sum_{k=1}^{n} u^m_{ik} \quad \forall i = 1, ..., c. \tag{4.36}$$

  Calculer U^l *par*

$$u^l_{ik} = \left[\sum_{k=1}^{c} \frac{\|x_k - v_i\|^{2/(m-1)}}{\|x_k - v_j\|^{2/(m-1)}} \right]^{-1}. \tag{4.37}$$

- *Une itération de DCA : exécuter une itération de* ***Fuzzy-DCA-1*** *ou* ***Fuzzy-DCA-2*** *(resp.* ***Fuzzy-DCA-3****) avec le point itinial* (U^l, Z^l) *(resp.* (T^l, V^l) *avec* $t^l_{ik} = \sqrt{u_{ik}}$*, pour* $i = 1, ..., c$ *et* $k = 1, .., n$*) pour obtenir* (U^{l+1}, V^{l+1}) *(resp.* (T^{l+1}, V^{l+1}) *et* $u^{l+1}_{ik} = (t^{l+1}_{ik})^2$*).*
- $l + 1 \leftarrow l$
- *Jusqu'à* $l = maxiter$.

Si nous utilisons l'algorithme combiné de FCM-DCA jusqu'à sa convergence, peut être l'efficacité de DCA n'est pas bien exploitée. Pour remédier à cette situation, nous proposons un algorithme de deux phases. Dans la première phase, nous exécutons quelques itérations de l'algorithme combiné de FCM-DCA pour trouver un bon point initial. Et dans la deuxième, à partir du point trouvé, nous appliquons DCA jusqu'à sa convergence. Comme on verra dans la suite, parmi différentes versions de DCA cet algorithme de deux phases est la meilleure.

4.4 Expériences numériques

Nous avons codé les algorithmes en C++ et exécuté sur une machine Pentium 4 CPU 2.8GHz 1.00Go RAM.

Les algorithmes sont testés sur deux ensembles de jeux de test. Le premier ensemble contient 5 jeux de données

- "PAPILLON" un jeu de données connu sous le nom "jeux de papillon ". Plusieur travaux ont utilisé ce jeu de données (voir Revue Modulad - Le Monde Des Utilisateurs de L'Analyse de Données, numéro 11, 1993, 7-44).
- "IRIS" est le classique jeu de données IRIS qui est peut-être le plus connu dans le domaine de reconnaissance de forme. IRIS contient 3 classes (Iris Setosa, Iris Versicolor, Iris Verginica), chacune a 50 objets dans R^4. La première classe est linéairement séparable aux deux autres qui ne sont pas linéairement séparable.
- "GENE" Un ensemble de 384 gènes disponible sur http ://faculty.washington.edu/kayee/cluster/
- "VOTE" Congressional Votes dataset (Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL : Congressional Quarterly Inc. Washington, D.C., 1985). Pour chaque vote, il y a trois possibilité de réponse : yes, nay et unknow. Il y a deux classe dans ce jeu de données : démocratie(267) et république(168).
- "ADN" L'ensemble de 3186 gènes disponible sur ftp ://genbank.bio.net, chaque gène est présenté par une séquence de 60 éléments. Ces gènes sont classés dans 3 clusters différents : "donors" (767 objets), "acceptors" (765 objets) et le rest.

Le deuxième ensemble de jeux de données sont les données de biopuces :

- "Yeast" est composé par 2945 gènes dans l'espace de dimension 15 et disponible sur http ://genomics.stanford.edu.
- "Serum" est composé par 517 gènes dans l'espace de dimension 12 et disponible sur http ://genomics.stanford.edu.
- "Human cancer" est composé par 60 cellule cancer dans l'espace de dimension 768 et disponible sur http ://discover.nci.nih.gov/nature2000/.
- "Breast Cancer" est composé par 84 cellule cancer dans l'espace de dimension 1753 et disponible sur http ://bioinformatics.upmc.edu/GE2/GEDA.html.

- "Ovarian Cancer" est composé parn 39 cellule cancer dans l'espace de dimension 7039 et récupérable sur http ://bioinformatics.upmc.edu/GE2/GEDA.html.

Dans la première expérimentation, nous comparons la performance de trois algorithmes DCA et l'algorithme standard FCM pour le premier ensemble de jeux de données. L'implémentation de FCM est disponible sur http ://www-igbmc.u-strasbg.fr/projets/fcm/. Dans cette expérimentation, on choisit $m = 2$. La tolérance ϵ_{DCA} est égale à 10^{-7}. Tous les algorithmes commencent avec le même point initial généré aléatoirement. Le résultat est présenté dans le tableau 4.1.

Dans la deuxième expérimentation, nous testons les algorithmes sur les données biopuces avec différentes valeur de paramètre m ce qui un majeur problème sur ce type de données. Le travail de Dembélé ([194]) a montré qu'en pratique $m = 2$ n'est pas adapté pour quelques jeux de données et la valeur optimale de m est très variée d'un jeu de données à l'autre. Nous testons avec $m \in [1.1, 5]$ sur 3 algorithmes DCA (avec la procédure FCM-DCA pour la recherche du point initial). Les résultats sont présenté dans tableau 4.2, 4.3, 4.4, 4.5 et 4.6. Dans la procédure FCM-DCA, nous prenons $maxiter = 5$.
Le diagramme dans la figure 4.2 représente la performance de quatre algorithmes sur les données "Yeast".

Dans la troisième expérimentation, nous nous intéressons sur l'effet de la recherche d'un bon point initial par la procédure FCM-DCA. Nous comparons deux versions de Fuzzy-DCA-3 (avec et sans la procédure FCM-DCA) sur le jeux de données "Humans". Le résultat est reporté dans la figure 4.3.

Dans ces tableaux nous utilisons les notations suivantes :

– N^{o}it : nombre d'itérations.
– Time : temps de calcul en seconde.
– PWPO : le pourcentage de point bien classé.
– $J_c = \sum_{i=1..n} \min_{k=1..c} \|x_i - v_k\|^2$, le coût du cluster (ce critère n'est pas le même que (4.2) !).
– N/A pour les cas ($m < 2$) où Fuzzy-DCA-2 n'est pas applicable.

Data	Fuzzy-DCA-1			Fuzzy-DCA-2			Fuzzy-DCA-3			FCM		
(n,p,c)	N^{o}it	Time	PWPO	N^{o}it	Time	PWPO	N^{o}it	Time	PWPO	N^{o}it	Time	PWPO
PAPILON (23,4,5)	20	0.003	91.3	10	0.002	91.3	2	0.001	91.3	18	0.002	91.3
IRIS (150,4,3)	23	0.03	91.77	5	0.01	91.77	4	0.01	91.77	15	0.03	89.33
VOTE (435,16,2)	16	0.05	87.9	3	0.01	87.9	3	0.01	89.8	19	0.06	83.7
GENE (384,17,5)	16	0.67	88.3	8	0.20	90.7	8	0.20	96.2	35	0.73	85.8
ADN (3186,60,3)	8	0.78	92	8	0.62	92.4	6	0.55	94	25	1.95	89.8

Tab. 4.1 – Résultat sur premier ensemble de jeux de données

Yeast data set (n=2945, p=15, c=16)												
	Fuzzy-DCA-1			Fuzzy-DCA-2			Fuzzy-DCA-3			FCM		
m	J_c	N^{o}it	Time	J_c	N^{o}it	Time	J_c	N^{o}it	Time	J_c	N^{o}it	Time
1.1	64831	31	314	N/A	N/A	N/A	**64061**	120	34	65868	179	564
1.3	64144	357	64	N/A	N/A	N/A	**62593**	114	30	62681	543	1886
1.5	43398	54	301	N/A	N/A	N/A	**43341**	155	68	44367	123	213
1.7	44981	42	187	N/A	N/A	N/A	**43792**	94	30	43939	48	118
1.9	45012	65	84	N/A	N/A	N/A	**43956**	72	29	44643	42	87
2	**43687**	61	70	**43687**	37	32	**43687**	44	24	43738	37	57
3	43710	45	34	**43687**	32	24	**43687**	42	22	43738	24	19
4	45012	20	19	43722	23	18	**43676**	17	16	43738	21	18
5	**43687**	21	18	**43687**	19	15	**43687**	15	14	43738	19	16

TAB. 4.2 – Résultat sur jeu de données "Yeast"

Serum data set (n=517, p=12, c=8)												
	Fuzzy-DCA-1			Fuzzy-DCA-2			Fuzzy-DCA-3			FCM		
m	J_c	N^{o}it	Time	J_c	N^{o}it	Time	J_c	N^{o}it	Time	J_c	N^{o}it	Time
1.1	12237	56	2.2	N/A	N/A	N/A	**12234**	47	0.92	13262	191	1.98
1.3	**9572**	78	5.2	N/A	N/A	N/A	**9572**	95	2.0	10231	176	5.3
1.5	8642	52	6.7	N/A	N/A	N/A	**8019**	109	2.4	9431	71	3.1
1.7	6034	41	4.1	N/A	N/A	N/A	**6013**	56	1.2	6068	29	1.4
1.9	6079	85	2.2	N/A	N/A	N/A	**6009**	55	1.3	6081	11	0.7
2	6023	17	0.76	**6001**	19	0.68	**6001**	19	0.67	6083	13	0.6
3	**6069**	12	0.61	**6069**	8	0.52	**6069**	10	0.43	6199	11	0.6
4	6079	17	0.67	**6069**	15	0.60	**6069**	15	0.52	6201	15	0.7
5	6092	18	0.87	6092	13	0.57	**6073**	11	0.49	6201	19	0.8

TAB. 4.3 – Résultat sur jeu de données "Serum"

Human cancer data set (n=60, p=728, c=9)												
	Fuzzy-DCA-1			Fuzzy-DCA-2			Fuzzy-DCA-3			FCM		
m	J_c	N^{o}it	Time	J_c	N^{o}it	Time	J_c	N^{o}it	Time	J_c	N^{o}it	Time
1.1	70300	156	25.1	N/A	N/A	N/A	**72235**	87	15.5	70315	118	22.4
1.3	**77599**	178	30.2	N/A	N/A	N/A	**77599**	94	19.4	77638	124	25.1
1.5	107792	237	39.6	N/A	N/A	N/A	**107824**	123	29.3	107858	280	52.4
1.7	**124698**	35	8.4	N/A	N/A	N/A	**124698**	42	7.9	**124698**	50	10.2
1.9	124602	85	13.4	N/A	N/A	N/A	**124496**	36	5.9	124698	36	8.6
2	124662	55	10.5	**124654**	28	5.2	**124654**	36	5.8	124697	31	8.4
3	**124654**	19	5.1	**124654**	17	4.6	**124654**	15	3.5	124697	21	6.3
4	124692	23	5.8	**124654**	19	4.7	**124654**	19	4.5	124699	19	5.9
5	**124654**	22	5.8	**124654**	22	4.8	**124654**	19	4.5	124699	18	5.2

TAB. 4.4 – Résultat sur jeu de données "Human cancer"

Breast cancer data set (n=84, p=1753, c=2)												
	Fuzzy-DCA-1			Fuzzy-DCA-2			Fuzzy-DCA-3			FCM		
m	J_c	N^{o}it	Time	J_c	N^{o}it	Time	J_c	N^{o}it	Time	J_c	N^{o}it	Time
1.1	**1878.3**	18	1.45	N/A	N/A	N/A	**1878.3**	15	1.12	1892.5	22	1.42
1.3	**1897.8**	16	1.45	N/A	N/A	N/A	**1897.8**	15	1.10	1902.8	19	1.40
1.5	1909.8	13	1.1	N/A	N/A	N/A	**1909.5**	12	1.0	1929.1	19	1.4
1.7	**2011.3**	20	1.6	N/A	N/A	N/A	**2011.3**	17	1.2	2040.4	50	3.8
1.9	**2302.3**	25	1.9	N/A	N/A	N/A	**2302.3**	18	1.3	2317.8	141	13.8
2	**2302.3**	20	1.6	**2302.3**	16	1.3	**2302.3**	18	1.3	2317.9	85	6.2
3	**2302.3**	19	1.6	**2302.3**	17	1.4	**2302.3**	15	1.2	2317.9	30	1.98
4	**2302.3**	20	1.8	**2302.3**	19	1.4	**2302.3**	19	1.4	2317.9	24	1.57
5	**2302.3**	16	1.4	**2302.3**	13	1.2	**2302.3**	12	1.2	2317.9	22	1.53

TAB. 4.5 – Résultat sur jeu de données "Breast cancer"

Ovarian cancer data set (n=39, p=7039, c=2)												
	Fuzzy-DCA-1			Fuzzy-DCA-2			Fuzzy-DCA-3			FCM		
m	J_c	N°it	Time	J_c	N°it	Time	J_c	N°it	Time	J_c	N°it	Time
1.1	**253.67**	18	2.65	N/A	N/A	N/A	**253.67**	17	2.17	263.54	47	5.9
1.3	263.67	19	2.65	N/A	N/A	N/A	**253.59**	15	2.10	263.15	31	3.9
1.5	254.85	18	2.64	N/A	N/A	N/A	**253.59**	13	2.0	263.25	26	3.2
1.7	**253.77**	20	2.70	N/A	N/A	N/A	**253.77**	17	2.0	263.72	25	3.1
1.9	**255.21**	22	2.75	N/A	N/A	N/A	**255.21**	15	1.9	264.6	20	2.5
2	**255.24**	20	2.6	**255.24**	16	2.0	**255.24**	18	2.1	265.3	19	2.4
3	**271.43**	18	2.5	**271.43**	17	1.9	**271.43**	15	2.0	282.9	30	3.98
4	**303.82**	19	2.5	**303.82**	14	1.6	**303.82**	14	1.5	311.76	58	7.4
5	**345.26**	18	2.4	**345.26**	14	1.6	**345.26**	13	1.4	354.58	182	23.13

TAB. 4.6 – Résultat sur jeu de données "Ovarian cancer"

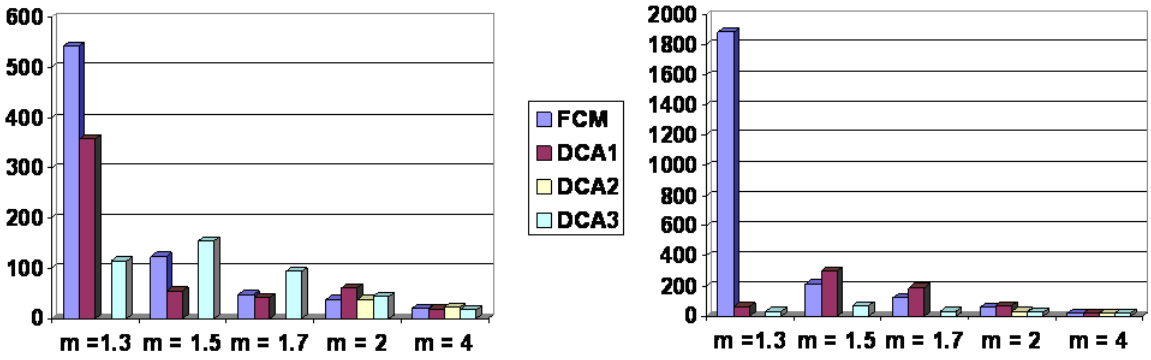

FIG. 4.2 – Résultat sur données "Yeast" : nombre d'itérations (gauche) et temps de calcul (droite)

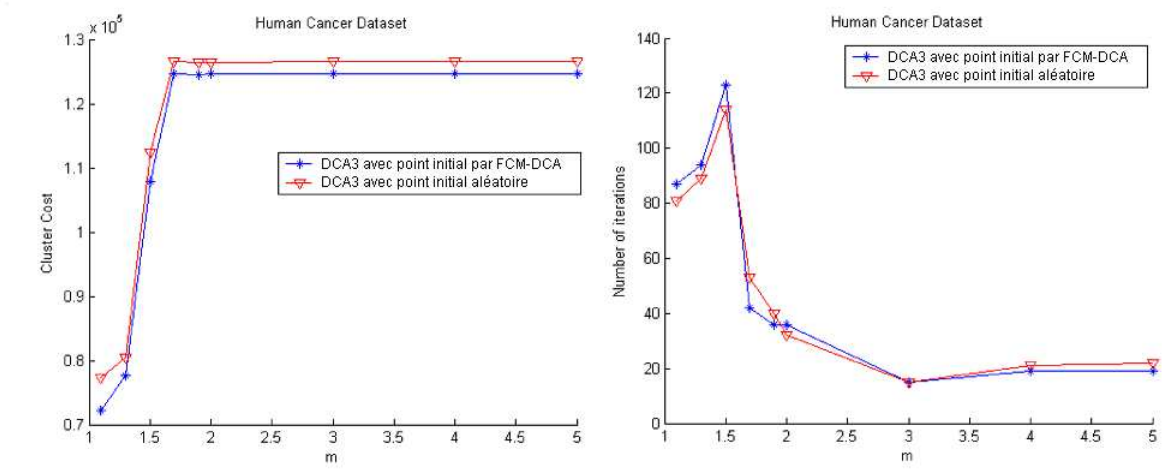

FIG. 4.3 – Fuzzy-DCA-3 avec et sans la procédure FCM-DCA, le coût de cluster (gauche) et nombre d'itérations (droite)

A partir des expérimentations, nous constatons que :

- DCA est efficace pour le problème FCM : tous les trois algorithmes DCA sont mieux que l'algorithme FCM standard.
- Fuzzy-DCA-3 est le meilleur parmi les trois algorithmes DCA en temps de calcul et coût de cluster.
- La procédure FCM-DCA est efficace pour trouver un bon point initial.
- FCM et DCA sont assez sensible au paramètre m.

4.5 Segmentation d'image par le clustering flou

4.5.1 Introduction

Nous présentons dans cette section une application de notre algorithme Fuzzy-DCA-3 à la segmentation d'image.

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

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

Dans ce domaine, de nombreuses méthodes basées sur différentes approches basées sur le contour, larégion, la texture,..., ont été développées au cours de- ces dernières années ([130]). On peut confirmer qu'aucune méthode ne semble prévaloir sur les autres, chacune ayant son domaine de prédilection. En absence de méthode universelle, il est classique de retrouver les différentes approches classifiées en quatre thèmes : clustering, approches contours, approches

régions et méthodes hybrides. Rajapakse ([132]) a classifié les différentes méthodes en tant que quatre catégories :

- Les méthodes classiques telles que le seuillage, la croissance de région, la segmentation basée sur les contours. La première méthode consiste à déterminer le seuil à appliquer à l'image : le seuillage permet de sélectionner les parties de l'image qui intéressent l'opérateur. La deuxième méthode propose à faire croître chaque région autour d'un pixel de départ dont l'agglomération n'exploite aucune connaissance a priori de l'image ou du bruit et la décision d'intégrer à la région un pixel voisin repose seulement sur un critère d'homogénéité imposé à la zone en croissance. Et la troisième méthode s'intéresse aux contours des objets extraits de l'image. La plupart du temps, ces contours sont morcelés et peu précis, il faut donc utiliser des techniques de reconstruction de contours par interpolation ou connaître a priori la forme de l'objet recherché.
- Les méthodes statistiques telles que la segmentation bayesienne ou la segmentation au maximum de vraisemblance sont basées sur les développements des chaînes de Markov. Dans [125], on peut trouver un algorithme de segmentation non supervisé "Multiple Resolution Segmentation" (MRS), formulé dans le contexte bayesien. La méthode utilise un modèle AR-2D causal (Gaussian Autoregressive) : l'image observée est considérée comme le mélange de champs aléatoires statistiquement homogénes. Le champ des classes est modélisé par un champ aléatoire markovien. La segmentation optimale est définie au sens du Maximum a Posteriori (MAP) et estimée grâce à un algorithme de minimisation locale (Iterated Conditional Mode).
- Les méthodes de réseaux de neurones : Une des stratégies possibles pour la segmentation est celle de la classification de pixels. Quelques exemples de segmentation d'images colorées par réseaux de neurones ont été publiés récemment. Dans [126], les réseaux utilisés étaient des réseaux de Hopfield, configurés à l'aide de l'histogramme des couleurs. Dans [129], les auteurs utilisent les réseaux auto-organisés du type Kohonen pour la segmentation.
- Les méthodes de clustering flou : ces techniques permetent d'obtenir une partition floue de l'image en donnant à chaque pixel de l'image un degré d'appartenance à une région donnée.

Un inconvénient du modèle standard de FCM dans la segmentation d'image est de ne pas tenir compte de l'information spatiale qui une relation entre le pixel et ses voisinages. Pourtant, cette information rend l'algorithme très sensible au bruit et à d'autres objets façonnés dans l'image. En fait, cette relation est une des caractéristiques importantes d'une image car les voisinages possèdent souvent les valeurs semblables, et la probabilité qu'ils appartiennent à la même partition est très élevée. Par ailleurs, si nous considérons une image bruitée, le FCM n'est pas une méthode permettant de surmonter le problème. Récemment, de nombreux chercheurs ont ajouté l'information spatiale à l'algorithme original de FCM pour améliorer l'efficacité de la segmentation d'image([131, 133, 128]).

Le but de notre travail est double. Le modèle de la classification dans la segmentation d'image est, en général, un problème de très grande dimension pour lequel, la recherche des méthodes efficaces est toujours d'actualité. Premièrement, nous proposons une nouvelle méthode de segmentation d'image via le modèle de FCM en utilisant notre approche DCA. Deuxièmement,

pour la segmentation d'image bruitée, nous considérons un modèle adaptatif de FCM (appelé FCM-Spatial) qui incorpore l'information spatiale à la fonction de clustering.

4.5.2 Modèle de FCM avec l'information spatiale

Dans la segmentation d'image par le modèle standard de FCM, chaque pixel $x_k \in \mathbb{R}^d$ représente les données multispectrales. Cependant, comme mentionné précédemment, une des caractéristiques importantes d'une image est que les voisinages de pixel possèdent les valeurs semblables, le rapport spatial est donc intéressant pour la segmentation d'image. L'information spatiale est la relation entre le pixel et ses voisinages. Il y a différentes manières d'incorporer l'information spatiale (voir la figure 4.4).

Dans notre cadre, nous considérons l'information spatiale de x_k comme une valeur moyenne de ses voisinages 3×3, et chaque point x_k dans (4.3) a deux groupes de valeurs : les valeurs du pixel et les valeurs moyennes de ses voisinages 3×3.
Soit N_k les voisinages 3×3 du pixel x_k. Les données entrées x_k dans notre modèle spatial de FCM sont $x_k = (x_{k1}, x_{k2})$ où x_{k1} représente les valeurs du pixel de k^{th} de l'image et $x_{k2} = (x_{k1} + \sum_{i \in N_k} x_{i1})/9$. D'où, le nombre de variables U n'est pas changé et V devient une matrice de $c \times 2d$ dont la i^{eme} ligne est, $v_i \in \mathbb{R}^{2d}$, le centre de la région C_i.
Le modèle spatial de FCM dans notre approche n'est pas tellement différent par rapport à son modèle standard FCM (4.3) sauf le fait que chaque $x_k \in \mathbb{R}^d$ est remplacé par $x_k = (x_{k1}, x_{k2}) \in \mathbb{R}^{2d}$. Par conséquent, n'importe quel algorithme pour le modèle standard de FCM (4.3) peut être appliqué au modèle spatial de FCM. Au point de vue numérique, le problème spatial de FCM est plus difficile car le nombre de variables V est doublé.

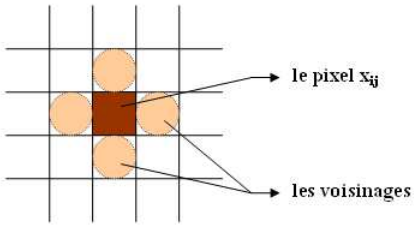

FIG. 4.4 – Le pixel et ses 4 voisinages

4.5.3 Expériences numériques

Pour vérifier l'efficacité de notre méthode proposée, nous utilisons d'abord l'image originale où les contours et les régions sont parfaitement localisées. Ensuite, dans la même image, nous ajoutons le bruit gaussien avec le rapport différent de signal-bruit pour démontrer les efficacités de chaque méthode proposée. Troisièmement, nous comparons le temps de calcul

entre l'algorithme accéléré de DCA et l'algorithme standard de FCM. Tous les tests ont été réalisés sur un ordinateur de Pentium[R] 4 CPU 3.00GHz 1.00Go RAM. A la fin, nous observons les résultats de ces méthodes sur l'image **Blume** qui contient plusieurs régions.
Pour comparer la performance de **Algorithme 4.4** (avec la recherche de point intial par FCM-DCA) selon le temps de calcul, nous appliquons **Algorithme de FCM** et **Algorithme 4.4** sur les mêmes images avec les mêmes paramètres initiaux. Le tableau 4.7 donne le temps de calcul de chaque méthode. Nous utilisons les notations suivantes :
– $Taille$: la taille de l'image.
– c : le nombre de régions d'image.
– $N°F$: le nombre d'itérations dans **Algorithme de FCM**.
– q : le nombre d'itérations de FCM-DCA pour la recherche de point initial de **Algorithme 4.4**.
– $N°D$: le nombre d'itérations de DCA dans la deuxième phase dans **Algorithme 4.4**.
– $Temps$: le temps de calcul de chaque algorithme en second.

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

– Dans plusieurs images, notre algorithme donne une segmentation presque parfaite. En plus, sans information spatiale, nos **Algorithme 4.5** et **Algorithme 4.4** peuvent surmonter la segmentation d'image bruitée dans certains cas.
– Avec l'information spatiale, **Algorithme 4.4** fonctionne bien sur toutes les images bruitées. Il peut supprimer les bruits de manière efficace. Nous pourrons confirmer, dans ce cas, que les deux algorithmes DCA apportent l'image de meilleure qualité par rapport à l'algorithme FCM.
– Dans la plupart de jeux d'essais, **Algorithme 4.5** est plus rapide que **Algorithme de FCM** parce que le calcul de projection des points sur les boules Euclidiennes est explicite. De plus, **Algorithme 4.4** permet d'avoir un gain de calcul important si le choix du paramètre q est bon. Le tableau 4.7 indique l'efficacité du choix du paramètre q en temps de calcul.

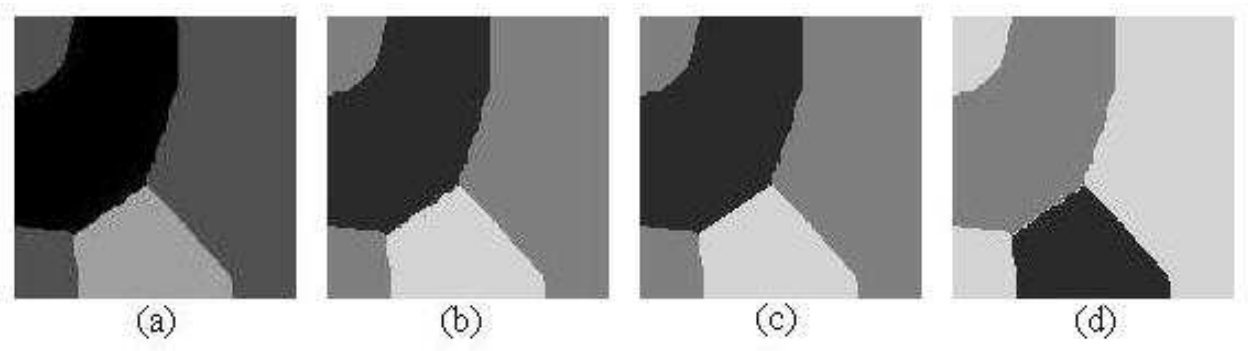

FIG. 4.5 – L'image originale et les résultats de segmentation (c=3).
(a) : L'image originale (b) : Le résultat de **Algorithme de FCM** (c) : Le résultat de **Algorithme 4.5** (d) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA)

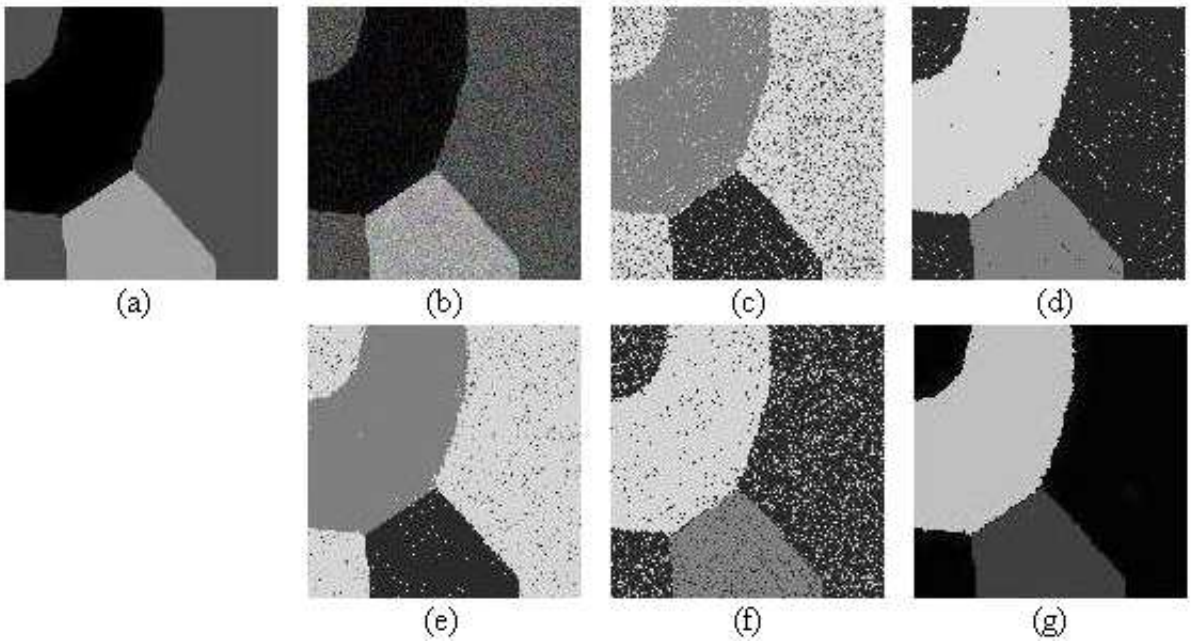

FIG. 4.6 – L'image originale avec le bruit et les résultats de segmentation (c=3).
(a) : L'image originale (b) : L'image originale avec le bruit Gaussien (c) : Le résultat de **Algorithme de FCM** (d) : Le résultat de **Algorithme de FCM** avec l'information spatiale. (e) : Le résultat de **Algorithme 4.5** (f) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA) (g) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA) avec l'information spatiale.

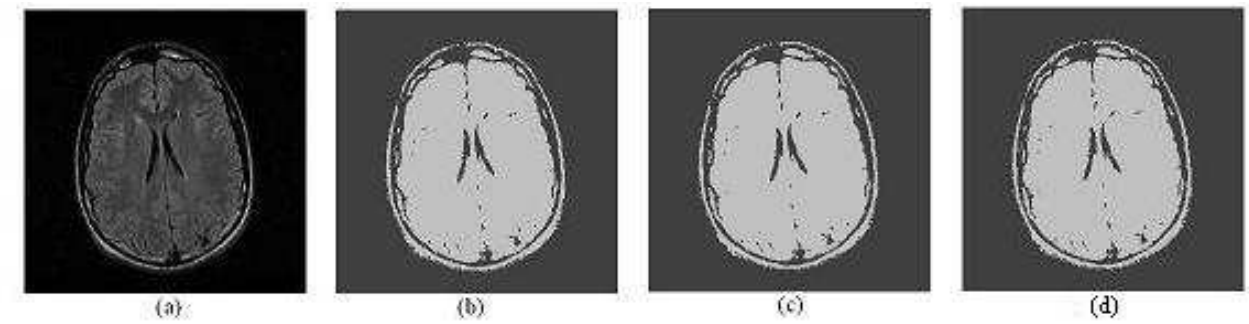

FIG. 4.7 – L'image médical originale et les résultats de segmentation (c=2).
(a) : L'image originale (b) : Le résultat de **Algorithme de FCM** (c) : Le résultat de **Algorithme 4.5** (d) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA)

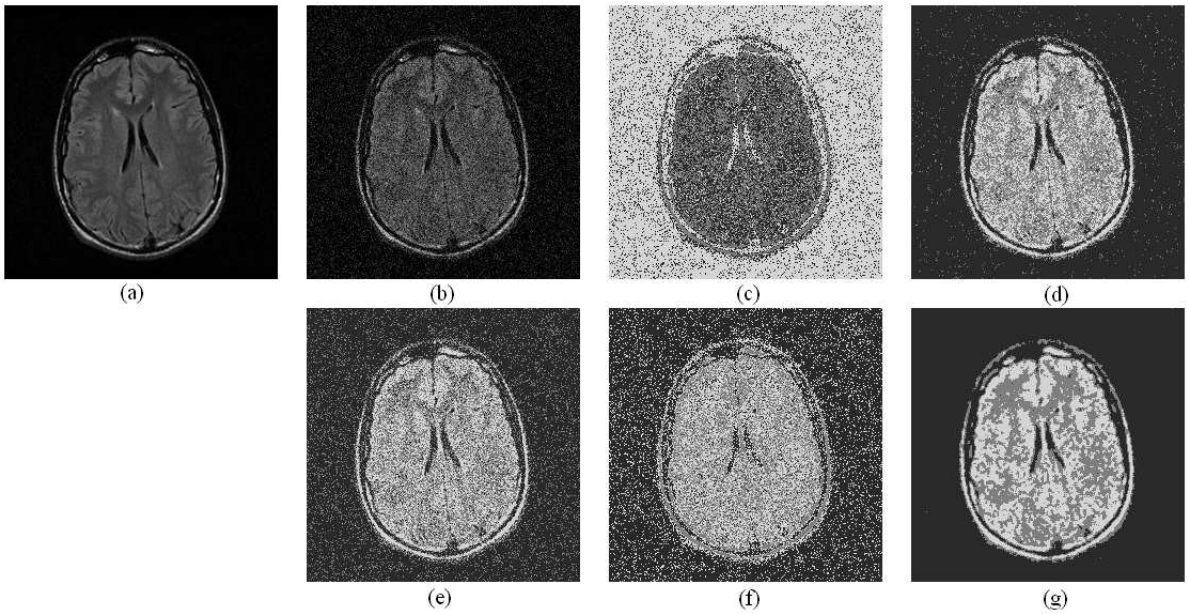

FIG. 4.8 – L'image médicale avec le bruit Gaussien et les résultats de segmentation (c=3).
(a) : L'image originale (b) : L'image originale avec le bruit Gaussien (c) : Le résultat de **Algorithme de FCM** (d) : Le résultat de **Algorithme de FCM** avec l'information spatiale. (e) : Le résultat de **Algorithme 4.5** (f) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA) (g) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA) avec l'information spatiale.

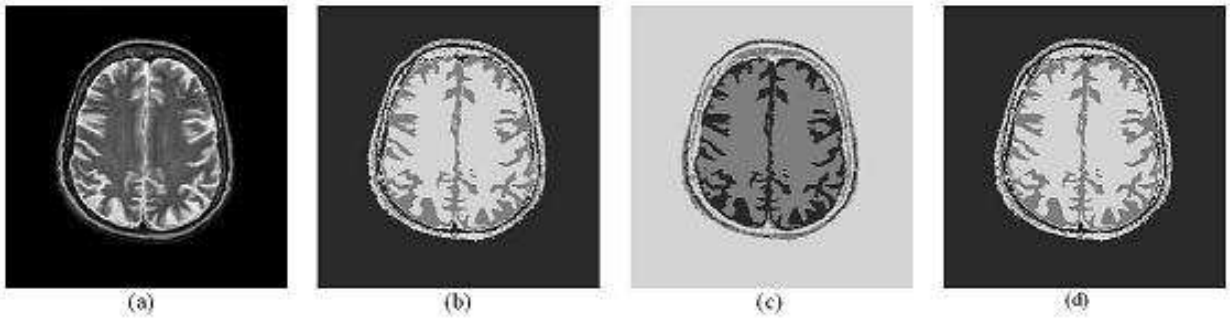

FIG. 4.9 – L'image médical originale et les résultats de segmentation (c=3).
(a) : L'image originale (b) : Le résultat de **Algorithme de FCM** (c) : Le résultat de **Algorithme 4.5** (d) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA)

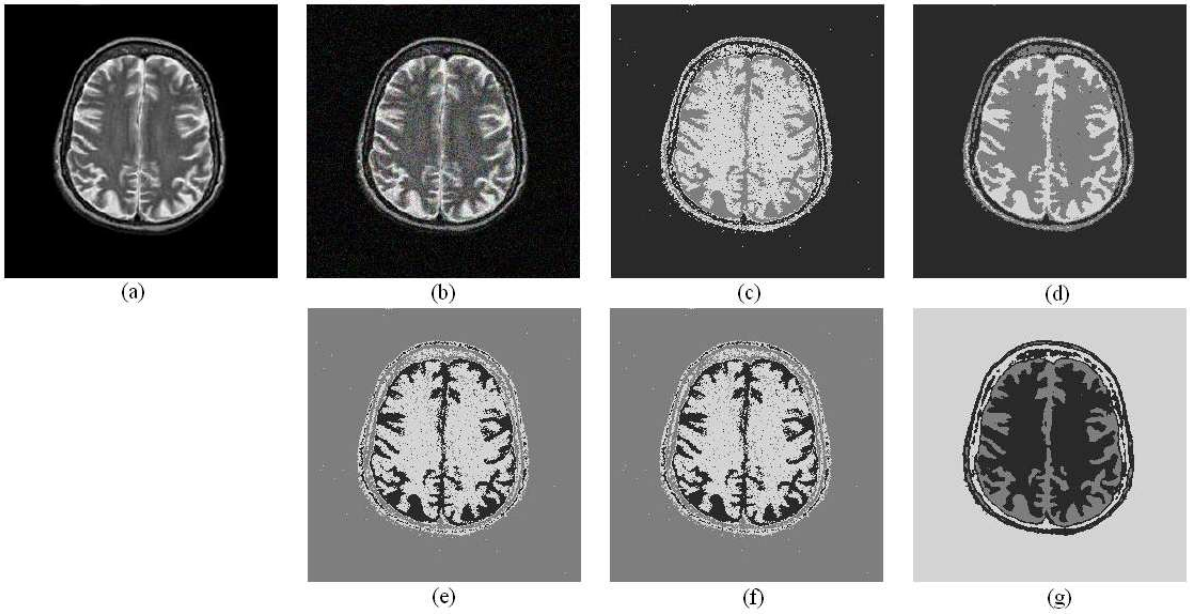

FIG. 4.10 – L'image médicale avec le bruit Gaussien et les résultats de segmentation (c=3).
(a) : L'image originale (b) : L'image originale avec le bruit Gaussien (c) : Le résultat de **Algorithme de FCM** (d) : Le résultat de **Algorithme de FCM** avec l'information spatiale. (e) : Le résultat de **Algorithme 4.5** (f) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA) (g) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA) avec l'information spatiale.

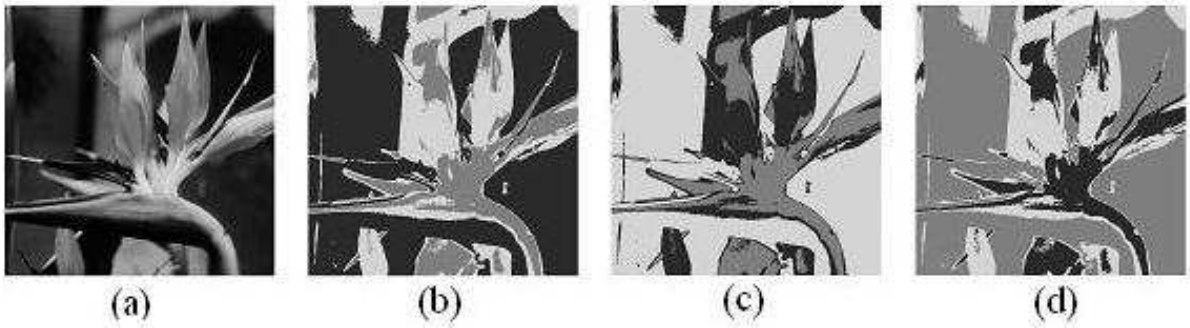

FIG. 4.11 – L'image **Blume** et les résultats de segmentation (c=3).
(a) : L'image originale (b) : Le résultat de **Algorithme de FCM** (c) : Le résultat de **Algorithme 4.5** (d) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA)

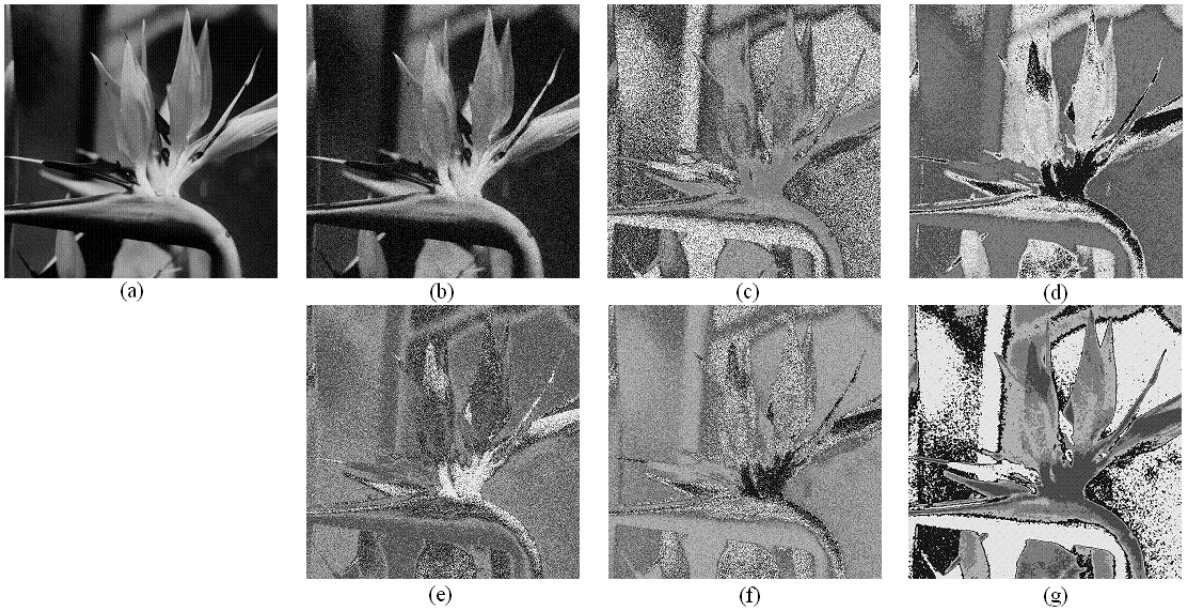

FIG. 4.12 – L'image médical avec le bruit Gaussien et les résultats de segmentation (c=5).
(a) : L'image originale (b) : L'image originale avec le bruit Gaussien (c) : Le résultat de **Algorithme de FCM** (d) : Le résultat de **Algorithme de FCM** avec l'information spatiale. (e) : Le résultat de **Algorithme 4.5** (f) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA) (g) : Le résultat de **Algorithme 4.4** (avec la recherche de point initial par FCM-DCA) avec l'information spatiale.

Data			**FCM**		**Algorithm 4.5**		**Algorithm 4.4**		
$N°$	Size	c	N^oF	Time	N^oI	Time	q	N^oD	Time
1	128^2	2	24	1.453	16	1.312	12	10	1.219
2	128^2	2	17	1.003	12	0.985	10	2	0.765
3	256^2	3	36	15.340	24	13.297	20	2	10.176
4	256^2	3	75	31.281	57	30.843	30	12	26.915
5	256^2	3	39	15.750	27	14.687	20	14	13.125
6	256^2	5	91	84.969	75	86.969	40	78	61.500
7	256^2	3	73	31.094	62	34.286	15	21	24.188
8	256^2	3	78	34.512	52	32.162	20	13	29.182
9	512^2	3	49	92.076	41	102.589	30	46	74.586
10	512^2	5	246	915.095	196	897.043	120	86	691.854

TAB. 4.7 – Comparaison sur le temps de calcul de **Algorithme FCM** et **Algorithme 4.5, 4.4**.

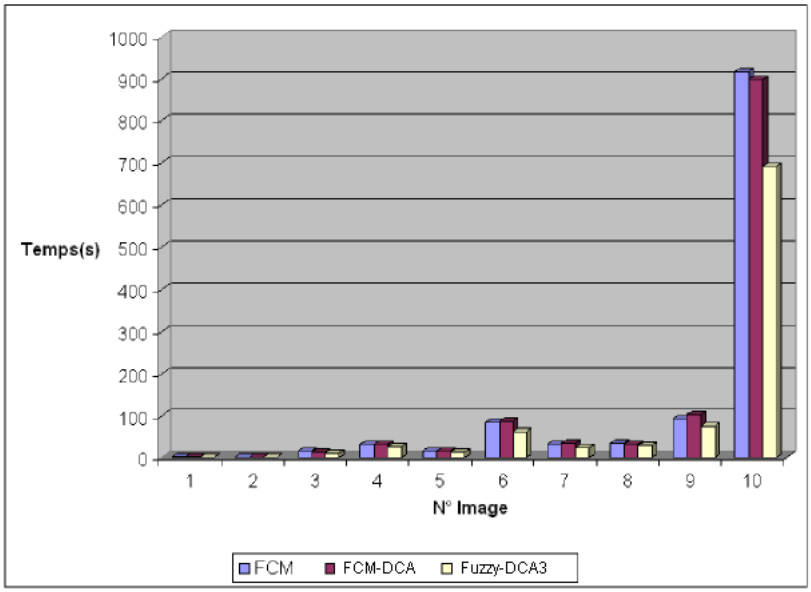

FIG. 4.13 – Comparaison sur le temps de calcul de **Algorithme FCM** et **Algorithme 4.5, 4.4**.

Conclusion. Dans ce chapitre, nous avons étudié trois formulations pour le modèle FCM en clustering flou. Nous avons montré, via ces trois différentes formulations, l'importance du choix de décomposition DC du problème afin de rendre le calcul plus simple et moins coûteux. La deuxième et la troisième formulation sont intéressantes car elles requièrent seulement des projections d'un points sur un simplexe/une boule et ces projections sont déterminées explicitement. En particulier, le **Fuzzy-DCA-3** est très simple et rapide : le gain de temps de calcul monte jusqu'à 62 fois (voire tableau 4.2, $m = 1.3$). L'effet du point initial a été également étudié. En effet, par une procédure alternative FCM-DCA, nous trouvons un "bon" point initial pour améliorer le résultat de nos algorithmes. Les expériences numériques sur différents jeux de données (les données biopuces, le problème de segmentation d'image, ...) ont montré l'efficacité de nos algorithmes sur les deux plans :la qualité de solution et le temps de calculs.

Chapitre 5

Clustering via la programmation DC pour la détermination d'arbre hiérarchique de multidiffusion

Résumé *Ce chapitre est consacré à l'étude des nouvelles méthodes de clustering hiérarchique à deux niveaux basées sur la programmation DC et DCA. Plusieurs modèles d'optimisation (non convexe, non différentiables) sont proposés, ce qui donne naissance à différents DC programmes. Les schémas DCA correspondants sont ainsi développés. Les expériences numériques comparatives avec des algorithmes d'optimisation existants prouvent la robustesse, la performance et la supériorité de nos algorithmes.*

5.1 Introduction

Le clustering hiérarchique à multiniveaux consiste au regroupement des objets de données dans une hiérarchie des clusters. Une des applications les plus importantes du clustering hiérarchique est le problème d'identification de la topologie d'un réseau.

Citons, par exemple, la construction d'un réseau de communication multicast. Son but principal est de fournir un service de communication entre les participants du groupe tout en assurant une bonne qualité de service et minimisant le coût total de communication. Les applications de la communication multidiffusion sont nombreuses : Systèmes de Groupware pour faciliter la conception et l'évaluation de collaboration, Vidéo conférence, Diffusion en temps réel des événements internationaux, Peer-to-peer applications pour le partage des données ou traitement, etc.
Un arbre hiérarchique de multidiffusion contient une source (le centre total) et plusieur niveaux de hiérarchie. Un noeud est relié à un noeud de niveau hiérarchique supérieur (sauf la source) et ses noeuds de niveau hiérarchique inférieur (s'il en existe). La communication multidiffusion fonctionne de façon suivante : le noeud central total envoie le message

aux autres centres de chaque sous-ensemble (centres de niveau 1), et à son tour, chaque centre retransmet le message reçu aux autres noeuds de niveau hiérarchique inférieur dans son sous-ensemble. Ce processus se répète jusqu'à ce que le message arrive aux noeuds termimaux (les noeuds les plus bas dans l'arbre hiérarchique). La multidiffusion (multicast) offre à la communication de groupe un gain considérable d'efficacité, en particulier pour des grands groupes. Les hiérarchies aident à réduire la complexité, le nombre de messages échangés entre les participants. Une des approches est d'utiliser les algorithmes de clustering pour déterminer l'arbre hiérarchique de multicast. Les algorithmes clustering aident à diviser des populations d'utilisateur selon une variété de critères.

Les méthodes standard de clustering hiérarchique sont souvent basées sur les algorithmes de clustering par partitionnement, couplage avec des stratégies itératives de contrôle pour re-ordonner, re-clustering dans le but de trouver le meilleur arbre hiérarchique. A notre connaissance, tandis qu'il existe plusieurs modèles d'optimisation pour le clustering par partitionnement, il y en a peu pour le clustering hiérarchique. La seule approche d'optimisation déterministe que nous trouvons dans la littérature est celle de L. Jia, A. Bagirov, I. Ouveysi et A.M. Rubinov ([244]) dont le but est de construire un arbre hiérarchique d'un réseau de multicast de deux niveaux. Ces auteurs ont proposé deux modèles d'optimisation non convexes non différentiables en choisissant la norme euclidienne comme distance et utilisé une méthode de la programmation non convexe appelée gradient discrète pour les résoudre.

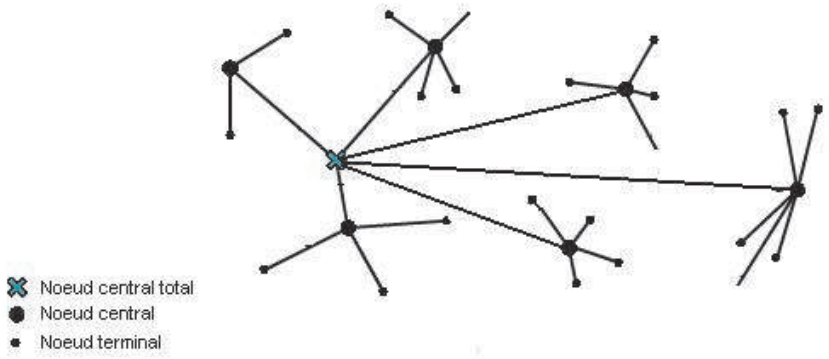

Fig. 5.1 – L'arbre hiérarchique de 2 niveaux

Notre travail a pour l'objectif de résoudre le même problème : déterminer d'un arbre hiérarchique de deux niveaux dont le coût total est minimal. Le problème peut être décrit comme suivant. Etant donné un ensemble $\mathcal{A}$ de p objets $\mathcal{A} := \{a_j \in \mathbb{R}^n : j = 1, ..., p\}$, et un entier k. Le but est de construire un arbre hiérarchique de deux niveau en minimisant son coût, le premier niveau (niveau 0) contient un seul objet appelé le centre total (la racine de l'arbre), et le deuxième (niveau 1) contenant les autres objets est un partition de **k** clus-

ters disjoints. Chaque cluster $\mathcal{A}_i$ est représenté par son centre étant parmi les points qui le constituent. Le coût total de cet arbre est la somme des coûts de k clusters et des coûts de connexions entre le centre total et les centres du niveau 1.

Le problème revient donc à choisir $k+1$ objets dans $\mathcal{A}$ dont un pour être le centre total et k autres pour être centres de k disjoints clusters, et déterminer ces clusters. Le centre total est défini comme étant l'objet le plus proche aux k centres (e.g la somme de distance du centre total aux k autres centres est minimum). Puisque chaque cluster est représenté par un de ses objets, il s'agit d'un problème d'optimisation discrète. Si on supprime le niveau 0 (centre total) on retrouve le problème de k-médian dans le chapitre 3. Il est clair que la présence du centre total rend le problème beaucoup plus difficile.

Nous proposons dans ce travail trois modèles d'optimisation pour ce problème. Les deux premiers sont inspirés par les travaux dans [244] sur la façons d'introduire le centre total dans la formulation, mais nous choisissons le carré de la norme euclidienne comme distance dans le but d'obtenir des schémas DCA plus simples. Les variables dans ces modèles sont les centres du niveau 1 et le centre total est calculé en fonction de ces derniers une fois qu'ils sont déterminés. La troisième formulation, une approche complètement différente des autres, considère tous les centres cherchés comme variables (on cherche simultanément le centre total (niveau 0) et les centres du niveau 1). Cette dernière approche nous semble la meilleure. Tous les trois problèmes sont transformés en premier lieu, via la pénalité, en des problèmes d'optimisation continue non convexes non différentiables que nous reformulons ensuite sous la forme des programmes DC. Plus précisément, chaque formulation DC nous ramène à la minimisation de la différence d'une fonction convexe quadratique simple et une fonction convexe non différentiable. Il s'avère que les DCA appliqués a ces problèmes sont explicites est non coûteux. Les résultats numériques prouvent l'efficacité de DCA par rapport aux méthodes existantes.

Le reste du chapitre est organisé de la façon suivante. Dans la deuxième section, nous présentons différents modèles d'optimisation du problème. La troisième partie décrit notre méthode de résolution par DCA et une procédure pour la recherche d'un point initial en combinant alternativement DCA - k-means. Finalement les résultats numériques sont reportés dans la dernière partie.

5.2 Formulation du problème

Notons par $\{x_1, \ldots, x_k\}$ les centres du niveau 1. Une fois connus ces centres, les clusters sont déterminés selon la formule suivante :

$$\mathcal{A}_i = \{a \in \mathcal{A} : \|x_i - a\|^2 < \min_{j=1,\ldots,k, i\neq j} \|x_j - a\|^2\}.$$

Le centre total, noté x_{k+1}, est défini par

$$x_{k+1} = \arg\min_{j=1,\dots,p} \sum_{i=1}^{k} \|x_i - a_j\|^2. \tag{5.1}$$

Dans les deux premières approches on cherche les centres du niveau 1 puis en déduit le centre total en utilisant (5.1).

5.2.1 Première approche : un centre total artificiel

Dans cette approche on considère un centre total *artificiel*, noté x^*, qui est le centre de gravité des centres du niveau 1 :

$$x^* = \frac{1}{k} \sum_{i=1}^{k} x_i. \tag{5.2}$$

On introduit alors le coût de connexions entre le centre artificiel et les autres centres dans la fonction objectif. Le premier modèle d'optimisation s'écrit comme :

$$(P1) \quad \begin{cases} \min \left(\frac{1}{2} \sum_{j=1}^{p} \min_{i=1,\dots,k} \|x_i - a_j\|^2 + \frac{1}{2} \sum_{i=1}^{k} \|x_i - \frac{1}{k} \sum_{l=1}^{k} x_l\|^2 \right) \\ \text{tel que :} \\ \quad \sum_{i=1}^{k} \min_{j=1,\dots,k} \|x_i - a_j\|^2 = 0. \end{cases}$$

Le premier terme de la fonction objectif est le coût du clustering au niveau 1 et le second est le coût de connexion entre le *centre total* et les autres centres (pour simplifier les calculs dans nos algorithmes nous ajoutons le coefficient $\frac{1}{2}$ devant chaque terme). La contrainte

$$\sum_{i=1}^{k} \min_{j=1,\dots,k} \|x_i - a_j\|^2 = 0 \tag{5.3}$$

impose que les centres du niveau 1 doivent appartenir à l'ensemble $\mathcal{A}$.

C'est un problème d'optimisation non convexe et non différentiable très difficile à traiter efficacement par les approches standard. Il est doublement non convexe par sa fonction objectif et sa contrainte. Une des techniques pour surmonter cette double difficulté est de pénaliser la contrainte. Ce qui donne naissance au problème suivant ($\tau > 0$ est le paramètre de pénalité) :

$$(P1_\tau) \quad \begin{cases} \min \left(\frac{1}{2} \sum_{j=1}^{p} \min_{i=1,\dots,k} \|x_i - a_j\|^2 + \frac{1}{2} \sum_{i=1}^{k} \|x_i - \frac{1}{k} \sum_{l=1}^{k} x_l\|^2 + \frac{\tau}{2} \sum_{i=1}^{k} \min_{j=1,\dots,p} \|x_i - a_j\|^2 \right) \\ \text{tel que :} \\ \quad x_i \in \mathbb{R}^n, i = 1, \dots k. \end{cases}$$

Dans le cas où une solution optimale de ce dernier problème ne vérifie pas la contrainte (5.3), on détermine, par la façon la plus naturelle, les centres *réels* en prenant les noeuds dans $\mathcal{A}$ qui leur sont les plus proches.

Nous utilisons cette même technique de pénalité pour les approches présentées ci-après.

5.2.2 Deuxième approche : le centre total réel

Dans cette approche, on introduit le coût de connexions entre le centre total *réel* et les autres centres dans la fonction objectif. Le deuxième modèle d'optimisation est le suivant :

$$(P2) \quad \begin{cases} \min \left(\tilde{C} := \frac{1}{2} \sum_{j=1}^{p} \min_{i=1,\dots,k} \|x_i - a_j\|^2 + \min_{j=1,\dots,p} \sum_{i=1}^{k} \|x_i - a_j\|^2 \right) \\ \text{tel que :} \\ \quad \sum_{i=1}^{k} \min_{j=1,\dots,k} \|x_i - a_j\|^2 = 0 \end{cases}$$

qui devient, après la pénalité la contrainte et l'ajout du coefficient de pondération $\tau_2 > 0$ au second terme de $\tilde{C}$ $(\tau_1, \tau_2 > 0)$:

$$(P2_\tau) \quad \begin{cases} \min \left(\frac{1}{2} \sum_{j=1}^{p} \min_{i=1,\dots,k} \|x_i - a_j\|^2 + \frac{\tau_1}{2} \sum_{i=1}^{k} \min_{j=1,\dots,p} \|x_i - a_j\|^2 + \frac{\tau_2}{2} \min_{j=1,\dots,p} \sum_{i=1}^{k} \|x_i - a_j\|^2 \right) \\ \text{tel que :} \\ \quad x_i \in \mathbb{R}^n, i = 1, \dots k. \end{cases}$$

5.2.3 La troisième approche : chercher simultanément tous les centres

L'inconvénient des approches précédentes est de ne pas chercher simultanément le centre total et les autres centres. Même si on trouvait une solution optimale exacte de $(P1)$ et $(P2)$ puis en déduisant le centre total selon (5.1), on n'obtiendrait pas forcément un arbre hiérarchique optimal. Dans la troisième approche, nous considérons tous les centres comme variables, i.e., nous cherchons simultanément le centres total et les autres centres. Le problème s'écrit comme suit :

$$(P3) \quad \begin{cases} \min \sum_{j=1}^{p} \min_{i=1,\dots,k} \|x_i - a_j\|^2 + \sum_{i=1}^{k} \|x_{k+1} - x_i\|^2 \\ \text{tel que :} \\ \quad \sum_{i=1}^{k+1} \min_{j=1,\dots,p} \|x_i - a_j\|^2 = 0. \end{cases}$$

Ce modèle tient compte du coût de connexion entre le centre total et les autres centres et assure que tous les centres soient dans $\mathcal{A}$. Il est intéressant de constater que cette formulation n'est pas plus "compliquée" que les deux précédentes ! Sur le plan algorithmique, on verra dans la suite que le troisième schéma de DCA est le plus efficace, tant sur la qualité de solution que le temps de calcul.
En pénalisant la contrainte, on obtient ($\tau > 0$ est le paramètre de pénalité) :

$$(P3_\tau) \qquad \begin{cases} \min \left(\frac{1}{2} \sum_{j=1}^{p} \min_{i=1,\ldots,k} \|x_i - a_j\|^2 + \frac{1}{2} \sum_{i=1}^{k} \|x_{k+1} - x_i\|^2 + \frac{\tau}{2} \sum_{i=1}^{k+1} \min_{j=1\ldots p} \|x_i - a_j\|^2 \right) \\ \text{tel que :} \\ \quad x_i \in \mathbb{R}^n, i = 1, \ldots k. \end{cases}$$

Nous allons montrer que ces trois problèmes (P1$_\tau$), (P2$_\tau$) et (P3$_\tau$) peuvent être reformulés sous la forme des programmes DC et développer les schémas de DCA pour leur résolution.

5.3 Résolution de (P1$_\tau$), (P2$_\tau$) et (P3$_\tau$) par DCA

Dans toute la suite nous allons travailler dans un espace matriciel, ce qui est plus commode pour les calculs. La variable sera une matrice X de type $m \times n$ dont la $i^{\text{ème}}$ ligne X_i est égale à x_i (m est égale à k dans les problèmes (P1$_\tau$) et (P2$_\tau$), et à $k+1$ dans (P3$_\tau$)). L'espace de travail est donc l'espace $\mathcal{M}_{m,n}(\mathbb{R})$ des matrices à coefficients réels ayant m lignes et n colonnes. Sa structure hilbertienne est définie à l'aide du produit scalaire usuel

$$\begin{aligned} \mathcal{M}_{m,n}(\mathbb{R}) &\ni X \longleftrightarrow (X_1, X_2, \ldots, X_m) \in (\mathbb{R}^n)^m,\ X_i :\in \mathbb{R}^n, (i = 1, .., m), \\ \langle X, Y \rangle &:= Tr(X^T Y) = \sum_{i=1}^{m} \langle X_i, Y_i \rangle = \sum_{i=1}^{m} \|X_i\|^2 \end{aligned}$$

et sa norme euclidienne

$$\|X\|^2 := \sum_{i=1}^{m} \langle X_i, X_i \rangle = \sum_{i=1}^{m} \|X_i\|^2 .$$

5.3.1 Résoudre (P1$_\tau$) par DCA

5.3.1.1 Décomposition DC de (P1$_\tau$)

A l'aide de la propriété ([2], [45])

$$\min_{i=1,\ldots,k} \|x_i - a_j\|^2 := \sum_{i=1}^{k} \|x_i - a_j\|^2 - \max_{i=1,\ldots,k} \sum_{r=1, r\neq i}^{k} \|x_r - a_j\|^2, \tag{5.4}$$

la fonction objectif de (P1$_\tau$) peut s'écrire comme

$$\begin{aligned} F_{1\tau}(X) = \; & \tfrac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{k}\|X_i - a_j\|^2 - \tfrac{1}{2}\sum_{j=1}^{p}\max_{i=1,\ldots,k}\sum_{r=1,r\neq i}^{k}\|X_r - a_j\|^2 \\ & + \tfrac{1}{2}\sum_{i=1}^{k}\|X_i - \tfrac{1}{k}\sum_{l=1}^{k}X_l\|^2 + \tfrac{\tau}{2}\sum_{j=1}^{p}\sum_{i=1}^{k}\|X_i - a_j\|^2 - \tfrac{\tau}{2}\sum_{j=1}^{p}\max_{j=1,\ldots,p}\sum_{s=1,s\neq j}^{p}\|X_i - a_s\|^2 \end{aligned}$$

qui est une fonction DC avec la décomposition suivante :

$$\begin{aligned} F_{1\tau}(X) = \; & G_{1\tau}(X) - H_{1\tau}(X), \\ G_{1\tau}(X) := \; & \tfrac{(\tau+1)}{2}\sum_{j=1}^{p}\sum_{i=1}^{k}\|X_i - a_j\|^2 + \tfrac{1}{2}\sum_{i=1}^{k}\|X_i - \tfrac{1}{k}\sum_{l=1}^{k}X_l\|^2, \\ H_{1\tau}(X) := \; & \tfrac{\tau}{2}\sum_{i=1}^{k}\max_{j=1,\ldots,p}\sum_{s=1,s\neq j}^{p}\|X_i - a_s\|^2 + \sum_{j=1}^{p}\max_{i=1,\ldots,k}\sum_{r=1,r\neq i}^{k}\tfrac{1}{2}\|X_r - a_j\|^2. \end{aligned} \tag{5.5}$$

(P1$_\tau$) est donc un programme DC dans l'espace matriciel $\mathbb{R}^{k\times n}$:

$$(\mathrm{P1}_\tau) \qquad \min\left\{G_{1\tau}(X) - H_{1\tau}(X) : X \in \mathbb{R}^{k\times n}\right\}.$$

La résolution de (P1$_\tau$) par DCA consiste en la détermination de deux suites $\{X^{(l)}\}$ and $\{Y^{(l)}\}$ dans $\mathbb{R}^{k\times n}$ telles que :

$$\begin{aligned} Y^{(l)} \; & \in \; \partial H_{1\tau}(X^{(l)}) \\ X^{(l+1)} \; & \in \; \partial G^*_{1\tau}(Y^{(l)}). \end{aligned}$$

5.3.1.2 Calculer $\partial H_{1\tau}(X)$

On a

$$\begin{aligned} H_{1\tau}(X) \; & := \tfrac{\tau}{2}\sum_{i=1}^{k}\max_{j=1,\ldots,p}\sum_{r=1,r\neq j}^{p}\|X_i - a_r\|^2 + \sum_{j=1}^{p}\max_{i=1,\ldots,k}\sum_{r=1,r\neq i}^{k}\tfrac{1}{2}\|X_r - a_j\|^2 \\ & := \tau H_1(X) + H_2(X), \end{aligned}$$

où

$$H_1(X) = \sum_{i=1}^{k}\max_{j=1,\ldots,p}\sum_{r=1,r\neq j}^{p}\|X_i - a_r\|^2, \tag{5.6}$$

$$H_2(X) = \sum_{j=1}^{p}\max_{i=1,\ldots,k}\sum_{r=1,r\neq i}^{k}\frac{1}{2}\|X_r - a_j\|^2. \tag{5.7}$$

Les règles usuelles de calcul de sous-différentiels de fonctions convexes ([243], [246]) permettent d'obtenir les résultats suivants

$$\partial H_{1\tau}(X) = \tau \partial H_1(X) + \partial H_2(X), \tag{5.8}$$

$\partial H_1(X)$ et $\partial H_2(X)$ peuvent être explicitement déterminés par (co signifie l'enveloppe convexe) :

$$\partial H_1(X) = \sum_{i=1}^{k} \partial h_i^1(X),\ \partial h_i^1(X) = co\{\partial h_{i,j}^1(X) : h_{i,j}^1(X) = h_{i,j}^1(X)\}, \tag{5.9}$$

où

$$h_{i,j}^1(X) \ : \ = \frac{1}{2} \sum_{r=1, r\neq j}^{p} \|X_i - a_r\|^2 \ , \ h_i^1(X) := \max_{j=1,\ldots,p} h_{i,j}^1(X). \tag{5.10}$$

Les fonctions $h_{i,j}^1$ sont différentiables et

$$[\nabla h_{i,j}^1(X)]_\ell = (p-1)X_i - \sum_{r=1, r\neq j}^{p} a_r \text{ si } \ell = \text{i},\ 0 \text{ sinon.} \tag{5.11}$$

De manière analogue, soient

$$h_{j,i}^2(X) := \frac{1}{2} \sum_{s=1, s\neq i}^{k} \|X_s - a_j\|^2 \text{ et } h_j^2(X) := \max_{i=1,\ldots,k} h_{j,i}^2(X), \tag{5.12}$$

alors on a

$$\partial h_j^2(X) = co\{\partial h_{j,i}^2(X) : h_{j,i}^2(X) = h_{j,i}^2(X)\} \text{ et } \partial H_2(X) = \sum_{j=1}^{p} \partial h_j^2(X). \tag{5.13}$$

Comme dans (5.11), les fonctions $h_{j,i}^2$ sont différentiable et de dérivée :

$$[\nabla h_{j,i}^2(X)]_l = X_l - a_j \text{ si } l \neq i,\ 0 \text{ sinon.} \tag{5.14}$$

ce qui détermine explicitement $\partial H_2(X)$.

5.3.1.3 Calculer $\partial G_{1\tau}^*(X)$

D'après (5.5), on a

$$G_{1\tau}(X) := \frac{(\tau+1)}{2} G_1(X) + \frac{1}{2} G_2(X),$$

où

$$G_1(X) := \sum_{j=1}^{p} \sum_{i=1}^{k} \|X_i - a_j\|^2 \text{ et } G_2(X) := \sum_{i=1}^{k} \|X_i - \frac{1}{k} \sum_{l=1}^{k} X_l\|^2 .$$

Soit $A^{(j)} \in \mathbb{R}^{k\times n}$ une matrice dont toutes les lignes sont égales à a_j

$$[A^{(j)}]_i := a_j, i = 1, ..., k \tag{5.15}$$

et T une matrice de taille $k \times k$ définie par

$$T := \frac{1}{k} ee^T \text{ (avec } T = T^2), \tag{5.16}$$

où $e \in \mathbb{R}^k$ est le vecteur dont les composantes sont égales à 1. On peut alors écrire

$$G_1(X) = \sum_{j=1}^{p} \left\| X - A^{(j)} \right\|^2 \tag{5.17}$$

et

$$G_2(X) = \|X - TX\|^2 = \|(I - T)X\|^2 . \tag{5.18}$$

$G_{1\tau}$ est alors une fonction quadratique définie positive et de dérivée

$$\nabla G_{1\tau}(X) = [(1 + (\tau + 1)p)I - T]X - A \tag{5.19}$$

avec

$$A := \sum_{j=1}^{p} A^{(j)}, \quad \text{i.e., } \; A_i = \sum_{j=1}^{p} a_j \;\; \text{for } i = 1, ..., k. \tag{5.20}$$

Par conséquent, $G^*_{1\tau}$ est également une fonction quadratique définie positive et

$$X = \nabla G^*_{1\tau}(Y) \text{ ssi } Y = \nabla G_{1\tau}(X).$$

Le calcul de $\nabla G^*_{1\tau}(Y)$ nous ramène au calcul de la matrice inverse de $(1 + (\tau + 1)p)I - T$. Cette matrice possède seulement deux valeurs propres $(\tau+1)p$ et $1+(\tau+1)p$ correspondant aux deux vecteurs propres e et $V := \{v \in \mathbb{R}^k : e^T v = 0\}$. Plus précisément :

$$\begin{aligned} [(1 + (\tau + 1)p)I - T]\, e &= (\tau + 1)pe \\ [(1 + (\tau + 1)p)I - T]\, v &= (1 + (\tau + 1)p)v, \; \forall v \in V. \end{aligned} \tag{5.21}$$

Alors la matrice inverse de $[(1 + (\tau + 1)p)I - T]$ a également deux valeurs propres $1/((\tau+1)p)$ et $1/(1 + (\tau + 1)p)$ correspondant aux deux vecteurs propres e et V. On a :

$$[(1 + (\tau + 1)p)I - T]^{-1} = \frac{1}{1 + (\tau + 1)p} I + \lambda T,$$

où λ est calculé par

$$[\frac{1}{1 + (\tau + 1)p} I + \lambda T]e = \frac{1}{(\tau + 1)p} e.$$

Par conséquent

$$\frac{1}{1 + (\tau + 1)p} + \lambda = \frac{1}{(\tau + 1)p},$$

et

$$\lambda = \frac{1}{(\tau+1)p} - \frac{1}{1+(\tau+1)p} = \frac{1}{[1+(\tau+1)p](\tau+1)p}.$$

Alors

$$[(1+(\tau+1)p)I - T]^{-1} = \frac{1}{1+(\tau+1)p} I + \frac{1}{[1+(\tau+1)p](\tau+1)p} T,$$

et

$$X = \nabla G^*_{1\tau}(Y) = [\frac{1}{1+(\tau+1)p} I + \frac{1}{[1+(\tau+1)p](\tau+1)p} T](Y+A). \tag{5.22}$$

5.3.1.4 Le schéma DCA pour résoudre le ($P1_\tau$)

Le schéma DCA pour résoudre le ($P1_\tau$) s'écrit comme suivant :

Algorithme 5.1
CHBN-DCA-1 : DCA appliqué au problème ($P1_\tau$)

Initialisation **:** *Choisir une matrice* $X^{(0)} \in \mathbb{R}^{k\times n}$. *Soit une tolérance* $\epsilon_{DCA} > 0$, $0 \leftarrow l$.

Répéter
- *Calculer* $Y^{(l)} \in \partial H_{1\tau}(X(^{(l)})$ *par les formules (5.8) - (5.14) ;*
- *Calculer* $X^{(l+1)} \in \partial G^*_{1\tau}(Y^{(l)})$ *par (5.22) ;*
- $l+1 \leftarrow l$

Jusqu'à $\left\|X^{(l)} - X^{(l-1)}\right\| \leq \epsilon_{DCA}(\left\|X^{(l)}\right\| + 1)$ ***ou*** $\left|F(X^{(l)}) - F(X^{(l-1)})\right| \leq \epsilon_{DCA}(\left|F(X^{(l)})\right| + 1)$.

5.3.2 Résoudre ($P2_\tau$) par DCA

5.3.2.1 Décomposition DC de ($P2_\tau$)

De même façon que pour $F_{1\tau}$, avec l'aide de la formule (5.4), on peut décomposer la fonction objectif de ($P2_\tau$) sous la forme :

$$\begin{aligned} F_{2\tau}(X) := \; & \sum_{j=1}^{p}\sum_{i=1}^{k} \|X_i - a_j\|^2 - \sum_{j=1}^{p} \max_{i=1,\dots,k} \sum_{r=1,r\neq i}^{k} \|X_r - a_j\|^2 \\ & + \tfrac{\tau_1}{2} \sum_{i=1}^{k}\sum_{j=1}^{p} \|X_i - a_j\|^2 - \tfrac{\tau_1}{2} \sum_{i=1}^{k} \max_{j=1,\dots,p} \sum_{s=1,s\neq j}^{p} \|X_i - a_s\|^2 \\ & + \tfrac{\tau_2}{2} \sum_{j=1}^{p}\sum_{i=1}^{k} \|X_i - a_j\|^2 - \tfrac{\tau_2}{2} \max_{j=1,\dots,p} \sum_{u=1,u\neq j}^{p} \sum_{i=1}^{k} \|X_i - a_u\|^2. \end{aligned}$$

Soit $G_{2\tau}$ et $H_{2\tau}$, deux fonctions définies sur $\mathbb{R}^{k\times n}$

$$\begin{aligned} G_{2\tau}(X) = &\quad (\tau_1+\tau_2+1)\sum_{j=1}^{p}\sum_{i=1}^{k}\tfrac{1}{2}\|X_i-a_j\|^2, \\ H_{2\tau}(X) := &\sum_{j=1}^{p}\max_{i=1,\ldots,k}\sum_{r=1,r\neq i}^{k}\tfrac{1}{2}\|X_r-a_j\|^2+\tau_1\sum_{i=1}^{k}\max_{j=1,\ldots,p}\sum_{s=1,s\neq j}^{p}\tfrac{1}{2}\|X_i-a_s\|^2 \\ &+\tau_2\max_{j=1,\ldots,p}\sum_{u=1,u\neq j}^{p}\sum_{i=1}^{k}\tfrac{1}{2}\|X_i-a_u\|^2. \end{aligned} \tag{5.23}$$

Alors

$$F_{2\tau}(X) = G_{2\tau}(X) - H_{2\tau}(X).$$

Il est facile de vérifier que les deux fonctions $G_{2\tau}$ et $H_{2\tau}$ sont convexes. Par conséquent, ($P2_\tau$) est un programme DC

$$(P2_\tau) \qquad \min\left\{G_{2\tau}(X) - H_{2\tau}(X) : X \in \mathbb{R}^{k\times n}\right\}.$$

5.3.2.2 Calculer $\partial H_{2\tau}(X)$

Par la définition, on a

$$H_{2\tau} = \tau_1 H_1(X) + H_2(X) + \tau_2 H_3(X), \tag{5.24}$$

où $H_1(X)$ et $H_2(X)$ sont définies respectivement par (5.6) et (5.7), et

$$H_3(X) := \max_{j=1,\ldots,p}\sum_{u=1,u\neq j}^{p}\sum_{i=1}^{k}\frac{1}{2}\|X_i-a_u\|^2. \tag{5.25}$$

Par suite

$$\partial H_{2\tau}(X) = \tau_1\partial H_1(X) + \partial H_2(X) + \tau_2\partial H_3(X). \tag{5.26}$$

On a déjà montré que l'on peut calculer explicitement $\partial H_1(X)$ et $\partial H_2(X)$. De même façon, on a pour $\partial H_3(X)$:

$$\partial H_3(X) = co\{\partial h_j^3(X) : h_j^3(X) = H_3(X)\} \text{ avec } h_j^3(X) = \sum_{u=1,u\neq j}^{p}\sum_{i=1}^{k}\frac{1}{2}\|X_i-a_u\|^2. \tag{5.27}$$

Les fonctions h_j^3 sont différentiables et de dérivée

$$[\nabla h_j^3(X)]_l = (p-1)X_l - \sum_{u=1,u\neq j}^{p} a_u. \tag{5.28}$$

Par conséquent $\partial H_3(X)$ est également explicitement déterminé.

5.3.2.3 Calculer $\partial G^*_{2\tau}(X)$

A partir de (5.23), on a

$$G_{2\tau}(X) = (\tau_1 + \tau_2 + 1)\sum_{j=1}^{p} \frac{1}{2}\left\|X - A^{(j)}\right\|^2, \tag{5.29}$$

où $A^{(j)} \in \mathbb{R}^{k\times n}$ est déjà define par (5.15). La dérivée de $G_{2\tau}$ est donc donnée par :

$$\nabla G_{2\tau}(X) = (\tau_1 + \tau_2 + 1)\sum_{j=1}^{p}(X - A^{(j)}) = (\tau_1 + \tau_2 + 1)(pX - A) \tag{5.30}$$

avec A définie par (5.20). Puisque

$$X = \nabla G^*_{2\tau}(Y) \text{ ssi } Y = \nabla G_{2\tau}(X),$$

on obtient finalement

$$\nabla G^*_{2\tau}(Y) = \frac{1}{(\tau_1 + \tau_2 + 1)p}Y + \frac{1}{p}A. \tag{5.31}$$

5.3.2.4 Le schéma DCA pour résoudre le ($\mathbf{P2_\tau}$)

Le schéma DCA pour résoudre le (P2$_\tau$) s'écrit comme suivant :

Algorithme 5.2
CHBN-DCA-2 : DCA appliqué au problème (P2$_\tau$)

Initialisation **:** *Choisir une matrice $X^{(0)} \in \mathbb{R}^{k\times n}$. Soit une tolérance $\epsilon_{DCA} > 0$, $0 \leftarrow l$.*

Répéter
– *Calculer $Y^{(l)} \in \partial H_{2\tau}(X^{(l)})$ par les formules (5.8) - (5.14), (5.26) - (5.28) ;*
– *Calculer $X^{(l+1)} \in \partial G^*_{2\tau}(Y^{(l)})$ par (5.31) ;*
– $l + 1 \leftarrow l$
Jusqu'à $\left\|X^{(l)} - X^{(l-1)}\right\| \leq \epsilon_{DCA}(\left\|X^{(l)}\right\| + 1)$ ***ou*** $\left|F(X^{(l)}) - F(X^{(l-1)})\right| \leq \epsilon_{DCA}(\left|F(X^{(l)})\right| + 1)$.

5.3.3 Résoudre (P3$_\tau$) par DCA

5.3.3.1 Décomposition DC de (P3$_\tau$)

De même façon, nous reformulons le problème (P3$_\tau$) avec variable X comme matrice de type $(k+1)\times n$ dont la $i^{\text{ème}}$ ligne X_i est égale à x_i. Selon la propriété (5.4), la fonction objectif

de (P3$_\tau$) peut s'écrire comme :

$$\begin{aligned} F_{3\tau}(X) := \; & \tfrac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{k}\|X_i - a_j\|^2 - \tfrac{1}{2}\sum_{j=1}^{p}\max_{i=1,\ldots,k}\sum_{r=1,r\neq i}^{k}\|X_r - a_j\|^2 \\ & + \tfrac{\tau}{2}\sum_{i=1}^{k+1}\sum_{j=1}^{p}\|X_i - a_j\|^2 - \tfrac{\tau}{2}\sum_{i=1}^{k+1}\max_{j=1,\ldots,p}\sum_{s=1,s\neq j}^{p}\|X_i - a_s\|^2 \\ & + \tfrac{1}{2}\sum_{i=1}^{k}\|X_{k+1} - X_i\|^2 \\ = \; & \tfrac{\tau+1}{2}\sum_{j=1}^{p}\sum_{i=1}^{k+1}\|X_i - a_j\|^2 + \tfrac{1}{2}\sum_{i=1}^{k}\|X_{k+1} - X_i\|^2 \\ & - \tfrac{1}{2}\sum_{j=1}^{p}\max_{i=1,\ldots,p}\sum_{r=1,r\neq i}^{k}\|X_r - a_j\|^2 - \tfrac{\tau}{2}\sum_{i=1}^{k+1}\max_{j=1,\ldots,p}\sum_{s=1,s\neq j}^{p}\|X_i - a_s\|^2 \\ & - \tfrac{1}{2}\sum_{j=1}^{p}\|X_{k+1} - a_j\|^2 \end{aligned}$$

qui est une fonction DC sur l'espace matriciel $\mathcal{M}_{k+1,n}(\mathbb{R})$ avec la décomposition DC suivante :

$$F_{3\tau} := \; G_{3\tau}(X) - H_{3\tau}(X),$$

où

$$\begin{aligned} G_{3\tau}(X) = \; & \tfrac{\tau+1}{2}\sum_{j=1}^{p}\sum_{i=1}^{k+1}\|X_i - a_j\|^2 + \tfrac{1}{2}\sum_{i=1}^{k}\|X_{k+1} - X_i\|^2, \\ H_{3\tau}(X) = \; & \tfrac{1}{2}\sum_{j=1}^{p}\max_{i=1,\ldots,k}\sum_{r=1,r\neq i}^{k}\|X_r - a_j\|^2 + \tfrac{\tau}{2}\sum_{i=1}^{k+1}\max_{j=1,\ldots,p}\sum_{s=1,s\neq j}^{p}\|X_i - a_s\|^2 \\ & + \tfrac{1}{2}\sum_{j=1}^{p}\|X_{k+1} - a_j\|^2. \end{aligned} \tag{5.32}$$

Il est clair que $G_{3\tau}$ et $H_{3\tau}$ sont convexes. (P3τ) est alors un programme DC de la forme :

$$\min\left\{G_{3\tau}(X) - H_{3\tau}(X) : X \in \mathbb{R}^{(k+1)\times n}\right\}. \tag{5.33}$$

De même façon que les deux premières approches, nous avons à calculer $\partial H_{3\tau}(X)$ et $\partial G^*_{3\tau}(Y)$.

5.3.3.2 Calculer $\partial H_{3\tau}(X)$

On a

$$\partial H_{3\tau}(X) = \partial H_4(X) + \partial H_5(X) + \partial H_6(X) \tag{5.34}$$

où

$$H_4(X) := \sum_{i=1}^{p} h_j^4(X), \qquad h_j^4(X) := \max_{i=1,\ldots,k} h_{j,i}^4(X), \qquad h_{j,i}^4(X) := \frac{1}{2}\sum_{r=1,r\neq i}^{k}\|X_r - a_j\|^2, \tag{5.35}$$

$$H_5(X) := \sum_{i=1}^{k+1} h_i^5(X), \qquad h_5^2(X) := \max_{j=1,\dots,p} h_{i,j}^5(X^\circ, \qquad h_{i,j}^5(X) := \frac{\tau}{2} \sum_{s=1, s\neq j}^{p} \|X_i - a_s\|^2, \tag{5.36}$$

et

$$H_6(X) := \frac{1}{2} \sum_{j=1}^{p} \|X_{k+1} - a_j\|^2. \tag{5.37}$$

Les fonctions $h_{j,i}^4$ sont différentiables et la dérivée est définie par

$$[\nabla h_{j,i}^4(X)]_l = \begin{cases} 0 \text{ si } l \in \{i, k+1\}, \\ X_l - a_j \text{ sinon.} \end{cases} \tag{5.38}$$

Par conséquent, le sous-différentiel de H_4 est explicitement donné par :

$$\partial H_4(X) = \sum_{i=1}^{p} \partial h_j^4(X), \quad \partial h_j^4(X) = co\{\partial h_{j,i}^4(X) : h_{j,i}^4(X) = h_j^4(X)\}. \tag{5.39}$$

De même façon

$$\partial H_5(X) = \sum_{j=1}^{k+1} \partial h_i^2(X), \quad \partial h_i^5(X) = co\{\partial h_{i,j}^5(X) : h_{i,j}^5(X) = h_i^5(X)\}, \tag{5.40}$$

où les fonctions $h_{i,j}^5$ est différentiables et de dérivée :

$$[\nabla h_{i,j}^5(X)]_l = \begin{cases} (p-1)X_l - \sum\limits_{s=1, s\neq j}^{p} a_s \text{ si } l = i, \\ 0 \text{ sinon.} \end{cases} \tag{5.41}$$

Le sous-différentiel de H_5 est également explicitement défini.

Finalement, pour H_6, on a immédiatement

$$[\nabla H_6(X)]_l = \begin{cases} pX_{k+1} - \sum\limits_{j=1}^{p} a_j \text{ si } l = k+1, \\ 0 \text{ sinon.} \end{cases} \tag{5.42}$$

5.3.3.3 Calculer $\partial G_{3\tau}^*(X)$.

Soient G_3 et G_4 deux fonctions dans n $\mathbb{R}^{(k+1)\times n}$ définies par

$$G_3(X) := \frac{1}{2} \sum_{j=1}^{p} \sum_{i=1}^{k+1} \|X_i - a_j\|^2, \quad G_4(X) := \frac{1}{2} \sum_{i=1}^{k} \|X_{k+1} - X_i\|^2. \tag{5.43}$$

D'après (5.32), on a :

$$G_{3\tau}(X) = (\tau + 1)G_3(X) + G_4(X). \tag{5.44}$$

Soit $A^{(j)} \in \mathbb{R}^{(k+1)\times n}$ définie par (5.15). On peut écrire G_3 sous la forme :

$$G_3(X) = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{k+1}\|X_i - a_j\|^2 = \frac{1}{2}\sum_{j=1}^{p}\|X - A^{(j)}\|^2. \tag{5.45}$$

D'un autre côté, G_4 peut s'écrire comme

$$G_4(X) = \frac{1}{2}\sum_{i=1}^{k}\|X_{k+1} - X_i\|^2 = \frac{1}{2}\sum_{i=1}^{k+1}\|X_{k+1} - X_i\|^2 = \frac{1}{2}\|WX\|^2, \tag{5.46}$$

où $W = (w_{ij}) \in \mathbb{R}^{(k+1)\times(k+1)}$ est une matrice définie par

$$w_{ij} = \begin{cases} -1 \text{ si } i = j, \text{ pour } j = 1, \ldots, k \\ 1 \text{ si } j = k+1, \text{ pour } i = 1, \ldots, k \\ 0 \text{ sinon.} \end{cases} \tag{5.47}$$

$G_{3\tau}(X)$ est donc une fonction quadratique définie positive sur $\mathbb{R}^{(k+1)\times n}$. Sa dérivée est donnée par :

$$\begin{aligned} \nabla G_{3\tau}(X) &= (\tau+1)\sum_{j=1}^{p}(X - A^{(j)}) + W^TWX \\ &= [(\tau+1)pI + W^TW]X - (\tau+1)A, \end{aligned} \tag{5.48}$$

avec

$$A := \sum_{j=1}^{p} A^j, \quad \text{i.e.,} \quad A_i = \sum_{j=1}^{p} a_j, : i = 1, ..., k+1.$$

Puisque $X = \nabla G_{3\tau}^*(Y)$ ssi $Y = \nabla G_{3\tau}(X)$, on a

$$Y = [(\tau+1)pI + W^TW]X - (\tau+1)A,$$

ou

$$[(\tau+1)pI + W^TW]X = Y + (\tau+1)A.$$

On obtient alors explicitement X :

$$X_i = \frac{B_i + X_{k+1}}{1+c} \text{ pour } i = 1...k, \tag{5.49}$$

$$X_{k+1} = \frac{(1+c)B_{k+1} + \sum_{l=1}^{k} B_l}{(1+c)(k+c) - k}, \tag{5.50}$$

avec $B = Y + (\tau+1)A$ et $c = (\tau+1)p$.

5.3.3.4 Le schéma DCA pour résoudre (P3$_\tau$)

Le schéma DCA pour résoudre (P3$_\tau$) se décrit comme suivant :

Algorithme 5.3
CHBN-DCA-3 : DCA appliqué au problème (P3$_\tau$)

Initialisation : Choisir une matrice $X^{(0)} \in \mathbb{R}^{(k+1)\times n}$. Soit une tolérance $\epsilon_{DCA} > 0$, $0 \leftarrow l$.

Répéter
- Calculer $Y^{(l)} \in \partial H_{3\tau}(X(^{(l)})$ par les formules (5.34) - (5.42) ;
- Calculer $X^{(l+1)} \in \partial G^*_{3\tau}(Y^{(l)})$ par (5.49) et (5.50) ;
- $l+1 \leftarrow l$

Jusqu'à $\left\|X^{(l)} - X^{(l-1)}\right\| \leq \epsilon_{DCA}(\left\|X^{(l)}+1\right\|)$ ***ou*** $\left|F(X^{(l)}) - F(X^{(l-1)})\right| \leq \epsilon_{DCA}(\left|F(X^{(l)})+1\right|)$.

Remarque 5.1 *Tous les trois problèmes (P1$_\tau$),(P2$_\tau$) et (P3$_\tau$) peuvent être reformulés sous la forme d'un programme DC qui est la minimisation de la différence d'une fonction convexe quadratique simple et une fonction convexe non différentiable. Ce programme DC est très convenable pour la résolution par DCA, qui consiste à résoudre explicitement une suite de problèmes quadratiques (la solution des problèmes quadratiques est explicite).*

5.3.4 Retrouver les centres réels

Dans le cas où la solution fournie par DCA n'appartient pas à $\mathcal{A}$, on retrouve les centres réels comme suivant :

- Soit X^* la solution fournie par **CHBN-DCA-1** ou **CHBN-DCA-2** et soient $x_i^* = (X^*)_i$, $i = 1, ..., k$. Alors les centres réels $\overline{x}_i$ pour $i = 1, ..., k$ sont déterminés par

$$\overline{x}_i = \text{ argmin } \{\|x_i^* - a_j\|^2 : j = 1, ..., p\}. \tag{5.51}$$

  Le centre total est obtenu par :

$$\min_{j=1,...,p} \sum_{i=1}^{k} \|\overline{x}_i - a_j\|^2. \tag{5.52}$$

- Soit X^* la solution fournie par **CHBN-DCA-3** et soient $x_i^* = (X^*)_i$, $i = 1, ..., k+1$. Les centres réels $\overline{x}_i$ pour $i = 1, ..., k+1$ sont obtenus par

$$\overline{x}_i = \text{ argmin } \{\|x_i^* - a_j\|^2 : j = 1, ..., p\}. \tag{5.53}$$

5.3.5 Chercher un point initial pour DCA

Comme nous avons déjà mentionné dans le chapitre 1, la recherche du point initial est une question cruciale de DCA. De manière analogue à la procédure alternative FCM-DCA dans

le chapitre précédent, nous proposons une combinaison de DCA - k-means pour initialiser nos algorithmes. A partir d'un point $X^{(0)}$ dont $X_i^{(0)}$ sont choisis aléatoirement parmi l'ensemble $\mathcal{A}$, on effectue une itération de DCA (calculer $Y^{(0)} \in \partial H(X^{(0)})$ et $Z^{(1)} \in \partial G^*(Y^{(0)})$). En suite, on réalise une itération de k-means à partir de $Z^{(1)}$ pour obtenir $X^{(1)}$. On répète cette procédure un certains nombre de fois pour obtenir un point initial pour les schémas DCA.

Cette procédure appelée **IP** est décrite comme ci-dessous. Pour simplifier l'écriture, on utilise la forme générale d'un programme DC pour tous les trois modèles :

$$F(X) = G(X) - H(X).$$

Algorithme 5.4
IP : Procédure DCA - k-means pour initialiser DCA

Initialisation : *Soit q un entier positif. Soit matrice $X^{(0)}$ dont $X_i^{(0)}$ sont choisis aléatoirement parmi les points dans $\mathcal{A}$.*

Pour $\mathbf{r = 0, 1, ..., q}$
- *Calculer $Y^{(r)}$ et $X^{(r+1)}$ selon le schéma de* ***CHBN-DCA-1*** *ou* ***CHBN-DCA-2*** *ou* ***CHBN-DCA-3***
- *Assigner les points $a_j \in \mathcal{A}$ aux clusters $\pi_i, i = 1, ...k$ (π_i est le cluster dont le centre est $X_i^{(r+1)}$).*
- *Pour chaque $i \in \{1, \ldots, k\}$, on recalcule Z_i comme étant le centre du cluster π_i.*

$$Z_i := \arg\min \left\{ \sum_{a_j \in \pi_i} \|y - a_j\|^2 : y \in \mathbb{R}^n \right\}.$$

- *Mettre à jour $X_i^{(r+1)} := Z_i$ pour $i = 1, ...t$.*

Notons que chaque itération de **CHBN-DCA-3** donne $(k + 1)$ centres tandis que k-means fournit seulement k (sauf le centre total) et le centre total est déterminé par :

$$Z_{k+1} = \arg\ \min_{a_j \in} \sum_{i=1}^{k} \|Z_i - a_j\|^2.$$

Après nombreuses expérimentations numériques, nous avons constaté que, pour la phase d'initialisation de DCA, la procédure IP (avec $3 \leq q \leq 5$) est meilleur que l'algorithme k-means exécuté jusqu'à sa convergence.

5.4 Expériences numériques

Les algorithmes sont implémentés en C++ et exécutés sur un ordinateur Pentium 4 2.6 GHz,1Gb Ram. Nous testons les algorithmes sur deux ensembles de jeux de données. Le

premier est un jeu de données réel qui contient les coordonnées de 51 villes north américaines. Ce jeu de données a été étudié dans plusieurs travaux [260], [261], [262], [244] avec $k = 6$. Le deuxième ensemble de données est généré aléatoirement.

Nos expérimentations contiennent trois parties. Dans la première expérimentation, nous comparons les trois schémas de DCA (utilisant la procédure IP pour la recherche d'un point initial) et la méthode basée sur k-means (noté **OKM**). Pour la procédure **IP**, on choisit $q = 5$ et $\tau = \tau_1 = \tau_2 = 2$ (d'après les test, pour τ, τ_1, τ_2 dans $[0.5, 2]$, on obtient le même résultat).

Pour **OKM**, on utilise le code disponible sur *http : //www.fas.umonteral.ca/biol/legendre/* pour trouver les centres de niveau 1. Les villes les plus proches des ces centre seront choisies pour les centres réels ($\overline{x}_i$ for $i = 1, ..., k$). Le centre total $\overline{x}_{k+1}$ est déterminé via la formule (5.52).

Puisque k-means est une méthode heuristique qui est très sensible au point initial, nous exécutons dix fois DCA et **OKM** à partir du même point initial aléatoirement choisi dans $\mathcal{A}$. Les résultats de **CHBN-DCA-1**, **CHBN-DCA-2**, **CHBN-DCA-3** et **OKM** sont présentés dans le tableau 5.1. Les critères de comparaison sont les suivants : le temps de calcul (en seconde), le nombre d'itérations et le coût total qui est calculé par :

$$\sum_{i=1}^{k} \sum_{j \in \mathcal{A}_{ii}} \|\overline{x}_i - a_j\| + \sum_{i=1}^{k} \|\overline{x}_{k+1} - \overline{x}_i\|, \tag{5.54}$$

où $\mathcal{A}_i$ est le cluster dont le centre est $\overline{x}_i$, pour $i = 1, \ldots, k$.

Dans le tableau 5.2, nous présentons le meilleur résultat fourni par les algorithmes développés dans [261] (**KMC**), dans [244] (**1-km**) et **DCA** pour le premier jeu de données. Dans [261], l'algorithme **KMC** a été proposé pour la détermination d'un arbre hiérarchique de multi-niveaux en appliquant itérativement k-means à chaque niveau et décomposant successivement des niveaux. Quelques variantes de **KMC** sont proposés dans [262].

Dans [244], les auteurs ont proposé quatre variantes de leur méthode pour résoudre le problème de détermination de l'arbre hiérarchiques de 2 niveaux. Le meilleur résultat dans ce travail a été obtenu par **1-km**(une variante de leur algorithme en prenant le point initial qui est la solution de k-means).

Les meilleurs résultats donnés par **CHLB-DCA-3-IP** et **1-km** après les dix tests sur différents point initiaux sont reportés dans le tableau 5.2.

Dans la deuxième expérimentation, nous comparons les DCA et **OKM** sur les données générées aléatoirement. Pour créer les données nous choisissons d'abord k centres et ensuite générons aléatoirement les points dans les cercles dont les centres sont créés précédemment. Les résultats sont présentés dans le tableau 5.3 (le même point initial pour tous les algorithmes).

Pour étudier l'efficacité de la procédure **IP**, nous testons deux variantes de **CHBN-DCA-3**

Point initial	**CHBN-DCA-1-IP**			**CHBN-DCA-2-IP**			**CHBN-DCA-3-IP**			**OKM**	
	Cost	iter	Time	Cost	iter	Time	Cost	iter	Time	Cost	Time
1	317	100	0.005	314	91	0.006	**310**	76	0.005	320	0.002
2	314	105	0.008	318	137	0.011	**305**	83	0.005	357	0.002
3	314	95	0.005	313	125	0.009	**298**	80	0.005	318	0.002
4	317	80	0.004	313	120	0.009	**303**	70	0.004	368	0.002
5	314	105	0.008	314	122	0.010	**313**	83	0.005	320	0.002
6	318	102	0.005	314	100	0.010	**314**	82	0.005	342	0.002
7	320	110	0.010	318	133	0.012	**314**	80	0.005	320	0.002
8	314	90	0.008	314	120	0.010	**310**	76	0.005	325	0.002
9	318	100	0.007	317	100	0.008	**313**	78	0.005	320	0.002
10	320	105	0.008	313	125	0.009	**314**	81	0.005	325	0.002

TAB. 5.1 – Résultat numérique sur les données de 51 villes north américaines

CHBN-DCA-3	**OKM**	**1-km**([244])	**KMC** ([261])
298	318	308	345

TAB. 5.2 – Comparaison avec les autres méthodes

- avec et sans la procédure **IP**. Les résultats sont reportés dans le tableau 5.4.

Commentaire : à partir des résultats numériques, on constate que
– la solution fournie par les trois méthodes basées sur DCA est meilleur que les autres méthodes sur les deux ensembles de jeux de données.
– **CHBN-DCA-3** est la plus efficace des trois méthodes DCA. Cela pourrait venir du fait que le troisième modèle d'optimisation est le plus adéquat.
– La procédure **IP** est efficace pour initialiser les trois schémas de DCA.
– DCA n'est pas coûteux en temps de calcul car les trois algorithmes sont explicites.

Conclusion : Nous avons proposé les trois nouveaux modèles d'optimisation pour le clustering (dur) hiérarchique de deux niveaux et ensuite les résolu par des méthodes basées sur la programmation DC et DCA. Il s'avère que chaque programme DC introduit est la minimisation de la différence d'une forme quadratique définie positive et d'une fonction convexe non différentiable, chose fort intéressante pour l'application de DCA qui s'interprète alors comme une technique qui consiste à résoudre une suite de programmes quadratiques convexes approximant dont les solutions se calculent de manière explicite. Les solutions données par DCA permettent alors de retrouver des centres réels et le centre total grâce à des procédures simples de tri. Nous avons également introduit une procédure alternative DCA - k-means pour chercher un point initial de DCA. Les simulations numériques montrent la qualité des

Data			CHBN-DCA-1-IP			CHBN-DCA-2-IP			CHBN-DCA-3-IP			OKM		
p	n	k	Cost	N^{o}it	Time	Cost	N^{o}it	Time	Cost	N^{o}it	Time	Cost	N^{o}it	Time
100	2	5	**322**	13	0.010	323	6	0.005	**322**	10	0.010	330	8	0.005
500	2	8	334	68	0.042	**333**	69	0.042	**333**	80	0.053	348	12	0.030
1000	8	10	**183**	24	0.100	**183**	10	0.057	**183**	10	0.086	228	5	0.030
2000	3	20	1968	28	0.100	**1965**	10	0.100	**1965**	10	0.100	2391	5	0.050
5000	5	10	**3851**	83	0.840	**3851**	74	0.75	**3851**	72	0.740	4428	20	0.420
5000	20	6	**18244**	47	2.6	**18244**	47	2,6	**18244**	46	2,4	21612	28	1.2
10000	20	6	**43699**	66	7.79	**43699**	67	8,20	**43699**	66	7,80	45239	62	6.2
20000	30	12	**107987**	73	37	**107987**	40	8,20	**107987**	74	37	119829	92	39
50000	30	20	**282099**	182	378	**282099**	183	419	**282099**	182	351	345553	179	332

TAB. 5.3 – Résultats numériques sur les données aléatoires

Data			CHBN-DCA-3-IP			CHBN-DCA-3		
p	n	q	Cost	N^{o}it	Time	Cost	N^{o}it	Time
51	2	6	**298**	80	0.010	320	75	0.010
100	2	5	**322**	10	0.010	323	10	0.010
500	2	8	**333**	80	0.053	**333**	82	0.060
1000	8	10	**183**	10	0.086	196	12	0.092
2000	3	20	**1965**	10	0.1	2024	15	0.16
5000	5	10	**3851**	72	0.74	4108	87	0.99
5000	20	6	**18244**	46	2,4	19342	50	2.6
10000	20	7	**43699**	66	7,80	43879	63	7.8
2000	30	12	**107987**	74	37	108124	74	37
50000	30	20	**282099**	182	351	289987	189	371

TAB. 5.4 – Comparaison de **CHBN-DCA-3-IP** et **CHBN-DCA-3**

solutions obtenues par nos algorithmes, leur robustesse et leur performance par rapport aux méthodes existantes.

Troisième partie

Modélisation et Optimisation Globale basée sur DCA, GA et une méthode de coupes en Cryptologie

Chapitre 6

Introduction à la cryptologie

On peut dire que la cryptologie est un art ancien et une science nouvelle : un art ancien car Jules César l'utilisait déjà ; une science nouvelle parce que ce n'est que depuis les années 1970 qu'elle devient un thème de recherche scientifique académique. Cette science englobe la cryptographie - l'écriture secrète - et la cryptanalyse - l'analyse de cette dernière. Dans toute l'histoire, le combat entre la cryptographie et la cryptanalyse a joué un grand rôle. Pour commencer ce chapitre, nous donnons une définition succincte des principaux termes utilisés dans la suite.

- **Chiffrement** : transformation appliquée à un message pour assurer sa confidentialité.
- **Clé** : donnée qui personnalise un protocole cryptographique. Cette donnée peut être confidentielle, on parle dans ce cas de "clé secrète" ou privée. Elle peut également être connue de tous, on parle alors de "clé publique".
- **Confidentialité** : garantie que seules les personnes autorisées ont accès à l'information. Pour cela, on utilise des procédés de chiffrement.
- **Contrôle d'accès, Identification** : preuve d'identité.
- **Déchiffrement** : transformation inverse du chiffrement qui consiste à retrouver l'information initiale contenue dans le message chiffré à l'aide d'une clé de "déchiffrement".
- **Décryptage** : action de "casser" le chiffrement d'une information, et donc de retrouver le texte clair d'un chiffré, sans connaître la clé de "déchiffrement".

6.1 Objectifs de la cryptographie

Longtemps, la cryptographie est un art réservé aux militaires, construite principalement sur la base mathématique. Au cours de ces dernières années, avec la croissance de l'informatique, cryptographie est devenue une science populaire basée sur la base mathématique et informatique.

Historiquement, l'objectif fondamental de la cryptographie est de permettre à deux personnes, appelées traditionnellement Alice et Bob, de communiquer de telle sorte qu'une troisième

personne ne puisse pas comprendre ce qui est échangé. L'information qu'Alice souhaite transmette à Bob, appelée *text clair*, peut être un texte simple, une donnée numérique ou n'importe quoi d'autre. Avant d'envoyer le message à Bob, Alice chiffre son *texte clair* avec sa clé. Bob reçoit le message venant d'Alice et déchiffre le message avec la bonne clé, ce que ne possède pas la troisième personne essayant de suivre leurs conversations.

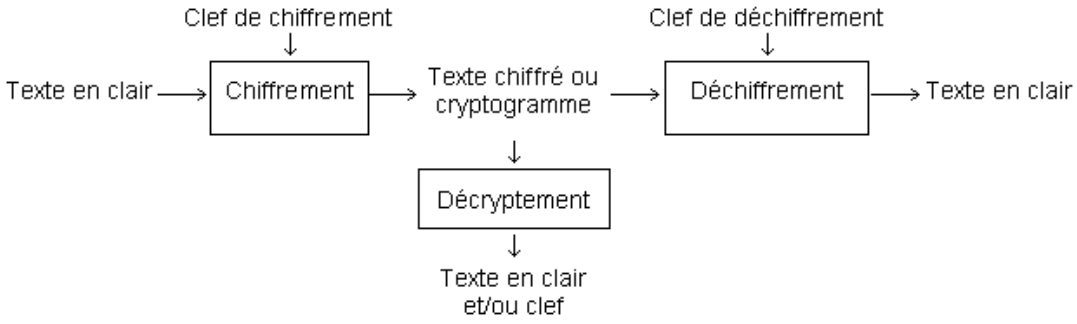

FIG. 6.1 – Le schéma de base de la cryptographie

La cryptographie de nos jours a pour objectifs suivants :

1. **Confidentialité** : le tout premier objectif était et reste la confidentialité qui permet de protéger le contenu des informations sauvegardées ou transmises. Seules les personnes autorisées doivent pouvoir y accéder. Ce besoin peut être satisfait en utilisant les algorithme de chiffrement et déchiffrement.
2. **Intégrité** : le problème d'intégrité se pose avec le partage d'informations. Par exemple, un document est partagé pour un groupe de travail. Le document peut donc être modifié par n'importe quel membre du groupe. On souhaite pouvoir détecter toute modification. Pour cela, on utilise une *fonction de hachage*, qui permet d'associer le document à une donnée de taille fixée (appelée "empreinte" ou "condensé"). En principe, une fonction de hachage doit satisfaire deux propriétés :
   – *Sens unique* : on doit assurer qu'à partir de la valeur de fonction de hachage, il est impossible de retrouver le message initial.
   – *Sans collisions* : pour deux messages différents, la fonction de hachage doit fournir deux valeurs différentes.
3. **Authenticité** : de nos jours, avec le réseau informatique, tout individu peut accéder à des machines dans le monde entier. Il faut donc, pour la sécurité, identifier la personne souhaitant consulter les informations secrètes. Les protocoles d'indentification sont donc employés dans lesquels le prouveur tente de convaincre le vérifieur son identité sans relever aucune information sur les information secrètes. Le protocole doit assurer qu'il est impossible qu'une personne malveillante, écoutant la conversation, puisse ensuite se faire passer pour le prouveur.

On peut classer les méthodes de cryptographie en trois grandes classes :

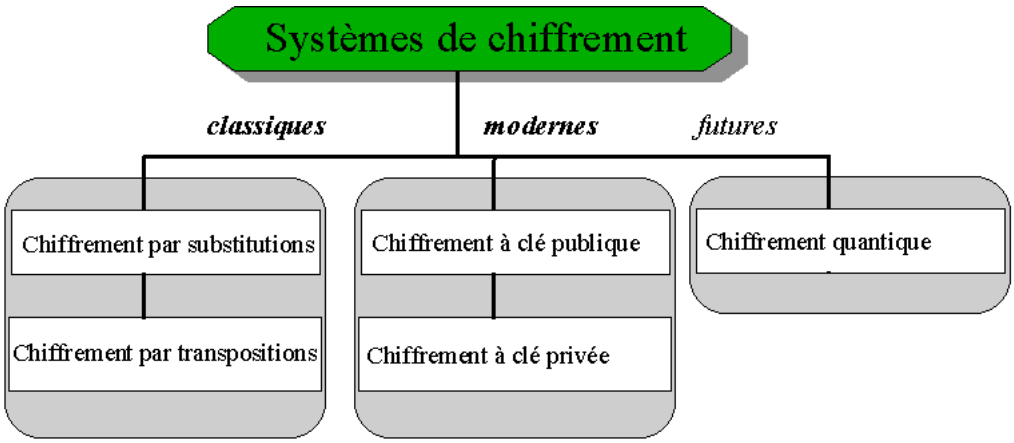

FIG. 6.2 – Les principales méthodes de cryptographie

Dans la suite, nous allons présenter brièvement ces principales méthodes de la cryptographie.

6.2 Les méthodes historiques de la cryptographie

Historiquement, le premier message chiffré connu remonte à l'Antiquité. Il s'agit d'une tablette d'argile, retrouvée en Irak. Un potier y avait gravé sa recette secrète en supprimant des consonnes et en modifiant l'orthographe des mots.

Après, entre le $X^{\text{ième}}$ et $VII^{\text{ième}}$ siècle av. J.-C., les Grecs utilisent une technique de chiffrement par transposition en utilisant une scytale, également appelée bâton de Plutarque. Autour du bâton, ils enroulent en spires jointives une bande de cuir et y inscrivent le message. Une fois déroulé, le message est envoyé au destinataire qui possède un bâton identique nécessaire au déchiffrement.

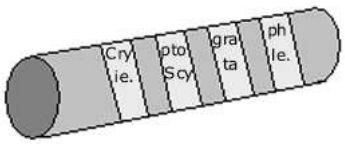

FIG. 6.3 – Batôn de Plurtaque

http ://tpecrypto.free.fr/origines.htm

Les premiers vrais systèmes de cryptographie sont apparus au alentours des années -200. Il existe 4 types de méthodes :
– Substitution par mono-alphabétique ;

– Substitution par poly-alphabétique ;
– Substitution homophonique ;
– Substitution par polygrammes.

6.2.1 La cryptographie par substitution mono-alphabétique

Le codage par substitution mono-alphabétique est le plus simple à imaginer. Il s'agit de remplacer chaque lettre ou symbole par un caractère différent. Son principal avantage est la rapidité de la mise en œuvre.

Par exemple, on définit la clé suivante :
clair A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
codé G Q B J S D P I F T L A R M V K Z X C W H Y N E U O

Le message "**alice aime bien bob**" sera codé par "**gafbs gfrs qfsm qvq**".

Un des désavantages de cette méthode est la longueur de la clé (qui est égale au nombre de caractères dans l'alphabet). On a donc inventé des alternatives :
– Le chiffre de César.
– Le chiffre At Bash.
– Le carré de Polybe.
De plus, la plus grande faiblesse de cette méthode est que : dans les langues, toutes les lettres n'ont pas la même fréquence d'apparition. Cette propriété rend illusoire le nombre de clés possibles (théoriquement égal à 26!, ce qui en soit est un chiffre énorme).

6.2.1.1 Chiffre de César

Le chiffre de César est la première méthode cryptographique par substitution mono-alphabétique. Il consiste en un code de substitution mono-alphabétique avec décalage. L'on recopie les lettres dans l'ordre alphabétique en commençant à un terme n de l'alphabet. Ce type de code a été conçu dans le but de faciliter la mémorisation de la clé de cryptage.

Le chiffre de César est un système peu sur, car il n'y a que 26 façons différentes de crypter un message. Il est donc très facile de tester de façon exhaustive toutes les clés possibles.

Par exemple, une clé de chiffre de César avec un décalage de 6 lettres :
clair A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
codé G H I J K L M N O P Q R S T U V W X Y Z A B C D E F

Le message "**alice aime bien bob**" sera codé par "**groik gosk hort huh**".

6.2.1.2 Chiffre de At Bash

Le chiffre At Bash, consiste simplement à écrire l'alphabet à l'envers. Cette méthode est très facile à casser car il n'y a pas de choix possible pour la clé, il suffit de connaître l'algorithme

de codage pour pouvoir décoder immédiatement.

6.2.1.3 Le carré de Polybe

Cette méthode a été créée par POLYBE, un historien grec qui vécut environ de -205 avant JC jusque -125 avant JC. Les lettres alphabétiques sont placées dans un tableau 5*5 (les lettres "**i**" et "**j**" sont dans la même cage) :

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I,J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Pour chiffrer le message, on remplace alors chaque lettre par ses coordonnées dans le tableau, en écrivant d'abord la ligne, puis la colonne.

Le message "**alice aime bien bob**" sera codé par "**1131241315 11243215 12241533 123412**".

Le carré de Polybe permet de réduire le nombre de symboles utilisés pour le codage, ce qui rend son analyse plus difficile. C'est en cela un précurseur des méthodes modernes.

Une amélioration du carré de Polybe est le chiffre de Delastelle, qui est un mélange de substitution et transposition. On commence par regrouper les lettres du message à coder 5*5, puis on utilise le carré de Polybe.

6.2.2 La cryptographie par substitution polyalphabétique

Même si l'on connaissait les faiblesses de la cryptographie par substitution, il n'y eut pas entre César et le $XVI^{\text{ième}}$ de véritable nouvelle méthode. Il fallait attendre jusqu'à 1586, Blaise de VIGENÈRE présente dans son livre *Traité des chiffres ou secrètes manières d'écrire* une nouvelle façon de chiffrer les messages qui domina 3 siècles durant. L'idée de VIGENÈRE est d'utiliser un chiffre de César, mais où le décalage utilisé change de lettres en lettres. Pour cela, on utilise une table composée de 26 alphabets, écrits dans l'ordre, mais décalés de ligne en ligne d'un caractère. On écrit encore en haut un alphabet complet, pour la clé, et à gauche, verticalement, un dernier alphabet, pour le texte à coder :

Pour coder un message, on choisit une clé de longueur arbitraire. On écrit ensuite cette clé sous le message à coder. Si la clé n'est pas assez longue pour couvrir tout le mot on la répète. Exemple :

Clair	F	A	N	T	A	S	I	O
Codé	S	U	P	I	N	F	O	S

On regarde ensuite à l'intersection de la lettre à coder (pris en abscisse), et de la lettre de la

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

FIG. 6.4 – Le carré de Vigenere

clé prise en ordonnée. Le choix de prendre les lettres de la clé en ordonnée et les lettres du texte en clair en ordonnée est arbitraire.

Par exemple, pour coder la lettre F, on va à la lettre F de la ligne du clair puis on va à la lettre S de la première colonne. L'intersection correspond à la lettre codée soit X.

On obtient après chiffrage : **XUCBNXXWG**

Cet algorithme de cryptographie possède deux principaux avantages :
– Une lettre peut être codée par différents façon, ce qui le rend invulnérable à une attaque par analyse statistique.
– L'on a une infinité de clés et il n'y a pas de règle quant au choix de la clé.
Le chiffrement de Vigenère n'a été décrypté qu'en 1854 par Charles BABBAGE.

6.2.3 Le chiffre de substitution homophonique

Pour échapper à l'analyse de fréquences, la *substitution homophonique*(ou *substitution à représentations multiples*) a été proposée. Son principe consiste à remplacer une lettre non pas par un symbole unique, mais par un symbole choisi au hasard parmi plusieurs, cela empêche la mise en correspondance des lettres les plus fréquentes avec les symboles les plus représentés dans le texte chiffré. Dans la version la plus sophistiquée, on choisit un nombre des symboles proportionnel à la fréquence d'apparition de la lettre, on parle alors de

renversement des fréquences.

Par exemple, dans le tableau ce-dessus, la lettre "a" sera chiffrée par "07 ou "21" ou "22" ou "23".

a	07	21	22	23
c	09	10		
e	08	20		
l	03			
i	05			

Le message "**alice**" peut donc être codé par "**0703050920**" ou "**2203050910**" ou bien d'autres.

6.2.4 La substitution par polygrammes

La substitution par polygrammes est moins connue que les trois méthodes que nous venons de présenter. Dans cette méthode, au lieu de substituer des caractères, on substitue par des polygrammes (souvent par des groupes de lettres).

Le système de "Playfair", inventé par Sir Charles WHEATSTONE, popularisé par L. PLAYFAIR, utilise ce stratagème au moyen d'une table. Cet algorithme remplace chaque paire de lettre du texte en clair par une autre paire.

Le Chiffre Playfair est beaucoup plus dur à casser par les attaques par analyse fréquentielle habituellement grâce aux 600 digrammes possibles au lieu des 26 lettres de l'alphabet. Cela fait que l'analyse de fréquence des digrammes reste toujours possible mais elle est considérablement plus difficile.

Comme la plupart des chiffrements anciens, de nos jours, le Chiffre de Playfair peut facilement être décrypté si l'on dispose de suffisamment d'échantillons et utilise l'attaque à texte clair connu.

6.3 Les méthodes modernes de la cryptographie

Parmi les méthodes modernes de la cryptographie, on distingue deux familles selon le principe de fonctionnement :
– Méthode à clé secrète (Cryptographie symétrique)
– Méthode à clé privée (Cryptographie asymétrique).

6.3.1 Cryptographie symétrique

Un système de chiffrement à clé secrète repose sur le partage entre deux personnes en communication, d'une même clé secrète utilisée à la fois pour le chiffrement des données et pour

son déchiffrement. Plus précisément, pour transmettre un message, Alice réalise au préalable le chiffrement en utilisant une clé secrète. A la réception, Bob reçoit le message crypté et applique un algorithme de déchiffrement avec la même clé qu'Alice pour le message en clair.

Le grand avantage des algorithmes symétriques est la rapidité. Ce qui explique le fait que malgré ses limites, les algorithmes symétriques sont beaucoup plus utilisés dans les applications réelles.

Le cryptage à clé symétrique possède des inconvénients dont le principal est le problème de la transmission de la clé au destinataire. En effet si la clé est interceptée par une troisième personne, elle a donc aucune difficulté de découvrir la conversation. Deuxième problème important, dans le cas d'échange entre N personnes susceptibles de communiquer, il faut distribuer $N * (N-1)/2$ clés, en outre il faut considérer les temps de chiffrement pour chaque clé qui implique un temps global important.

Les méthodes les plus connues dans cette famille sont : DES et AES.

6.3.1.1 Data Encryption Standard - DES

Dans les années 1970, pour les besoins civils, le NBS (National Bureau of Standards) lança un appel d'offres dans le Federal Register pour la création d'un système cryptographique. Le cahier des charges était le suivant :
- L'algorithme repose sur une clé relativement petite.
- L'algorithme doit être facile à implémenter et doit être très rapide.
- Le chiffrement doit avoir un haut niveau de sûreté, uniquement lié à la clé, et non à la confidentialité de l'algorithme.

Pour répondre à cet appel d'offre, IBM et NSA (National Security Agency) ont proposé DES. Cet algorithme de chiffrement est devenu le plus utilisé au monde durant le dernier quart du $XX^{\text{ième}}$ siècle.

DES utilise une chaîne de 64 bits, mais en fait seuls 56 bits servent réellement à définir la clé. Les bits 8, 16, 24, 32, 40, 48, 56, 64 sont des bits de parité (bits de détection d'erreur).

L'algorithme DES transforme un bloc de 64 bits en un autre bloc de 64 bits en quatre étapes :

1. Le texte est découpé en blocs de 64 bits. On fabrique à partir de K 16 sous-clés $K_1, ..., K_{16}$ à 48 bits. Les sous-clés K_i sont composées de 48 bits de K, pris dans un certain ordre.
2. Pour chaque bloc de 64 bits x du texte, on calcule une permutation finie $y = P(x)$. y est représenté sous la forme $y = G_0D_0$, où G_0 est les 32 bits à gauche de y, D_0 est 32 bits à droite.
3. On applique 16 rondes d'une même fonction. A partir de $G_{i-1}D_{i-1}$, on calcule G_iD_i en posant : $G_i = D_{i-1}, D_{i-1} = G_{i-1}XORf(D_{i-1}, K_i)$.
4. On applique à $G_{16}D_{16}$ l'inverse de la permutation initiale. $Z = P^{-1}(G_{16}D_{16})$ est le bloc de 64 bits chiffré à partir de x.

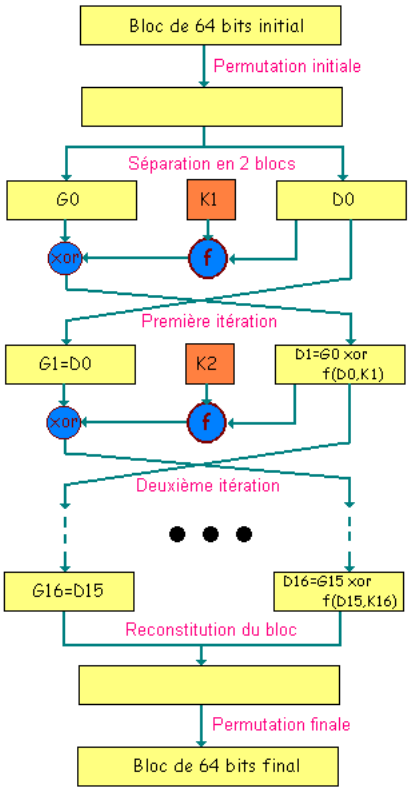

FIG. 6.5 – DES

http ://www.bibmath.net/crypto/moderne/des.php3

De nos jours, DES est complètement obsolète. Le 17 juin 1997, le DES est cassé en 3 semaines par une fédération de petites machines sur Internet. Et on estime très officiellement à cette date à quelques secondes le temps nécessaire à un Etat pour casser un système utilisant le DES.



FIG. 6.6 – Triple-DES

http ://www.bibmath.net/crypto/moderne/des.php3

Une alternative est donc d'utiliser un système composant de 3 sous-système DES, le *Triple-DES*. Le temps de calcul est donc trois fois plus que DES simple et ne répond plus au besoin des applications réelles.

6.3.1.2 Advanced Encryption Standard - AES

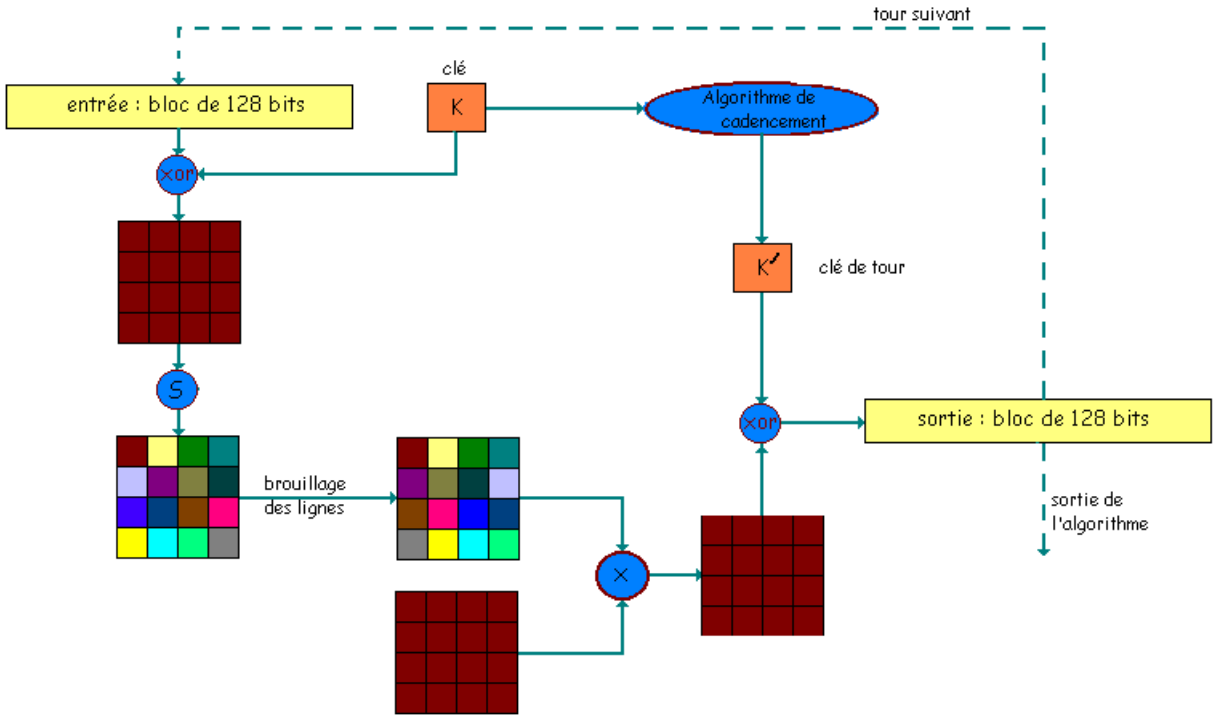

FIG. 6.7 – AES

http ://www.bibmath.net/crypto/moderne/des.php3

Comme nous venons de voir que le DES n'est plus sûr et adapté au besoin. En janvier 1997, le NIST (National Institute of Standards and Technologies) des Etats-Unis lance un appel d'offres pour une nouvelle méthode, appelée, Advanced Encryption System (AES). Cette nouvelle méthode devrait satisfaire les besoins suivants :

- Une grande sécurité.
- Une large portabilité : il est destiné à servir aussi bien dans les cartes à puces, aux processeurs de 8 bits peu puissants, ...
- La rapidité.
- Une lecture facile de l'algorithme.
- La méthode doit utiliser des clés plus grandes que DES.

Le 2 octobre 2000, le Rijndael a été officiellement choisi. Cette méthode est mis au point par deux belges, J. DAEMEN et V. RIJMEN.

Le Rijndael procède par blocs de 128 bits, avec une clé de 128, 192 ou 256 bits. Les 128 bits en entrée sont permutés selon une table définie au préalable et ensuite placés dans une matrice de 4x4 éléments et ses lignes subissent une rotation vers la droite. Une transformation linéaire est

ensuite appliquée sur la matrice, elle consiste en la multiplication binaire de chaque élément de la matrice avec des polynômes issus d'une matrice auxiliaire. Finalement, on applique un XOR entre la matrice et une autre matrice pour obtenir une matrice intermédiaire. Ces différentes opérations sont répétées plusieurs fois et définissent un tour. Pour une clé de 128, 192 ou 256, AES nécessite respectivement 10, 12 ou 14 tours.

6.3.2 Cryptographie asymétrique

Le concept de cryptographie asymétrique (ou à clé publique) fut inventé pas W. Diffie et M. Hellman. L'idée neuve de cette méthode est que les clés peuvent être des paires : une clé pour le chiffrement et une autre pour le déchiffrement. On doit assurer qu'il est impossible de générer une clé à partir de l'autre. S. Diffie et Hellman n'ont pas eux-même proposé des systèmes concrets pour ce concept. Depuis 1976, de nombreux algorithmes ont été proposés. Nombre d'entre eux ne sont pas sûrs. Parmi ceux qui restent, nombre d'entre eux ne sont pas pratiques. En 1977, D. Rivest, A. Shamir et L. Adleman ont proposé une méthode, la meilleure et la plus utilisée jusqu'à ce jour, le système RSA (le sigle RSA désigne les lettres initiales de leur nom).

6.3.3 Le RSA

Principe de fonctionnement de RSA :

1. Création des clés : Bob crée 4 nombres p,q, e et d :
   - p et q sont deux grands nombres premiers distincts.
   - e est un entier premier avec le produit $(p-1)(q-1)$.
   - d est tel que $ed = 1 \bmod (p-1)(q-1)$.
2. Distribution des clés : Le couple (n,e) constitue la clé publique de Bob. Il la rend disponible à tout le monde. Le couple (n,d) constitue sa clé privée qu'il garde pour lui seul.
3. Si Alice souhait envoyer un message codé à Bob.
   (a) Alice représente son message sous la forme d'un ou plusieurs entiers M compris entre 0 et $n-1$.
   (b) Alice calcule $C = M^e \bmod$ n et envoie C à Bob.
4. Bob reçoit C, et il calcule grâce à sa clé privée $D = C^d(\bmod$ n). D'après un théorème du mathématicien Euler, $D = M^{de} = M(\bmod$ n). Il a donc reconstitué le message initial.

La sécurité de cet algorithme repose sur deux conjectures : casser RSA nécessite la factorisation du nombre n et la factorisation est un problème difficile. Actuellement, les attaques actuelles du RSA se font essentiellement en factorisant l'entier n. En 2005, le plus grand nombre factorisé était de 663 bits. Cependant, les clefs RSA sont habituellement de longueur

comprise entre 1024 et 2048 bits. On peut donc dire que le RSA est une méthode de chiffrement assez sûre, mais il ne faut toutefois pas verser dans un optimisme béat, une mauvaise utilisation de la cryptographie RSA (choix d'un exposant e trop petit, mauvaise complétion des blancs,...) la rend vulnérable.

Concernant la vitesse de RSA, on peut dire que RSA est beaucoup plus lent que les algorithmes symétriques. En général, le DES est cent fois plus vite que RSA. Dans certains cas particuliers, DES est mille fois plus rapide que RSA.

6.3.4 Comparaison entre cryptographie asymétrique et symétrique

Vitesse de chiffrement
La vitesse est la plus grande cause de différence entre deux familles de méthodes. On peut dire que RSA est beaucoup plus lent que les algorithmes symétriques. En général, le DES est cent fois plus vite que RSA. Dans certaine cas particuliers, DES est mille fois plus rapide que RSA.
Dans les applications réelles, les méthodes symétriques sont beaucoup plus utilisés. La cryptographie asymétrique, peu utilisé, sert principalement à authentifier les clés secrètes.

Longueur des clés
Nous avons vu que DES utilise des clés de longueur maximale 256 bits, cependant RSA opère habituellement avec les clés de longueur 1024 ou 2048bits.

Facilité de distribution de clés
La distribution de clé est le plus grand inconvénient des méthodes symétriques, ce qui n'est pas dans le cas des méthodes asymétriques. Par contre, les clés publique doivent être protégées aux attaques de type *BlackHat*.

6.4 Une méthode du future : Cryptographie quantique

La cryptographie quantique n'est pas un algorithme de chiffrement à proprement parler : elle permet simplement de mettre en œuvre un algorithme de cryptographie classique, le chiffrement de Vernam. Le code de Vernam est le seul qui soit mathématiquement reconnu comme inviolable, mais le fait de devoir utiliser une clef différente pour chaque chiffrement est à l'origine d'un problème pratique difficile (le problème de *distribution des clefs secrètes*). Les échanges de clés posent des problèmes de sécurité aussi importants que la transmission du message en lui-même. Ce problème, en effet, limite le domaine d'application du code de Vernam. Plusieurs solutions ont été proposées, basées principalement sur mathématique (système de clé publique). Les physiciens y ont travaillé également et ont proposé une nouvelle méthode basée sur la mécanique quantique.

6.4.1 Photons polarisés

Le protocole de cryptographie quantique est entièrement fondé sur les propriétés quantiques des photons polarisés.

Un photon peut être polarisé selon un axe quelconque. La polarisation est mesurée par un angle entre de 0° à 180°. Dans le protocole proposé par C.H. Bennett et G. Brassard ([137],[138]), la polarisation peut prendre quatre valeurs : 0°, 45°, 90°, 135°. Pour les photons polarisés de 0° à 90°, on parle de polarisation rectiligne, pour ceux polarisés de 45° à 135°, de polarisation diagonale.



Fig. 6.8 – Photon polarisé

http ://www.bibmath.net/crypto/moderne/quantique.php3

Pour détecter la polarisation du photon, on utilise un filtre polarisant suivi d'un détecteur de photons.
- Si un photon polarisé à 0° rencontre un filtre polarisant orienté à 0°, il traverse ce filtre polarisant.
- Si un photon polarisé à 90° rencontre le même filtre, il est stoppé.
- Si le photon est polarisé diagonalement (45° ou 135°), une fois sur deux, il traverse le filtre, et une fois sur deux, il est stoppé. Il est donc impossible de distinguer une polarisation à 45° et à 135° avec un filtre orienté à 0°.

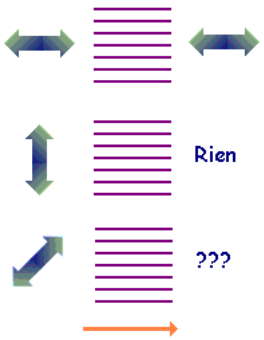

Fig. 6.9 – Traversée d'un filtre rectiligne

http ://www.bibmath.net/crypto/moderne/quantique.php3

De la même façon, on peut utiliser un filtre polarisant orienté à 45° : il laisse passer les

photons polarisés à 45°, stoppe ceux polarisés à 135°, et se comporte aléatoirement avec ceux à 0° et 90° !

6.4.2 Protocole de transmission de la clé

Voici un exemple de protocole permettant de transmettre une clef composée de bits d'Alice à Bob (http ://www.bibmath.net/crypto/moderne/quantique.php3). Ils possèdent deux canaux : un quantique avec lequel ils peuvent échanger des photons polarisés et un canal classique non sécurisé où ils échangent la conversation. On suppose que les photons polarisés à 0° ou 45° représentent bit 0, et ceux polarisés à 90° ou 135° représentent 1. Alice émet, sur le canal quantique, une suite de photons polarisés au hasard parmi 0°, 45°, 90° et 135°. A l'autre bout, Bob reçoit les photons et mesure aléatoirement ou leur polarisation rectiligne (filtre placé à 0°), ou leur polarisation diagonale (filtre placé à 45°). Si le photon traverse le filtre, Bob note 0, sinon il note 1.

Certaines mesures de Bob (en moyenne, une sur deux) n'ont pas d'intérêt : il a pu essayer de mesurer la polarisation rectiligne d'un photon polarisé à 45°, ce qui n'a pas de sens et donne un résultat aléatoire (par exemple, le photon a été bloqué par le filtre, Bob note donc 1 alors qu'Alice avait envoyé 0). Pour éliminer ces bits sans sens, il indique à Alice, par le canal radio, quel type de mesure (rectiligne ou diagonale) il a faite pour chaque photon. Par le même canal radio, Alice lui indique quelles sont les mesures correctes (photon polarisé à 0° ou 90° avec filtre rectiligne, photon à 45° ou 135° avec filtre diagonal), dans l'exemple ci-dessous la 1, la 3, la 4, et la 7. Les bits 1,3,4,7 sont désormais connus à la fois de Bob et d'Alice, et constituent leur clé secrète commune :

Alice émet des photons Valeur en bits	 0	 0	 1	 1	 1	 0	 0	 1
Bob reçoit les photons a tra--vers un filtre Passage? Valeur en bits	 OUI 0	 NON 1	 NON 1	 NON 1	 NON 1	 OUI 0	 OUI 0	 OUI 0
Canal radio: Bob: mon filtre: Alice: correct?	Diagonal oui	Diagonal non	Droit oui	Droit oui	Droit non	Droit non	Droit oui	Diagonal non
Clef reconstituée	0		1	1			0	

FIG. 6.10 – Protocole de transmission de clé dans cryptographie quantique

http ://www.bibmath.net/crypto/moderne/quantique.php3

Théoriquement, la cryptographie quantique est très prometteuse mais notamment, il reste

de nombreux problèmes techniques à résoudre :
– On ne sait pas encore émettre des photons un par un.
– Il est difficile de garder la polarisation sur les grandes distances, le record d'échange de clés a été effectué sur la distance maximale à 67 km par fibre optique.

6.5 Cryptanalyse : quelques attaques connues

À l'opposé des procédés de chiffrement se trouve la cryptanalyse, qui consiste à endosser le rôle d'un espion, tenter de déchiffrer le message sans avoir la connaissance complète des paramètres qui ont permis de le chiffrer. Il existe principalement trois méthodes de cryptanalyse :
– Cryptanalyse à chiffré seul : en plus de la connaissance sur le système plus ou moins complète dont il dispose, le cryptanalyste ne possède qu'un chiffré. Il peut faire des hypothèses sur les messages originaux qu'il ne possède pas. C'est la cryptanalyse plus ardue de par le manque d'informations à disposition.
– Cryptanalyse à clair connu : le cryptanalyste possède, dans ce cas, un couple (clair, chiffré) ; La cryptanalyse linéaire fait partie de cette catégorie.
– Cryptanalyse à clair choisi : le cryptanalyste possède des messages en clair, il a alors la liberté de chiffrer des messages qu'il choisit, et d'observer le cryptogramme correspondant. La cryptanalyse différentielle est un exemple d'attaque à texte clair choisi.

Il existe plusieurs familles d'attaques cryptanalytiques, les plus connues étant l'analyse fréquentielle, la cryptanalyse différentielle, la cryptanalyse linéaire et la cryptanalyse de Jakobsen.
– *Cryptanalyse fréquentielle :* elle examine les répétitions des lettres du message chiffré afin de trouver la clé. Elle est inefficace contre les chiffrements modernes tels que DES, RSA. Elle est principalement utilisée contre les chiffrements mono-alphabétiques qui substituent chaque lettre par une autre et qui présentent un biais statistique.
– *Cryptanalyse linéaire* : introduite par Matsui en 1993 ([153]). C'est une attaque à clairs connus qui s'applique aux chiffrements itératifs par blocs. Cette approche est intéressante car elle se ramène à considérer un problème de codage, plus précisément un problème de reconstruction multivarié.
– *Cryptanalyse différentielle* : proposée par E. Biham et A. Shamiret à la fin des années 1980, cette méthode est une analyse statistique des changements dans la structure de la méthode de chiffrement après avoir légèrement modifié les entrées. Avec un très grand nombre de perturbations, il est possible d'extraire la clé. C'est une méthode générique de cryptanalyse qui peut être appliquée aux algorithmes de chiffrement itératif par blocs, mais également aux algorithmes de chiffrement par flots et aux fonctions de hachage.
– *Cryptanalyse de Jakobsen([148])* les polynômes sont utilisés pour casser les systèmes de chiffrements.

6.6 Optimisation en Cryptologie et nos contributions

Si la théorie des nombres est une approche largement étudiée et utilisée en cryptologie, l'optimisation constitue des outils potentiels et prometteurs qui sont de plus en plus populaires dans ce domaine. Nous donnons dans cette section un bref panorama des problèmes d'optimisation issus des techniques de la cryptologie et les méthodes de résolution existantes.

6.6.1 Cryptanalyses des chiffrements classiques par les techniques d'optimisation méta-heuristique

Comme nous avons vu précédemment, les chiffrements classiques sont basés autour des notions *substitution* et *transposition*. Les systèmes cryptographiques modernes ont maintenant supplanté les chiffrements classiques. Cependant les concepts de base de *substitution* et *transposition* sont toujours largement utilisées (sur les blocs ou bits au lieu des caractères). La cryptanalyse des chiffrements classiques est ainsi, naturellement, l'application la plus populaire des techniques d'optimisation méta-heuristique en cryptologie. Les progrès dans ce domaine étaient assez lents jusqu'à 1993, quand les articles sur les algorithmes génétiques ([152], [165]) sont parus. Spillman et al. ([165]) ont montré comment un chiffrement par substitution peut être attaqué par un algorithme génétique. De manière indépendante, Matthews ([152]) a proposé un algorithme génétique pour les chiffrements par transposition. Depuis cette date, de nombreux travaux ont vu le jour : J. Giddy et R. Safavi-Naini ([149]) ont utilisé le recuit simulé pour attaquer des chiffrements par transposition simple, Jakobsen ([148]) a proposé les attaques des chiffrements par substitution simple et polyalphabétique via les algorithmes hill-climbing. J. Clark et Dawson ont des contributions les plus approfondies dans ce domaine. Leurs travaux couvrent les applications des algorithmes génétiques, de recuit simulé, et les techniques de recherche tabou pour attaquer les chiffrements par substitution, transposition et polyalphabétique ([140], [141], [142], [143]).

6.6.2 Schémas de cryptanalyses basés sur les problèmes standard difficiles

6.6.2.1 Cryptanalyse des schémas de cryptage de knapsack

Le cryptage basé sur le problème de sac à dos est le premier cryptosystème à clef publique. Merkle et Hellman ont proposé d'utiliser un sac à dos (knapsac) croissant comme clé privée, et de le camoufler sous un sac à dos général pour en faire une clé publique. Plusieurs algorithmes génétiques ont été proposés pour la cryptanalyse des chiffrements par knapsacks ([140], [151], [161], [166], ...). Il est à noter que ces algorithmes peuvent traiter seulement des problèmes de petite taille.

6.6.2.2 Protocole d'identification basé sur des problèmes NP-complets

Une preuve à divulgation nulle de connaissance est un concept utilisé en cryptologie dans le cadre de l'authentification et de l'identification. Cette expression désigne un protocole sécurisé dans lequel une entité nommée prouveur, prouve mathématiquement à une autre entité, le vérificateur, qu'une proposition est vraie sans toutefois révéler une autre information que la véracité de la proposition. En pratique, ce schéma se présente souvent sous la forme d'un protocole de type stimulation/réponse. Le vérificateur et le fournisseur de preuve s'échangent des informations et le vérificateur contrôle si la réponse finale est positive ou négative. Ce protocole est proposé par GOLDWASSER et al. en 1985 ([150]) et développé ensuite par FIAT et SHAMIR ([249]). En 1995 POINTCHEVAL a proposé un schéma basé sur le problème "Permuted Perceptron" (PPP) en se basant sur le problème $\mathcal{NP}$-complet "Perceptron" (PP), connu comme un problème difficile, qui doit son existence au fameux problème perceptron dans les réseaux de neurones. Il existe entre autres des schémas d'identification basés sur les problèmes difficiles tels que "syndrome decoding" ([258]) et "Permuted Kernel Problem" ([257]). De nombreux algorithmes métaheuristiques ont été développés pour la résolution de ces problèmes $\mathcal{NP}$-complets difficiles.

6.6.3 Optimisation en cryptographie du $20^{\text{ième}}$ siècle : stratégies de concevoir des systèmes cryptographiques

Pour concevoir des systèmes de chiffrements modernes les techniques S-box et les fonctions Booléennes sont des outils indispensables. La construction des fonctions Booléennes et les S-box ayant des propriétés cryptographiques souhaitées (haut degré de non linéarité, équilibrée, faible de degré d'autocorrélation ...) est un sujet important de la recherche en cryptographie durant des années 90. L'application des méthodes d'optimisation heuristique à cette tâche est un thème émergeant depuis 1995 jusqu'à 2000 (voir par exemple [155], [156], [157], [158]) .

6.6.4 Optimisation dans les systèmes cryptographiques du $21^{\text{ième}}$ siècle : améliorations de conception des systèmes

Depuis 2000, plusieurs travaux importants et intéressants en optimisation heuristique sont développés pour améliorer les techniques S-box. Citons, par exemple, la technique de Hill-climbing de BURNETT et al. ([139]) pour améliorer davantage la performance des S-boxes choisis par IBM, les travaux, de CLARK et JACOB ([144]), MILLAN et al. ([159]), CLARK et al. ([145] ...)

6.6.5 Nos contributions

Nous avons vu dans les paragraphes précédents que l'optimisation constitue des outils efficaces, à la fois classiques et prometteurs pour la cryptologie. La plupart des méthodes d'optimisation existantes dans ce domaine se situent dans le cadre de l'optimisation heuristique. Notre ambition est de construire les modèles et développer les méthodes d'optimisation déterministe (et l'hybridation de deux approches déterministe/métaheuristique), pour la résolution des problèmes difficiles de très grande dimension en cryptologie moderne. Dans cette thèse nous avons choisi en connaissance de cause les trois problèmes importants et d'actualité. Le premier est la construction des fonctions Booléennes équilibrées de haut degré non linéarité - une des techniques de cryptographie du $21^{\text{ième}}$ siècle. Le le deuxième (resp. le troisième) est le problème Perceptron (PP) (resp. "Permuted Perceptron" (PPP)) qui sont parmi les sujets les plus difficiles en cryptanalyse de nos jours.

Chapitre 7

Contributions aux techniques cryptographiques modernes via la construction des fonctions booléennes équilibrées de haut degré de non-linéarité

Résumé *Nous nous sommes intéressés dans ce chapitre à la génération des fonctions booléennes équilibrées de haut degré de non-linéarité qui joue un rôle très important dans la cryptographie moderne. Nous proposons un modèle d'optimisation déterministe qui est la minimisation d'une fonction convexe polyédrale en variables binaires sur un polyèdre convexe. En utilisant un nouveau résultat sur la pénalité exacte nous reformulons ce dernier sous la forme d'une programmation DC et développons DCA pour la résolution. Les techniques d'hybridation de DCA et les algorithmes génétiques sont investies pour la phase d'initialisation de DCA. Les résultats numériques montrent que DCA est une approche prometteuse pour ce problème difficile.*

7.1 Introduction

Les techniques S-box (Substitution boxes) jouent un rôle important dans la cryptographie moderne. S-box désigne une table de substitution (fonction de substitution) utilisée dans un algorithme de chiffrement symétrique. Une S-box prend en entrée une variables de m bits et produit en sortie une variable de n bits (les entrées et les sorties n'ont pas forcément la même taille). Dans un algorithme de chiffrement par blocs, le choix de la fonction de substitution est extrêmement important car il conditionne la résistance du système aux attaques classiques telles que la cryptanalyse différentielle et la cryptanalyse linéaire. Par exemple, DES (Data Encryption Standard) - un des algorithmes les plus connus, qui consiste à faire une substitution et permutation à chaque itération grâce à 8 S-box de taille 16×4.

Ces S-box contribuent à la "confusion" en rendant l'information originale inintelligible, par exemple en cassant la linéarité de la structure de chiffrement. Les valeurs présentées dans les S-Box doivent être choisies de manière à éviter les attaques, par divers moyens parce que la puissante des méthodes de chiffrement dépend de la puissante de la fonction S-box utilisée. Plusieurs études et développements ont été réalisés sur ce problème ces dernières années. Mais parallèlement avec les avancements dans l'utilisation de S-box, l'apparition d'une variété de techniques de cryptanalyse a prouvé que des S-box doivent être conçues avec grand soin.

Dans ce travail, nous construisons les fonctions booléennes (la S-box la plus simple qui donne un seul bit à la sortie) qui constituent des éléments essentiels dans des algorithmes de chiffrement à flots (e.g A5, RC4,...) et des algorithmes de chiffrements par blocs (e.g DES, AES,...). La mise en œuvre d'un tel système qui soit aussi résistant que possible aux attaques nécessite une bonne connaissance des fonctions booléennes qui doivent satisfaire certains critères cryptographiques importants.

Si les critères que doit satisfaire une fonction booléenne varient avec le type de chiffrement dans lequel elle est utilisée, on constate néanmoins que dans presque tous les cas elle doit présenter une haute non-linéarité (elle doit être la plus éloignée possible des fonctions affines). L'étude de la non linéarité des fonctions booléennes a été commencée par les travaux de .F. Dillon([231]) et O.S. Rothaus ([240]) où ils ont introduit les fonctions courbes. Depuis, de nombreuses recherches ont été réalisées ([229], [230], [238], [232]). Toutefois, si la non linéarité est un critère essentiel, il n'est pas le seul. Par exemple, lors qu'une fonction booléenne est utilisée dans un système de *registre à décalage à rétroaction linéaire*, elle doit également être équilibrée et sans corrélation.

La conception d'une fonction booléenne est une tâche difficile, on doit tenir compte un maximum nombre possible de critères. Cependant, ces critères ne sont pas compatibles entre eux, ils ne peuvent pas être satisfaits simultanément. La question est alors de réaliser le meilleur compromis possible entre ces critères.

Dans ce travail, nous considérons les critères de non-linéarité et l'équilibre. Ces deux critères ont été largement étudiés dans [230], [238]. Ils ont été étudiés et résolus avec succès par des approches heuristiques ([229]). Néanmoins il n'existe pas à ce jour des méthodes déterministes. Nous proposons dans ce travail le premier modèle déterministe pour ce problème. Nous le formulons, en premier temps, sous la forme d'un problème de minimisation d'une fonction convexe polyédrale en variables binaires sur un polyèdre convexe. En utilisant la technique de pénalité exacte ([16], [30]), nous le transformons ensuite en une programmation DC et développons DCA pour la résolution. Nous proposons également deux techniques d'hybridation de DCA et un algorithme génétique (AG) pour chercher un bon point initial pour DCA.

Le reste du chapitre est organisé de la façon suivante. Dans la deuxième section, nous introduisons les définitions et propriétés importantes des fonctions booléennes en cryptographie. La troisième section concerne la formulation du problème étudié. La résolution de ce problème par DCA est présentée dans la section 4 tandis que les techniques d'hybridation DCA-AG

sont décrites dans la section 5. Finalement, les résultats numériques sont reportés dans la dernière section.

7.2 Préliminaire

Définition 7.1 *Notons* $B := \{0,1\}$ *et* $B^n := \{0,1\}^n$. *Une fonction booléenne*

$$f : B^n \to B$$

est une application de n variables binaires à une variable binaire.

En général, on présente une fonction booléenne par une tableau de vérité
Exemple : $f : B^2 \to B^1$

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

Si on fixe le tableau de $x = (x_1, x_2, ..., x_n)$, la fonction booléenne f peut être déterminée par la dernière colonne de la table de vérité, autrement dit, par un vecteur zéro-un de taille 2^n. Dans toute la suite, nous considérons une fonction booléenne f comme un vecteur de taille 2^n. La fonction f dans l'exemple ci-dessus n'est rien d'autre que le vecteur $(0,1,1,1)^T$.

Noté $F := \{f : B^n \to B\}$, l'ensemble des fonctions booléennes. Cet ensemble F coïncide donc avec ensemble B^{2^n}.

Définition 7.2 *La fonction polaire d'une fonction booléenne* $f(x)$, *notée* $\widehat{f} = 1 - 2f(x)$, *où* $\widehat{f} \in \{1, -1\}$.

Définition 7.3 *Une fonction booléenne linaire* $L_w(x)$, *paramétrée par* w, *est écrite sous la forme*

$$L_w(x) = wx = w_1x_1 \oplus w_2x_2 \oplus ...w_nx_n. \tag{7.1}$$

Définition 7.4 *Une fonction affine est définie par*

$$A_w(x) = wx \oplus c \text{ où } c \in Z_2. \tag{7.2}$$

Deux notions fondamentales de fonctions booléennes sont la distance Hamming et le poids Hamming.

Définition 7.5 *Le poids Hamming d'une fonction booléenne f est le nombre de* 1 *dans le tableau de vérité (ou dans vecteur f)*

$$hwt(f) := \sum_{x \in B^n} f(x). \tag{7.3}$$

Définition 7.6 *La distance Hamming entre deux fonctions booléennes est définie par le nombre de positions pour lesquelles les deux fonctions sont différentes :*

$$\begin{aligned} d(f,g) &:= \sum_{x \in B^n} f(x) \oplus g(x) \\ &:= (2^n - \sum_{x \in B^n} \widehat{f}(x)\widehat{g}(x)). \end{aligned} \tag{7.4}$$

Définition 7.7 *Le déséquilibre d'une fonction booléenne est définie par :*

$$I_f := \frac{1}{2} \left| \sum_{x \in B^n} \widehat{f}(x) \right|. \tag{7.5}$$

Pour la cryptographie, il est préférable que le nombre de 1 soit égal au nombre de 0 dans la tableau de vérité, elle est dite *équilibrée* dans ce cas. L'équilibre est un critère cryptographique important dans la mesure où une fonction non-équilibrée est corrélée à des fonctions constantes.

Définition 7.8 *La non-linéarité d'une fonction est définie par la distance minimale entre elle et l'ensemble des fonctions affines.*

La non-linéarité est considérée comme étant la plus importante propriété en cryptographie car les fonctions linéaires peuvent être cassées facilement par une variété des méthodes de cryptanalyse. Ainsi la non linéarité est un indicateur important pour évaluer une fonction booléenne.

La transformation Walsh-Hadamard (**W**alsh-**H**adamard **T**ransform) est souvent utilisé pour calculer la non-linéarité car cette méthode offre une complexité $O(n2^n)$ au lieu d'une complexité $O(2^{2n})$ pour la méthode "naïve".

Définition 7.9 *Le WHT d'une fonction Booléene f est défini par :*

$$\widehat{F}(w) := \sum_{x} \widehat{f}(x)\widehat{L}_w(x). \tag{7.6}$$

La non linéarité est donnée par :

$$\begin{aligned} N_f &:= \min_{w \in B^n} \tfrac{1}{2}(2^n - \widehat{F}(w)) \\ &= \min_{w \in B^n} \tfrac{1}{2}(2^n - \sum_{x \in B^n} \widehat{f}(x)\widehat{L}_w(x)) \\ &= 2^{n-1} - \tfrac{1}{2} \max_{w \in B^n} \sum_{x \in B^n} \widehat{f}(x)\widehat{L}_w(x)). \end{aligned} \tag{7.7}$$

7.3 Formulation mathématique et reformulation du problème

Dans la partie précédente, nous avons défini les propriétés d'une fonction booléenne et vu que la non linéarité et l'équilibre sont des indicateurs importants en cryptographie. Le but de ce chapitre est de *trouver une fonction booléenne équilibrée f qui maximise la non linéarité.* La formulation mathématique de ce problème s'écrit comme :

$$\max_{f\in F} N_f = \max_{f\in F}\min_{w\in B^n} \frac{1}{2}\left(2^n - \mid \sum_{x\in B^n} \widehat{f}(x)\widehat{L}_w(x) \mid\right). \tag{7.8}$$

Soit

$$\Phi(\widehat{f}) := \max_{w\in B^n} \mid \sum_{x\in B^n} \widehat{f}(x)\widehat{L}_w(x)\mid, \tag{7.9}$$

on a

$$\max_{f\in F} N_f = 2^{n-1} - \frac{1}{2}\min_{f\in F}\Phi(\widehat{f}). \tag{7.10}$$

Nous voyons que l'opérateur "XOR" intervient dans la définition de $\widehat{L}(w)$, (voir Def.7.4). C'est un opérateur binaire et il n'est pas convenable pour l'optimisation continue. Nous devons exprimer $\widehat{L}(w)$ par une autre formule. Avec une démonstration récursive, nous obtenons

$$\begin{aligned}\widehat{L}_w(x) &= 1 - 2wx = 1 - 2(w_1x_1 \oplus w_2x_2 \oplus ...w_nx_n)\\ &= \begin{cases} 1 \text{ si } \langle w, x\rangle = \sum_{i=1}^{n} w_ix_i \text{ est un nombre pair,}\\ -1 \text{ sinon.}\end{cases}\end{aligned} \tag{7.11}$$

Pour tout w, $x \in B^n$, notons $a_{w,x} := \widehat{L}_w(x) \in \{-1, 1\}$, alors la fonction Φ s'écrit comme :

$$\Phi(\widehat{f}) = \max_{w\in B^n}\left|\sum_{x\in B^n} a_{wx}\widehat{f}(x)\right|. \tag{7.12}$$

Par suite, en remplaçant $\widehat{f}$ par $2f_x - 1$, nous avons

$$\Phi(\widehat{f}) = \max_{w\in B^n}\left|\sum_{x\in B^n} a_{wx}(2f_x - 1)\right| = 2\max_{w\in B^n}\left|\sum_{x\in B^n} a_{wx}f_x - \frac{1}{2}\sum_{x\in B^n} a_{wx}\right|. \tag{7.13}$$

Ce qui donne, en combinant avec (7.10) :

$$\max_{f\in F} N_f = 2^{n-1} - \min_{u\in B^n}\Psi(f), \tag{7.14}$$

où

$$\Psi(f) := \max_{w\in B^n}\left|\sum_{x\in B^n} a_{wx}f_x - \frac{1}{2}\sum_{x\in B^n} a_{wx}\right|. \tag{7.15}$$

Donc, maximiser N_f revient à minimiser $\Psi(f)$ sur B^n.

Pour le critère d'équilibre, on ajoute la contrainte :

$$| \sum_{x \in B^n} f_x - 2^{n-1} | \leq b, \tag{7.16}$$

avec b un nombre non négatif. Si $b = 0$, le critère (7.16) devient *équilibre*. Finalement, nous avons le problème :

$$\beta := \min\{\Psi(f) : 2^{n-1} - b \leq \sum_{x \in B^n} f_x \leq 2^{n-1} + b,\ f \in B^n\}. \tag{7.17}$$

Il est facile de voir que Ψ est une fonction convexe polyédrale. Problème (7.17) est donc la minimisation d'une fonction convexe polyédrale sous contraintes linéaires et en variables binaires.

Reformulation.

Nous venons de voir que (7.17) est un problème d'optimisation en variables binaires. Nous l'allons reformuler en un problème de variables continues. Définissons la fonction :

$$\begin{array}{rccl} p: & I\!R^{2^n} & \rightarrow & I\!R \\ & f & \mapsto & p(f) := \sum\limits_{x \in B^n} min\{f_x, 1 - f_x\}. \end{array} \tag{7.18}$$

Il est clair que p est une fonction concave, non négative sur $[0,1]^n$, et

$$p(f) = 0 \quad \text{ssi} \quad f \in B^n.$$

Par suite le problème (7.17) s'écrit comme :

$$\min \left\{ \Psi(f) : 2^{n-1} - b \leq \sum_{x \in B^n} f_x \leq 2^{n-1} + b,\ p(f) \leq 0 \right\}. \tag{7.19}$$

En utilisant la technique de pénalité exacte ([30], [16]), nous obtenons le problème d'optimisation continue équivalent suivant ($t > 0$ est le paramètre de pénalité) :

$$(Q) \quad \begin{cases} \beta = \min \Psi(f) + tp(f) \\ \text{tel que :} \\ \quad 2^{n-1} - b \leq \sum\limits_{x \in B^n} f_x \leq 2^{n-1} + b, \\ \quad 0 \leq f_x \leq 1,\ \forall x \in B^n. \end{cases}$$

(7.19) et (Q) sont équivalents dans le sens qu'ils possèdent la même valeur optimale et le même ensemble de solutions optimales.

7.4 Résolution du problème (Q) par la programmation DC et DCA

Nous allons, dans le premier temps, reformuler (Q) sous la forme d'un programme DC. Soit K, l'ensemble des solutions réalisables de (Q) :

$$K := \left\{ \begin{array}{l} f : 2^{n-1} - b \leq \sum\limits_{x \in B^n} f_x \leq 2^{n-1} + b, \\ 0 \leq f_x \leq 1, \ \forall x \in B^n \end{array} \right\}. \tag{7.20}$$

Notons χ_K, fonction indicatrice de K :

$$\chi_K(f) = \left\{ \begin{array}{ll} 0 & \text{si } f \in K \\ +\infty & \text{sinon.} \end{array} \right. \tag{7.21}$$

Puisque K est un ensemble convexe, χ_K est une fonction convexe sur $I\!R^{2^n}$.

La décompostion suivante de (Q) nous semble naturelle :

$$\Psi(f) + tp(f) := G(f) - H(f),$$

où $G(f) := \chi_K(f) + \Psi(f)$ et $H(f) := -tp(f)$ (il est clair que G et H sont convexe). (Q) est ainsi un problème DC de la forme

$$(Q_{dc}) \quad \beta := \min\{G(f) - H(f) : \quad f \in I\!R^{2^n}\}.$$

Définissons la fonction ψ_w par

$$\psi_w(f) := \left| \sum_{x \in B^n} a_{wx} f_x - \frac{1}{2} \sum_{x \in B^n} a_{wx} \right|, \tag{7.22}$$

ψ_w est convexe et $\Psi(f) = \max_{w \in B^n} \psi_w(f)$. Ψ est ainsi une fonction convexe polyédrale, par suite G l'est également (χ_K est une fonction polyédrale car K est un polyèdre). D'autre part, comme Ψ, $H := -tp$ est une fonction convexe polyédrale. Finalement (Q_{dc}) est un problème DC polyédral dont les deux composantes DC sont les fonctions polyédrales. Nous verrons dans la suite que cette propriété est intéressante pour la convergence de DCA.

Selon la description de DCA dans le chapitre 1, la résolution de $(Q)_{dc}$ par DCA consiste en la détermination de deux suites $\{v^k\}$ et $\{f^k\}$ telles que :

$$\begin{array}{rcl} v^k & \in & \partial H(f^k) \\ f^{k+1} & \in & \partial G^*(v^k). \end{array}$$

Par la définition de H, nous pouvons choisir v^k comme :

$$v_x^k := \begin{cases} -t & \text{si} \quad f_x^k \leq 0.5, \\ t & \text{sinon.} \end{cases} \tag{7.23}$$

D'autre part, la condition $f^{k+1} \in \partial G^*(v^k)$ est équivalente à :

$$f^{k+1} = \text{argmin}\{\Psi(f) - \langle v^k, f\rangle : f \in K\}. \tag{7.24}$$

De plus, puisque Ψ est convexe polyédrale, le problème (7.24) est en fait équivalent à une programmation linéaire. En effet :

$$\min\{\Psi(f) - \langle v^k, f\rangle : f \in K\} = \min_{f\in K} \max_{w\in B^n} \big(\psi_w(f) - \langle v^k, f\rangle\big)$$

$$\Leftrightarrow \begin{cases} \min & \xi - \langle v^k, f\rangle \\ \text{tel que :} & \\ & \psi_w(f) \le \xi, \forall w \in B^n \\ & f \in K \end{cases} \tag{7.25}$$

$$\Leftrightarrow \begin{cases} \min & \xi - \langle v^k, f\rangle \\ \text{tel que :} & \\ & \sum\limits_{x\in B^n} a_{wx} f_x - \frac{1}{2} \sum\limits_{x\in B^n} a_{wx} \le \xi, \forall w \in B^n \\ & -\sum\limits_{x\in B^n} a_{wx} f_x + \frac{1}{2} \sum\limits_{x\in B^n} a_{wx} \le \xi, \forall w \in B^n \\ & 2^{n-1} - b \le \sum\limits_{x\in B^n} f_x \le 2^{n-1} + b, \\ & 0 \le f_x \le 1, \ \forall x \in B^n. \end{cases} \tag{7.26}$$

Finalement le schéma DCA appliqué à la résolution de (Q_{dc}) décrit comme suivant :

Algorithme 7.1
FB-DCA : DCA appliqué au problème (Q_{dc}).
Initialisation : Choisir $f^0 \in \mathbb{R}^{2^n}$ et une tolérance $\epsilon_{DCA} > 0$; $0 \leftarrow l$.

Répeter
- *Déterminer $v^k \in \partial H(f)$ par (7.23).*
- *Résoudre le problème linéaire (7.26) pour obtenir f^{k+1}.*
- *$k + 1 \leftarrow k$.*

Jusqu'à $\| f^k - f^{k-1} \| < \epsilon_{DCA}$.

Convergence de l'algorithme. Notons Ω, l'ensemble de solutions réalisables du problème linéaire (7.26) et $V(\Omega)$ l'ensemble des sommets de Ω. Soit f^* la solution obtenue par **FB-DCA**. Les propriétés de convergence de **FB-DCA** sont décrites dans le théorème suivant :

Théorème 7.1 *(La convergence de **FB-DCA**)*
*(i) **FB-DCA** génère une suite $\{f^k\}$ appartenant à $V(\Omega)$ telle que la suite $\{\Psi(f^k) + tp(f^k)\}$ est décroissante.*
(ii) Pour un nombre t suffisamment grand, si à l'itération r on obtient $f^r \in \{0,1\}^{2^n}$, alors $f^k \in \{0,1\}^{2^n}$ pour tout $k \ge r$.

(iii) La suite $\{f^k\}$ converge vers $\{f^*\} \in V(\Omega)$ après un nombre fini d'itérations. Le point f^* est un point critique du problème (Q_{dc}). En plus, si $f_x^* \neq \frac{1}{2}$ pour tout $x \in B^n$, alors f^* est une solution locale de (Q_{dc}).

Preuve. i) Ce résultat est la conséquence du Théorème de convergence de DCA (voir le chapitre 1 sur DCA).

ii) Soit

$$t > t_1 := \max\left\{\frac{\Psi(f) - \eta}{\theta} : f \in V(\Omega),\ p(f) \leq 0\right\},$$

où $\eta := \min\{\Psi(f) : f \in V(\Omega)\}$ et $\theta := \min\{p(f) : f \in V(\Omega)\}$.
Soit $\{f^k)\} \subset V(\Omega)$ $(k \geq 1)$, la suite générée par **FB-DCA**. Si $V(\Omega) \subset \{0,1\}^{2^n}$, alors la démonstration est triviale. Sinon, soit $f^r \in \{0,1\}^{2^n}$ et $f^{r+1} \in V(\Omega)$, une solution optimale du problème linéaire (7.26). A partir de (i), on a

$$\Psi(f^{r+1}) + tp(f^{r+1}) \leq \Psi(f^r) + tp(f^r).$$

Par suite, si $p(f^r) = 0$ on a :

$$tp(f^{r+1}) \leq \Psi(f^r) - \Psi(f^{r+1}) \leq \Psi(f^r) - \eta.$$

Supposons, par contradiction, que $p(f^{r+1}) > 0$. Alors

$$t \leq \frac{\Psi(f^r) - \Psi(f^{r+1})}{p(f^{r+1})} \leq \frac{\Psi(f^r) - \eta}{\theta} \leq t_1$$

ce qui est contradiction avec le fait que $t > t_1$.

iii) Puisque (Q_{dc}) est une programmation DC polyédrale, FB-DCA a une convergence finie (voir le chapitre 1 la partie concernant DCA). Selon le théorème général de convergence de DCA,$\{f^*\}$ est un point critique de $G - H$, i.e.

$$\partial G(f^*) \cap \partial H(f^*) \neq \emptyset. \tag{7.27}$$

Si $f_x^* \neq 1/2, \forall x \in B^n$, alors H est différentiable en f^* et la condition (7.27) devient $\partial H(f^*) \subset \partial G(f^*)$. Cette inclusion sous-différentielle est une condition nécessaire et suffisante de l'optimalité locale d'une programmation DC polyédrale dont le deuxième composant H est une fonction convexe polyédrale. La preuve est alors complète. ■

7.5 Combinaison de DCA et Algorithmes Génétiques pour la résolution de (Q_{dc})

Les algorithmes évolutionnaires ont été beaucoup développés durant ces dernières années, parmi lesquels le plus connu et utilisé est l'algorithme génétique qui est inspiré de la loi d'évolution naturelle.

Dans cette partie, nous allons étudier l'algorithme génétique dans le but de trouver un point initial pour DCA. Nous proposons deux schémas de deux phases. Dans le premier schéma nous utilisons un algorithme génétique classique dans la phase 1 puis FB-DCA dans la phase 2, tandis que le deuxième consiste à une combinaison de DCA et algorithme génétique dans la phase 1 puis FB-DCA dans la phase 2.

Pour définir un algorithme génétique, nous avons à définir ses éléments principaux : le codage, la fonction fitness, la population initiale, les opérateurs génétiques (croisement et mutation).

Représentation : Comment représenter les individus est un point important dans les algorithmes génétiques. Nous avons déjà vu qu'une fonction booléenne est présentée par un vecteur binaire. Donc naturellement, nous utilisons le codage binaire pour présenter une fonction booléenne.

Population initiale : Le codage utilisé est le codage binaire, on peut alors utiliser la formule suivant pour déterminer la taille de la population initiale :

$$N \simeq \lceil 1 + \log(-l/\ln P_2^*)/\log 2 \rceil$$

où P_2^* est la probabilité pour qu'au moins un allèle se présente à chaque locus dans la population.

Rappelons que nous considérons les fonctions booléennes équilibrées. Nous générons aléatoirement donc une population initiale qui contient des fonctions équilibrées.

Fonction d'évaluation : Notre but est de maximiser la non linéartié, donc naturellement nous utilisons la définition de la non linéarité comme la fonction d'évaluation.

Sélection :

La sélection utilisée dans notre algorithme génétique est la sélection par tournoi avec $n = 2$. Plus précisément, à chaque fois, on choisit deux individus, celui qui possède meilleure valeur de non-linéartié sera utilisé pour produire les nouveaux individus.

Croisement : Dans le but de conserver l'équilibre des fonctions, nous avons à adopter un croisement tel que à partir de deux fonctions produites à partir de deux fonctions équilibrées le sont également. La procédure est simple, dans chaque parent, on choisit une sous-séquence qui est équilibrée et le croisement est réalisé sur ces deux sous-séquences.

La probabilité de croisement est fixé à $p_c = 0.6$.

Mutation : De même façon que l'opérateur croisement, la mutation est réalisée de telle sorte que la fonction produite reste équilibrée. Pour cela, nous réalisons un nombre pair de mutation de bit. La probabilité est fixé à $p_m = 0.005$.

Condition de terminaison : Comme dans la plupart des algorithmes génétiques, nous utilisons un nombre maximal de générations pour la condition de terminaison. Nous avons constaté à partir des tests numériques qu'après environs 30 itérations, le résultat de notre algorithme génétique n'évolue quasiment plus. On fixe donc le nombre de générations à 40.

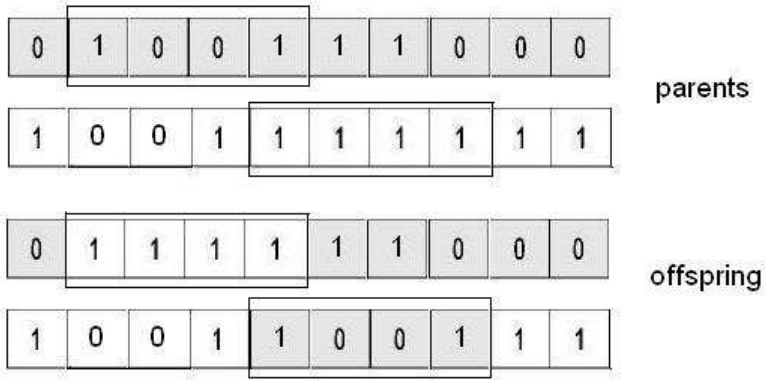

FIG. 7.1 – Opérateur croisement

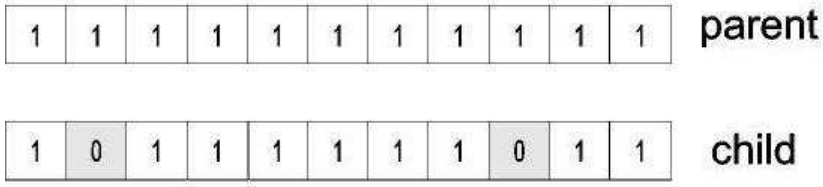

FIG. 7.2 – Opérateur mutation

Combiner DCA-GA

Les éléments principaux de l'algorithme génétique sont définis, nous sommes prêt à l'utiliser. Dans le premier schéma, nous appliquons l'algorithme génétique jusqu'à la fin et lancer DCA avec la solution obtenue précédemment.

Algorithme 7.2
FB-DCA-AG-1 : Combiner DCA et Algorithme Génétique pour la résolution de Q_{dc}

La première phase :
- *Initialisation*
  Choisir le nombre maximal de générations nbIterMax
  Choisir le nombre d'éléments etilismes N_e *(*$0 < N_e < N$*).*
  Générer la population initiale P_t *contenant* N *fonctions équilibrées.*
  Evaluer la population P_t*.*
- *Pour nbIteration allant de 1 à nbIterMax*
  - *Pour i allant de* 1 *à* $(N - N_e)$
    Sélectionner deux individus i_1 *et* i_2 *dans* P_t*.*
    Appliquer le croisement i' = Crossover(i_1*,* i_2*) avec une probabilité* $p_c = 0.6$*.*
    Appliquer la mutation i" = Mutate(i') avec une probabilité $p_m = 0.005$*.*
    Insérer le nouvel individu i" dans P_{t+1}*.*

- *Insérer N_e meilleur individus de P_t à P_{t+1}*
- *$P_t \leftarrow P_{t+1}$*
- *Evaluer la population P_t.*

– *Choisir le meilleur individu de la population finale T^**

La deuxième phase : *Appliquer l'algorithme DCA à partir de point initial T^* jusqu'à sa convergence.*

Dans la deuxième combinaison de DCA et algorithme génétique, nous intégrerons DCA dans le schéma d'algorithme génétique afin de profiter des avantages de DCA.

Algorithme 7.3

FB-DCA-AG-2 : Combiner DCA et Algorithme Génétique pour la résolution de (Q_{dc})

La première phase :

– *Initialisation*
  Choisir le nombre maximal de générations nbIterMax
  Choisir le nombre d'éléments élitismes N_e ($0 < N_e < N$).
  Générer la population initiale P_t contenant N fonctions équilibrées.
  Evaluer la population P_t.

– *Pour nbIteration allant de 1 à nbIterMax*
  - *Pour i allant de 1 à $(N - N_e - 2)$*
    Sélectionner deux individus i_1 et i_2 dans P_t.
    Appliquer le croisement i' = Crossover(i_1, i_2) avec une probabilité $p_c = 0.6$.
    Appliquer la mutation i" = Mutate(i') avec une probabilité $p_m = 0.005$.
    Insérer le nouvel individu i" dans P_{t+1}.
  - *Insérer N_e meilleur individus de P_t à P_{t+1}*
  - *Choisir deux meilleur individus T_1 et T_2 de P_{t+1}.*
  - *Soit N_f^* la meilleur non linéarité actuelle.*
  - *Si $N_f(T_1) \geq N_f^*$(resp. $N_f(T_2) \geq N_f^*$) alors on effectue q itérations de DCA à partir de T_1 (resp. T_2) pour obtenir la solution T_1^*(resp. T_2^*) et l'insérer à P_{t+1}*
  - *$P_t \leftarrow P_{t+1}$*
  - *Evaluer la population P_t.*

– *Choisir le meilleur individu de la population finale T^**

La deuxième phase : *Appliquer l'algorithme DCA à partir de point initial T^* jusqu'à sa convergence.*

Les opérateurs génétiques (croisement et mutation) que nous utilisons dans notre algorithme génétique conservent l'équilibre des fonctions, donc T_1 et T_2 sont des solutions réalisables du problème (7.19). Par ailleurs, puisque DCA est une méthode de descente, la solution T_1^* (resp. T_2^*) est toujours meilleur que T_1 (resp. T_2) en terme de non linéarité.

7.6 Expériences numériques

Nous comparons nos algorithmes avec deux algorithmes efficaces développés dans ([229]) : Hill Climbing (**R HC**) et **GA HC** - une combinaison de algorithme génétique et Hill Climbing. Dans ce travail, Clark (qui est maintenant un grand spécialiste de l'optimisation heuristique pour la cryptologie) a proposé plusieurs approches et prouvé, par de nombreux expériences numériques, que l'algorithme **GA HC** est le meilleur.

Nous utilisons $b = 0$ dans la contrainte d'équilibre, ce qui signifie que les fonctions trouvées sont équilibrées. Pour résoudre le problème linéaire (7.26), nous utilisons le logiciel CPLEX version 10.1.

Nos algorithmes sont implémentés en C++ et testés sur un ordinateur quadruple processeurs Xeon (3.6 MHz, 8MB cache), 32 GM Ram sous système d'exploitation Debian Linux.

Après les premiers tests, nous avons constaté que, dans **FB-DCA-AG-1** et **FB-DCA-AG-2**, nous pouvons arrêter l'algorithme génétique avant 40 itérations. Plus précisément, dans la plupart des cas, quand le nombre d'itérations de l'algorithme génétique dans la phase 1 varie dans l'intervalle $[20, 40]$, nous obtenons le même résultat après la phase 2. A titre d'un exemple, nous présentons dans le tableau 7.1 ci-dessous les résultats de nos algorithmes sur différentes valeurs de nbIterMax dans le cas $n = 9$.

N_g	GA			FB-DCA-AG-1			FB-DCA-AG-2		
	N_f	N^{o}it	Time	N_f	N^{o}it	Time	N_f	N^{o}it	Time
10	228	10	1mn	236	7	3mn	**238**	7	4mn
20	230	20	1.5mn	**238**	6	3mn	**238**	6	4mn
30	232	30	2mn	**238**	6	4mn	**238**	8	5mn
40	232	40	2mn	**238**	6	4mn	**238**	6	5mn

Tab. 7.1 – Résultat sur différentes valeurs de nbIterMax dans l'algorithme génétique ($n = 9$).

Dans le deuxième tableau, nous présentons la meilleure valeur de non linéarité obtenue par les algorithmes avec n allant de 8 à 15. Notons que les résultats de **R HC** et **GA HC** sont obtenus après avoir testé 10000 fonctions, tandis que le nombre des fonctions booléennes générées durant d'exécution de DCA est faible : dans FB-DCA cette quantité est égale au nombre d'itérations de FB-DCA plus le total des nombres d'itérations de CPLEX pour les programmations linéaires (7.26) ; elle ne dépasse pas 100 dans tous les dimensions n allant de 8 à 15.

Dans le troisième tableau, nous présentons le nombre d'itérations de chaque algorithme.

A partir des résultats numériques, nous constatons que :

– DCA est une approche efficace pour ce problème : théoriquement nos schémas de DCA convergent après un nombre fini d'itérations ; pratiquement, le nombre maximum des itérations est 12. Par ailleurs, le degré de non linéarité des fonctions booléennes générées par DCA sont plus haut que celui obtenu par **R HC** et **GA HC**.

n	DCA		FB-DCA-AG-1		FB-DCA-AG-2		GA HC	R HC
	N_f	Time	N_f	Time	N_f	Time	N_f	N_f
8	**120**	1mn	**120**	2mn	**120**	2mn	114	114
9	236	2mn	**238**	4mn	**238**	4mn	236	232
10	486	15mn	486	21mn	**490**	24mn	482	476
11	984	1h	984	1.25h	**988**	2.5h	980	968
12	1984	3h	1998	3.5h	**2008**	5.5h	1980	1961
13	4008	6h	4022	7h	**4024**	8h	3994	3968
14	**8092**	8h	**8092**	9h	**8092**	10.5h	8036	7996
15	16168	11h	16202	12.5h	**16246**	14h	16144	16085

TAB. 7.2 – Non linearité $N_f(7.7)$ pour n de 8 à 15

n	DCA		FB-DCA-AG-1		FB-DCA-AG-2	
	N_f	N^{o}it	N_f	N^{o}it	N_f	N^{o}it
8	**120**	8	**120**	7	**120**	7
9	236	4	**238**	6	**238**	5
10	486	12	486	13	**490**	8
11	984	8	984	5	**988**	5
12	1984	10	1998	13	**2008**	12
13	4008	5	4022	7	**4024**	7
14	**8092**	5	**8092**	3	**8092**	3
15	16168	3	16202	2	**16246**	2

TAB. 7.3 – Nombre d'iterations

– La combinaison **FB-DCA-AG-2** est la meilleure en terme de qualité de solution par rapport aux autres méthodes.
– Pour la phase 1 de **FB-DCA-AG-1** et **FB-DCA-AG-2** il suffit d'executer une vingtaine de générations de AG.

Conclusion.

Pour la première fois dans ce domaine, nous avons pu bâtir un modèle d'optimisation déterministe pour la construction des fonctions booléennes équilibrées de haut degré de non-linéarité. En utilisant un nouveau résultat sur la pénalité exacte, nous avons reformulé le problème en terme d'une programmation DC, ce qui nous permet de développer DCA pour sa résolution numérique. L'algorithme FB-DCA, qui consiste à la résolution d'un problème linéaire à chaque itération, a une convergence finie et fournit une solution entière malgré qu'il fonctionne sur un domaine continu. Ces propriétés fortes intéressantes de FB-DCA justifient son efficacité sur le plan numérique. Pour trouver un bon point initial de FB-DCA nous avons proposé certaines versions de combinaison de DCA et AG. Ces procédures sont efficaces : elles peuvent améliorer les résultats de FB-DCA.

Chapitre 8

Contribution à la cryptanalyse via la résolution des problèmes "Perceptron" et "Permuted Perceptron"

Résumé *Ce chapitre est consacré à l'étude des techniques de cryptanalyse d'un schéma d'identification basé sur les deux problèmes "Perceptron" (PP) et "Perceptron Permuté" (PPP).*

8.1 Introduction

L'identification est un des objectifs essentiels de la cryptologie. Elle permet d'assurer de l'authentification des partenaires et de l'origine des messages. De nous jours, elle s'impose dans tout système de communication.

Le protocole d'identification, le plus courant, est le contrôle par le mot de passe. Le schéma est simple.

– Alice choisit son mot de passe x. Elle envoie x au serveur. Le serveur crypte x par une fonction à sens unique : $y = f(x)$. L'identité d'Alice est stockée alors sur le serveur par le coupe (Alice,y).
– Lorsque Alice souhaite se connecter au serveur, elle fournit son mot de passe x_t. Si $y = f(x_t)$ alors Alice peut accéder au serveur.

Ce protocole est simple et très utilisé. Mais malheureusement, il n'est pas sûr car un simple espionnage permet de récupérer le mot de passe qui circule en clair sur le réseau. Pour améliorer ce protocole, en 1981, Lamport ([253]) a proposé de renouveler le mot de passe à chaque utilisation. Le protocole est le suivant :

– Alice choisit un mot de passe x_0. Elle calcule ensuite $x_1 = f(x_0)$, $x_2 = f(x_1)$, ..., $x_N = f(x_{N-1})$, où f est une fonction à sens unique. Alice envoie x_N au serveur, son identité sur

le serveur est alors (Alice,x_N).
- Lors de la première tentative de se connecter au serveur, Alice envoie x_{N-1} au serveur. Le serveur vérifie son identité et remplace le mot de passe d'Alice par x_{N-1} (l'identité d'Alice après la première connexion au serveur est alors (Alice,x_{N-1})). A la deuxième fois, Alice enverra x_{N-2} au serveur et ensuit de suite.

Le protocole de LAMPORT est évidemment plus sur que le protocole classique mais il possède encore des inconvénients. L'inconvénient le plus important est du au nombre maximal d'utilisation N.

La nouvelle vague de protocoles d'identification est basée sur les preuves interactives *divulgation nulle de connaissance (zero-knowlegde)*. Le but du prouveur est de convaincre le vérificateur qu'il connaît un témoin sans toutefois révéler aucune autre information que la véracité du témoin. En pratique, ce schéma se présente souvent sous la forme d'un protocole de type stimulation/réponse. Le vérificateur et le prouveur s'échangent des informations et le vérificateur contrôle si la réponse finale est positive ou négative. Les protocoles à zero-knowlegde doivent vérifier les propriétés suivantes :

- Consistance : si le prouveur et le vérificateur suivent le protocole alors le vérificateur doit toujours accepter la preuve.
- Solidité : si le témoin est faux, aucun prouveur malicieux ne peut convaincre le vérificateur que le témoin est vrai et ceci avec une forte probabilité.
- Aucun apport d'information (zero knowledge) : le vérificateur n'apprend de la part du prouveur rien de plus que la véracité de témoin, il n'obtient aucune information qu'il ne connaissait déjà sans l'apport du prouveur.

Les deux premières propriétés servent à définir un système de preuve interactive, tandis que la troisième fait la divulgation nulle de connaissance.

Le premier protocole zero-knowlegde, nommé Fiat - Shamir , fut présenté en 1987 par A. FIAT et A. SHAMIR ([249]). Ce protocole est basé sur la théorie des nombres. Son inconvénient principal est qu'il est très coûteux en temps de calcul et en puissance de machine. Il existe également quelques variantes de Fiat-Shamir ([251], [254]) qui tentent d'améliorer la performance en temps de calcul.

Depuis 1989, de nombreux nouveaux schémas sont apparus. Au contraire des protocoles de type Fiat-Shamir, les nouveaux protocoles sont basés sur des problèmes $\mathcal{NP}$-complets et nécessitent que des opérations sur des petits entiers, voire sur des bits : Permuted Kernel Problem ([257]), Syndrome Decoding ([258], [250]), Constrained Linear Equations ([259]), Permuted Perceptrons Problem ([256]).

Dans notre travail, nous nous intéressons au protocole Permuted Perceptrons Problem (PPP), présenté par D. POINTCHEVAL en 1995. PPP est basé sur le problème $\mathcal{NP}$-complet Perceptron Problem (PP) qui doit son existence au fameux problème Perceptron dans les réseaux de neurones. Le problème PP et PPP sont définis ainsi.

Soit $\mathcal{E} := \{-1, 1\}$. Une matrice (resp. vecteur) est dite $\mathcal{E} - matrice$ (resp. $\mathcal{E} - vecteur$) si toutes les composantes de la matrice (resp. vecteur) appartiennent à $\mathcal{E}$.

Définition 8.1.1 ***Perceptron Problem***
Donnée : A : une $\mathcal{E}-$ matrice de taille (m,n).
Objectif : Trouver un $\mathcal{E}$ - vecteur x de taille n tel que $(Ax)_i \geq 0 \quad \forall i = 1..m$. □

Définition 8.1.2 ***Permuted Perceptron Problem***
Donnée : A : une $\mathcal{E}-$ matrice de taille (m,n).
S : Un multi-ensemble des entiers positifs de taille m.
Objectif : Trouver un $\mathcal{E}$ - vecteur x de taille n tel que $\{\{(Ax)_i \mid i = 1..m\}\} = S$. □

Les schémas d'identification PP et PPP nécessitent seulement l'addition et la soustraction sur des entiers, ce qui les rends peu coûteux en temps de calcul et puissance de machine. C'est pour cette raison, PP et PPP sont particulièrement adaptés pour les implantations sur les cartes à microprocesseur.

Dans ces deux problèmes, la matrice A et le multi-ensemble S sont utilisés comme la clef publique d'Alice, tandis que le vecteur x sera gardé pour la clef secrète. La sûreté de ces protocoles repose alors sur la difficulté de trouver la clef secrète (vecteur x) à partir de la clef publique (matrice A et multi-ensemble S).

Pour mettre à l'épreuve la sécurité de schéma PP et PPP, D. Pointcheval a proposé plusieurs types d'attaque, principalement basés sur la méthode recuit simulé ([255]). Evidemment la taille du problème (la taille de la matrice A et multi set S) conditionne la sûreté du problème. Si m et n sont trop grands, le temps de calcul sera élevé. Par contre, si on choisit m et n assez petits, le nombre de solutions possibles sera grand sachant que pour une matrice A et multi-ensemble S donnés, il est forte possible qu'il existe plusieurs vecteurs vérifiant la condition $\{\{(Ax)_i \mid i = 1..m\}\} = S$. D. Pointcheval a pu calculer ce nombre de solutions possibles en fonction de la taille de A et S. Il est donc naturel de choisir ces tailles de telle sorte que le nombre de solutions soit minimum. Pour cela, l'auteur a proposé de choisir $n \approx m + 16$.

La deuxième question concerne la taille minimale que l'on doit considérer. Bien évidemment, cette taille minimale dépend l'efficacité des attaques. Avec ses attaques, D. Pointcheval a réussit à casser un système PPP avec $m = 71, n = 87$. Selon D. Pointcheval, pour une utilisation sûre de PPP, la taille minimale $m = 101, n = 117$ est conseillée. Le challenge est donc de casser le système PPP de taille minimum $m = 101, n = 117$.

En 1999, L.R. Knudsen et W. Meier ont mis en œuvre une nouvelle attaque sur PPP ([252]). Dans ce travail, les auteurs ont proposé une nouvelle méthode recuit simulé suivie par une procédure de recherche avancée. Avec cette nouvelle méthode, les auteurs ont réussit dans certain cas de casser le PPP de la taille $m = 101, n = 117$. De plus, le temps de calcul a été amélioré considérablement par rapport à celui estimé par D. Pointcheval.

Plus récemment, en 2002, J.A. Clark et J.L. Jacob ([248]) ont proposé une étude préliminaire sur une nouvelle attaque basée sur "timing side channel". Les premiers résultats sont intéressants mais il reste beaucoup de choses à améliorer.

Jusqu'à ce jour, parmi les tentatives d'attaques sur les problèmes PP/PPP, aucun modèle d'optimisation déterministe n'a été proposé. Dans ce travail, nous proposons un modèle déterministe et ensuite chercher à le résoudre par une méthode appropriée.

Le reste du chapitre est organisé de la façon suivante. La résolution du problème PP par une approche déterministe basée sur DCA est présentée dans la deuxième section. La troisième section concerne le problème PPP. En plus de DCA avec plusieurs procédures de recherche d'un point initial, nous introduisons une combinaison de DCA et une méthode de coupes.

8.2 La résolution du "Perceptron Problem" (PP)

8.2.1 Modélisation

Rappelons que dans le problème PP, le but est simplement de trouver un $\mathcal{E}$ - vecteur x de taille n tel que $(Ax)_i \geq 0 \quad \forall i = 1..m$. Soient $\mathcal{P}$ l'ensemble des solutions du problème PP et $\mathcal{P}'$ le domaine continu relaxé de $\mathcal{P}$, i.e.

$$\mathcal{P} := \{x \in \{-1,1\}^n : (Ax)_i \geq 0 \quad \forall i = 1..m\}, \quad \mathcal{P}' := \{x \in [-1,1]^n : (Ax)_i \geq 0 \quad \forall i = 1..m\}.$$

Nous pouvons formuler PP comme un problème d'optimisation grâce au résultat suivant :

Proposition 8.1 *Si $\mathcal{P}$ est non vide alors PP est équivalent à l'un des deux problèmes suivants*

$$0 = \min \left\{ p_1(x) := \sum_{i=1}^{n} (1 - x_i^2) : x \in \mathcal{P}' \right\} \tag{8.1}$$

$$0 = \min \left\{ p_2(x) := \sum_{i=1}^{n} \min\{(1 + x_i), (1 - x_i)\} : x \in \mathcal{P}' \right\}. \tag{8.2}$$

Preuve : Si $\mathcal{P}$ est non vide alors $\mathcal{P}'$ l'est également. Dans ce cas les problèmes (8.1) et (8.2) admettent une solution optimale car pour tout x dans $\mathcal{P}'$ on a

$$p_1(x) \geq 0,\ p_2(x) \geq 0.$$

De plus

$$p_1(x) = p_2(x) = 0 \Leftrightarrow x \in \{-1,1\}^n.$$

Par suite x^* est une solution optimale de (8.1) (resp. de (8.2)) si et seulement si $x^* \in \mathcal{P}$.

Notons qu'il est très commode à utiliser DCA à ces deux problèmes car on connaît a priori la valeur optimale, ce qui permet de vérifier la globalité d'une solution donnée par DCA.

Le problème (8.1) est une programmation quadratique concave sous les contraintes linéaires pour laquelle DCA a été intensivement développé dans les travaux de H.A Le Thi et T. Pham Dinh (voir par exemple [5], [6]). Dans ce travail nous considérons le problème (8.2) qui est en fait une programmation DC dont tous les deux composantes DC sont les fonctions polyédrales. Ce qui offre des propriétés intéressantes à DCA, comme on a vu dans le chapitre précédent.

8.2.2 Résolution de (8.2) par DCA

8.2.2.1 Formulation DC de (8.2)

Soit $\chi_{\mathcal{P}'}$ la fonction indicatrice de $\mathcal{P}'$:

$$\chi_{\mathcal{P}'}(x) = \begin{cases} 0 & \text{si } x \in \mathcal{P}' \\ +\infty & \text{sinon.} \end{cases} \tag{8.3}$$

$\mathcal{P}'$ est un ensemble convexe, donc $\chi_{\mathcal{P}'}$ est une fonction convexe sur $\mathbb{R}^n$. Le problème (8.2) s'écrit comme

$$0 = \min \{F_{PP}(x) := \chi_{\mathcal{P}'}(x) - (-p_2(x)) : x \in \mathbb{R}^n\}. \tag{8.4}$$

Une décomposition naturelle de $F_{PP}(x)$ serait

$$F_{PP}(x) = G_{PP}(x) - H_{PP}(x),$$

avec

$$G_{PP}(x) = \chi_{\mathcal{P}'}(x) \tag{8.5}$$

et

$$H_{PP}(x) = -p_2(x) = -\sum_{i=1}^{n} \min\{(1 - x_i), (1 + x_i)\}. \tag{8.6}$$

Les fonctions $G_{PP}(x)$ et $H_{PP}(x)$ sont convexes, donc (PP) est un programme DC. Selon la description de DCA dans le chapitre 1, la résolution de (PP) par DCA consiste en la détermination de deux suites $\{x^l\}$ et $\{y^l\}$ telles que :

$$\begin{array}{lll} y^l & \in & \partial H_{PP}(x^l) \\ x^{l+1} & \in & \partial G^*_{PP}(y^l). \end{array}$$

8.2.2.2 Calculer $y^l \in \partial H_{PP}(x^l)$

Par la définition de H_{PP}, un de ses sous-gradient peut s'écrire comme :

$$y_i^l = \begin{cases} -1 \text{ si } x_i^l \leq 0 \\ 1 \text{ sinon} \end{cases} \quad \text{pour i=1..n.} \tag{8.7}$$

8.2.2.3 Calculer $x^{l+1} \in \partial G^*_{PP}(y^l)$

La condition $x^{l+1} \in \partial G^*_{PP}(y^l)$ est équivalent à :

$$x^{l+1} \quad = \text{argmin}\{G_{PP}(x) - \langle y^l, x\rangle : x \in \mathcal{P}'\}. \tag{8.8}$$

Autrement dit, x^{l+1} est une solution optimale du problème linéaire suivant

$$\min\{-\langle y^l, x\rangle : x \in \mathcal{P}'\}. \tag{8.9}$$

8.2.2.4 Schéma DCA pour la résolution de (8.2)

Algorithme 8.1
PP-DCA-1 : DCA appliqué au problème (8.2)

Initialisation **:** *Choisir un vecteur $x^{(0)} \in \mathbb{R}^n$. Soit une tolérance $\epsilon_{DCA} > 0$, $0 \leftarrow l$.*

Répéter
- *Calculer y^l via (8.7) ;*
- *Calculer $x^{(l+1)}$ en résolvant le problème linéaire (8.9) :*

$$\left\{\min -\langle y^l, x\rangle : x \in \mathcal{P}'\right\}.$$

- $l+1 \leftarrow l$

Jusqu'à $\left\|x^l - x^{(l-1)}\right\| \leq \epsilon_{DCA}(\left\|x^l\right\| + 1)$ ***ou*** $F_{PP}(x^l) = 0$.

Les propriétes de convergence de **PP-DCA-1** se trouvent dans le théorème suivant dont la preuve est analogue à celle du théorème 7.1 du chapitre précédent.

Théorème 8.1 *(La convergence de **PP-DCA-1**)*
(i) ***PP-DCA-1*** *génère une suite $\{x^l\}$ appartenant à l'ensemble des sommets de $\mathcal{P}'$ noté $V(\mathcal{P}')$ telle que la suite $\{p_2(x^l)\}$ soit décroissante.*
(ii) La suite $\{x^l\}$ converge vers $x^* \in V(\mathcal{P}')$ en un nombre fini d'itérations. x^* est un point critique du problème (8.2). En plus, si $x^*_i \neq 0$ (resp. $x_i \in \{-1, 1\}$) pour tout $i \in \{1, ..n\}$, alors x^* est une solution locale (resp. globale) de (8.2).

8.2.2.5 Un schéma de redémarrage de DCA

Nous avons constaté, à travers des résultats numériques de **PP-DCA-1**, qu'après quelques itérations, **PP-DCA-1** fournit une solution dont la plupart des composantes sont dans $\{-1, 1\}$. Pour obtenir une solution globale, nous proposons une procédure qui aide **PP-DCA-1** à échapper aux optimaux locaux en changeant les composantes de la solution courante n'étant pas dans $\{-1, 1\}$. La procédure est décrite comme suivant :

Algorithme 8.2
PP-DCA-2 : DCA appliqué au problème (PP)

Initialisation **:** *Choisir un vecteur* $x^0 \in \mathbb{R}^n$. *Soit une tolérance* $\epsilon_{DCA} > 0$, $0 \leftarrow l$.

Répéter
- *Calculer* y^l *via (8.7);*
- *Calculer* $x^{(l+1)}$ *en résolvant le problème linéaire (8.9) :*

$$\left\{\min -\langle y^l, x\rangle; x \in \mathcal{P}'\right\}.$$

- *Soit* i *le premier indice pour lequel* x_i *n'est pas dans* {*-1,1*}
  Si $0 \leq x_i < 1$ *alors* $x_i \leftarrow 1$
  Si $0 > x_i > -1$ *alors* $x_i \leftarrow -1$
- $l + 1 \leftarrow l$

Jusqu'à $\left\|x^l - x^{(l-1)}\right\| \leq \epsilon_{DCA}(\left\|x^l\right\| + 1)$ ***ou*** $F_{PP}(x^l) = 0$.

8.2.3 Expériences numériques

Nos algorithmes sont implémentés en C++ et testés avec un ordinateur Pentium 4 2.8 MHz, 1 GM Ram. Le logiciel CPLEX version 9.1 est utilisé pour la résolution des sous-problèmes linéaires.

Nous comparons nos algorithmes avec la méthode recuit simulé présentée par D. Pointcheval ([255]).

Dans le tableau 8.1, nous présentons le résultat de test de deux méthodes pour le problème PP avec les tailles différentes. Pour chaque problème, 100 instances ont été créées de façon aléatoire. Les critères de comparaison sont : Time - le temps de calcul (le temps moyen en seconde dans les cas de réussite) et PR - le pourcentage de réussite sur 100 tests.

On constate que ,sur le plan de qualité des solutions, les deux algorithmes sont comparables. Par contre, concernant la rapidité, **DCA-PP-2** représente un avantage incontestable : jusqu'à 40 fois plus rapide que **RC**.

8.3 La résolution du "Permuted Perceptron Problem" (PPP)

8.3.1 Modélisation

A la différence de PP, dans le PPP, nous avons une contrainte de plus :

$$\{\{(Ax)_i \mid i = 1..m\}\} = S.$$

m	n	PP-DCA-2		RC	
		Time(s)	PR(%)	Time(s)	PR(%)
101	117	0.26	100	3.2	100
121	137	0.32	97	4.5	100
151	167	0.51	100	6.2	100
171	187	0.87	99	6.0	100
201	217	1.1	95	42	100
301	317	3.85	96	114.5	100
401	417	6.23	91	181.3	98
501	517	12.21	100	287.3	97
701	717	27.75	91	800	95
851	867	42	88	1620	91
1001	1017	98.2	99	2981	93

TAB. 8.1 – Comparaison entre **PP-DCA-2**, **RC** sur 100 instances du problème PP

Cette contrainte rend le problème plus difficile. Pour modéliser PPP nous représentons le multi-ensemble S par les deux vecteurs suivants :
- s^* : vecteur qui contient l'ensemble des éléments distincts de S. On note p, le nombre des composantes de s^*.
- c : vecteur qui contient le nombre d'occurrence de chaque élément distinct de S. Plus précisément, pour chaque composante s_i^* de s^*, c_i est le nombre d'occurrence de s_i^* dans S. La taille de c est bien égale à celle de s^*.

Par exemple : le multi-ensemble $S = \{1\ 3\ 3\ 5\ 1\ 3\}$ est représenté par les deux vecteurs :

$$\begin{array}{rcl} s^* & = & \{1\ \ 3\ \ 5\} \\ c & = & \{2\ \ 3\ \ 1\}. \end{array}$$

Pour exprimer la relation entre $\{\{(Ax)_i \mid i = 1..m\}\}$ et S, nous introduisons une nouvelle variable - une matrice y de taille (m,p) telle que :

$$y_{i,j} = \begin{cases} 1 \text{ si } A_i x = s_j^* \\ 0 \text{ sinon} \end{cases} \quad \text{i=1..m, j=1..p.} \tag{8.10}$$

Par exemple : pour le multi-ensemble $S = \{1\ 3\ 3\ 5\ 1\ 3\}$ et $Ax = \{1\ 5\ 3\ 3\ 5\ 4\}$, alors la matrice y correspondante est la suivante :

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Pour que $\{\{(Ax)_i \mid i = 1..m\}\} = S$, chaque $(Ax)_i$ doit correspondre à une et une seule composante de s^*, ce qui signifie que sur chaque ligne de la matrice y on doit avoir un et un seul élément égale à 1, i.e.

$$\sum_{j=1}^{p} y_{i,j} = 1, \quad \forall i = 1..m. \tag{8.11}$$

Cette condition assure que $\{(Ax)_i \mid i = 1..m\}$ est un sous ensemble de S. D'un autre côté, pour chaque s_j^*, on doit avoir exactement c_j éléments dans $\{(Ax)_i\}$ tels que $(Ax)_i = s_j^*$. Cette contrainte est exprimée par :

$$\sum_{i=1}^{m} y_{i,j} = c_j, \quad \forall j = 1..p. \tag{8.12}$$

Soient $s_{max} := \max\{s_j^* : j = 1..p\}$ et $s_{min} := \min\{s_j^* : j = 1..p\}$. La relation (8.10) est exprimée par :

$$\begin{cases} A_i x \leq & s_{max} - (s_{max} - s_j^*) y_{i,j} \quad \forall i = 1..m, j = 1..p \\ A_i x \geq & s_{min} + (s_j^* - s_{min}) y_{i,j} \quad \forall i = 1..m, j = 1..p \end{cases} \tag{8.13}$$

C'est à dire : si $y_{i,j} = 1$, alors on a forcément $(Ax)_i = s_j^*$. Dans le cas $y_{i,j} = 0$, on impose simplement que la valeur de $(Ax)_i$ doit être comprise dans $[s_{min}, s_{max}]$.

Soit K_{PPP} l'ensemble des solutions de PPP. On peut écrire alors

$$K_{PPP} = \left\{ \begin{array}{l} (x, y) \in \{-1, 1\}^n \times \{0, 1\}^{m \times p} : \\ \sum\limits_{k=1}^{n} a_{i,k} x_k \leq s_{max} - (s_{max} - s_j^*) y_{i,j} \quad \forall i = 1..m, j = 1..p \\ \sum\limits_{k=1}^{n} a_{i,k} x_k \geq s_{min} + (s_j^* - s_{min}) y_{i,j} \quad \forall i = 1..m, j = 1..p \\ \sum\limits_{j=1}^{p} y_{i,j} = 1, \quad \forall i = 1..m \\ \sum\limits_{i=1}^{m} y_{i,j} = c_j, \quad \forall j = 1..p \end{array} \right\} \tag{8.14}$$

De manière analogue au problème PP, on peut facilement démontrer que PPP est équivalent au problème d'optimisation suivant :

$$0 = \min \left\{ \sum_{i=1}^{n} \min\{(1 + x_i), (1 - x_i)\} : (x, y) \in K'_{PPP} \right\} \tag{8.15}$$

où K'_{PPP} est le domaine relaxé relative à variable x de K_{PPP}, i.e. on remplace dans (8.14) la contrainte $x \in \{-1, 1\}^n$ par $x \in [-1, 1]^n$.

Dans la suite, nous allons reformuler (8.15) sous la forme d'un programme DC polyédrale et appliquer DCA pour la résolution.

8.3.2 Résolution de (8.15) par DCA

Dans (8.15), x est une variable continue, mais y reste encore discrète. Dans l'objectif de transformer ce problème en un problème de variable continue, nous définissions la fonction pénalité de y :

$$\begin{array}{rccl} p: & I\!R^{m\times p} & \to & I\!R \\ & y & \mapsto & p(y) := \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{p} \min\{y_{i,j}, (1-y_{i,j})\}. \end{array} \tag{8.16}$$

p est une fonction concave non négative sur $[0,1]^{m\times p}$, de plus :

$$p(y) = 0 \quad \text{ssi} \quad y \in \{0,1\}^{m\times p}.$$

En utilisant le résultat de pénalité exacte ([33]) on obtient le problème suivant qui est équivalent à (8.15) ($t > 0$ est le paramètre de pénalité) :

$$(PPP_t) \qquad \min\left\{ \sum_{i=1}^{n} \min\{(1+x_i),(1-x_i)\} + t\sum_{i=1}^{m}\sum_{j=1}^{p} \min\{y_{i,j},(1-y_{i,j})\} : (x,y) \in \mathcal{K}_{PPP} \right\}$$

où

$$\mathcal{K}_{PPP} = \left\{ \begin{array}{l} (x,y) \in [-1,1]^n \times [0,1]^{m\times p} : \\ \sum\limits_{k=1}^{n} a_{i,k}x_k \le s_{max} - (s_{max} - s_j^*)y_{i,j} \quad \forall i = 1..m, j = 1..p \\ \sum\limits_{k=1}^{n} a_{i,k}x_k \ge s_{min} + (s_j^* - s_{min})y_{i,j} \quad \forall i = 1..m, j = 1..p \\ \sum\limits_{j=1}^{p} y_{i,j} = 1, \quad \forall i = 1..m \\ \sum\limits_{i=1}^{m} y_{i,j} = c_j, \quad \forall j = 1..p \end{array} \right\}. \tag{8.17}$$

8.3.2.1 Formulation DC de (PPP_t)

Le problème (PPP_t) peut s'écrire encore sous la forme

$$(PPP_t) \quad \min\left\{ \chi_{\mathcal{K}_{PPP}}(x,y) - H_{PPP}(x,y) : (x,y) \in I\!R^n \times I\!R^{m\times p} \right\},$$

où

$$H_{PPP}(x,y) = -\sum_{i=1}^{n} \min\{(1+x_i),(1-x_i)\} - t\sum_{i=1}^{m}\sum_{j=1}^{p} \min\{y_{i,j},(1-y_{i,j})\}. \tag{8.18}$$

La fonction objectif de (PPP_t), notée F_{PPP}, a une décomposition DC naturelle suivante :

$$F_{PPP}(x,y) = G_{PPP}(x,y) - H_{PPP}(x,y),$$

où

$$G_{PPP}(x,y) = \chi_{\mathcal{K}_{PPP}}(x,y). \tag{8.19}$$

Il est clair que $G_{PPP}(x,y)$ et $H_{PPP}(x,y)$ sont les fonctions convexes polyédrales, (PPP_t) est donc un problème DC polyédral dont tous les deux composantes DC sont polyédrales.
Selon la description de DCA dans le chapitre 1, la résolution de (PPP_t) par DCA consiste en la détermination de deux suites $\{(x,y)^l\}$ et $\{(u,v)^l\}$ telles que :

$$\begin{array}{rcl}(u^l, v^l) & \in & \partial H_{PPP}(x^l, y^l)\\ (x^{l+1}, y^{l+1}) & \in & \partial G^*_{PPP}(u^l, v^l).\end{array}$$

8.3.2.2 Calculer $(u^l, v^l) \in \partial H_{PPP}(x^l, y^l)$

Par la définition de H_{PPP}, on peut déterminer (u^l, v^l) comme suit :

$$u_i^l = \begin{cases} -1 \text{ si } x_i^l \leq 0 \\ 1 \text{ sinon} \end{cases} \quad \text{pour i=1..n} \tag{8.20}$$

et

$$v_{i,j}^l = \begin{cases} -t \text{ si } y_{i,j}^l \leq 0.5 \\ t \text{ sinon} \end{cases} \quad \text{pour i=1..m, j=1..p.} \tag{8.21}$$

8.3.2.3 Calculer $(x^{l+1}, y^{l+1}) \in \partial G^*_{PPP}(u^l, v^l)$

La condition $(x^{l+1}, y^{l+1}) \in \partial G^*_{PPP}(u^l, v^l)$ est équivalent à :

$$(x^{l+1}, y^{l+1}) \in \text{argmin}\{-\langle (u^l, v^l), (x,y)\rangle : (x,y) \in \mathcal{K}_{PPP}\}. \tag{8.22}$$

8.3.2.4 Schéma DCA pour la résolution de (PPP_t)

PPP-DCA : DCA appliqué au problème (PPP_t)

Algorithme 8.3

Initialisation : Choisir un vecteur $x^0 \in \mathbb{R}^n$. Calculer la matrice y^0 à partir de x^0 via (8.10). Soit une tolérance $\epsilon_{DCA} > 0$, un paramètre de pénalité $t > 0$, $0 \leftarrow l$.

Répéter

– *Calculer* (u^l, v^l) *selon (8.20) et (8.21) ;*
– *Calculer* $(x^{(l+1)}, y^{(l+1)})$ *en résolvant le problème linéaire suivant :*

$$\min\left\{-\langle x, u^l\rangle - \langle y, v^l\rangle : (x,y) \in \mathcal{K}_{PPP}\right\}.$$

– $l+1 \leftarrow l$
Jusqu'à $\left\|(x^l, y^l) - (x^{(l-1)}, y^{(l-1)})\right\| \leq \epsilon_{DCA}(\left\|(x^l, y^l)\right\| + 1)$ ***ou*** $F_{PPP}(x^l, y^l) = 0$

De manière analogue aux théorèmes 8.1 et 7.1, on a

Théorème 8.2 *(La convergence de* ***PPP-DCA****)*
(i) ***PPP-DCA*** *génère une suite* $\{(x^l, y^l)\}$ *appartenant à l'ensemble des sommets de* $\mathcal{K}_{PPP}$ *noté* $V(\mathcal{K}_{PPP})$ *telle que la suite* $\{F_PPP(x^l, y^l)\}$ *soit décroissante.*
(ii) La suite $\{(x^l, y^l)\}$ *converge vers* $(x^*, y^*) \in V(\mathcal{K}_{PPP})$ *après un nombre fini d'itérations.* (x^*, y^*) *est un point critique du problème* (PPP_t)*. En plus, si* $x^*_k \neq 0$ *et* $y^*_{i,j} \neq 0.5$ *(resp.* $x^*_k \in \{-1, 1\}$ *et* $y^*_{i,j} \in \{0, 1\}$*) pour tout* $k \in \{1, ..n\}$*,* $i = 1..m, j = 1..p$*, alors* (x^*, y^*) *est une solution locale (resp. globale) de* (PPP_t)*.*
(iii) Pour un nombre t *suffisamment grande, si à l'itération* r *on obtient* $y^r \in \{0,1\}^{m\times p}$*, alors* $y^l \in \{0,1\}^{m\times p}$ *pour tout* $l \geq r$*.*

8.3.3 Chercher un point initial

La recherche d'un bon point initial est toujours une question importante dans la résolution d'une programmation DC par DCA. Elle dépend de la structure du problème. Dans cette partie, nous allons étudier différentes méthodes pour chercher un point initial.

8.3.3.1 Le vecteur majorité

On définit un vecteur majorité M comme suit :

$$\begin{cases} M_j = 1 si \mid \{i \mid A_{i,j} = 1\} \mid > \frac{m}{2} \\ M_j = -1 \text{ sinon.} \end{cases} \tag{8.23}$$

Ce vecteur, proposé par D. Pointcheval, est considéré comme étant la première tentative d'attaque aux problème PP et PPP. En effet, D. Pointcheval a montré que le cas où m et n sont impair, la corrélation entre vecteur majorité M et vecteur secret x est très grande.

$$\{j \mid M_j = x_j, j = 1..n\} \approx 0.8n.$$

La première attaque consiste donc à changer 20% des composantes du vecteur M. Bien évidemment, il y a $\begin{pmatrix} n \\ 0.2n \end{pmatrix}$ possibilité de choisir les composantes de M à changer. Cette attaque n'est pas efficace pour les problèmes de grande taille. Par exemple, pour $n \geq 95$, la

complexité de cette attaque est supérieure à 2^{64}. D. POINTCHEVAL a également proposé une variante de cette attaque en groupant en priorité les composantes de M dont les valeurs sont litigieuses.

Malgré que cette attaque n'est pas efficace, la forte corrélation entre M et vecteur secret pourrait apporter un atout à nos algorithmes. On nomme alors **IP1** la procédure qui choisit le vecteur majorité M comme le point initial.

8.3.3.2 La solution du problème relaxé de (8.15)

Dans plusieurs problèmes en variables binaires, nous avons constaté qu'une façon efficace de choisir le point initial est de prendre la solution du problème relaxé du problème considéré. Dans ce travail nous choisissons, comme le point initial, la solution du problème relaxé de (8.15) suivant :

$$\min \left\{ \sum_{i=1}^{n} \min\{(1+x_i),(1-x_i) : (x,y) \in \mathcal{K}_{PPP} \right\}. \tag{8.24}$$

Nous utilisons encore DCA pour la résolution de ce problème.

On nomme **IP2**, cette procédure pour choisir le point initial.

8.3.3.3 Algorithme génétique

Vu le succès de l'approche d'hybridation DCA-GA du chapitre précédent nous développons un schéma de l'algorithme génétique pour la recherche d'un poin initial de PPP-DCA. Nous allons maintenant définir les éléments principaux de notre algorithme génétique, à savoir : le codage, la fonction fitness, la population initiale, les opérateurs génétiques (croisement et mutation).

Représentation :

Tout d'abord, nous allons effectuer une transformation de variable x en une variable zéro-un. Les fonctions de transformation sont définies comme suivant :

$$\begin{array}{rccc} \phi: & \mathcal{E} & \to & \mathcal{B} \\ & x & \mapsto & \frac{x+1}{2} \end{array} \qquad \begin{array}{rccc} \varphi: & \mathcal{B} & \to & \mathcal{E} \\ & x & \mapsto & 2x-1 \end{array} \tag{8.25}$$

où $\mathcal{B} = \{0,1\}$.

x et y sont maintenant les variables zéro-un, donc le codage binaire est convenable pour présenter les individus. Un individu sera présenté par un vecteur binaire de taille $n+mp$ dont :

– n premiers composantes correspondent au vecteur x.

– mp composantes suivantes correspondent aux m lignes de la matrice y placées une après l'autre.

Population initiale : Le codage utilisé est le codage binaire, on peut alors utiliser la formule suivant pour déterminer la taille de la population initiale :

$$N \simeq \lceil 1 + \log(-l/\ln P_2^*)/\log 2 \rceil .$$

où P_2^* est la probabilité pour qu'au moins un allèle se présente à chaque locus dans la population.

D'autre part, la matrice y est déterminée à partir de x, donc pour créer un individu, on génère seulement un vecteur x et ensuite calcule y à partir de x via (8.10).

Fonction d'évaluation : ayant plusieurs contraintes nous ne pouvons pas assurer que les individus respectent bien ces contraintes. Alors, la fonction d'adaptation doit prendre en compte la pénalisation de ces contraintes. On peut définir la fonction d'adaptation comme suit :

$$\begin{aligned} F_a(x,y) \;=\; & \sum_{i=1}^{n} x_i(1-x_i) + \sum_{i=1}^{n}\sum_{j=1}^{p} y_{i,j}(1-y_{i,j}) + \\ & \sum_{i=1}^{m}\sum_{j=1}^{p}\left(2\sum_{k=1}^{n} a_{i,k}x_k - \sum_{k=1}^{n} a_{i,k} - s_{max} + (s_{max}-s_j^*)y_{i,j}\right) + \\ & \sum_{i=1}^{m}\sum_{j=1}^{p}\left(-2\sum_{k=1}^{n} a_{i,k}x_k + \sum_{k=1}^{n} a_{i,k} + s_{min} + (s_j^*-s_{min})y_{i,j}\right) + \\ & \sum_{i=1}^{m}\left(1-\sum_{j=1}^{p} y_{i,j}\right) + \sum_{j=1}^{p}\left|c_j - \sum_{i=1}^{m} y_{i,j}\right| . \end{aligned} \tag{8.26}$$

Sélection : La sélection utilisée dans notre algorithme génétique est la sélection par tournoi avec $n = 2$. Plus précisément, à chaque fois, on choisit deux individus, celui qui possède la plus petite valeur de fonction d'adaptation sera utilisé pour produire les nouveaux individus.

Croisement : On applique un croisement en deux points classique sur la partie x. La partie y est calculée en suite par via 8.10. La probabilité de croisement est fixé à $p_c = 0.7$.

Mutation : De même façon que l'opérateur croisement, la mutation est réalisée seulement sur la partie x. La probabilité est fixé à $p_m = 0.005$.

Condition de terminaison : Comme dans la plupart des algorithmes génétiques, nous utilisons un nombre maximal de générations pour la condition de terminaison. On fixe le nombre d'itérations à 100.

Algorithme 8.4
IP3 : Algorithme Génétique pour chercher un point initial pour PPP

– *Initialisation*
Choisir le nombre d'éléments etilismes N_e ($0 < N_e < N$).

Générer la population initiale P_t contenant N individus.
Evaluer la population P_t.
- *Pour nbIteration allant de 1 à nbIterMax*
  - *Pour i allant de 1 à $(N - N_e)$*
    Sélectionner deux individus i_1 et i_2 dans P_t.
    Appliquer le croisement i' = Crossover(i_1, i_2) avec une probabilité $p_c = 0.7$.
    Appliquer la mutation i" = Mutate(i') avec une probabilité $p_m = 0.005$.
    Insérer le nouveau individu i" dans P_{t+1}.
  - *Insérer N_e meilleur individus de P_t à P_{t+1}*
  - *$P_t \leftarrow P_{t+1}$*
  - *Evaluer la population P_t.*

8.3.4 DCA et méthode de coupes

Dans cette partie, dans le but de trouver une solution globale du problème, nous utilisons une approche d'optimisation globale. Les techniques d'optimisation globales ont connu de très nombreux développements importants au cours de ces dernières années. Théoriquement elles permettent d'obtenir une solution globale, néanmoins elles possèdent deux inconvénients majeurs : leur lourdeur (encombrements en places mémoires) et leur coût trop important.

Récemment, en 2005, une nouvelle méthode globale DCA&CUT basée sur des nouvelles coupes et DCA a été introduite dans [34], [51] pour la résolution d'une programmation DC polyédrale en variables mixtes zéro-un. L'idée d'utilisation d'une approche continue (en utilisant un problème équivalent de variables continues via la pénalité exacte) pour un problème combinatoire a donné naissance à ces coupes. Dans cette approche, DCA joue un rôle crucial : d'une part, il peut fournir des très bonnes solutions réalisables du problème combinatoire bien qu'il travaille sur un domaine continu, (dans plusieurs cas DCA trouve des solutions optimales globales, par exemple, pour le problème de Complémentarité Linéaire) ; d'autre part, les coupes sont déterminées à partir des solutions fournies par DCA. Réciproquement, ces coupes, en réduisant le domaine de la relaxation continue, renforcent la convergence de DCA vers une solution globale, et finalement prouvent la globalité de DCA. DCA&CUT a été appliqué avec succès dans plusieurs modèles d'optimisation, à savoir : programmation linéaire en variables mixtes zéro-un, programmation DC polyédrale en variables mixtes zéro-un, programmation quadratique en variables mixtes zéro-un.

En considérant PPP comme le problème d'optimisation (8.15) qui n'est rien d'autre qu'une programmation DC polyédrale en variables zéro-un, nous développons dans la suite un schéma DCA&CUT pour le résoudre.

DCA&CUT pour la résolution de (8.15)

Rappelons le problème (8.15) à résoudre

$$0 = \min \left\{ \sum_{i=1}^{n} \min\{(1 + x_i), (1 - x_i)\} : (x, y) \in K'_{PPP} \right\}$$

et notons que $K'_{PPP} = \{0,1\}^{m\times p} \cup \mathcal{K}_{PPP}$.

L'algorithme DCA&CUT est appliqué au problème continu (PPP_t) obtenu par la pénalité exacte :

$$(PPP_t) \quad \min\left\{\sum_{i=1}^{n}\min\{(1+x_i),(1-x_i)\} + t\sum_{i=1}^{m}\sum_{j=1}^{p}\min\{y_{i,j},(1-y_{i,j})\} : (x,y)\in\mathcal{K}_{PPP}\right\}.$$

Dans le schéma DCA&CUT, la solution fournie par DCA sera utilisée pour déterminer les coupes. La construction de coupes peut être résumée comme ci-dessus :

Soit $z^* = (x^*, y^*)$, une solution de $(PPP)_t$ donnée par DCA. On distingue deux cas possibles.

– Dans le cas où $p(z^*)$ n'est pas entière, la coupe est déterminée par le théorème suivant :
Théorème 8.3 *Soit $z^* = (x^*, y^*)$ un minimum local du problème (PPP_t) tel que la valeur $p(z^*)$ non entière. Alors l'inégalité :*

$$l_{z^*}(z) \geq \lfloor p(z^*)\rfloor + 1$$

est une coupe séparant strictement z^* de K'_{PPP},

où la foncton $l_{z^*}(z)$ est définie par :

$$l_{z^*}(z) = l_{z^*}(y) = \sum_{(i,j)\in I_0(z^*)} y_{ij} + \sum_{(i,j)\in I_1(y^*)}(1-y_{ij}) \tag{8.27}$$

avec

$$\begin{array}{ll} I_0(z^*) & = \{(i,j)\in\{\{1,2,...,m\}\times\{1,2...,p\}\} : y^*_{i,j}\leq 1/2\} \\ I_1(z^*) & = \{\{1,2,...,m\}\times\{1,2...,p\}\}\backslash I_0 \end{array} \tag{8.28}$$

– Dans le cas contraire, c'est à dire $p(z^*)$ est entière, en appliquant la procédure suivante, nommée Procédure P, on obtient soit une coupe soit une solution réalisable de (8.15) :
Algorithme 8.5
Procédure P
Entrée : un mininum local non réalisable $z^* = (x^*, y^*)$ tel que $p(z^*) > 0 \in Z$.
Sortie : soit une coupe, soit une solution réalisable.
Etape 1
Poser $k \leftarrow 0, K_0 \leftarrow \mathcal{K}_{PPP}$.
Si $p(z^*) \notin Z$, déterminer la coupe via le théorème 8.3, fin de la procédure.
Sinon, poser $I_F \leftarrow I_F(z^*)$.
où $I_F(z^*) = \{(i,j)\in I = \{1,2,...,m\}\times\{1,2,...,p\} : y^*_{i,j}\notin\{0,1\}\}$, l'ensemble des indices pour lesquels $y^*_{i,j}$ n'appartiennent pas à $\{0,1\}$.
Etape 2
Si $I_F = \emptyset$, déterminer la coupe via le théorème 8.3, fin de la procédure.
Sinon
Poser $k \leftarrow k+1$ et choisir $(i_k, j_k)\in I_F$.

$\mu_0 \leftarrow min\{l_{z^*} : z \in K_k, y_{i_k,j_k} = 0, \}(P_{k0})$. *Soit* $v^0 = argmin(P_{k0})$.
$\mu_1 \leftarrow min\{l_{z^*} : z \in K_k, y_{i_k,j_k} = 1, \}(P_{k1})$. *Soit* $v^1 = argmin(P_{k1})$.
$\mu \leftarrow \min\{\mu_0, \mu_1\}$.
Si $v^0 \in K'_{PPP}$ *ou* $v^1 \in K'_{PPP}$, *fin de la procédure. Sinon passer à l'etape 3.*

Etape 3
Si $l_{z^*} < \mu$, *déterminer la coupe via le théorème 8.3, fin de la procédure.*
Sinon passer à l'étape 4.

Etape 4
Si $\mu_0 = \mu_1$ *alors :*
Soit $\overline{v} \in \{v_0, v_1\}$ *tel que* $\overline{v} \notin R(z^*)$.
Trouver un minimum local $\hat{v}$ *satisfaisant* $p(\hat{v}) < p(z^*)$.
Si $\hat{v} \in K'_{PPP}$, *fin de la procédure. Sinon retourner à l'étape 1.*
Sinon
Si $\mu_0 < \mu_1$, *poser* $K_{k+1} \leftarrow \{z \in K_k : y_{i_k,j_k} = 0\}$.
Sinon, poser $K_{k+1} \leftarrow \{z \in K_k : y_{i_k,j_k} = 1\}$.
Retourner à l'étape 2.

Dans notre problème (PPP), nous connaissons déjà la valeur optimale du problème. On impose alors la borne inférieure égale à la valeur optimale de (PPP). La borne superieure sera mise à jour à chaque itération par DCA. Le schéma DCA&CUT appliqué au problème (PPP) est décrit comme suivant :

Algorithme 8.6
Schéma de DCA&CUT pour résoudre (PPP)

Etape 0 - Initialisation :
Poser $k \leftarrow 0$, $K \leftarrow \mathcal{K}_{PPP}$.
Poser borne inférieure $\beta_0 \leftarrow 0$.
Poser borne supérieure $\gamma_0 \leftarrow \infty$.
Passer à l'étape 1.

Etape 1 - Appliquer DCA pour chercher soit une coupe soit une solution réalisable :
1.1 Appliquer DCA au problème (PPP_t) *pour obtenir sa solution* $z^k_{DCA} = (x^k_{DCA}, y^k_{DCA})$
1.2a Si $z^k_{DCA} \in K'_{PPP}$, *poser* $z* \leftarrow z^k_{DCA}$ *et passer à l'étape 2.*
1.2b Sinon
Si $p(z^k_{DCA}) \notin Z$
Déterminer la coupe $l_k(z) \geq \mu_k$ *via théorème 8.3*
Passer à l'étape 4.
Sinon
Appliquer la Procédure P.
Si Procédure P fournit une solution réalisable, noté par z^*, *passer à l'étape 2.*
Sinon, soit $l_k(z) \geq \mu_k$ *la coupe obtenue par Procédure P, passer à l'étape 4.*
Passer à l'étape 4.

Etape 2 - Mettre à jour la borne supérieure et la meilleure solution réalisable :

Si $F_{PPP}(x^k_{DCA}, y^k_{DCA}) < \gamma_k$, *poser* $\gamma_k = F_{PPP}(x^k_{DCA}, y^k_{DCA})$ *et* $z^k \leftarrow z^*$.
Passer à l'étape 3.

Etape 3 - Séparation la solution réalisable :
Déterminer la contrainte de séparation $\phi_k(z) \geq v_k$ *:*

$$\phi_k(z) = \phi_k(x) = \sum_{j:x^*_j=0} x_j + \sum_{j:x^*_j=1} (1 - x_j).$$

Poser $K_{k+1} \leftarrow K_k \bigcap \{z : \phi(z) \geq v_k\}$.
Passer à l'étape 5.

Etape 4 - Ajouter la coupe aux contraintes :
Poser $K_{k+1} \leftarrow K_k \bigcap \{z : l_k(z) \geq \mu_k\}$.
Passer à l'étape 5.

Etape 5 - Test d'arrêt :
Si $\gamma_k - \beta_k \leq 0$ *alors on s'arrete.*
Sinon, poser $k \leftarrow k + 1$ *et passer à l'étape 1.*

La convergence de l'algorithme 8.6 est exprimée dans le théorème ci-dessous :

Théorème 8.4 *Si* $\mathcal{K}_{PPP}$ *est un polyèdre convexe borné alors l'algorithme 8.6 converge vers une solution optimale globale après un nombre fini d'itérations.*

Preuve. La preuve est analogue à celle du théorème de la convergence de DCA&CUT pour la programmation quadratique en variables mixtes zéro-un (voir [51]).

8.3.5 Expériences numériques

Nos algorithmes sont implémentés en C++ et testés avec un ordinateur Pentium 4 2.8 MHz, 1 GM Ram. Le logiciel CPLEX version 9.1 est utilisé pour la résolution des sous-problèmes linéaires.

Nous présentons ci-dessous les résultats de deux algorithmes PPP-DCA, DCA&CUT pour différentes valeurs de n et m. Avec PPP-DCA, nous utilisons tous les trois méthodes IP1, IP2, IP3 pour la recherche du point initial. Le nombre maximal de coupes dans DCA&CUT est fixé à 200.

De même façon que pour PP, nous exécutons les algorithmes sur 100 exemples générés aléatoirement. Le temps de calcul (le temps moyen en seconde dans les cas de réussite) et le pourcentage de réussite sont utilisés pour la comparaison.

Nous constatons que les résultats numériques sur le problème PPP sont assez modestes. Le pourcentage de réussite est petit pour une taille relativement petite par rapport au challenge sur le problème PPP. Pour les problèmes de taille plus grande, nos algorithmes n'arrivent pas

m	n	PPP-DCA-IP1		PPP-DCA-IP2		PPP-DCA-IP3		PPP-DCA&CUT	
		Time(s)	PR(%)	Time(s)	PR(%)	Time(s)	PR(%)	Time(s)	PR(%)
21	37	1.89	21	1.78	20	2.63	21	1.95	25
31	47	3.1	9	3.1	11	6.2	11	3.3	12
41	57	8.3	5	8.3	5	21.1	5	11.5	6
51	67	42.25	1	45.2	2	154.9	1	96.2	2

TAB. 8.2 – Résultat de test de **PPP-DCA**, **DCA&CUT** sur 100 instances du problème PPP

à trouver une solution (rappelons que DCA&CUT s'arrête après avoir introduit 200 coupes). L'apparition de la variable y augmente le nombre de variables et de contraintes de façon considérable et rend le problème très difficile.

Conclusion.

Certaines attaques basées sur le recuit simulé ont été développées depuis 1999 pour les schémas d'identification PP/PPP. Cependant aucun modèle d'optimisation déterministe n'a vu le jour pour ces deux problèmes. Nous avons formulé PP comme la minimisation d'une forme DC polyédrale sur un polyèdre convexe et utilisé DCA pour sa résolution. Notre algorithme **PP-DCA-2** permet de gagner un gain considérable en temps de calcul (jusqu'à 40 fois plus rapide que **RC**). De plus, nous avons testé avec les instances de grande taille jusqu'à 1001×1017 (en moins de 100 seconds).

La modélisation de PPP est beaucoup plus sophistiquée, mais nous obtenons à la fin également une programmation DC polyédrale en variables zéro-un pour laquelle nous développons DCA et une méthode de coupes. Les résultats numériques sur PPP sont assez modestes. Ce la est du au nombre important de variables binaires qui rendent le problème très difficile. La question porte sur le futur travail est de développer d'autres techniques pour le modèle proposé ou de trouver une autre modélisation plus convenable.

Il est à noter que tous les deux problèmes étudiés dans cette partie sont des problèmes combinatoires $\mathcal{NP}$-difficiles et de très grande dimension en pratique. Nos contributions propres portent à la fois sur la modélisation et les méthodes de résolution.

Conclusion et Perspectives

Nous avons étudié dans cette thèse les quatre classes des problèmes non convexes difficiles qui jouent un rôle important dans les deux domaines - fouille de données et cryptologie. L'épine dorsale de notre méthodologie est la programmation DC et DCA. Cette démarche est justifiée par la complexité d'une part, et la taille très grande d'autre part, des problèmes considérés. Notre ambition est de proposer les nouveaux algorithmes combinés efficaces basés sur DCA étant capables de traiter des problèmes de grande dimension.

"L'art" de modélisation occupe une place importante dans nos travaux. En effet, grâce aux techniques de formulation/reformulation nous avons pu mettre en évidence la forme DC des problèmes étudiés et ainsi unifier les méthodes de résolution en une seule DCA. Même pour le cas presque impossible de résoudre exactement le problème original (Clustering hiérarchique), nous avons trouvé un "compromis" entre la modélisation et les méthodes d'optimisation en considérant un problème "approché".

Par un constant souci d'exploiter l'efficacité des décompositions DC et des points initiaux de DCA, nous avons étudié avec soins la structure spéciale de chaque problème pour trouver des algorithmes bien adaptés, robustes et peu coûteux. Notre objectif est atteint pour les trois premières classes des problèmes.

Pour le clustering flou nous avons proposé, sur le même modèle de FCM, les trois différentes formulations DC, et montré l'importance du choix de décomposition DC afin de rendre le calcul plus simple et moins coûteux. Les deux algorithmes Fuzzy-DCA-2 et Fuzzy-DCA-3 sont intéressants car ils requièrent seulement des projections d'un points sur un simplexe/une boule Euclidienne et ces projections sont déterminées explicitement. En particulier, Fuzzy-DCA-3 est extrêmement rapide : le gain de temps par rapport à l'algorithme classique FCM monte jusqu'à 62 fois. L'effet du point initial a été également abordé par l'introduction d'une procédure alternative FCM-DCA.

Quant au problème de clustering hiérarchique, nous avons mis en évidence l'effet de la modélisation en proposant trois modèles d'optimisation qui donnent naissance aux trois schémas de DCA qui sont tous simples, peu coûteux, mais fournissent des solutions de qualité différente. Chaque programme DC introduit est la minimisation de la différence d'une forme quadratique définie positive et d'une fonction convexe non différentiable, chose fort intéressante pour l'application de DCA qui s'interprète alors comme une technique qui consiste à résoudre une suite de programmes quadratiques convexes approximants dont les

solutions se calculent de manière explicite. La procédure combinée DCA - k-means pour la recherche d'un bon point initial de DCA nous semble pertinente.

Pour la génération des fonctions booléennes non linéaires et équilibrées, nous avons pu bâtir un modèle d'optimisation déterministe (combinatoire) puis le transformer en un programme DC. L'algorithme FB-DCA, qui consiste à la résolution d'un problème linéaire à chaque itération, jouit des propriétés précieuses : il a une convergence finie et fournit une solution entière malgré qu'il fonctionne sur un domaine continu. La combinaison de DCA et les algorithmes génétiques apporte une nouveauté sur le plan algorithmique et cette technique est prometteuse.

Finalement, dans les expériences numériques de tous ces trois classes des problèmes, nous avons montré la supériorité des algorithmes proposés par rapport aux algorithmes standard.

Concernant les deux problèmes PP et PPP en cryptanalyse, qui sont très difficiles et attirent l'attention particulière de nombreux chercheurs dans ce domaine, nous avons abouti l'objectif pour le problème PP seulement. La formulation du problème PP simplement sous la forme de la minimisation d'une forme quadratique concave sous les contraintes de bornes nous amène à utiliser efficacement DCA, qui a été appliqué avec grand succès aux programmations quadratiques non convexes dans les travaux précédents. Nous avons pu traiter des problèmes PP de taille beaucoup plus grande que ceux résolus par les méthodes existantes. Le problème PPP reste néanmoins très difficile malgré nos efforts sur le plan algorithmique (plusieurs tentatives par différentes approches).

Perspectives

Plusieurs perspectives sont ouvertes à la suite de nos travaux.

Premièrement, en fouille de données et plus particulièrement en classification, le domaine constituant un cadre algorithmique enrichissant pour la programmation DC et DCA, vu l'efficacité, la robustesse et la performance des algorithmes développés dans cette thèse, nous pourrions

- appliquer ces algorithmes simples aux problèmes d'applications dans divers domaines, en particulier en Biologie médicale et Analyse d'image (clustering des gènes et imagerie médicale par exemple), ou encore dans les systèmes informatiques coopératifs et les réseaux de communication ;
- élargir le cadre d'application de la programmation DC et DCA aux autres types de problèmes du clustering : clustering par blocs (classification croisée), clustering hiérarchique à multi-niveaux (nous avons étudié le cas de biniveaux) ;
- développer la programmation DC et DCA pour les modèles de mélanges dans l'approche statistique en classification.

Deuxièmement, en cryptographie nous pourrions considérer d'autres critères des fonctions booléennes (auto corrélation par exemple) et investir les techniques de modélisation et

optimisation DC à la construction des "bonnes" fonctions ayant d'autres caractéristiques intéressantes dans les S-box.

Enfin, serait-il possible de trouver une autre modélisation du problème PPP pour laquelle on pourrait envisager des méthodes efficaces, ou faudrait-il développer d'autres techniques pour le modèle quadratique proposé ?

Tout cela fera l'objet de nos futurs travaux.

ANNEXE

Bibliographie

[1] H.A Le Thi, *Analyse numérique des algorithmes de l'optimisation DC. Approches locale et globale. Codes et simulations numériques en grande dimension. Applications.* Thèse de Doctorat de l'Université de Rouen, 1994.

[2] H.A Le Thi, *Contribution à l'optimisation non convexe et l'optimisation globale : Théorie, Algorithmes et Applications*, Habilitation à Diriger des Recherches, Université de Rouen, 1997.

[3] H.A Le Thi, *An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints*, Mathematical Programming, Ser. A, Vol.87, N°.3, 401–426, 2000.

[4] H.A Le Thi, *Solving large scale Molecular distance geometry problem by a smoothing technique via the Gaussian transform an DC programming*, Journal of Global Optimization, Vol.27, 375–397, 2003.

[5] H.A Le Thi and T. Pham Dinh, *Solving a class of linearly constrained indefinite quadratic problems by DC algorithms* Journal of Global Optimization, Vol.11, 253–285, 1997.

[6] H.A Le Thi and T. Pham Dinh, *A Branch-and-Bound method via DC Optimization Algorithm and Ellipsoidal techniques for Box Constrained Nonconvex Quadratic Programming Problems*, Journal of Global Optimization, Vol.13, 171–206, 1998.

[7] H.A Le Thi and T. Pham Dinh, *DC programming approach for large-scale molecular optimization via the general distance geometry problem*, Nonconvex Optimization and Its Applications 40, in Optimization in Computational Chemistry and Molecular Biology : Local and Global Approaches, Kluwer Academic Publishers, 301–339, 2000.

[8] H.A Le Thi and T. Pham Dinh, *Large Scale Molecular Conformation via the Exact Distance Geometry Problem*, In Optimization, Lecture Notes in Economics and Mathematical Systems, Vol.481, Heidelberg, Springer-Verlag, 260-277, 2000.

[9] H.A Le Thi and T. Pham Dinh, *A continuous Approach for Globally Solving Linearly Constrained Quadratic Zero - One Programming Problems*, Optimization, Vol.50, 93–120, 2001.

[10] H.A Le Thi and T. Pham Dinh, *DC optimization approaches via Markov models for restoration of signals (1-D) and (2-D)*, Nonconvex Optimization and Its Applications

54 : In Advances in Convex Analysis and Global Optimization, Kluwer Academic Publishers, 300–317, 2001.

[11] H.A Le Thi and T. Pham Dinh, *DC programming approach and solution algorithm to the multidimensional scaling problem*, Nonconvex Optimization and Its Applications 53 : In From Local to Global Optimization, Kluwer Academic Publishers, 231–276, 2001.

[12] H.A Le Thi and T. Pham Dinh, *DC Programming Approach for Multicommodity Network Optimization Problems with Step Increasing Cost Functions*, Special Issue of Journal of Global Optimization (dedicated to Professor R. Horst on the occasion of his 60 th birthday), Vol.22, 204–233, 2002.

[13] H.A Le Thi and T. Pham Dinh, *Dc Programming. Theory, Algorithms, Applications : The State of the Art*, First InternationalWorkshop on Global Constrained Optimization and Constraint Satisfaction, October 2-4, 2002, Valbonne-Sophia Antipolis, France, Research Report, Laboratory of Modeling, Optimization & Operations Research, Insa-Rouen, France, 2002.

[14] H.A Le Thi and T. Pham Dinh, *Large scale molecular optimization from distances matrices by a DC optimization approach*, SIAM Journal of Optimization, Vol.14, N°.1, 77–116, 2003.

[15] H.A Le Thi and T. Pham Dinh, *A new algorithm for solving large scale molecular distance geometry problems, Applied Optimization : in Hight Performance Algorithms and Software for Nonlinear Optimization*, Kluwer Academic Publishers, 276–296, 2003.

[16] H.A Le Thi and T. Pham Dinh, *The DC (Difference of Convex functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems*, Annals of Operations Research, Vol.133, 23–46, 2005.

[17] H.A Le Thi and T. Pham Dinh, *A continuous approach for the concave cost supply problem via DC Programming and DCA*, to appear in Discrete Applied Mathematics.

[18] H.A Le Thi, T. Belghiti and T. Pham Dinh, *A new efficient algorithm based on DC programming and DCA for Clustering*, July 2006, Journal of Global Optimization.

[19] H.A Le Thi, T. Belghiti and T. Pham Dinh, *Clustering via DC programming and DCA*, Proceeding de 13ème Rencontres de la Société Francophone de Classification SFC'06, p.133-139, Metz, France Septembre 2006.

[20] H.A Le Thi, M. Le Hoai and T. Pham Dinh, *Optimization based DC programming and DCA for Hierarchical Clustering*, European Journal of Operational Research, Vol. 183, p. 1067-1085, 2007.

[21] H.A Le Thi, M. Le Hoai and T. Pham Dinh, *Une nouvelle approche basée sur la programmation DC et DCA pour la classification floue*, EGC 2007, RNTI, 703–714, 2007.

[22] H.A Le Thi, T.P Nguyen and T. Pham Dinh, *A Continuous DC Programming Approach To The Strategic Supply Chain Design Problem From Qualified Partner Set* In Press, Available Online June 2006, European Journal of Operational Research.

[23] , H.A Le Thi, M. Le Hoai, T.P Nguyen and T. Pham Dinh, *Noisy Image Segmentation by Fuzzy C-Means Clustering based DCA*, Submitted, 2007.

[24] , H.A Le Thi, T. Pham Dinh and H. Dinh Nho, *Towards Tikhonov regularization of nonlinear ill-posed problems : a DC programming approach*, C.R. Acad. Sci. Paris, Ser. I, Vol.335, 1073–1078, 2002.

[25] H.A Le Thi, T. Pham Dinh and H. Dinh Nho, *Solving inverse problems for an elliptic equations by DC (Difference of Convex functions) programming*, Journal of Global Optimization, Vol.25, 407–423, 2003.

[26] H.A Le Thi, T. Pham Dinh and H. Dinh Nho, *On the ill-posedness of the trust region Subproblem*, Journal of Inverse and Ill-posed Problems, Vol.11, 545–577, 2003.

[27] H.A Le Thi, T. Pham Dinh and N. Huynh Van, *Exact penalty techniques in DC Programming*, Submitted, 2004.

[28] H.A Le Thi, T. Pham Dinh and M. Le Dung, *Numerical solution for optimization over the efficient set by DC optimization algorithm*, Operations Research Letters, Vol.19, 117–128, 1996.

[29] H.A Le Thi, T. Pham Dinh and M. Le Dung, *A combined DC Optimization : Ellipsoidal Branch-and-Bound Algorithm for Solving Nonconvex Quadratic Programming Problems*, Journal of Combinatorial Optimization, Vol.2, N°.1, 9–28, 1998.

[30] H.A Le Thi, T. Pham Dinh and M. Le Dung, *Exact Penalty in DC. Programming*, Vietnam Journal of Mathematics, Vol.27, N°.2, 169–178, 1999.

[31] H.A Le Thi, T. Pham Dinh and M. Le Dung, *Simplicially Constrained DC Optimization over the Efficient Set and Weakly Efficient Sets*, Journal of Optimization, Theory and Applications, Vol.117, N°.3, 503-531, 2003.

[32] H.A Le Thi, T. Pham Dinh and T. Nguyen Van, *Combination between Local and Global Methods for Solving an Optimization Problem over the Efficient Set*, European Journal of Operational Research, Vol.142, 257–270, 2002.

[33] H.A Le Thi, T. Pham Dinh and N. Huynh Van, *Exact Penalty Techniques in DC Programming*, SIAM Conference on Optimization, May 15-19, 2005, Stockholm, Sweden (Special invited session).

[34] H.A Le Thi, T. Pham Dinh and V. Nguyen Van, *A Combined DCA and New Cutting Plane Techniques for Globally Solving Mixed zero-one Programming*, SIAM Conference on Optimization, May 15-19, 2005, Stockholm, Sweden (Special invited session).

[35] 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.

[36] 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.

[37] T. Pham Dinh, *Contribution à la théorie de normes et ses applications à l'analyse numérique*, Thèse de Doctorat d'Etat Es Science, Université Joseph Fourier- Grenoble, 1981.

[38] T. Pham Dinh, *Convergence of subgradient method for computing the bound norm of matrices*, Linear Alg. and Its Appl., Vol.62, 163–182, 1984.

[39] T. Pham Dinh, *Algorithmes de calcul d'une forme quadratique sur la boule unité de la norme maximum*, Numer. Math., Vol.45, 377–440, 1985.

[40] 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.

[41] T. Pham Dinh, *Duality in DC (difference of convex functions) optimization. Subgradient methods*, Trends in Mathematical Optimization, International Series of Numer Math., Vol 84, 277–293, 1988.

[42] T. Pham Dinh and H.A Le Thi, *Stabilité de la dualité lagrangienne en optimisation DC (différence de deux fonctions convexes)*, C.R. Acad. Paris, P.318, Série I, 379–384, 1994.

[43] T. Pham Dinh and H.A Le Thi, *Lagrangian stability and global optimality in nonconvex quadratic minimization over Euclidean balls and spheres*, Journal of Convex Analysis, Vol.2, 263–276, 1995.

[44] T. Pham Dinh and H.A Le Thi, *DC optimization algorithms for globally minimizing nonconvex quadratic forms on Euclideans balls and spheres*, Operations Research Letters, Vol.19, 207–216, 1996.

[45] T. Pham Dinh and H.A Le Thi, *Convex analysis approach to d.c. programming : Theory, Algorithms and Applications*, Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, Vol.22, N°.1, 289–355, 1997.

[46] T. Pham Dinh and H.A Le Thi, *DC optimization algorithms for solving the trust region subproblem*, SIAM Journal of Optimization, Vol.8, N°.2, 476–505, 1998.

[47] F.B Akoa, *Approches de points intérieurs et de programmation DC en optimisation non convexe. Code et simulations numériques industrielles.*, Thèse de Doctorat de l'Université de Rouen, 2005.

[48] T.Q Phong, *Analyse Numerique des Méthodes d'Optimisation Globale*, Thèse de Doctoral, Université de Rouen, 1994.

[49] T.Q Phong, T. Pham Dinh and H.A Le Thi, *A New Method for Solving DC Programming Problems. Application to Fuel Mixture Nonconvex Optimization Problem*, Journal of Global Optimal, Vol.6, 87–105, 1995.

[50] P.T Thach, *DC sets, DC functions and nonlinear equations*, Mathematical Programming, Vol.58, 415–428, 1993.

[51] V. Nguyen Van, *Méthodes exactes pour l'optimisation DC polyédrale en variables mixtes 0-1 basées sur DCA et des nouvelles coupes*, Thèse de Doctorat de l'Université de Rouen, 2006.

[52] R. Collobert, F. Sinz, J. Weston and L. Bottou, *Trading Convexity for Scalability*, Proceedings of the 23rd ICML, 2006.

[53] N. Krause and Y. Singer, *Leveraging the margin more carefully*, International Conference on Machine Learning ICML, 2004.

[54] Y. Liu, X. Shen and H. Doss, *Multicategory ψ-Learning and Support Vector Machine : Computational Tools*, Journal of Computational and Graphical Statistics, Vol.14, 219–236, 2005.

[55] Y. Liu and X. Shen, *Multicategory ψ-Learning*, Journal of the American Statistical Association, Vol.101, 500–509, 2006.

[56] J. Neumann, C. Schnörr and G. Steidl, *SVM-based Feature Selection by Direct Objective Minimisation*, Pattern Recognition, Proc. of 26th DAGM Symposium, LNCS, Springer, August 2004.

[57] J. Neumann, C. Schnörr and G. Steidl, *Combined SVM-Based Feature Selection and Classification Machine Learning*, in Press (published online) http ://www.cvgpr.uni-mannheim.de/Publications/ML-05.pdf.

[58] C. Ronan, S. Fabian, W. Jason and B. Léon, *Trading Convexity for Scalability*, International Conference on Machine Learning ICML, 2006.

[59] X. Shen, *From large margin classification to ψ-learning*, School of Statistics, University of Minnesota.

[60] X. Shen, G.C Tseng, X. Zhang, and W.H Wong, *On ψ-Learning*, Journal of American Statistical Association, Vol.98, 724–734, 2003.

[61] T. Schüle, C. Schnörr, S. Weber and J. Hornegger, *Discrete Tomography by convex-concave regularization and DC programming*, Discrete Applied Mat., Vol.151, 229–243, 2005.

[62] T. Schüle, S. Weber and C. Schnörr, *Adaptive Reconstruction of Discrete-Valued Objects from few Projections*, Electr. Notes in Discr. Math., Vol.20, 365–384, 2005.

[63] S. Weber, T. Schüle, J. Hornegger and C. Schnörr, *Binary Tomography by Iterating Linear Programs from Noisy Projections*, IWCIA, LNCS 3322, 38–51, 2004.

[64] S. Weber, C. Schnörr, T. Schüle and J. Hornegger, *Binary Tomography by Iterating Linear Programs*, R. Klette, R. Kozera, L. Noakes and J. Weickert (Eds.), Computational Imaging and Vision - Geometric Properties from Incomplete Data, Kluwer Academic Press 2005.

[65] S. Weber, T. Schüle and C. Schnörr, *Prior Learning and Convex-Concave Regularization of Binary Tomography*, Electr. Notes in Discr. Math., Vol.20, 313–327, 2005.

[66] S. Weber, A. Nagy, T. Schüle, C. Schnörr and A. Kuba, *A Benchmark Evaluation of Large-Scale Optimization Approaches to Binary Tomography*, DGCI 2006, LNCS 4245, 146–156, 2006.

[67] S. Weber, T. Schüle, A. Kuba and C. Schnörr, *Binary Tomography with Deblurring*, IWCIA 2006, LNCS 4040, 375–388, 2006.

[68] A.L Yuille and A. Rangarajan, *The Convex Concave Procedure (CCCP)*, Advances in Neural Information Processing System 14, Cambrige MA : MIT Press.

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

[70] E. Balas, *Disjunctive programming : Properties of the convex hull of feasible points*, Technical Report MSRR No.330, Carnegie Mellon University, 1974.

[71] E. Balas, *Disjunctive programming : Properties of the convex hull of feasible points*, Discrete Applied Mathematics,Vol. 89,p. 1-44, 1998.

[72] H.P Benson *Concave minimization : theoty, applications and algorithms* in Handbook of Global Optimisation, R. Horst and P. Padalos eds., Kluwer Academic Publishers, 43–148, 1995.

[73] H.P Benson *Deterministic algorithm for constrained concave minimization : a unified critical survey* Naval Research Logistics, Vol.43, 765–795, 1996.

[74] H.P Benson and R. Horst *A branch and bound - outer approximation for concave minimization over a convex set* Journal of Computers and Mathematics with applications, Vol.21, 67–76, 1991.

[75] G.B. Dantzig, D.R. Fulkerson and S.M. Johnson, emphSolution of a large scale traveling salesman problem, Operations Recheach, Vol. 2, p. 393-410, 1954.

[76] V. Chvátal, *Edmonds polytopes and a hierarchy of combinatorial problem*, Discrete Mathematics, Vol. 4 , p. 305-337, 1973.

[77] E.W. Cheney and A.A. Goldstein, *Newton's method for convex programming and Tchebycheff approximation*, Numerische Mathematics, Vol. 1, p. 253-268, 1959.

[78] R.W Cottle and G.B Dantzig *Complementary pivot theory of mathematical programming* Linear algebra and its applications, Vol.1, 103–125, 1968.

[79] G.B Dantzig *Linear Programming and Extensions* Princeton University Press, Princeton, New Jersey, 1963.

[80] G.B Dantzig, D.R Fulkerson and S.M Johnson *Solution of a large scale traveling salesman problem* Operations Recheach, Vol.2, 393–410, 1954.

[81] , J.E. Falk and K.L. Hoffman, *A successive underestimating method foe concave programming problems*, Mathematics of Operation Research, Vol. 1, p. 251-259, 1976.

[82] R. Fletcher *Pratical methods of Optimization* John Wiley, New York, 1980.

[83] R. E. Gomory, *An Algorithm for the integer solutions of linear programs*, In R. Graves and P. Wolfe, editors, Recent Advances in Mathematical Programming, p. 269-302, McGraw-Hill, Newyork, 1963.

[84] R. Horst, N.V Thoai and H. Tuy *On a outer approximation concept in global optimisation* Optimization, Vol.20, 255–264, 1989.

[85] R. Horst, N.V Thoai and H.P Benson *Concave minimization via conical partitions and polyhedral outer approximation* Mathematical Programming, Vol.50, 259–276, 1991.

[86] R. Horst, P.M Pardalos and N.V Thoai *Introduction to Global Optimization* Kluwer Academic Publishers, 1995.

[87] R. Horst and H. Tuy *Global Optimization : Deterministic Approaches* Third edition, Springer-Verlag, 1996.

[88] R. Horst and N.V Thoai *DC Programming : Overview* Journal of Optimization Theory and Applications, Vol.2103, 1–43, 1999.

[89] P.J Laurent *Approximation et optimisation* Paris : Hermann, 1972.

[90] K. Levenberg *A method for the solution or certain nonlinear problems in least quares* Quart. Appl. Math., Vol.2, 1944.

[91] D.W. Marquardt *An algorithm for least squares estimation of nonlinear parameters* SIAM J. Appl. Math., Vol.11, 1963.

[92] J.J Moré *Recent developpements in algorithm and software for trust region methods* Mathematique Programming, The state of the art, Springer-Verlag, Berlin, 258–287, 1983.

[93] J.J Moré and D.C Sorensen *Computing a trust region step* SIAM J. Sci. Statist. Comput., Vol.4, 553–572, 1983.

[94] M.W. Padberg, *On the factial structure of set packing polyhedra*, Mathematical Programming, Vol. 5, p. 199-215, 1973.

[95] B. T Polyak *Introduction to optimization* Inc., Publications Division, 1987.

[96] R.T Rockafellar *Convex analysis* Princeton University Press, 1970.

[97] R.T Rockafellar *Monotone operators and the proximal point algorithm* SIAM Journal on Control and Optimization, Vol.14, 877–898, 1976.

[98] S.J Sadjadi and K. Ponnambalam *Advances in trust region algorithms for constrained optimization* Applied Numerical Mathematics, Vol.29, 423–443, 1999.

[99] D.C Sorensen *Newton's method with a model trust region modification* SIAM J. Numer. Anal., Vol.19, N°.2, 409–426, 1982.

[100] S. Steihaug *The conjugate gradient method and trust region in large scale optimization* SIAM J. Numer. Anal., Vol.20, 626–637, 1983.

[101] J.F Toland *Duality in nonconvex optimization* Journal of Mathematical Analysis and Applications, Vol.66, 399-415, 1978.

[102] J.F Toland *On subdifferential calculus and duality in nonconvex optimization* Bull. Soc. Math. France, Mémoire 60, 177–183, 1979.

[103] P.L Toint *Towards an efficient sparsity exploiting Newton method for minimization* Duff, I., ed., Sparse Matrices and Their Uses, 57–88, Academic Press, 1981.

[104] H. Tuy *Concave programming under linear constraints* Translated Soviet Mathematics, Vol.5, 1437–1440, 1964.

[105] H. Tuy *On outer approximation methods for solving concave minimization problems* Acta Mathematica Vietnamica, Vol.8, 3–34, 1983.

[106] H. Tuy *Global Minimization of a Difference of Two Convex Functions* Mathematics Programming Study, Vol.30, 150–182, 1987.

[107] H. Tuy *Global Optimization : Deterministic Approaches* 2nd revised edition, Springer-Verlag, Berlin, 1993.

[108] H. Tuy *DC Optimisation : Theory, Methods and Algorithms, Handbook of Global Optimisation* Horst and Pardalos eds, Klaver Academic Publishers, 149–216, 1995.

[109] H. Tuy *Convex Analysis and Global Optimization* Kluwer Academic Publishers, 1998.

[110] J.B.H Urruty *Generalized differentiability, duality and optimization for problem dealing with differences of convex functions*Lecture Notes in Economics and Mathematical Systems, Vol.256, Heidelberg, Springer-Verlag, 260–277, 1985.

[111] J.B.H Urruty *Conditions nécessaires et suffisantes d'optimalité globale en optimisation de différences de deux fonctions convexes* CRAS, Vol.309, Série I, 459–462, 1989.

[112] F. Rendl and H. Wolkovicz *A semidefinite framework to trust region subproblems with application to large scale minimization* CORR Report 94-32, University of Waterloo, 1994.

[113] C. R.Reeves, J. E.Rowe, *Genetic algorithms-Principles and perspectives*, ISBN : 0306480506, Kluwer Academic Pub, 2002.

[114] C. Darwin, *On the Origin of Species by means of natural selection, or the Preservations of favored races in the struggle of life*, 1859.

[115] L.J Fogel, A.J Owens and M.J Walsh, *Artificial Intelligence Through Simulated Evolution*, Wiley and sons. NY, 1966.

[116] D.E. Goldberg, *Genetic Algorithms in Search*, Optimisation and Machine Learning, Addison Wesley, 1989. ISBN : 0-201-15767-5.

[117] C.L. Bridges and D.E Goldberg, *An analysis of multipoint crossover*, Proceedings of the Foundation of Genetic Algorithms, 1991.

[118] J.H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, 1975.

[119] Z. Michalewicz and C.Z Janikov, *Handling constraints in genetic algorithms*, Proceedings of the Fourth International Conference on Genetic Algorithm, Edited by Belew R.K et Booker L.B. ICGA, Morgan Kaufman, p.151-157, 1991.

[120] W.B. Langdon and R. Poli, *Foundations of Genetic Programming*, ISBN :35404245512, Springer, 2002.

[121] G. Syswerda, *Uniform crossover in genetic algorithms*, Proceedings of the third European conference on Genetic Algorithm, 1989.

[122] G.R. Raidl, *A unified view on hybrid metaheuristics*, Proceedings of the Hybrid Metaheuristics Workshop, LNCS,4030 pp.1-12, Springer, 2006.

[123] M.N Ahmed, S.M Yamany, N. Mohamed, A.A Farag and T. Moriarty *A modified fuzzy C-means algorithm for bias field estimation and segmentation of MRI data* IEEE Trans. on Medical Imaging, Vol.21, 193–199, 2002.

[124] J.C. Bezdek, L.O. Hall and L.P. Clake *Review of MR image segmentation techniques using pattern recognition* Medical Physics, Vol.20, 1033–1048, 1993.

[125] C. Bouman and B. Liu *Segmentation of Textured Images Using a Multiple Resolution Approach* Proc. IEEE Int'l Conf. on Acoust., Speech and Sig. Proc., NewYork, NY, April 11-14, 1124–1127, 1988.

[126] P. Campadelli, D. Medici and R. Schettini *Color Image Segmentation using Hopfield Networks* Image and Vision Computing, Vol.15, N°.3, 161–166, 1997.

[127] J.C Dunn *A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters* Journal of Cybernetics, Vol.3, 32–57, 1973.

[128] W.L Hung, M.S Yang and D.H Chen *Parameter selection for suppressed fuzzy c-means with an application to MRI segmentation* Pattern Recognition Letters, Vol.27, 424–438, 2006.

[129] E. Littman and E. Ritter *Adaptative Color Segmentation - A Comparison of Neural and Statistical Methods* IEEE Transactions on Neural Networks, Vol.8, N°.1, 175–185, 1997.

[130] J.P. Pal and S.K. Pal *A review on image segmentation techniques* Pattern Recognition, Vol.26, 1277–1294, 1993.

[131] D.L Pham *Fuzzy Clustering with spatial constraints* Proc.IEEE Intern. Conf. on Image Processing, New Yord, USA.

[132] J.C Rajapakse, J.N Giedd and J.L Rapoport *Statistical Approach to Segmentation of Singke-Chanel Cerebral MR Images* IEEE Trans. On Medical Imaging, Vol.16, 2004.

[133] D.Q Zhang and S.C Chen *A novel kernelized fuzzy C-means algorithm with application in medical image segmentation* Artificial Intelligence in Medicine, Vol.32, 37–50, 2004.

[134] S. Nascimento, B. Mirkin and F. Moura-Pires, *Modeling Proportional Membership in Fuzzy Clustering*, IEEE Transactions on Fuzzy Systems, Vol. 11, N° 2, April 2003.

[135] W. Glenn, X. Ying Wang and J.M. Garibaldi, *The Application of a Simulated Annealing Fuzzy Clustering Algorithm for Cancer Diagnosis*, SIP 2005, Japan.

[136] M.E.S. Mendes Rodrigues and L. Sacks, *A scalable hierarchical fuzzy clustering algorithm for text mining*, In : Proc. of the 4th International Conference on Recent Advances in Soft Computing, RASC'2004, pp.269-274, Nottingham, UK, Dec. 2004.

[137] C.H. Bennett, *Quantum cryptography : Uncertainly in the service of privacy*, Science, 257 :752-753,August 1992.

[138] C.H. Bennett, G. Brassard, A. Ekert, *La cryptographie quantique*, Pour La Science Hors-Série, l'Art du Secret, juillet-octobre 2002.

[139] L. Burnett, G. Carter, E. Dawson and W. Millan., *Efficient Methods for Generating MARS-like S-Boxes*, In Bruce Schneier, editor, Fast Software Encryption 2000, LNCS Vol. 1978, p. 300-313. Springer-Verlag, April 2000.

[140] A. Clark, Ed Dawson and H. Bergen, *Combinatorial optimisation and the knapsack cipher*, Cryptologia, 20(1) :85-93, January 1996.

[141] A. Clark, Ed Dawson and H. Nieuwland, *Cryptanalysis of polyalphabetic substitution ciphers using a parallel genetic algorithm*, In 1996 IEEE International Symposium on Information Theory and Its Applications, volume 1, p. 339-342, Victoria, Canada, 1996. IEEE.

[142] A. Clark, Ed Dawson, *Discrete optimisation : A powerful tool for cryptanalysis*, In PRAGOCRYPT'96 - Proceedings of the 1st International Conference on the Theory and Applications of Cryptology, volume 1, pages 425-451, Prague, Czech Republic, 1996.

[143] A. Clark, Ed Dawson, *A parallel genetic algorithm for cryptanalysis of the polyalphabetic substitution cipher*, Cryptologia, 21(2) :129-138, 1997.

[144] A. Clark and J.L Jacob, *Two Stage Optimisation in the Design of Boolean Functions*, In Ed Dawson, Andrew Clark, and Colin Boyd, editors, 5th Australasian Conference on Information Security and Privacy, ACISP 2000, p. 242-254, Springer Verlag LNCS 1841, july 2000.

[145] A. Clark et al, *Almost Boolean Functions : the Design of Boolean Functions by Spectral Inversion*, In Proceedings of Conference on Evolutionary Computation, CEC 2003, Canberra, Australia, 2003.

[146] J. Daemen and V. Rijmen, *The design of Rijndael*, ISBN :3540435802, Springer, 2002.

[147] Lerville Edmond, *Les cahiers secrets de la cryptographie*, Editions du rocher, 1972.

[148] T. Jakobsen, *Cryptanalysis of block cipher with probabilistic non linear relations of low degree*, Crypto'98, LNCS Vol. 1462, p. 347-362, Springer, 1998

[149] J. Giddy,R. Safavi-Naini, *Automated cryptanalysis of transposition ciphers*, The Computer Journal 37(5) :429-436, 1994.

[150] S. Goldwasser, S. Micali and C. Rackoff, *Knowledge Complexity of Identification Proof Schemes*, In 17th ACM Symposium on the Theory of Computing STOC, p. 291-304., SACM, 1985.

[151] O. Levbedko and A. Topchy, *On Efficiency of Genetic Cryptanalysis for Knapsack Ciphers*, In Poster Proceedings of ACDM 98, 1998.

[152] R.A.J. Matthews *The use of genetic algorithms in cryptanalysis*, Cryptologia, 17(2) :187-201, 1993.

[153] M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology - Eurocrypt'93, pp. 386-396, Springer Verlag, Lofthus, Norvège, 1993.

[154] H.X. Mel and D. Baker, *La cryptographie décryptée*, ISBN :274401155X, Campus-Press, Paris, 2001.

[155] W. Millan, *How to Improve the Non-linearity of Bijective S-boxes*, In C. Boyd and E. Dawson, editors, 3rd Australian Conference on Information Security and Privacy, LNCS 1438, p. 181-192, Springer-Verlag, April 1998.

[156] W. Millan, A. Clark and Ed Dawson, *Heuristic Design of Cryptographically Strong Balanced Boolean Functions*, In Advances in Cryptology EUROCRYPT'98, LNCS 1403, p. 489-499. Springer Verlag, 1998.

[157] W. Millan, L. Burnett, G. Carter, A. Clark and E. Dawson, *Evolutionary Heuristics for Finding Cryptographically Strong S-Boxes*, In ICICS 99, 1999.

[158] W. Millan, A. Clark and E. Dawson, *Boolean Function Design Using Hill Climbing Methods*, In Bruce Schneier, editor, LNCS 1978 4th Australian Conference on Information Security and Privacy. Springer-Verlag, 1999.

[159] W. Millan et al, *Evolutionary Generation of Bent Functions for Cryptography*, In Proceedings of Conference on Evolutionary Computation, CEC 2003, Canberra, Australia, 2003.

[160] R. Oppliger, *Contemporary Cryptography*, ISBN :1580536425, Artech House INC, 2005.

[161] , I.F.T. Yaseen and H.V. Sahasrabuddhe, *Breaking Multiplicative Knapsack Ciphers Using a Genetic Algorithm*, In International Conference on Knowledge Based Computer Systems, p. 129-139, 1998.

[162] D. Stinson, *Cryptographie : Theorie et pratique*, ISBN :2711786757, Vuibert, Paris, 2001.

[163] B. Schneier, *Cryptographie appliquée*, ISBN :2711786765, Vuibert, Paris, 2001.

[164] N. Ferguson and B. Schneier, *Practical Cryptography*, ISBN :047122357, Wiley Publishing, 2003.

[165] R. Spillman, M. Janssen, B. Nelson and M. Kepner, *Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers*, Cryptologia, 17(1) :31-44, 1993.

[166] R. Spillman, *Cryptanalysis of Knapsack Ciphers Using Genetic Algorithms* Cryptologia, XVII(4) :367-377, 1993.

[167] G. Celeux and G. Govaert, *A classification EM algorithm for clustering and two stochastic versions*, Computational Statistics and Data Analysis,14 : 315-332, 1992.

[168] G. Celeux and J. Diebolt, *The SEM algorithm : a probabilistic teacher algorithm from the EM algorithm for the mixture problem*, Computational Statistics Quarterly, 2(1) :73 92, 1985.

[169] A.P. Dempster, N.M. Laird and D.B. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of Royal Statistical Society B, vol.39, p. 1-38, 1977.

[170] E. Diday, *Optimisation en classiffication automatique et reconnaissance de formes*, Note Scient. INRIA n°6, 1972.

[171] E.W. Holman, *Evolutionary and psychological effects in pre-evolutionary classiffcations*, Journal of Classiffcation, vol.2, 1985, pp. 29-39.

[172] M. Dash, K. Choi, P. Scheuermann and H. Liu *Feature selection for clustering - A Filter Solution*, In : proceedings of IEEE International Conference of Data Mining (ICDM 2002). Maebashi City, Japan, December 2002.

[173] H. Liu and R. Setiono, *A Probabilistic Approach to Feature Selection - A Filter Solution*, In : International Conference on Machine Learning, pp. 319-327.Bari, Italy, 1996.

[174] M. Law, M. Figueiredo and A.K. Jain, *Feature selection in mixture-based clustering*, In : Advances in Neural Information Processing Systems 15 (NIPS 2002), pp. 609-616. Vancouver, Canada, 2002.

M. Nadif and F.X. Jollois, *Accéelération de EM pour données qualitatives : étude comparative de différentes versions*, In : EGC 2004, 4th French-Speaking Conference

on Knowledge Discovery and Knowledge Management. pp. 253-264.Clermont-Ferrand, France, Janvier 2004.

[175] J. MacQueen, *Some methods for classiffcation and analysis of multivariate observations*, In : Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability. pp. 281-297. Berkeley, 1967.

[176] A. Meyerson, *A k-Median Algorithme with running time Independent of Data Size*, Machine Learning, Kluwer Acadimic Publishers, Manufactured in The Netherlands 56, p.61-87,2004.

[177] P. Mitra, C.A Murthy and S.K. Pal *Unsupervised Feature Selection Using Feature Similarity*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.24, No 4, 2002.

[178]

[179] C. Muller, X. Polanco, J. Royaute and Y. Toussaint, *Acquisition et structuration des connaissances en corpus : éléements méthodologiques*, Technical Report nNo RR-3198, Inria, Institut National de Recherche en Informatique et en Automatique, Juin 1997.

[180] F.Z. Smadja, *Retrieving Collocations from Text : Xtract*, Computational Linguistics, vol.19(1), 1994, pp. 143-177.

[181] C.R. Rao, *he utilization of multiple measurements in problems of biological classiffication*, T. Journal of Royal Statistical Society B, Vol.10, p. 159-203, 1948.

[182] N. Alon and J.H Spencer, *The probabilistic method*, NewYork, NY : Wiley, 1991.

[183] S. Arora and R. Kannan, *Learning mixtures of arbitrary Gaussians*, In Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, pp. 247-27, 2001.

[184] B.S. Bradley and O.L. Mangasarian, *Feature selection via concave minimization and support vector machines*, In J.Shavlik, editor, Machine Learning Proceedings of the Fifteenth International Conferences (ICML'98), pp. 82-90, San Francisco, California, MorganKaufmann, 1998.

[185] J.C Bezdek, *Fuzzy mathematics in pattern classification*, PH.D dissertation, Cornell Univ., Ithaca, NY, 1973.

[186] J.C Bezdek, *Numerical taxonomy with fuzzy sets*, Journal of Mathematical Biology, 1, pp. 57-71, 1974.

[187] J.C Bezdek, *A Convergence Theorem for the Fuzzy ISODATA Clustering Algorithms* IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-2, no. 1, pp. 1-8, January 1980.

[188] J.C Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithm*, New York, NY. Plenum Press, 1991.

[189] J.C. Bezdek, C. Coray, R. Gunderson and J. Watson, *Detection and characterization of cluster substructure*, SIAM J. Appl. Math., vol. 40, n. 2, pp. 339-372, 1981.

[190] J.C Bezdek, R.J. Hathaway, M.J. Sabin and W.T. Tucker, *Convergence Theory for Fuzzy c-Means : Counterexamples and Repairs*, IEEE Trans. Syst. Man Cyber., vol. SMC-17, n.5, pp.873-877, September/October 1987.

[191] C. Bouveyron, *Modélisation et classification des données de grande dimension - application à l'analyse d'image*, Thèse de doctorat, Universitée Joseph Fourier Grenoble, 2006.

[192] G. Cleuziou, *Une méthode de classiddication non-supervisée pour l'apprentissage de règles et la recherche d'information*, Thèse de doctorat, Universitée d'Orléans 2004.

[193] R.N. Davé, *Characterization and detection of noise in clustering* Pattern Recognition Letters, Vol. 12, n. 11, pp. 657-664, 1991.

[194] D. Dembélé and P. Kastner, *Fuzzy C-means Clustering method for clustering microarray data*, Bioinformatics Vol. 19, No 8, pp. 573-580, 2003.

[195] I.S. Dhilon, J. Korgan and C. Nicholas, *Feature Selection and Document Clustering*, In M.W. Berry, editor, A Comprehensive Survey of Text Mining, pages 73-100, Springer-Verlag, 2003

[196] R.O. Duda and P.E. Hart *Pattern classification and scene analysis*, Wiley, 1972.

[197] J.C. Dunn, *A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact Well-Separated Clusters*, J. Cybernetics, vol. 3, No. 3, pp. 32-57, 1973.

[198] T. Feder and D. Greene, *Optimal algorithms for approximate clustering*, In Proc. STOC, 1988.

[199] D. Fisher, *Knowledge acquisition via incremental conceptual clustering*, Machine Learning, 2, pp. 139-172, 1987.

[200] K. Fukunaga, *Statistical Pattern Recognition*, AcademicPress, NY

[201] I. Gitman and M.D. Levine, *An Algorithm for Detecting Unimodal Fuzzy Sets and its Application as a Clustering Technique*, IEEE Trans. Comput., vol. c-19, no. 7, pp. 583-593, july 1970.

[202] R. Hathaway and J.C. Bezdek, *Local convergence of the fuzzy c-means algorithms*, Pattern Recognition 19, pp. 477-480, 1986.

[203] R. Hathaway, J.C. Bezdek and W. Tucker, *An improved convergence theorem for the fuzzy c -means clustering algorithms*, in The Analysis of Fuzzy Information, J. C. Bezdek, ed, Boca Raton, FL : CRC Press, vol. 3, ch. 8, 1987.

[204] R. Hathaway and J.C. Bezdek, *Recent Convergence Results for the Fuzzy c-Means Algorithms*, Journal of Classification, 5, pp. 237-247, 1988.

[205] M.A. Ismail and S.A. Selim, *On the local optimality of the fuzzy ISODATA clustering algorithm* IEEETrans. Pattern Anal. Machine Intell., Vol. PAMI-8, no. 2, pp. 284-288, 1986.

[206] M.A. Ismail and S.A. Selim, *Fuzzy c-means : Optimality of solutions and effective termination of the algorithm*, Pattern Recognition, vol. 19, n. 6, pp. 481-485, 1986.

[207] F. Klawonn and F. Hoppner, *What is Fuzzy About Fuzzy Clustering ? - Understanding and Improving the Concept of the Fuzzifier*, In : M.R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, C. Borgelt (eds.) : Advances in Intelligent Data Analysis, Berlin, 254-264, Springer, 2003.

[208] T. Kim, J.C. Bezdek and R. Hathaway, *Optimality test for fixed points of the fuzzy c-means algorithm*, Pattern Recognition, vol. 21, n. 6, pp. 651-663, 1988.

[209] R. Krishnapuram and J. Keller, *A possibilistic Approch to Clustering*, IEEE Trans. on Fuzzy Systems, vol. 1, n. 2, May 1993.

[210] R. Krishnapuram and J. Keller, *The Possibilistic C-Means Algorithm : Insights and Recommandations*, IEEE Trans. on Fuzzy Systems, vol. 4, n. 3, pp. 385-393, August 1996.

[211] L. Khoja, *Contribution à la Classification Floue non Supervisée*, Université de Savoie, 1997.

[212] Y. Liu, X. Shen and H. Doss, *Multicategory ψ-Learning and Support Vector Machine : Computational Tools.*, Journal of Computational and Graphical Statistics, 14, pp.219-236, 2005.

[213] Y. Liu and X. Shen, *Multicategoryψ-Learning*, Journal of the American Statistical Association, 101, pp.500-509, 2006.

[214] O.L. Mangasarian, *Mathematical Programming in Data Mining*, Data Mining and Knowledge Discovery 1, pp.183-201, 1987.

[215] B. Polyak, *Introduction to Optimization*, Optimization Software, Inc., Publication Division.New York, 1987.

[216] W. Pedrycz, *Conditional fuzzy c-means*, Pattern Recognition Letters 17, pp. 625-631, 1996.

[217] J.C Rajapakse, J.N Giedd and J.L Rapoport *Statistical Approach to Segmentation of Singke-Chanel Cerebral MR Images*, IEEE Trans. On Medical Imaging, 16, 2004.

[218] X. Shen, G.C. Tseng, X. Zhang and W. H. Wong, *ψ-Learning*, Journal of American Statistical Association, 98, pp.724-734, 2003.

[219] W.T. Tucker, *Counterexamples to the convergence theorem of fuzzy ISODATA clustering algorithms*, in The Analysis of Fuzzy Information, J. C. Bezdek, ed, Boca Raton, FL : CRC Press, vol. 3, ch. 7, 1987.

[220] A.L. Yuille and A. Rangarajan, *The Convex Concave Procedure (CCCP)*, Advances in Neural Information Processing System 14,Cambrige MA : MIT Press, 2002.

[221] . G. Whitwell, X Ying Wang and J.M. Garibaldi, *The Application of a Simulated Annealing Fuzzy Clustering Algorithm for Cancer Diagnosis*, SIP 2005, Japan.

[222] M.J. Sabin, *Convergence and consistency of fuzzy c-means/ISODATA algorithms*, IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-9, no. 5, pp. 661-668, September 1987.

[223] M. Rodrigues and L. Sacks, *scalable hierarchical fuzzy clustering algorithm for text mining*, In : Proc. of the 4th International Conference on Recent Advances in Soft Computing, RASC'2004, pp.269-274, Nottingham, UK, Dec. 2004.

[224] P.J. Rousseeuw, *Discussion : Fuzzy Clustering at the Intersection*, Technometrics, Vol. 37, n. 3, pp.283-286, August 1995.

[225] E.R. Ruspini, *A New Approach to Clustering*, Inform. Control, vol. 15, no. 1, pp. 22-32, July 1969.

[226] E.R. Ruspini, *Numerical Methods for Fuzzy Clustering*, Information Science, 2, pp. 319-350, 1970.

[227] M.S. Yang and K.F. Yu, *On stochastic convergence theorems for the fuzzy c-means clustering procedure*, Int. J. General Systems 16, pp. 397-411, 1990.

[228] M.S. Yang *Convergence properties of the generalized fuzzy c-means clustering algorithms*, Computers Math. Applic., vol. 25, n. 12, pp. 3-11, 1993.

[229] A.J. Clark, *Optimisation Heuristics for Cryptology*, PhD thesis, Faculty of Information Technology, Queensland University of Technology (1998).

[230] A. Canteaut, C. Carlet, P. Charpin and C.Fontaine, *Propagation characteristic and correlation-immunity of hight nonlinenar boolean function*, Advances in Cryptographie - EUROCRYPT 2000, Number 1807 in LNCS, Springer Verlag, 2000.

[231] J.F. Dillon, *Elementary Hadamard Difference sets* PhD Thesis, University of Maryland, 1974.

[232] S. Hirose and K. Ikeda, *Relationships among Nonlinearity Criteria of Boolean Functions*, IEICE Trans. on Fundamentals, vol. E78-A, no. 2, pp. 235-243, Feb. 1995.

[233] S. Hirose and K. Ikeda, *Complexity of Boolean Functions Satisfying the Propagation Criterion*, IEICE Trans. on Fundamentals, vol. E78-A, no. 4, pp. 470-478, Apr. 1995.

[234] S. Hirose and K. Ikeda, *Propagation Characteristics of Boolean Functions and Their Balancedness*, IEICE Trans. on Fundamentals, vol. E78-A, no. 1, pp. 11-18, Jan. 1995.

[235] M. Le Hoai,H.A. Le Thi, T. Pham Dinh and P. Bouvry, *A deterministic optimization approach for generating highly nonlinear balanced boolean function in cryptography*, High Performance Scientific Computing, Hanoi, Vietnam, 6-10 March 2006.

[236] S. Maitra, *On Nonlinearity and Autocorrelation Properties of Correlation Immune Boolean Functions*, Journal of Infomation Science and engineering 20,305-323,2004

[237] A. Menezes, P. Van Oorschot and S. Vanstone, *Handbook of Applied Cryptography.*, CRC Press 1996.

[238] J. Seberry, X.M. Zhang and Y. Zheng, *Nonlinearly balanced boolean functions and their propagation characteristics*, Advances in Cryptographie - EUROCRYPT 2000, In Advances in Cryptology - CRYPT0'93, Springer Verlag 1994.

[239] W. Stallings, *Cryptography and Network Security*, 3rd ed, Prentice Hall 2003.

[240] O.S. Rothaus, *On bent functions*, Journal of Combinatorial Theory, Series A, 20 :300-305, 1976.

[241] D. Fisher, *Iterative optimization and simplification of hierarchical clusterings*, Journal of Artificial Intelligence Research, vol. 4, pp. 147-180, 1996.

[242] A.K. Jain, M.N. Murty and P.J. Flynn, *Data clustering : a review*, ACM Computing Surveys, vol. 31, no. 3, pp. 264-323, 1999.

[243] J.B. Hiriart Urruty, C. Lemarechal, *Convex analysis and Minimization Algorithms I &II*, Springer Verlag, 1993.

[244] L. Jia, A. Bagirov, I. Ouveysi and A.M. Rubinov, *Optimization based clustering algorithms in Multicast group hierarchies*, Proceedings of the Australian Telecommunications, Networks and Applications Conference (ATNAC), 2003, Melbourne Australia, (published on CD, ISNB 0-646-42229-4).

[245] F. Murtagh, *A survey of recent advances in hierarchical clustering algorithms*, The Computer Journal, vol. 26, no. 4, 1983.

[246] R.T. Rockafellar, *Convex analysis*, Princeton, Princeton University Press, 1970.

[247] T.Wong, R. Katz, S. McCanne, *A Preference Clustering Protocol for Large-Scale Multicast Applications*, Proceedings of the First International COST264 Workshop on Networked Group Communication, pp 1-18, 1999.

[248] J.A. Clark and J.L. Jacob, *Fault Injection and a Timing Channel on an Analysis Technique* Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques : Advances in Cryptology, p. 181-196, LNCS Vol. 2332, 2002, Springer Verlag.

[249] A. Fiat and A. Shamir *How to prove yourself : practical solutions od identification ans signature problems* In Adlyzo, A.M..Ed.., Advances in Cryptology, Proceedings of CRYPTO'86, Vol 263 of Lecture Notes in Computer Science, p. 186-194, Santa-Barbara, Californie 1987, Springer Verlag.

[250] M. Finiazs, *Syndrome decoding in the Non-Standard Cases*, My-Crypt 2005.

[251] L. Guillou and J. Quisquater, *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory*, Gunter C.G., Ed., Advances in Cryptology - Proceedings of EUROCRYPT'88, LNCS Vol. 330, p. 123-128, 1988, Davos Suisse, Springer-Verlag

[252] L.R. Knudsen and W. Meier, *Cryptanalysis of an Identifcation Scheme Based on the Permuted Perceptron Problem*, Eurocrypt'99, 1999.

[253] L . Lamport *Password Authentication with Insecure Communication* Communication of ACM 24, 11 :770-771, 1981

[254] K. Ohta and T. Okamoto, *A Modification of the Fiat-Shamir Scheme* In Gold-wasser S.Ed, Advances in Cryptology - Proceedings of CRYPTO'88, LNCS, Vol.403, p.232-242, Santa-Barbara, Californie, 1989, Springer-Verlag.

[255] D. Pointcheval, *Les preuves de connaissance et leurs preuves de sécurité* Thèse de doctoral, Université de Caen, 1996.

[256] , D. Pointcheval, *A new identification scheme based on the perceptron problem*, Eurocrypt'95, 1995

[257] A. Shamir, *An effcient identification scheme based on permuted kernels*, Springer Verlag, 1998.

[258] J. Stern, *A new identification scheme based on syndrome decoding*, Advences in Cryptology-CRYPTO'93, 1990.

[259] J. Stern, *Designing Identification Schemes with Keys of Short Size* In Desmedt Y.G, Advances in Cryptology - Proceedings of CRYPTO'94, LNCV Vol. 839, p. 164-173, Santa-Barbara, Californie, 1994, Springer Verlag.

[260] G. Waters, *Hierarchies for network evolution*, Sixteenth UK Teletraffic Symposium on Management of Quality of Service - the New Challenge,Harlow, UK, May 2000.

[261] G. Waters and S. Guan Lim, *Applying clustering algorithms to multicast group hierarchies*, Technical Report No. 4-03 August 2003.

[262] G. Waters, J. Crawford and S. Guan Lim, *Optimising multicast structures for grid computing*, Computer Communications27 1389-1400, 2004.