Non convex optimization techniques based on DC programming and DCA and evolution methods for clustering

Minh Thuy Ta

THÈSE

en vue de l'obtention du titre de

DOCTEUR DE L'UNIVERSITÉ DE LORRAINE

(arrêté ministériel du 7 Août 2006)

Spécialité Informatique

présentée par

Ta Minh Thuy

Titre de la thèse :

Techniques d'optimisation non convexe basée sur la programmation DC et DCA et méthodes évolutives pour la classification non supervisée

soutenue le 04 Juillet 2014

Composition du Jury :

Rapporteurs	Jin-Kao HAO	*Professeur, Université d'Angers*
	Mustapha LEBBAH	*MCF, HDR, Université Paris 13*
Examinateurs	Pierre GANÇARSKI	*Professeur, Université de Strasbourg*
	Tao PHAM DINH	*Professeur, INSA-Rouen*
	Guillaume CLEUZIOU	*MCF, Université d'Orléans*
Directrice de thèse	Hoai An LE THI	*Professeur, Université de Lorraine*
Co-encadrant	Lydia BOUDJELOUD-ASSALA	*MCF, Université de Lorraine*

Thèse préparée au sein de laboratoire
d'Informatique Théorique et Appliquée (LITA)
Université de Lorraine, Metz, France

Remerciements

Je souhaite remercier en premier lieu ma directrice de thèse, Madame, le Professeur LE THI Hoai An, pour l'aide qu'elle m'a apportée, pour sa patience, pour son soutien permanent et pour ses précieux conseils, pour les discussions que l'on a pu avoir qui se sont toujours révélées très intéressantes et instructives. Son œil critique m'a été très précieux pour structurer le travail. Un grand merci pour les nuits sans sommeil pour corriger mes articles et ma thèse. Je voudrais aussi sincèrement la remercier pour son intérêt pour ma vie et ma famille, en France et au Vietnam. Merci de m'avoir enseigné ses connaissances de la science et de la vie.

J'adresse toute ma gratitude à Madame Lydia BOUDJELOUD-Assala, Maître de conférence, ma co-encadrante, pour avoir partagé ses connaissances scientifiques, pour toute l'aide prodiguée, pour sa patience et sa sympathie, malgré toutes les difficultés de communication rencontrées lors des premiers mois de thèse. Merci de m'avoir encouragé à étudier pendant plus de trois ans loin de mon pays.

J'adresse également mes sincères remerciements à Monsieur PHAM DINH Tao, Professeur à l'INSA de Rouen pour ses conseils, et son suivi dans mes recherches. Je voudrais lui exprimer toute ma reconnaissance pour les discussions très intéressantes qu'il a menées et pour m'avoir suggéré de nouvelles voies de recherche.

Je souhaite vivement remercier Monsieur Jin-Kao HAO, Professeur à l'Université d'Angers et Monsieur Mustapha LEBBAH, MCF, HDR, Université Paris 13, de m'avoir fait l'honneur d'accepter la charge de rapporteur de ma thèse, ainsi que d'avoir participé au jury.

Je souhaite également exprimer ma gratitude à Monsieur Pierre GANÇARSKI, Professeur à l'Université de Strasbourg, Monsieur Guillaume CLEUZIOU, MCF à l'Université d'Orléans de participer au jury.

Je remercie particulièrement le Docteur LE Hoai Minh pour les discussions intéressantes que nous avons eu lors de nos collaborations. Je le remercie également pour le partage dans la vie.

Mes remerciements s'adressent également au Gouvernement Vietnamien qui a financé mes études pendant 3 ans.

Un grand merci à mes collègues du LITA, qui m'ont aidé pendant plus de trois ans. Je n'oublie pas de remercier USTH (University of Science and Technology of Ha Noi) pour son soutien.

Je remercie tous mes collègues français et vietnamiens rencontrés à Metz, pour les moments agréables lors de mon séjour en France. J'associe à ces remerciements Anh Vu, Bich Thuy, Manh Cuong, Xuan Thanh, Tran Thuy, Duy Nhat, Anh Son, Duc Quynh, Trong Phuc,... pour leurs partages dans le travail et dans la vie.

Je n'oublie pas de témoigner mon affectation, ma reconnaissance à ma famille et mon fils pour les sacrifices qu'ils ont faits, pour leur soutien lors des moments difficiles.

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.

Résumé

Nous nous intéressons particulièrement, dans cette thèse, à quatre problèmes en apprentissage et fouille de données : clustering pour les données évolutives, clustering pour les données massives, clustering avec pondération de variables et enfin le clustering sans connaissance a priori du nombre de clusters avec initialisation optimale des centres de clusters.

Les méthodes que nous décrivons se basent sur des approches d'optimisation déterministe, à savoir la programmation DC (Difference of Convex functions) et DCA (Difference of Convex Algorithms), pour la résolution de problèmes de clustering cités précédemment, ainsi que des approches évolutionnaires élitistes.

Nous adaptons l'algorithme de clustering DCA–MSSC pour le traitement de données évolutives par fenêtres, en appréhendant les données évolutives avec deux modèles : fenêtres fixes et fenêtres glissantes.

Pour le problème du clustering de données massives, nous utilisons l'algorithme DCA en deux phases. Dans la première phase, les données massives sont divisées en plusieurs sous-ensembles, sur lesquelles nous appliquons l'algorithme DCA–MSSC pour effectuer un clustering. Dans la deuxième phase, nous proposons un algorithme DCA-Weight pour effectuer un clustering pondéré sur l'ensemble des centres obtenues à la première phase.

Concernant le clustering avec pondération de variables, nous proposons également deux approches: clustering dur avec pondération de variables et clustering floue avec pondération de variables. Nous testons notre approche sur un problème de segmentation d'image.

Le dernier problème abordé dans cette thèse est le clustering sans connaissance a priori du nombre des clusters. Nous proposons pour cela une approche évolutionnaire élitiste. Le principe consiste à utiliser plusieurs algorithmes évolutionnaires (EAs) en même temps, de les faire concourir afin d'obtenir la meilleure combinaison de centres initiaux pour le clustering et par la même occasion le nombre optimal de clusters. Les différents tests réalisés sur plusieurs ensembles de données de grande taille sont très prometteurs et montrent l'efficacité des approches proposées.

Abstract

This thesis focus on four problems in data mining and machine learning: clustering data streams, clustering massive data sets, weighted hard and fuzzy clustering and finally the clustering without a prior knowledge of the clusters number.

Our methods are based on deterministic optimization approaches, namely the DC (Difference of Convex functions) programming and DCA (Difference of Convex Algorithm) for solving some classes of clustering problems cited before. Our methods are also, based on elitist evolutionary approaches.

We adapt the clustering algorithm DCA–MSSC to deal with data streams using two windows models: sub–windows and sliding windows.

For the problem of clustering massive data sets, we propose to use the DCA algorithm with two phases. In the first phase, massive data is divided into several subsets, on which the algorithm DCA–MSSC performs clustering. In the second phase, we propose a DCA–Weight algorithm to perform a weighted clustering on the obtained centers in the first phase.

For the weighted clustering, we also propose two approaches: weighted hard clustering and weighted fuzzy clustering. We test our approach on image segmentation application.

The final issue addressed in this thesis is the clustering without a prior knowledge of the clusters number. We propose an elitist evolutionary approach, where we apply several evolutionary algorithms (EAs) at the same time, to find the optimal combination of initial clusters seed and in the same time the optimal clusters number. The various tests performed on several sets of large data are very promising and demonstrate the effectiveness of the proposed approaches.

TA Minh Thuy

Né le 13 Octobre, 1980 (Viet Nam)

Tél: 06 49 98 92 33

E-mail: minh-thuy.ta@univ-lorraine.fr

Adresse personnelle: 41 rue du Pont des Morts, 57000, Metz

Adresse professionnelle: LITA – Université de Lorraine, Ile du Saulcy, 57045 Metz

Situation Actuelle

Depuis décembre 2010	Doctorant au Laboratoire d'Informatique Théorique et Appliquée (LITA EA 3097) de l'Université de Lorraine. Encadré par Prof. Le Thi Hoai An et MCF. Lydia Boudjeloud–Assala. Sujet de thèse : **"Techniques d'optimisation non convexe basée sur la programmation DC et DCA et méthodes évolutives pour la classification non supervisée"**, bourse des gouvernements vietnamo–française.

Experience Professionnelle

2007–2010	Centre des technologies de l'information de la Banque – Banque internationale du Vietnam (VIB), Ha Noi.
2002–2007	Chercheur (Image, Approximation, E-Learning), Institut de technologie de l'information, Université Nationale du Vietnam, Ha Noi (VNUH).

Diplôme et Formation

2010–à ce jour	Doctorant en Informatique. LITA–Université de Lorraine, Metz.
2002–2004	Master Mathématiques. Sujet: **"Image Deblur by Tikhonov Regularization Methods"**. Faculté de Mathématique – Mécanique – Informatique, Université de Sciences de Hanoi, Vietnam.
1998–2002	Licence Mathématiques appliquées et Informatiques. Sujet: **"Explicit Pseudo Two–Step Runge–Kutta–Nystrom Method"**. Faculté de Mathématique – Mécanique – Informatique, Université de Sciences de Hanoi, Vietnam.

Publications

Refereed international journal papers

[1]. Le Hoai Minh, Ta Minh Thuy. *DC programming and DCA for solving Minimum Sum–of–Squares Clustering using weighted dissimilarity measures.* Special Issue on Optimization and Machine Learning, Transactions on Computational Intelligence XIII, ISBN: 978–3–642–54454–5 (Print) 978–3–642–54455–2 (Online) (2014).

[2]. Lydia Boudjeloud–Assala, Ta Minh Thuy. *A clustering algorithm based on elitist evolutionary approach.* 27 pages, Submitted to Journal of Classification.

Refereed papers in books / Refereed international conference papers

[1]. Lydia Boudjeloud–Assala, Ta Minh Thuy. *Determine optimal number of clusters with an elitist evolutionary approach*, Advances in Knowledge Discovery and Data Mining, PAKDD'2014, Lecture Notes in Computer Science, Volume 8444, pp 324–335 (2014).

[2]. Ta Minh Thuy, Hoai An Le Thi, and Lydia Boudjeloud-Assala. *An Efficient Clustering Method for Massive Dataset Based on DC Programming and DCA Approach.* 20th International Conference on Neural Information Processing, ICONIP'2013, 3–7 November, Part II, Lecture Notes in Computer Science Volume 8227, pp. 538–545 (2013).

[3]. Hoai Minh Le, Bich Thuy Nguyen Thi, Minh Thuy Ta, Hoai An Le Thi. *Image Segmentation via Feature Weighted Fuzzy Clustering by a DCA Based Algorithm.* Advanced Computational Methods for Knowledge Engineering, Studies in Computational Intelligence. Volume 479, Springer, ISSN: 1860–949X (Print) 1860–9503 (Online), pp 53–63 (2013).

[4]. Minh Thuy Ta, Hoai An Le Thi, Lydia Boudjeloud–Assala. *Clustering data streams over sliding windows by DCA.* Advanced Computational Methods for Knowledge Engineering, Studies in Computational Intelligence. Volume 479, Springer, ISSN: 1860–949X (Print) 1860–9503 (Online), pp. 65–75 (2013).

[5]. Le Hoai Minh, Ta Minh Thuy, Le Thi Hoai An and Pham Dinh Tao. *DC Programming and DCA for clustering using weighted dissimilarity measures*, Proceedings of the 5th NIPS Workshop on Optimization for Machine Learning, Lake Tahoe, Nevada, USA, 5 pages (2012).

[6]. Minh Thuy Ta, Hoai An Le Thi, Lydia Boudjeloud–Assala. *Clustering Data Stream by a Sub-window Approach Using DCA*. Machine Learning and Data Mining in Pattern Recognition (MLDM2012), Lecture Notes in Computer Science Volume 7376, ISBN 978–3–642–31536–7, pp. 279–292 (2012).

Communications in national / International conferences

[1]. M.T. TA, H.A. Le Thi, L. Boudjeloud–Assala, *A DC programming approach for clustering massive data sets*. 25th European Conference on Operational Research (EURO2012), 8–11 July, Vilinus, Lithuania (2012).

[2]. M.T. TA, *Traitement de masses de données par une méthode d'optimisation non convexe*. Journées communes aux Groupes de Travail EGC et AFIHM, Fouille et Visualisation de Données Massives, Big Data Mining and Visualization, 18–19 Juin, Tours, France (2012).

[3]. M.T. TA, H.A. Le Thi, L. Boudjeloud–Assala, *Clustering data streams based on sub-windows: a DC programming approach*. 15th Austrian–French–German conference on Optimization (AFG11), 19–23 September, Toulouse, France (2011).

Contents

List of Figures

List of Tables

Introduction générale

Cadre général et nos motivations

De nos jours, pour répondre aux exigences du développement de la technologie, nous nous retrouvons face à de gros volumes de données pouvant évoluer dans le temps. Les méthodes classiques d'apprentissage et de fouille de données deviennent inadaptées pour appréhender de telles données. Il devient donc, nécessaire de proposer de nouvelles approches, plus efficaces, pouvant traiter des données massives et/ou évolutives.

Dans ce contexte nous avons choisi d'utiliser des approches d'optimisation déterministe, à savoir la programmation DC (Difference of Convex functions) et DCA (Difference of Convex Algorithms), pour la résolution de certaines classes de problèmes en classification non supervisée, un domain important d'apprentissage et de fouille de données.

On peut distinguer deux branches de l'optimisation déterministe: la programmation convexe et la programmation non convexe. Le principe d'un programme convexe ou d'un problème d'optimisation convexe consiste à la minimisation d'une fonction convexe (Objectif) sous des contraintes convexes. Lorsque la double convexité dans la fonction Objectif et dans les contraintes, n'est pas vérifiée, on est face à 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 structure du programme convexe en question 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 à 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 contrainte à une extension logique et naturelle à la non convexité et la non différentiabilité. Les méthodes

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

L'optimisation non convexe connaît une explosion spectaculaire depuis des années 90, 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. Il existe deux approches différentes mais complémentaires en programmation non convexe:

1. 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 d'évaluation. Elles ont connu de très nombreux développements importants au cours de ces dernières années à travers les travaux de H. Tuy (Horst and Tuy [1996]) (reconnu comme le pionnier), R. Horst R. Horst, H. Benson, P.M. Pardalos, H. Konno, Le Dung Muu, Le Thi Hoai An, Nguyen Van Thoai, Phan Thien Thach, and Pham Dinh Tao. 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.
2. Approches locales et globales d'analyse convexe qui sont basées sur l'analyse et l'optimisation convexe. Ici la programmation DC et DCA jouent un rôle central, car la plupart des problèmes d'optimisation non convexe sont formulés/reformulés sous la forme DC. Sur le plan algorithmique, l'essentiel repose sur les algorithmes de l'optimisation DC (DCA) introduits par Pham Dinh Tao en 1985 à l'état préliminaire et développés intensivement à travers de nombreux travaux communs de Le Thi Hoai An et Pham Dinh Tao depuis 1993 pour devenir maintenant classiques et de plus en plus utilisés par des chercheurs et praticiens de par le monde, dans différents domaines des sciences appliquées (pour ne citer que quelques-uns, voir la liste des références dans Le Thi [web site]).

La programmation DC et DCA considèrent le problème DC de la forme

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

où g et h sont des fonctions convexes définies sur $\mathbb{R}^n$ et à valeurs dans $\mathbb{R} \cup \{+\infty\}$, semi-continues inférieurement et propres. La fonction f est appelée fonction DC avec les composantes DC g et h, et $g - h$ est une décomposition DC de f. DCA est basé sur la dualité DC et des conditions d'optimalité locale. La construction de DCA implique les composantes DC g et h et non la fonction DC f elle-même. Or chaque fonction DC admet une infinité des décompositions DC qui influencent considérablement sur la qualité (la rapidité, l'efficacité, la globalité de la solution obtenue,...) de DCA. Ainsi, au point de vue algorithmique, la

recherche d'une "bonne" décomposition DC et d'un "bon" point initial est très importante dans le développement de DCA pour la résolution d'un programme DC.

Les travaux dans cette thèse sont basés sur la programmation DC et DCA. Cette démarche est justifiée par de multiple arguments (Le Thi [2012], Pham Dinh and Le Thi [2014]):

- La programmation DC et DCA fournissent un cadre très riche pour les problèmes d'apprentissage et de fouille de données (Machine Learning and Data Mining - MLDM): MLDM constituent *une mine des programmes DC* dont la résolution appropriée devrait recourir à la programmation DC et DCA. En effet la liste indicative (non exhaustive) des références dans Le Thi [web site] témoigne de la vitalité, la puissance et la percée de cette approche dans la communauté de MLDM.
- DCA est une philosophie plutôt qu'un algorithme. Pour chaque problème, nous pouvons concevoir une famille d'algorithmes basés sur DCA. La flexibilité de DCA sur le choix des décomposition DC peut offrir des schémas DCA plus performants que des méthodes standard.
- L'analyse convexe fournit des outils puissants pour prouver la convergence de DCA dans un cadre général. Ainsi tous les algorithmes basés sur DCA bénéficient (au moins) des propriétés de convergence générales du schéma DCA générique qui ont été démontrées.
- DCA est une méthode efficace, rapide et scalable pour la programmation non convexe. A notre connaissance, DCA est l'un des rares algorithmes de la programmation non convexe, non différentiable qui peut résoudre des programmes DC de très grande dimension. La programmation DC et DCA ont été appliqués avec succès pour la modélisation DC et la résolution de nombreux et divers problèmes d'optimisation non convexes dans différents domaines des sciences appliquées, en particulier en MLDM (voir par exemple la liste des références dans Le Thi [web site]).
  Il est important de noter qu'avec les techniques de reformulation en programmation DC et les décompositions DC appropriées, on peut retrouver la plupart des algorithmes existants en programmation convexe/non convexe comme cas particuliers de DCA.

En particulier, pour la communauté de Data Mining-Machine Learning, les méthodes suivantes EM: Expectation–Maximisation (Dempster et al. [1977]), SLA: Succesive Linear Approximation (Bradley and Mangasarian [1998]), CCCP: ConCave–Convex Procedure (Yuille and Rangarajan [2003]) ISTA (Iterative Shrinkage-Thresholding Algorithms) (Chambolle et al. [1998]), sont des cas particuliers de DCA.

Autre direction de recherche, dans notre travail, est l'approche évolutionnaire élitiste. Les algorithmes évolutionnaires (Abraham et al. [2006], Whitley [2001]) sont une famille d'algorithmes simulant l'évolution naturelle pour trouver des solutions à un problème d'optimisation. L'évolution est déterminée par la sélection naturelle et la concurrence entre les individus. Les meilleurs individus sont ceux qui essayent de survivre, de maintenir et/ou de propager leur génétique. Ces algorithmes manipulent des populations de solutions.

Les algorithmes évolutionnaires s'inspirent de l'évolution des êtres vivants, en considérant que

celle-ci tend à produire des organismes plus adaptés à leur environnement. Selon la théorie de l'évolution, plusieurs mécanismes sont à l'œuvre pour ce faire. Les caractéristiques d'un organisme sont en grande partie codées dans ses gènes, chaque population d'organismes est composée d'individus tous différents, les différences entre individus leur confèrent une adaptation plus ou moins grande à leur environnement. Les organismes transmettent une partie de leurs caractéristiques à leurs descendants et les individus les plus adaptés se reproduisent plus efficacement. Leurs caractéristiques ont donc tendance à davantage se répandre dans la population.

Les algorithmes évolutionnaires comportent certains avantages (Fogel [1997], Blickle [1996], Abraham et al. [2006]):

– Ils sont applicables à plusieurs types de problèmes: discontinus, non–linéaires, multi–modaux et particulièrement à des problèmes ayant des solutions multiples.
– Ils peuvent être parallélisés.
– Ils sont simples et faciles à mettre en œuvre.
– Ils sont robustes aux changements dynamiques.

Nous nous intéressons, dans cette thèse, à l'approche évolutionnaire élitiste. Une des difficultés concernant l'application des algorithmes évolutionnaires (EAs) est le paramétrage de ces derniers. Pour ne pas faire face à ce problème en particulier, nous proposons une approche élitiste faisant concourir plusieurs algorithmes évolutionnaires (EAs) afin d'obtenir la meilleure solution a un problème donné. Le principe de l'approche élitiste est donc, de mettre en concurrence plusieurs algorithmes évolutionnaires, les meilleurs solutions produites sont stockées dans une population *élite*. La solution optimale est représentée par l'individu *élite*.

Nos contributions

Dans ce travail, nous nous intéressons particulièrement à quatre problèmes en classification non supervisée: clustering pour les données évolutives, clustering pour les données massives, clustering avec pondération de variables et clustering sans connaissance a priori du nombre de clusters avec initialisation optimal des centres de clusters.

Nous proposons d'appréhender les données évolutives avec deux modèles: approche par des sous–fenêtre fixes et approche par sous–fenêtres glissantes. Les données évolutives représentent un défi en fouille de données, non seulement par l'importance du volume des données évolutives à traiter, mais aussi par le fait que les données peuvent être corrélées dans le temps. Ces corrélations permettent de détecter des caractéristiques importantes de l'évolution des données dans le temps. Une des stratégies les plus efficaces pour appréhender les données évolutives est l'analyse par paquets (Silva [2007]). L'approche consiste à séparer les données dans des sous–fenêtres (fixes ou glissantes), dans le but de détecter des modèles, ce qui n'aurait pas été possible avec une analyse globale du flux. Un algorithme de clustering est

ensuite, appliqué sur chaque fenêtre. Nous utilisons l'algorithme de clustering DCA–MSSC (Le Thi et al. [2007a]) pour le traitement de données évolutives par fenêtres via deux options: fenêtres fixes et fenêtres glissantes. Concernant l'approche par fenêtres fixes, un clustering global est effectué sur la totalité des données et un clustering local indépendant est effectué de façon indépendantes sur chaque fenêtres, avec une initialisation aléatoire des centres de clusters. Pour évaluer l'efficacité de cette approche, une première comparaison des résultats sur chaque fenêtre est faite avec les classes connues des données. Une seconde comparaison est réalisée avec les résultats du clustering global obtenue sur la totalité des données. Concernant la seconde approche (modélisation par fenêtres glissantes), nous prenons en compte que les éléments les plus récents. S'appuyant sur l'algorithme DCA–MSSC (Le Thi et al. [2007a]), nous proposons un algorithme DCA–Stream, pour le problème de clustering par fenêtres glissantes. Le principe de l'algorithme DCA–Stream est de proposer un clustering sur la fenêtre t en se basant sur le clustering effectué sur la fenêtre $t-1$. Pour évaluer l'efficacité de cet algorithme, une comparaison avec DCA–MSSC-random (en utilisant des points de départ aléatoires) et l'algorithme k-Means standard est effectuée.

Le deuxième problème traité dans cette thèse est le clustering de données massives. En s'appuyant sur l'approche proposée par Nittel et al. [2004], Nittel and Leung [2004], nous proposons d'utiliser l'algorithme DCA en deux phases. Dans la première phase, les données massives sont divisées en plusieurs sous-ensembles, sur lesquelles nous appliquons l'algorithme DCA–MSSC (Le Thi et al. [2007a]) pour effectuer un clustering. Dans la deuxième phase, nous appliquons DCA-Weight (une adaptation de DCA–MSSC sur le modèle MSSC en ajoutant un coefficient de poids) pour effectuer un clustering pondéré sur l'ensemble des centres obtenues à la première phase sur les différents sous-ensembles. Chaque centre de cluster est pondéré par le nombre de points appartenant au cluster. Des résultats expérimentaux sur des ensembles de données réelles et synthétiques, sont prometteurs et démontrent l'efficacité de notre approche.

Concernant le clustering avec pondération de variables, nous proposons également deux approches: clustering dur avec pondération de variables et clustering floue avec pondération de variables. Habituellement, les variables peuvent être divisés en trois catégories: les variables pertinentes, redondantes et non pertinentes. Les variables pertinentes sont essentielles pour le processus de classification, les variables redondantes n'apportent aucune nouvelle information au classifieur, tandis que les variables non pertinente ne fournissent aucune information utile. Chaque variable est affectée d'une valeur continue dans l'intervalle $[0, 1]$, nommée un poids (les variables pertinentes auront un poids élevé).

Dans la première approche, nous considérons le clustering dur où chaque objet appartient à une et une seule classe, avec deux formulations DC : programmation avec variable mixte zéro–un et programmation à deux niveaux. Nous nous basons sur les approches DC, développées dans Le Thi et al. [2007a] et Le Thi et al. [2014c]. Nous proposons une reformulation du problème d'optimisation en utilisant une technique de pénalité. Dans la seconde approche, nous considérons le clustering floue (où un objet peut appartenir à plusieurs classes) avec pondération de variables. Nous proposons un nouveau programme DC et un algorithme pour le clustering floue avec pondération de variables. Nous appliquons l'algorithme développé au

problème de segmentation d'image qui est un problème réel de grande taille.

Le dernier problème traité dans cette thèse est le clustering sans connaissance a priori du nombre des clusters. Dans la littérature, il existe de nombreuses approches pour résoudre ce problème, cependant, il reste encore un grand défi en fouille de données. Pour appréhender ce problème, nous proposons une approche évolutive élitiste. Le principe consiste à utiliser plusieurs algorithmes évolutionnaires (EAs) en même temps, de les faire concourir afin de trouver la meilleure solution. Une des difficultés concernant l'application des algorithmes évolutionnaires (EAs) est le paramétrage de ces derniers. Pour ne pas faire face à ce problème en particulier, nous exécutons plusieurs algorithmes évolutionnaires (EAs) avec plusieurs valeurs de paramètres que nous faisons concourir afin d'obtenir la meilleure combinaison de centres initiaux pour le clustering. En ayant un codage sous forme de combinaison de centres initiaux, nous pouvons ainsi proposer le nombre optimal de clusters. Nous proposons également, une nouvelle stratégie de mutation basée sur le voisinage. Le gène muté est échangé par un point (objet des données) n'appartenant pas au voisinage des autres points composant l'individu génétique. Pour déterminer le voisinage d'un point, nous proposons une méthode qui détermine la limite d'un cluster de façon automatique. La distance entre tous les points et le centre potentiel représenté par un gène, est calculée et ordonnée de la plus petite à la plus grande. Nous essayons, ensuite, de trouver une augmentation abrupte de la distance, indiquant, ainsi, la limite du cluster. Nous avons choisi d'utiliser la méthode de détection de pic présentée par Palshikar [2009]. L'approche proposée, peut être utilisée directement comme un algorithme de clustering ou comme initialisation de l'algorithme des k–Means. Les tests réalisés sur des ensembles de données de grande taille et multi–classes, sont très prometteurs et montrent l'efficacité de notre approche.

Organisation de la thèse

La thèse est composée de cinq chapitres.

Le premier chapitre décrit de manière succincte la programmation DC et DCA, Il présente les outils théoriques et algorithmiques servant des références aux autres chapitres. La première partie de ce chapitre, concerne la programmation DC et DCA tandis que la deuxième porte principalement, sur les algorithmes DCA pour le problème de clustering. Nous considérons particulièrement, l'algorithme de clustering DCA–MSSC Le Thi et al. [2007a], qui sera utilisé dans les chapitres suivants. Le second chapitre est consacré au problème du clustering des données évolutives. Deux approches - par fenêtres fixes et par fenêtres glissantes, sont proposées. Le clustering de données massive est traité dans le troisième chapitre. Le clustering dur et floue avec pondération de variables sont étudiés dans le quatrième chapitre, avec une application réelle pour la segmentation d'images. Nous abordons, enfin, le clustering sans connaissance a priori du nombre des clusters en proposant une approche évolutive élitiste, dans le dernier chapitre.

Chapter 1

Methodology

1.1 Introduction

DC programming and DCA, which constitute the backbone of nonconvex programming and global optimization, were introduced by Pham Dinh Tao in their preliminary form in 1985 (Pham Dinh [1986], Pham Dinh and Bernoussi [1988]), and extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1993 to become now classic and increasingly popular (see e.g. the list of reference in Le Thi [web site]). These theoretical and algorithmic tools are applied with great success in different renowned laboratories (to name a few: Princeton, Stanford, MIT, Berkeley, Carnegie Mellon, Cornell, Imperial College, Institut für Allgemeine Mechanik (IAM, RWTH-Aachen), California, Mannheim, Heidelberg, Courant Institute of Mathematical Sciences, Minnesota, The University of North Carolina at Chapel Hill, Michigan, Iowa, Florida, Tokyo Institute of Technology, Fribourg, EPFL, National-ICT-Australia-(NICTA), University of Sydney, Fudan, The Chinese University of Hongkong, Hanoi Institute of Mathematics, Coimbra, Vienna, Copenhague, Louvain, Pukyong, Namur, Google, Yahoo, Nasa, Siemens, etc) for modeling and solving nonconvex programs.

The theoretical results on DC Programming and DCA can be found in Le Thi [1994, 1997], Pham Dinh and Le Thi [1997, 1998], Le Thi et al. [1999], Le Thi and Pham Dinh [2002, 2005], Le Thi et al. [2012a] and Pham Dinh and Le Thi [2014] while numerical methods based on DCA for solving difficult problems such as bilevel programming problems, nonconvex quadratic programming problems, binary quadratic programming problems, optimization over efficient / weakly efficient set, trust region subproblems, linear complementarity problems, etc, can be found in various joint works of Le Thi Hoai An and Pham Dinh Tao (see e.g. Le Thi and Pham Dinh [1997], Le Thi et al. [2002, 2012b], Pham Dinh and Le Thi [1998] and the list of references in Le Thi [web site]).

DC Programming and DCA have been successfully used by researchers and practitioners to model and solve their nonconvex programs from many fields of Applied Science, whose Data Mining and Machine Learning remains, to date, the domain of predilection. In particular,

DCA is widely used in various areas of unsupervised learning (Pan et al. [2013], Le Thi et al. [2007a,b], Le Hoai et al. [2013b], Le Thi et al. [2014c, 2007c, 2008a, 2007b], Le Hoai and Ta [2014]), supervised learning (Wang et al. [2010], Le Thi et al. [2008e,b,c, 2013e,b]), semi-supervised learning (Wang et al. [2009], Tian et al. [2012], Yang and Wang [2013], Le Hoai et al. [2013a]), learning with sparsity/uncertainty (Mamadou et al. [2010], Le Thi et al. [2013e], Cheng Soon and Le Thi [2013], Le Thi et al. [2013c], Phan et al. [2014]), in dictionary learning (Fawzi et al. [2014]),... For other domains, to cite a few, in finance - risk management, portfolio selection (Mokhtar et al. [2014], Mohri and Medina [2014], Le Thi and Moeini [2014], Le Thi et al. [2012b], Le Thi and Tran [2012], Le Thi et al. [2009a,b, 2014a], Pham Dinh et al. [2014], Pham et al. [2013], Le Thi and Moeini [2006], Pham Dinh et al. [2009], Nguyen et al. [2011]); in transport-logistic (Zhong and Aghezzaf [2011, 2009], Ndiaye et al. [2008], in communication systems - routing Ta et al. [2012a], Nguyen and Le Thi [2011], Ta et al. [2010a, 2012b, 2010b], Le Thi and Pham Dinh [2014]), wireless networks (Wu et al. [2014], Vucic et al. [2010], Le Thi et al. [2008d]); in production management - supply chain design, scheduling (Le Thi et al. [2007d], Nguyen et al. [2007], Le Thi et al. [2009d], Le Hoai et al. [2012], Nguyen and Le Thi [2011], Le Thi et al. [2013d, 2009c], Le Thi and Tran [2014]); in bioinformatics (Ying et al. [2009], Le Thi et al. [2008b, 2013a, 2008e]); in data security - cryptography, anomaly detection (Le Hoai et al. [2008, 2010], Le Thi et al. [2009e, 2014b]); as well as in computer vision and image/signal processing (Weber et al. [2006], Schnörr [2007], Kokiopoulou et al. [2009], Gasso et al. [2009], Le Hoai et al. [2013c], Le Thi et al. [2008a]). See Le Thi [web site] for a more complete (but not exhaustive) list.

In the first part of this chapter we give a brief introduction of DC (Difference of Convex functions) programming and DCA (DC Algorithms) which constitutes the backbone of our work. In the second part we present the clustering algorithm DCA-MSSC which is background of our DCA based approaches developed in the chapters 2, 3 and 4.

1.2 DC programming and DCA

In this section, we give a brief presentation of DC programming and DCA. The materials presented in section are extracted from Le Thi [1997].

Their original key idea relies on the structure DC of objective function and constraint functions in nonconvex programs which are explored and exploited in a deep and suitable way. *The resulting DCA introduces the nice and elegant concept of approximating a nonconvex (DC) program by a sequence of convex ones: each iteration of DCA requires solution of a convex program.*

Their popularity resides in *their rich, deep and rigorous mathematical foundations, and the versality, flexibility, robustness, inexpensiveness and efficiency of DCA's compared to existing methods, their adaptation to specific structures of addressed problems and their ability to solve real-world large-scale nonconvex programs.* Recent developments in convex programming are mainly devoted to reformulation techniques and scalable algorithms in order to handle large-

scale problems. Obviously, they allow for enhancement of DC programming and DCA in high dimensional nonconvex programming.

For beginning, let us present some fundamental notations of convex analysis and DC programming

1.2.1 Fundamentals of DC Annalysis

Definitions and properties

This paragraph is devoted to a brief recall of convex analysis to facilitate the reading of its content. For more details, we refer to the work of Laurent [1972], of Rockafellar [1970] and of Auslender [1976].

Let X be the Euclidean space $\mathbb{R}^n$, equipped with the canonical inner product $\langle ., . \rangle$ and its Euclidean norm $\|x\| = \langle x, x \rangle^{\frac{1}{2}}$. The dual vector space of X is denoted by Y, which can be identified with X itself. We use the basic tools of modern convex analysis where a function can take the infinity value $+\infty$ (Rockafellar [1970]). For $f : X \longrightarrow \mathbb{R} \cup \{+\infty\}$, its effective domain of f, denoted by $\text{dom}(f)$, is

$$\text{dom}(f) = \{x \in X : f(x) < +\infty\} \tag{1.1}$$

and the epigraph of f, denoted by $\text{epi}(f)$, is

$$\text{epi}(f) = \{(x, \lambda) \in X \times \mathbb{R} : f(x) \leq \lambda\}.$$

If $\text{dom}(f) \neq \emptyset$ then we say that the function f is *proper*.

A proper function $f :\longrightarrow \mathbb{R} \cup \{+\infty\}$ is called *convex* if its epigraph is a convex set in $X \times \mathbb{R}$. This is equivalent to

$$f((1-\lambda)x^1 + \lambda x^2) \leq (1-\lambda)f(x^1) + \lambda f(x^2), \forall x^1, x^2 \in X \ , \ \forall \lambda \in]0, 1[. \tag{1.2}$$

It amounts to say that the finite function f is convex on its effective domain. In the sequence $\text{Conv}(X)$ denotes the set of convex proper functions on X. It is clear that $\text{Conv}(X)$ is a convex cone with apex at the origin.

In (1.2), if the strict inequality holds all $x^1, x^2 \in \text{dom } f$ with $x^1 \neq x^2$ then f is called *strictly convex* function on dom f .

A proper function f is called *strongly convex* on a convex set $C \subset \text{dom } f$ if there exists a number $\rho > 0$ such that

$$f((1-\lambda)x^1 + \lambda x^2) \leq (1-\lambda)f(x^1) + \lambda f(x^2) - (1-\lambda)\lambda\frac{\rho}{2}\|x^1 - x^2\|^2, \tag{1.3}$$

for all $x^1, x^2 \in C$, and for all $\lambda \in]0,1[$. It is equivalent to saying that $f - \frac{\rho}{2}\|.\|^2$ is convex on C. The *modulus of strong convexity* of f on C, denoted by $\rho(f,C)$ or $\rho(f)$ if $C = X$, is given by

$$\rho(f,C) = Sup\{\rho \geq 0 : f - \frac{\rho}{2}\|.\|^2 \text{ is convex on } C\} > 0. \tag{1.4}$$

Clearly, f is convex on C if and only if $\rho(f,C) \geq 0$. One says that f is *strongly convex* on C if $\rho(f,C) > 0$.

Remark 1.1 *f strongly convex $\Longrightarrow$ f strictly convex $\Longrightarrow$ f convex.*

Let f be a convex proper function on X, a vector $y_0 \in Y$ is called a *subgradient* of f at a point $x_0 \in \text{dom}(f)$ if

$$\langle y_0, x - x_0\rangle + f(x_0) \leq f(x) \quad \forall x \in X.$$

The set of all subgradients of f at x_0 is called the *subdifferential* of f at x_0 and is denoted by $\partial f(x_0)$.
Let $\epsilon > 0$, a vector y_0 is called ϵ-subgradient of f at point x_0 if

$$\langle y_0, x - x_0\rangle + f(x_0) \leq f(x) + \epsilon \quad \forall x \in X.$$

Then the set of all $\epsilon - subgradients$ of f at point x_0 is called the the ϵ-subdifferential of f at x_0 and is denoted by $\partial_\epsilon f(x_0)$.

A proper function $f : X \longrightarrow \mathbb{R}$ is called *lower semi-continuous* (l.s.c) at a point $x_0 \in X$ if

$$\lim_{x \to x_0} \inf f(x) \geq f(x_0).$$

or equivalently

$$\forall \epsilon > 0 \ \exists \eta > 0 \text{ such that } \|x - x_0\| \leq \eta \Longrightarrow f(x) \geq f(x_0) - \epsilon$$

Let $\Gamma_0(X)$ be the set of all l.s.c proper convex functions on X.

Definition 1.1 *The* conjugate function *$f^\star$ of $f \in \Gamma_0(X)$ is defined by*

$$f^\star(y) = \sup\{\langle x, y\rangle - f(x) : x \in X\}. \tag{1.5}$$

i.e., $f^\star$ is the pointwise supremum of the family of (continuous) affine functions $y \mapsto \langle x, y\rangle - f(x)$ on Y. .

The function f is polyhedral convex if it is the sum of a pointwise supremum of a finite collection of affine functions and the indicator function of a nonempty polyhedral convex set.

The main properties are summarized in the following proposition that will be needed for further:

Proposition 1.1 *If $f \in \Gamma_0(X)$ then:*

- $f \in \Gamma_0(X) \Longleftrightarrow f^\star \in \Gamma_0(Y)$. *Moreover* $f = f^{\star\star}$,
- $y \in \partial f(x) \Longleftrightarrow f(x) + f^\star(y) = \langle x, y\rangle$ *and* $y \in \partial f(x) \Longleftrightarrow x \in \partial f^\star(y)$,
- $\partial f(x)$ *is a closed convex set,*
- $\partial f(x)$ *is equal to a singleton* $\{y\}$ *iff* f *is differentiable at* x *and* $\nabla f(x) = y$,
- $f(x_0) = min\{f(x), x \in X\} \Longleftrightarrow 0 \in \partial f(x_0)$.

DC functions: A function $f : \Omega \mapsto \mathbb{R}$ defined on a convex set convex $\Omega \subset \mathbb{R}^n$ is called DC on Ω if it can be presented in the form of difference of two convex functions on Ω, i.e.

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

where g and h are convex functions on Ω, $g - h$ is called a DC decomposition of f. We denote by $DC(\Omega)$ be the set of all DC functions on Ω.

DC functions have many important properties, in particular $DC(\Omega)$ is closed with respect to frequently used operations in optimization. Specifically

Proposition 1.2 *(i)* $DC(\Omega)$ *is a vector space spanned by the convex cone* $Conv(\Omega)$: $DC(\Omega) = Conv(\Omega) - Conv(\Omega)$.
(ii) The pointwise supremum of a finite collection of finite DC functions on Ω *is DC on* Ω, *The pointwise infimum of a finite collection of finite DC functions on* Ω *is DC on* Ω,
(iii) Let $f \in DC(\Omega)$, *then* $|f(x)|, f^+(x) = max\{0, f(x)\}$ *and* $f^-(x) = min\{0, f(x)\}$ *belong to* $DC(\Omega)$.

These results have been generalized to convex functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ (Le Thi [1997]).

Remark 1.2 *Given a DC function* f *and a DC decomposition* $f = g - h$, *then for any finite convex function* φ, $f = (g + \varphi) - (h + \varphi)$ *gives another DC decomposition of* f. *Thus, a DC function has infinitely many DC decompositions.*

Classification of nonconvex programs

Due to the preponderance and wealthy properties of DC functions, the passage from Conv(Ω) to the vector space $DC(\Omega)$ permits to expand significantly convex programming to nonconvex programming. The field of optimization problems involving DC functions is very large and covers most of problems encountered in applications.

However, we cannot deal with any nonconvex differentiable/nondifferentiable program. The following classification is well-known:

$$(1) \quad \sup\{f(x) : x \in C\}, \text{where } \ f \text{ and } C \text{ are convex}$$

$$(2) \quad \inf\{g(x) - h(x) : x \in X\}, \text{ where } g \text{ and } h \text{ are convex}$$

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

where g, h, f_1, f_2 and C are convex, these seem to be large enough to contain substantially all nonconvex programs encountered in real life. Problem (1) is a special case of Problem (2) with $g = \chi_C$, the indicator function of C and $h = -f$. Problem (2) can be modified in the form equivalent to (1)

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

While Problem (3) can be transformed into the form (2) by using exact penalty related to the DC constraints $f_1(x) - f_2(x) \leq 0$. Its solution can also be reduced, under certain technical conditions, to that of a sequence of Problems (1). Problem (2) is called *a DC program*. It is a major interest both from practical and theoretical point of view. From the theoretical point of view, we can note that, as remarked above, the class of DC functions is remarkably closed with respect to operations frequently used in optimization. Moreover, there is an elegant duality theory (Pham Dinh [1975, 1976], Toland [1978], Urruty [1986], Le Thi [1994, 1997], Le Thi and Pham Dinh [1997]) which, as with Lagrangian duality in convex optimization, has profound practical implications for numerical methods.

DC programming and DCA (DC Algorithms) were introduced by Pham Dinh Tao (Pham Dinh [1986], Pham Dinh and Bernoussi [1988]) in their preliminary form. In fact, DCA is a generalization of subgradient algorithms which were studied by the same author on convex maximization (Pham Dinh [1975, 1986]). These theoretical and algorithmic tools are extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1994 (see e.g. Le Thi [1994, 1997], Le Thi and Pham Dinh [1997], Pham Dinh and Le Thi [1997, 1998], Le Thi and Pham Dinh [2001], Le Thi et al. [2002], Le Thi and Pham Dinh [2003, 2005] and Le Thi et al. [2011, 2012a], Pham Dinh and Le Thi [2014], Le Thi [web site]) to become now classic and increasingly popular.

DC Duality

In convex analysis, the concept of duality (conjugate function, dual problem, etc.) is a very powerful fundamental concept. For convex programs and in particular linear, a duality theory has been developed over several decades (Rockafellar [1970]). More recently, an important concept of duality in nonconvex analysis has been proposed and developed, first for convex maximization problems, before reaching the DC programming. DC duality introduced by Toland (1978) can be regarded as a natural and logical generalization of earlier works of Pham Dinh Tao (1975) on convex maximization. We will present below the main results on optimal conditions (local and global) and the DC duality. For more details, the reader is referred to the document of Le Thi and Pham Dinh [1997].

A standard DC program is of the form ($g, h \in \Gamma_0(X)$)

$$\alpha := \inf\{f(x) := g(x) - h(x) : x \in X\} \quad (P_{dc})$$

DC duality associates a primal DC program with its dual, which is also a DC program with the same optimal value,

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

by using the fact that every function $\varphi \in \Gamma_0(\mathbb{R}^n)$ is characterized as a pointwise supremum of a collection of affine functions, say

$$\varphi(x) = \sup\{\langle x, y\rangle - \varphi^*(y) : y \in \mathbb{R}^n\},\ \forall x \in \mathbb{R}^n,$$

There is a perfect symmetry between between (P_{dc}) and its dual (D_{dc}): the dual of (D_{dc}) is exactly (P_{dc}).

A standard DC program with a convex constraint C (a nonempty closed convex set in $\mathbb{R}^n$)

$$\alpha = \inf\{f(x) := g(x) - h(x)\ :\ x \in C\} \tag{1.6}$$

can be expressed in the form of (P_{dc}) by adding the indicator function χ_C of C ($\chi_C(x) = 0$ if $x \in C, +\infty$ otherwise) to the function g. The vector space of DC functions, $DC(\mathbb{R}^n) = \Gamma_0(\mathbb{R}^n) - \Gamma_0(\mathbb{R}^n)$, forms a wide class encompassing most real-life objective functions and is closed with respect to usual operations in optimization. DC programming constitutes so an extension of convex programming, sufficiently large to cover most nonconvex programs (Pham Dinh and Le Thi [1997, 1998], Le Thi and Pham Dinh [2003, 2005], Le Thi [web site] and references quoted therein), but not too in order to leverage the powerful arsenal of the latter.

1.2.2 DC optimality and DCA

Polyhedral DC program is a DC program in which at least one of the functions g and h is polyhedral convex. Polyhedral DC programming, which plays a central role in nonconvex optimization and global optimization and is the foundation of DC programming and DCA, has interesting properties (from both a theoretical and an algorithmic point of view) on local optimality conditions and the finiteness of DCA's convegence.

DC programming investigates the structure of $DC(\mathbb{R}^n)$, DC duality and local and global optimality conditions for DC programs. The complexity of DC programs clearly lies in the distinction between local and global solution and, consequently; the lack of verifiable global optimality conditions.

We have developed necessary local optimality conditions for the primal DC program (P_{dc}), by symmetry those relating to dual DC program (D_{dc}) are trivially deduced

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset \tag{1.7}$$

(such a point x^* is called critical point of $g - h$ or (1.7) a generalized Karusk-Kuhn-Tucker (KKT) condition for (P_{dc})), and

$$\emptyset \neq \partial h(x^*) \subset \partial g(x^*). \tag{1.8}$$

The condition (1.8) is also sufficient (for local optimality) in many important classes of DC programs. In particular it is sufficient for the next cases quite often encountered in practice:

– In polyhedral DC programs with h being a polyhedral convex function. In this case, if h is differentiable at a critical point x^*, then x^* is actually a local minimizer for (P_{dc}). Since a convex function is differentiable everywhere except for a set of measure zero, one can say that a critical point x^* is almost always a local minimizer for (P_{dc}).
– In case the function f is locally convex at x^*. Note that, if h is polyhedral convex, then $f = g - h$ is locally convex everywhere h is differentiable.

The transportation of global solutions between (P_{dc}) and (D_{dc}) is expressed by:

$$[\bigcup_{y^* \in \mathcal{D}} \partial g^*(y^*)] \subset \mathcal{P} \ , \ [\bigcup_{x^* \in \mathcal{P}} \partial h(x^*)] \subset \mathcal{D} \tag{1.9}$$

where $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of (P_{dc}) and (D_{dc}) respectively. The first (second) inclusion becomes equality if the function h (resp. g^*) is subdifferentiable on $\mathcal{P}$ (resp. $\mathcal{D}$). They show that solving a DC program implies solving its dual. Note also that, under technical conditions, this transportation also holds for local solutions of (P_{dc}) and (D_{dc}) (Pham Dinh and Le Thi [1997, 1998], Le Thi and Pham Dinh [2003, 2005], Le Thi [web site] and references quoted therein).

Based on local optimality conditions and duality in DC programming, the DCA consists in constructing of two sequences $\{x^k\}$ and $\{y^k\}$ of trial solutions of the primal and dual programs respectively, such that the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing, and $\{x^k\}$ (resp. $\{y^k\}$) converges to a primal feasible solution x^* (resp. a dual feasible solution y^*) satisfying local optimality conditions and

$$x^* \in \partial g^*(y^*), \quad y^* \in \partial h(x^*). \tag{1.10}$$

The sequences $\{x^k\}$ and $\{y^k\}$ are determined in the way that x^{k+1} (resp. y^{k+1}) is a solution to the convex program (P_k) (resp. (D_{k+1})) defined by ($x^0 \in$ dom ∂h being a given initial point and $y^0 \in \partial h(x^0)$ being chosen)

$$(P_k) \qquad \inf\{g(x) - [h(x^k) + \langle x - x^k, y^k\rangle] : x \in \mathbb{R}^n\}, \tag{1.11}$$

$$(D_{k+1}) \qquad \inf\{h^*(y) - [g^*(y^k) + \langle y - y^k, x^{k+1}\rangle] : y \in \mathbb{R}^n\}. \tag{1.12}$$

The DCA has the quite simple interpretation: at the k-th iteration, one replaces in the primal DC program (P_{dc}) the second component h by its affine minorization $h^{(k)}(x) := h(x^k) + \langle x - x^k, y^k\rangle$ defined by a subgradient y^k of h at x^k to give birth to the primal convex program (P_k), the solution of which is nothing but $\partial g^*(y^k)$. Dually, a solution x^{k+1} of (P_k) is then used to define the dual convex program (D_{k+1}) obtained from (D_{dc}) by replacing the second DC component g^* with its affine minorization $(g^*)^{(k)}(y) := g^*(y^k) + \langle y - y^k, x^{k+1}\rangle$ defined by the subgradient x^{k+1} of g^* at y^k : the solution set of $(D_{k+1})$is eaxctly $\partial h(x^{k+1})$. The process is repeated until convergence. DCA performs a double linearization with the

help of the subgradients of h and g^* and the DCA then yields the next scheme: (starting from given $x^0 \in \text{dom } \partial h$)

$$y^k \in \partial h(x^k); \quad x^{k+1} \in \partial g^*(y^k), \ \forall k \geq 0. \tag{1.13}$$

DCA's convergence properties:

DCA is a descent method without linesearch, but with global convergence, which enjoys the following properties: (C and D are two convex sets in $\mathbb{R}^n$, containing the sequences $\{x^k\}$ and $\{y^k\}$ respectively).

i) The sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing and

- $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ iff $y^k \in \partial g(x^k) \cap \partial h(x^k), y^k \in \partial g(x^{k+1}) \cap \partial h(x^{k+1})$ and $[\rho(g, C) + \rho(h, C)]\|x^{k+1} - x^k\| = 0$. Moreover if g or h are strictly convex on C then $x^k = x^{k+1}$.

  In such a case DCA terminates at the k^{th} iteration (finite convergence of DCA)
- $h^*(y^{k+1}) - g^*(y^{k+1}) = h^*(y^k) - g^*(y^k)$ iff $x^{k+1} \in \partial g^*(y^k) \cap \partial h^*(y^k), x^{k+1} \in \partial g^*(y^{k+1}) \cap \partial h^*(y^{k+1})$ and $[\rho(g^*, D) + \rho(h^*, D)]\|y^{k+1} - y^k\| = 0$. Moreover if g^* or h^* are strictly convex on D, then $y^{k+1} = y^k$.

  In such a case DCA terminates at the k^{th} iteration (finite convergence of DCA).

ii) If $\rho(g, C) + \rho(h, C) > 0$ (resp. $\rho(g^*, D) + \rho(h^*, D) > 0$)) then the series $\{\|x^{k+1} - x^k\|^2$ (resp. $\{\|y^{k+1} - y^k\|^2\}$ converges.

iii) If the optimal value α of problem (P_{dc}) is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded then every limit point x^*(resp. y^*) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g - h$ (resp. $h^* - g^*$).

iv) DCA has a linear convergence for general DC programs.

v) DCA has a finite convergence for polyhedral DC programs.

DCA's distinctive feature relies upon the fact that DCA deals with the convex DC components g and h but not with the DC function f itself. DCA is one of the rare algorithms for nonconvex nonsmooth programming. Moreover, a DC function f *has infinitely many DC decompositions which have crucial implications for the qualities* (convergence speed, robustness, efficiency, globality of computed solutions,...) of DCA. For a given DC program, the choice of *optimal* DC decompositions is still open. Of course, this depends strongly on the very specific structure of the problem being considered. In order to tackle the large-scale setting, one tries in practice to choose g and h such that sequences $\{x^k\}$ and $\{y^k\}$ can be easily calculated, *i.e.,* either they are in an explicit form or their computations are inexpensive. Very often in practice, the solution of (D_k) to compute the sequence $\{y^k\}$ is explicit because the calculation of a subgradient of h is explicitly obtained by using the usual rules for calculating subdifferential of convex functions. But the solution of the convex program (P_k), if not explicit, should be achieved by efficient algorithms well-adapted to its special structure, in order to handle the large-scale setting.

1.3 DCA for clustering problem

The goal of clustering problem is partition a data set into groups (or clusters) such that elements of the same group are similar and elements of different groups are dissimilar. DC programming and DCA have been developed in several works for hard clustering and fuzzy clustering (Le Thi et al. [2007a,b,c, 2008a, 2014c], Le Hoai et al. [2013b]).

The first paper in this domain was published in 2007 (Le Thi et al. [2007a]) for the hard clustering problem. A DCA scheme was proposed for the bilevel programming formulation of minimum sum of squares clustering problem (MSSC) where all computations are explicit and require only matrix–vector product. So this schema is very simple and inexpensive. Further, in Le Thi et al. [2007b], the authors introduced three DC programs for the hierarchical clustering problem. It fortunately turn out that their algorithms are explicit and inexpensive. The numerical results on several datasets proved that the proposed DCAs are efficient. DC Programming and DCA also succeeded in solving the block clustering model in Le Hoai et al. [2013b]. Recently in Le Thi et al. [2014c], the authors studied a DCA based algorithm for the mixed integer program formulation of the MSSC problem. Based on an exact penalty technique, the mixed integer program was reformulated as a continuous optimization problem which was then recast to a DC program. In the same work, a Gaussian kernel version of the bilevel formulation of the MSSC problem was also investigated. The considered problem was formulated as a DC program for which an appropriate DC decomposition was developed. This algorithm has been shown to be more efficient than the DCA–MSSC in several numerical test problems.

For fuzzy clustering, the Fuzzy C–Means (FCM) model was considered in Le Thi et al. [2007c]. Three DC formulations and corresponding DCAs schemes for FCM model were introduced. The numerical results demonstrated that the proposed algorithms are more efficient, more robust than related existing algorithms. In addition, DCA based algorithms on fuzzy model also achieved encouraging results in image segmentation (Le Thi et al. [2008a]).

In the next section, we will consider the DCA clustering algorithm, named DCA–MSSC, introduced in Le Thi et al. [2007a]. This algorithm is used in a useful way in two our problems: clustering data stream (Chapter 2) and clustering massive data sets (Chapter 3). Also, the way to construct DC decomposition and DCA–MSSC helped us in the design of DCA based algorithms for clustering with weighted features (Chapter 4).

1.3.1 Minimum Sum of Squares Clustering formulation (MSSC)

An instance of the partitional clustering problem consists of a data set $\mathcal{A} := \{a^1, ..., a^m\}$ of m points in $\mathbb{R}^n$, a measured distance, and an integer k; we have to choose k members x^ℓ $(l = 1, ...k)$ in $\mathbb{R}^n$) as "centroid" and assign each member of $\mathcal{A}$ to its closest centroid. The assignment distance of a point $a \in \mathcal{A}$ is the distance from a to the centroid which is assigned to. The objective function, which is to be minimized, is the sum of assignment distances. If the squared Euclidean distance is used, then the corresponding optimization formulation,

called MSSC problem, is expressed as:

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

with $\|.\|$ denotes the Euclidean norm.

The DCA applied to MSSC problem has been developed in Le Thi et al. [2007a]. Here, to simplify related computations in DCA for solving problem (MSSC) one works on the vector space $\mathbb{R}^{k\times n}$ of $(k \times n)$ real matrices. The variables are then $X \in \mathbb{R}^{k\times n}$ whose i^{th} row X_i is equal to x^i for $i = 1, ..., k$. The Euclidean structure of $\mathbb{R}^{k\times n}$ is defined with the help of the usual scalar product

$$\begin{aligned} \mathbb{R}^{k\times n} \ni X &\longleftrightarrow (X_1, X_2, \ldots, X_k) \in (\mathbb{R}^n)^k,\ X_i \in \mathbb{R}^n, (i = 1, .., k), \\ \langle X, Y \rangle &: = Tr(X^T Y) = \sum_{i=1}^{k} \langle X_i, Y_i \rangle \end{aligned}$$

and its Euclidean norm $\|X\|^2 := \sum_{i=1}^{k} \langle X_i, X_i \rangle = \sum_{i=1}^{k} \|X_i\|^2$ (Tr denotes the trace of a square matrix). We will reformulate the MSSC problem as a DC program in the matrix space $\mathbb{R}^{k\times n}$ and then describe DCA for solving it.

1.3.2 DC program for MSSC problem

According to the property

$$\min_{\ell=1,\ldots,k} \left\| x^\ell - a^i \right\|^2 = \sum_{\ell=1}^{k} \left\| x^\ell - a^i \right\|^2 - \max_{r=1,\ldots,k} \sum_{\ell=1,\ell\neq r} \left\| x^\ell - a^i \right\|^2$$

and the convexity of the functions

$$\sum_{\ell=1}^{k} \left\| x^\ell - a^i \right\|^2 \quad , \max_{r=1,\ldots,k} \sum_{\ell=1,\ell\neq r} \left\| x^\ell - a^i \right\|^2,$$

we can say that (MSSC) is a DC program with the following DC formulation:

$$(MSSC) \Leftrightarrow \min \quad \{F(X) := G(X) - H(X) :\ X \in \mathbb{R}^{k\times n}\}, \tag{1.14}$$

where the DC components G and H are given by

$$G(X) = \sum_{i=1}^{m} \sum_{\ell=1}^{k} G_{i\ell}(X), G_{i\ell}(X) = \frac{1}{2} \left\| X_\ell - a^i \right\|^2 \text{ for } i = 1,\ldots,m, \ell = 1,\ldots,k \tag{1.15}$$

and

$$H(X) = \sum_{i=1}^{m} H_i(X),\ H_i(X) = \max_{j=1,\ldots,k} H_{ij}(X); \tag{1.16}$$

$$H_{ij}(X) := \sum_{\ell=1,\ell\neq j}^{k} \frac{1}{2}\left\|X_\ell - a^i\right\|^2 \text{ for } i = 1,\ldots,m. \tag{1.17}$$

It is interesting to note that the function G is a strictly convex quadratic form. More precisely we have, after simple calculations:

$$G(X) = \frac{m}{2}\|X\|^2 - \langle B, X\rangle + \frac{k}{2}\|A\|^2 \tag{1.18}$$

where $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{k\times n}$ are given by

$$\begin{aligned} A_i &:= \quad a^i \quad \text{for} \quad i = 1,\ldots,m \\ B_\ell &:= \quad a = \sum_{i=1}^{m} a^i \text{ for } \ell = 1,\ldots,k. \end{aligned} \tag{1.19}$$

In the matrix space $\mathbb{R}^{k\times n}$, the DC program (1.14) then is minimizing the difference of the simplest convex quadratic function (1.18) and the nonsmooth convex one (1.16). This nice feature is very convenient for applying DCA, which consists in solving a sequence of approximate convex quadratic programs whose solutions are explicit.

1.3.3 DC Algorithm for solving MSSC problem

According to the description of DCA, determining the DCA scheme applied to (1.14) amounts to computing the two sequences $\{X^{(p)}\}$ and $\{Y^{(p)}\}$ in $\mathbb{R}^{k\times n}$ such that

$$Y^{(p)} \in \partial H(X^{(p)}), X^{(p+1)} \in \partial G^*(Y^{(p)}).$$

We shall present below the computation of $\partial H(X)$ and $\partial G^*(Y)$.

We first express the convex function H_{ij} by

$$\begin{aligned} H_{ij}(X) &= \sum_{\ell=1}^{k} \frac{1}{2}\left\|X_\ell - a^i\right\|^2 - \frac{1}{2}\left\|X_j - a^i\right\|^2 \\ &= \frac{1}{2}\left\|X - A^{[i]}\right\|^2 - \frac{1}{2}\left\|X_j - a^i\right\|^2 \end{aligned} \tag{1.20}$$

where $A^{[i]} \in \mathbb{R}^{k\times n}$ is the matrix whose rows are all equal to a^i.

That gives

$$\nabla H_{ij}(X) = X - A^{[i]} - e_j^{[k]}(X_j - a^i) \tag{1.21}$$

with $\{e_j^{[k]} : j = 1, ..., k\}$ being the canonical basis of $\mathbb{R}^k$.

Hence, according to (1.16) we get the following simpler matrix formula for computing ∂H

$$Y \in \partial H(X) \Leftrightarrow Y = \sum_{i=1}^{m} Y^{[i]} \text{ with } Y^{[i]} \in \partial H_i(X) \text{ for } i = 1, ..., k, \tag{1.22}$$

where $Y^{[i]}$ is a convex combination of $\{\nabla H_{ij}(X) : j \in K_i(X)\}$, i.e.,

$$Y^{[i]} = \sum_{j \in K_i(X)} \lambda_j^{[i]} \nabla H_{ij}(X) \text{ with } \lambda_j^{[i]} \geq 0 \text{ for } j \in K_i(X) \text{ and } \sum_{j \in K_i(X)} \lambda_j^{[i]} = 1, \tag{1.23}$$

with $K_i(X) := \{j = 1, \ldots, k : H_{ij}(X) = H_i(X)\}$.

In particular we can take for $i = 1, ..., m$

$$Y^{[i]} = X - A^{[i]} - e_{j(i)}^{[k]}(X_{j(i)} - a^i) \text{ for some } j(i) \in K_i(X), \tag{1.24}$$

and the corresponding $Y \in \partial H(X)$ defined by

$$Y = mX - B - \sum_{i=1}^{m} e_{j(i)}^{[k]}(X_{j(i)} - a^i). \tag{1.25}$$

Since the function G is strictly convex quadratic, its conjugate G^* is differentiable and we have from (1.18)

$$X = \nabla G^*(Y) \Longleftrightarrow Y = \nabla G(X) = mX - B,$$

or again

$$X = \frac{1}{m}(B + Y). \tag{1.26}$$

We are now in a position to describe the DCA for solving problem (MSSC) via the DC decomposition (1.14).

Algorithm DCA-MSSC:

Initialization: Let $\epsilon > 0$ be given, $X^{(0)}$ be an initial point in $\mathbb{R}^{k \times n}$, set $p := 0$;

Repeat

Calculate $Y^{(p)} \in \partial H(X^{(p)})$ by using (1.25)

$$Y^{(p)} = mX^{(p)} - B - \sum_{i=1}^{m} e_{j(i)}^{[k]}(X_{j(i)}^{(p)} - a^i)$$

and calculate $X^{(p+1)}$ according to (1.26)

$$X^{(p+1)} := \frac{1}{m}(B + Y^p). \tag{1.27}$$

Set $p + 1 \leftarrow p$

Until $\| X^{(p+1)} - X^{(p)} \| \leq \epsilon(\| X^{(p)} \| + 1)$ or $\left|F(X^{(p+1)}) - F(X^{(p)})\right| \leq \epsilon(\left|F(X^{(p)})\right| + 1)$.

1.4 Conclusion

In this chapter, we introduce in the first part, a brief presentation of DC programming and DCA. The fundamentals of DC analysis, such as DC functions, DC duality, global and local optimality in DC optimization are also presented. In the second part, we first review DCA based algorithms for clustering problems which have been efficiently applied for both hard clustering and fuzzy clustering, and then focus on the DCA–MSSC algorithm.

For a comprehensive study of DC programming and DCA, refer to Le Thi [1994], Le Thi [1997], Le Thi et al. [2002], Pham Dinh and Le Thi [1997, 1998, 2014] and the reference therein. The solution of a nonconvex problem by DC programming and DCA should have two tasks: looking for an appropriate DC decomposition and looking for a good starting point. In the next three chapters we will investigate DC programming and DCA to deal with hard clustering with data stream / massive data, and hard / fuzzy clustering with weighted feature measures.

Chapter 2

Clustering data streams[1]

In this chapter, we focus on two models to deal with data streams: sub–windows and sliding windows. In the first model, we are interested by an exploratory analysis for clustering data stream based on a sub–windows approach. Our approach consists to separate the data on different sub–windows and then apply DCA–MSSC clustering algorithm on each of them. Two clustering strategies are investigated: global clustering and independent local clustering. Global clustering is performed on the whole data set and the independent local clustering is performed in each windows with random center initialization. We study the consistency of the results. In the second model, we focus on the initialization strategy for the clustering algorithm performing on the new windows. Based on DCA–MSSC algorithm, we propose a DCA–Stream, for clustering data stream over the sliding windows model.

2.1 Introduction

Data stream is one emerging topic of data mining, it concerns many applications involving large and temporal data sets such as telephone records data, banking data, multimedia data,.... In stream models, the data is massive and evolve continuously, it can be read only once or a small number of times and it is impossible to load the entire data set into memory. Traditional data mining techniques are not useful for these applications, and it requires to develop new approaches to deal with data streams. A data stream is an ordered sequence of points $(x_1, x_2, ..., x_n)$ that must be accessed in one order and can be read only once or a small number of times. Each reading of the sequence is called a *linear scan* or a *pass* (Guha

1. This chapter is published under the titles:
[1]. Minh Thuy Ta, Hoai An Le Thi, Lydia Boudjeloud–Assala. *Clustering Data Stream by a Sub–window Approach Using DCA*. Machine Learning and Data Mining in Pattern Recognition (MLDM2012), Lecture Notes in Computer Science Volume 7376, ISBN 978–3–642–31536–7, pp. 279–292 (2012).
[2]. Minh Thuy Ta, Hoai An Le Thi, Lydia Boudjeloud–Assala. *Clustering data streams over sliding windows by DCA*. Advanced Computational Methods for Knowledge Engineering, Studies in Computational Intelligence. Volume 479, Springer, ISSN: 1860–949X (Print) 1860–9503 (Online), pp. 65–75 (2013).

et al. [2003]). The volume of such data set is so large that it may be impossible to store the data on a disk, or even when the data can be stored, it is impossible to deal with the whole data set. In addition, the data patterns may evolve continuously and can be depend on the time or on the events. Data streams differ from the conventional stored relation model in several ways (Babcock et al. [2002]):

1. The data elements in the stream arrive online.
2. The system has no control over the order in which data elements arrive to be processed.
3. Data streams are potentially unbounded in size.
4. Once an element from a data stream has been processed it is either discarded or archived.

Therefore, traditional techniques are not useful to deal with such data, new challenges are posed for data mining algorithms. For mining on stream data, one crucial and efficient strategy is analysis of packet data. The data set is divided into more significant sub-periods, called sub-windows, with the aim of detect data patterns evolution or emergence, which would not have been revealed by a global analysis in whole time period. A mining method is then applied on each sub-window. This strategy allows reduce costs (physical memory, CPU time, etc.) as we focus on the analysis of a portion of the data. The sub-window approaches are studied in Silva [2007] where some strategies of clustering data stream are studied: global clustering, independent local clustering, dependent local clustering. There, the data set is divided into sub-windows on each of which the K-Means based algorithm is then applied, and web usage data sets have been tested.

In the first part of this chapter, we are interested in an exploratory analysis of the independent local clustering strategy for clustering data stream based on a sub-window approach. Obviously, such an analysis depends on the performance of clustering algorithm to be used. Our approach consists of separating the data on different sub-windows and then apply DCA–MSSC clustering algorithm on each sub-window. Our aims are study:
– the efficiency of the independent local clustering strategy, say the adequation of this local clustering and real clusters,
– the adequation between the independent local clustering and global clustering strategies based on the same DCA clustering algorithm.
In the second part, we consider another model of data stream: sliding window. In this model, data elements arrive continually, and we only considered the most recent elements. The new stream will differ slightly from the current ones. It is important to study the initialization strategy of clustering algorithm performing on new windows. In this section, based on DCA–MSSC algorithm, we study an initialization strategy for clustering on the new window and design a streaming clustering algorithm for data streams over sliding windows.

Comparative experiments with clustering data stream using K-Means, a standard clustering method, on different data sets are reported. We also study the effect of the different strategies toward the speed of evolution of data, as well as the performance of DCA based approaches versus the standard K-Means algorithm.

2.2 State of the art

In the literature, the clustering data streams can be divided into three approaches: the divide–and–conquer approach, the two stages: online and offline approach and the density approach.

2.2.1 Divide–and–conquer-based methods

One of the first researches in this area is due to Guha et al. [2000]. The authors propose an algorithm, which uses divide–and–conquer strategy for clustering data stream. The core of algorithm is to use a randomized constant–factor approximation algorithm of Charikar and Guha [1999] to solve the k–Median clustering problem. In the first level, m points sampled from the stream are taken as subset. The constant–factor approximation algorithm is used to partition these m points into $2k$ median points. In the next level, m medians are clustered to define $2k$ medians (with assumption that m is a multiple of $2k$). Repeatedly, when level–i has m medians, these medians are clustered to $2k$ clusters of level–$(i+1)$. When all data points are considered, all intermediate medians are partitioned into k final clusters.

In 2002, O'Callaghan et al. propose a STREAM algorithm (O'Callaghan et al. [2002]). STREAM algorithm replaces the k–Median clustering problem by a variant called facility location problem. Then, the authors introduce a LSEARCH algorithm to solve this problem. The original data stream arrives in chunks $X_1, X_2, ..., X_n$, where each of chunk contains at most m data points. Then, STREAM executes clustering in each chunk X_i by LSEARCH algorithm and assigns to each resulting median, a weight equal to the number of member points belonging to it. When the number of medians exceeds m, a second level of clustering is applied to these sets. Finally, when the whole data points arrives, LSEARCH algorithm is performed on the remaining weighted medians. The clustering method LSEARCH is simple, fast, constant–factor approximation of k–Median subroutine. The authors also prove that the clustering performance of STREAM algorithm is better than BIRCH algorithm (Zhang et al. [1996]) as well as STREAM k–Means (O'Callaghan et al. [2002]). One of the disadvantage of this algorithm is that is not particularly sensitive to evolution in the underlying data stream. In some particular cases, the pattern is evolved and changed significantly.

Another approach is mining the data on sub–sets, which is proposed by Silva [2007]. The data set is divided into more significant sub–sets, with the aim of detecting data patterns evolution, which would not have been revealed by a global analysis in whole data set. A mining method is then applied on each sub–sets. The author also studies the data changes: merge, split, new cluster appears or disappears. The author presents 4 strategies: global clustering, independent local clustering, dependent local clustering and previous local clustering.

2.2.2 On/offline-based methods

The second direction research is two stages online and offline approach. Aggarwal et al. (Aggarwal et al. [2003]) developed a two–stages clustering algorithm, CluStream, to process the clustering data stream problem. This algorithm consists of an online micro-clustering component and an offline macro-clustering. Firstly, the micro-clustering creates q initial micro-clusters by k–Means algorithm. This phase will gather the statistics information from incoming data. At any moment, the algorithm takes q micro–clusters, where q is pre–defined such that it is larger than the natural number of clusters and smaller the number of data points arrive. When a new point arrives, the distances from this point to the micro–clusters centers are calculated, and this point is assigned to the nearest micro–cluster if the distance is smaller than maximum boundary. If not, it creates one new micro-cluster, and one old micro–cluster is deleted or two old micro–clusters are merged. These clusters are called micro–clusters, which are stored at snapshots in time which follow a pyramidal pattern. This frame is a good technique, because it helps to balance between the storage requirements and the ability to recall summary information. The offline component performs clustering on these summary statistics information to provide the understanding of clusters in data stream. These clusters are denoted by macro–clusters. In this phase, the algorithm uses the information provided by the user preferences such as time–horizon h and number of clusters k. The macro component can perform at any given time. By this way, algorithm can cluster the data of the specified previous time. However, CluStream is efficient with sphere–shaped, but the evolving data clusters are arbitrarily shaped. In addition, it is inefficient for handling the outlier and the number of clusters must be pre-defined.

HPStream algorithm proposed by Aggarwal et al. [2004], works with high dimensional data streams. This algorithm includes a fading cluster structure and a *projected clustering* technique. This technique continuously refines the subset of projected dimensions, and determines clusters on them. With fading structure, the algorithm can easily manage the historical information as well as current data. A fading cluster structure consists of $(2 \cdot d + 1)$ tuple, which is similar to a micro–cluster in CluStream algorithm. It maintains condensed representations of the clusters over time. In experiments, the authors found that the quality of HPStream algorithm is more efficient than the full dimensional CluStream algorithm. But we must choice some input parameters: the average projected dimensionality ℓ, the radius threshold and the decay rate λ.

2.2.3 Density-based methods

Another direction research is based on density methods. The first idea is presented by Cao et al. [2006]. The advantage of density–based approach is the ability to find clusters with any shape of data and it does not need to know the number of clusters. This algorithm combines micro–clustering with a density–estimation process for clustering. Firstly, based on the weight of data points, the algorithm detects the dense regions, that called core micro-clusters. Then, algorithm defines potential core micro–clusters (denoted by p–micro-clusters)

and outlier micro-clusters (denoted by o–micro–clusters). Note that, a micro–cluster in this algorithm has a structure that is similar to micro–cluster in CluStream (once includes weight, radius information and once includes the sum of data values, the sum of square values), but the different key is that: the number of micro–cluster in CluStream is fixed, but that in DenStream is not. By the manage of two micro–clusters: p–micro–cluster and o–micro–cluster, DenStream can both detect the number of clusters and manage the evolve of data. Finally, the final results of clustering is obtained by applying a DBSCAN (Ester et al. [1996]) variant algorithm on the set of potential micro–cluster. However, the performance of the algorithm still depends on the choice of parameters input.

Another density–based method is D–stream (Tu and Chen [2009]), which is a density based grid clustering algorithm. The algorithm uses an online component which maps incoming data into a grid and an offline component which calculates density of grid and then discards the data. Finally, the offline component clusters the grids based on their density. This algorithm includes three advantages. Firstly, it uses a density decaying technique to capture the evolution of data stream. Secondly, it represents the data by discrete fine grids, thus, the raw data can be discarded. D–Stream uses a novel concept - the *attraction* of grids, that characterizes the positional information of the data in each grid. By using both attraction information and density information, the algorithm can improve the quality of results. Nevertheless, for high dimensional data, it is not scalable because the number of grids increases exponentially with the number of dimensions.

By expanding the definitions of cluster feature structure from their research (Aggarwal et al. [2003]), in 2008, Aggarwal and Yu [2008] propose the UMicro algorithm for clustering uncertain data streams. In many applications, the estimated error of data is available, which can exists in the form of standard deviation or probability density functions. Such information can be used to improve quality of the results. UMicro algorithm put the uncertainty of data points into the structures of micro-clusters. The uncertainty information is quite important because of the varying relative behavior of the attributes and its effect on the calculate the distance for the assignments.

Continuing research on uncertain data stream, Zhang et al. [2009] introduce the PMicro algorithm. This algorithm proposes a novel solution for clustering on uncertain data streams based on the probability model. In UMicro, each tuple t has the values of some attributes that are uncertain, but in the point probability model, each tuple is affiliated with a probability $p(t)$. $p(t)$ is occurring probability of this tuple. This algorithm introduces Probability Cluster Feature (PCF) to handle uncertain data streams, which takes into account the distance between tuples, the uncertainty information of attributes, also handle the existence probability of tuples. In experiments, the authors found that the PMicro is better than UMicro in both quality and execution time.

2.2.4 Sliding windows methods

In the literature, there are several papers in the clustering data streams occuring over sliding windows. One of the first researches belongs to Babcock et al. [2003]. Their study is an extension Guha et al. [2000]'s to the sliding windows model. In their work, the authors use the Exponential Histogram (EH) data structure to group data elements into buckets that cover adjacent time intervals and maintain statistic information. Each bucket B_i maintains both timestamp t_i of the most recent data elements and statistics information in this bucket. When a new point arrives, algorithm checks and adds it into the newest bucket, or creates a new bucket. If the oldest bucket reaches $N + 1$, it means that this bucket expired. The algorithm then recalculates the summarized information.

One other related work is given by Beringer and Hullermeier [2006]. They develop an online version of k–Means algorithm for clustering data streams. The important improvement of this algorithm is the use of DFT (Discrete Fourier Transform) to improve the efficiency of calculating the distance function. In this method, instead of calculating on the w-dimensional space, it only calculates on the u-dimensional space ($u \ll w$). This approximation has two advantages: it captures only important properties, the noise properties will be filtered; and the calculation of the distance function becomes more efficient by reducing the number of dimensions. Besides, the authors introduce an adaptation with the optimal number of clusters problem. By using one criteria value, the algorithm compares the values at $(k, k + 1, k - 1)$. The maximum value corresponds to the optimal number. However, the number of clusters may has significant changes in the evolving data stream. It does not simply change by ± 1.

In 2008, Zhou et al. [2008] present the SWClustering algorithm. The authors introduce a novel data structure, the Exponential Histogram of Cluster Features (EHCF) that includes the exponential histogram and temporal cluster features. The first one maintains the in–cluster evolution, and the second one represents the change of the cluster distribution. This approach has several advantages such as: (1) the quality of the clusters is improved because the EHCF captures the distribution of recent records precisely; (2) The mechanism employed to adaptively maintain the in–cluster synopsis can track the evolution of cluster better than previous methods, while consuming much less memory; (3) the EHCF provides a flexible framework for analyzing the evolution of cluster and tracking a specific cluster efficiently without interfering with other clusters, thus reducing the consumption of computing resources for data stream clustering (Zhou et al. [2008]). Nevertheless, this algorithm seems only suitable for clustering with spherical shapes of data and it can not guarantee the global optimality of computed solutions.

Based on strategy two stages, Dang et al. proposed the SWEM algorithm (clustering data streams in a time based Sliding Window with Expectation Maximization technique) (Dang et al. [2009a,b]). The authors develop a method to adaptively split and merge micro components. This algorithm uses EM technique to clustering, it gives some advantage properties, such as filter noise or handle missing data.

Tang et al. [2008] present the MovStream algorithm. This research introduces some basic movements of evolution of an individual cluster. Algorithm then will monitor the clusters in evolving data streams based on the measurement of these changes. In the first step, MovStream uses k–Mean algorithm to initialize clusters. When a new point arrives, it will be added in a *ClusterTable* - a dynamic list data structure. Then, algorithm will detect the cluster's evolving by movement events. It focuses on 5 significant movements: move, split, merge, stretch and delete. The experimental results show that MovStream algorithm is well-known than CluStream algorithm. However, this algorithm is not good with arbitrary shape and it depends on the choice of parameter set: $\delta_i, i = 1 \ldots 5$.

Another approach is based on density methods. Ren et al. [2009] introduce the SDStream algorithm, which combines the ideas of both SWClustering algorithm (Zhou et al. [2008]) and DenStream algorithm (Cao et al. [2006]). Therefore, it not only takes the advantages of density–based algorithm discover clusters of arbitrary size and shape or does not need to know the number of clusters, but also it can satisfy the requirements of clustering over sliding windows by using the *Exponential Histogram of Cluster Feature* (EHCF) structure. This algorithm creates the core micro–clusters, potential micro–clusters and outlier micro–clusters as well as DenStream algorithm, but which are stored in the form of EHCF. In EHCF only the most recent N records are considered. When a data point arrives, it will be inserted into the nearest micro cluster. It belongs to potential micro–cluster if the new radius of potential micro–cluster is small or equal to ϵ (radius threshold), else it belongs to outlier micro–cluster if the new radius of outlier micro–cluster is less than ϵ. Otherwise, the algorithm creates a new micro cluster. Consequently, if two clusters are *density-reachable*, they will be merged. One micro–cluster having the value of time in EHCF greater than N (window size), it is outdated micro-cluster. This cluster will be deleted. Finally, this algorithm executes a modified DBSCAN algorithm (Ester et al. [1996]) on the potential micro-clusters.

In the field of clustering uncertain data streams, Jin et al. [2013] propose the cluUS algorithm to handle the sliding windows model. The authors study a novel structure - Uncertain Feature Histogram (UFH) to further summarize the disjoint UF structures at different intervals. There are five kinds of UFH considered: dead, partly dead, grown-up, on-growing, and fresh. When system requires the results of clustering, a k–Means variant is used. In comparison with the enhanced CluStream-WS and UMicro-SW (Jin et al. [2013]), cluUS algorithm only consumes a small memory space, while the accuracy is still close to the optimal results. However, this algorithm depends on many input parameters.

In the following, we present our strategies to deal with the data streams. In the one hand, we are interested by an exploratory analysis based on a sub–windows approach. On the other hand, we focus on the sliding windows based approach.

2.3 Clustering data streams based on sub-windows

For mining on stream data, one crucial strategy is analysis by packets. The data set is divided into more significant sub–periods, called sub–windows, with the aim of detect data patterns evolution or emergence, which would not have been revealed by a global analysis in whole time period. A mining method is then applied on each sub–window. This strategy allows reduce costs (physical memory, CPU time, etc.).

The sub–window approaches that are studied in Silva [2007] where the data set is divided into sub–windows on which the k–Means based algorithm is then applied. The model is presented in the figure 2.1.

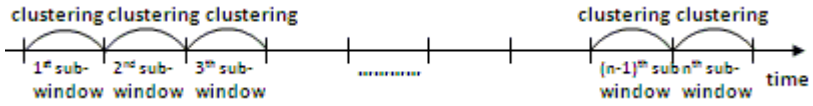


Figure 2.1: Analysis on significant sub–windows (ref. Silva [2007])

In this section, we are interested in an exploratory analysis of the independent local clustering strategy for clustering data stream based on a sub-window approach. Obviously, such an analysis depends on the performance of clustering algorithm to be used. Our approach consists of separating the data on different sub–windows and then apply **DCA–MSSC** clustering algorithm on each sub–window. Our aims are studying:

- The efficiency of the independent local clustering strategy, comparing with local clustering and real clusters,
- The adequation between the independent local clustering and global clustering strategies based on the same DCA clustering algorithm.

2.3.1 DCA–MSSC algorithm based on sub–windows

Two sub-window approaches are proposed for data stream in the literature: logical window and sliding window. In this section, we adopt the logical window approach in which data set is subdivided into separated sub–windows having the same clusters number, and the number of elements in each window is given.

In our study, we are concerned with an analysis of the two clustering strategies - global clustering and independent local clustering.

1. **Global clustering**: the clustering is performed on the whole data set.
2. **Independent local clustering**: the data set is subdivided into separated sub–windows, and the clustering is performed independently on each sub–window from a random starting point.

According to these strategies, two analyses are considered in order to analyze the efficiency of the independent local clustering approach. In our analyses we used DCA–MSSC algorithm, an efficient clustering algorithm is presented in the first chapter, for clustering the data. We also applied the k–Means classical algorithm to prove the efficiency of our algorithm.

2.3.1.1 Local independent clustering analysis

For this study, we divide data set into n sub–windows by ratio of the elements in each group, and then perform clustering on each sub–window from a random starting point. By comparison with the real clustering, we get the percent of the bad placed objects (PBPO). In experiments, we applied the DCA–MSSC algorithm in compare with k–Means classical.

The schema given in figure 2.2 describes this approach.

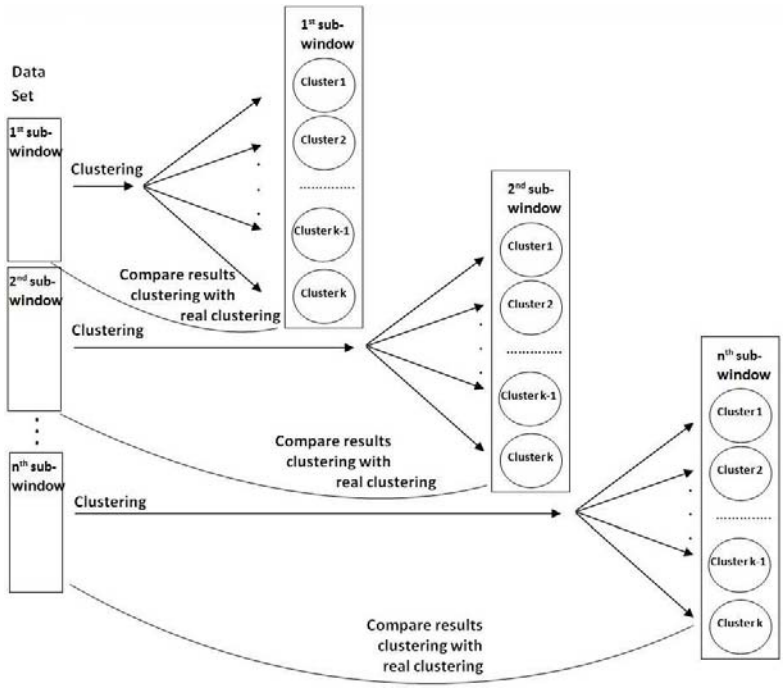


Figure 2.2: Analysis of the efficiency of independent local clustering (ref. Silva [2007])

2.3.1.2 Local independent clustering vs global clustering analysis

In the first step, we apply a clustering algorithm on the whole data set (global clustering). We get the results of global clustering with k groups and the ratio number of elements in each group. After that, we divide data set into n sub–windows by this ratio number of elements. Then, we perform a clustering algorithm on each sub–window (independent local clustering)

and compare the results with the global clustering (see the schema given in figure 2.3). We also use both DCA and k–Means clustering algorithms in our analysis.

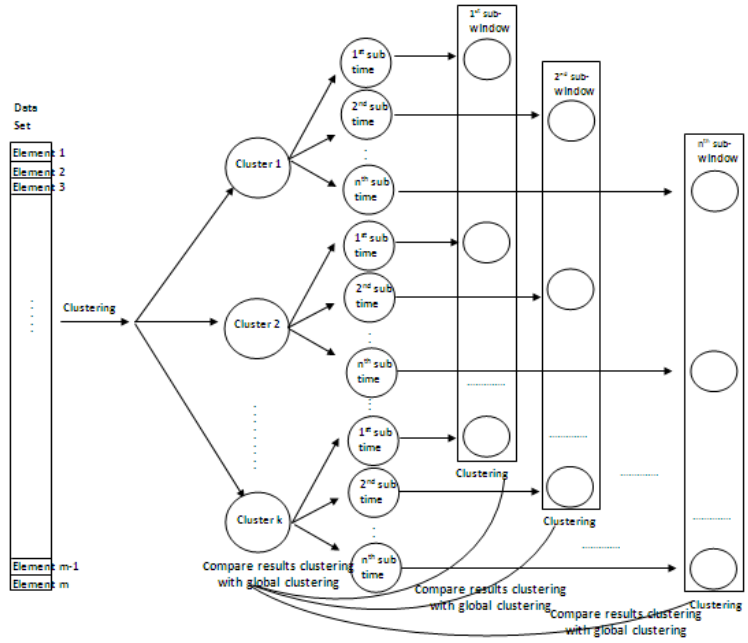


Figure 2.3: Analysis of the adequation between independent local clustering and global clustering (ref. Silva [2007])

2.3.2 Numerical experiments

We consider 6 real-world data sets: *Iris, Ionosphere, Pima, Wisconsin Breast, Waveform, Magic* from Newman and Merz. The global clustering and local clustering strategies based on DCA and k–Means algorithms have been implemented in the Visual C++ 2008, and run on a PC Intel i5 CPU650, 3.2 GHz of 4GB RAM. The information about data sets is summarized in table 2.1.

Table 2.1: Data Sets

Dataset	Points	Dimension	No. clusters	No. windows
Iris	150	4	3	2
Ionosphere	351	34	2	3
Pima	768	8	2	5
Wisconsin Breast	683	34	2	5
Waveform	5000	21	3	5
Magic	19020	10	2	7

2.3.2.1 First experiments

First, we divide data set into n sub windows by ratio of the elements in each group. For example, the *Ionosphere* data set has 351 elements partitioned in 2 groups: the first group has 125 elements; the second group has 226 elements. If we want to obtain 3 windows, we chose randomly 41 (125/3) elements in the first group and chose randomly 75 (226/3) elements in the second group to compose the window 1. We construct windows 2 and 3 using the similar process.

After, we perform the clustering algorithm on each sub–window. We get the percent of the bad placed objects (PBPO) in comparing with real clusters.

The table 2.2 presents the comparative results of independent local clustering strategy based on DCA–MSSC algorithm and k–Means algorithm. Here "Min PBPO" (resp. "Max PBPO") denotes the minimum (resp. the maximum) of PBPO over 10 executions of DCA–MSSC and/or k–Means algorithm on each sub–window. For studying the efficiency of clustering algorithm we also report the results of global clustering (clustering on the whole data set) in comparing with real clusters. The starting points of DCA–MSSC and k–Means are the same and they are randomly chosen from given objects.

From the results reported in Table 2.2 we observe that:

i) The independent local clustering strategy is as efficient as the global clustering strategy, in comparing with the real clusters. In other words, clustering by logical sub–windows approach can be efficiently used for clustering data stream and/or clustering on mass of data.

ii) Clustering of stream data based on DCA–MSSC is in general better than the one based on k–Means.

2.3.2.2 Second experiments

As in the first experiment, we perform clustering with two algorithms: DCA–MSSC and k–Means. We are interested in the adequation between local and global clustering strategies. We first perform the global clustering algorithm on the whole data set. After that we divide the data set on sub–windows with the same procedure used in the first experiment. In each sub–window, we run the clustering algorithm and then compare the obtained results with clusters furnished by the global clustering strategy.

Table 2.3 presents the comparative results of independent local clustering strategy based on DCA–MSSC and k–Means algorithm. Here "Min Error" (resp. "Max Error") denotes the minimum (resp. the maximum) of "errors" of the local clustering in comparing with clusters furnished by the global clustering over 10 executions of DCA–MSSC and/or k–Means algorithm on each sub–window.

From the results reported in Table 2.3 we see that:

Table 2.2: Comparative results of global and independent local clustering strategies using DCA and k–Means

Iris	k–Means Min PBPO	DCA Min PBPO	k–Means Max PBPO	DCA Max PBPO	k–Means Avg PBPO	DCA Avg PBPO	k–Means Avg Time	DCA Avg Time
Window1	10.6667%	**10.0000%**	47.000%	**46.5200%**	18.4000%	**18.2000%**	0.0020	0.0016
Window2	12.0000%	**4.0000%**	52.0000%	**45.3333%**	15.0000%	15.0000%	0.0026	0.0027
Dataset	10.6667%	**8.0000%**	54.6670%	**49.3333%**	18.4000%	**16.2000%**	0.0025	0.0038
Wisconsin Breast	k–Means Min PBPO	DCA Min PBPO	k–Means Max PBPO	DCA Max PBPO	k–Means Avg PBPO	DCA Avg PBPO	k–Means Avg Time	DCA Avg Time
Window1	2.2059%	2.2059%	2.2059%	2.2059%	2.2059%	2.2059%	0.0066	0.0070
Window2	1.4716%	1.4716%	1.4716%	1.4716%	1.4716%	1.4716%	0.0050	0.0058
Window3	5.1471%	5.1471%	5.8824%	5.8824%	5.5147%	**5.3677%**	0.0052	0.0062
Window4	5.1471%	**3.6765%**	6.6177%	6.6177%	5.8088%	**5.0735%**	0.0050	0.0062
Window5	4.3166%	4.3166%	5.0360%	**4.3166%**	4.4604%	**4.3166%**	0.0054	0.0062
Dataset	3,95315%	3.95315%	4,09956%	4.09956%	4.07028%	**4.0410%**	0.0091	0.0137
Ionosphere	k–Means Min PBPO	DCA Min PBPO	k–Means Max PBPO	DCA Max PBPO	k–Means Avg PBPO	DCA Avg PBPO	k–Means Avg Time	DCA Avg Time
Window1	25.8621%	25.8621%	34.4828%	**32.7586%**	28.7931%	**27.8448%**	0.0100	0.0130
Window2	32.7586%	32.7586%	34.4828%	**33.6207%**	33.6207%	**33.2759%**	0.0110	0.0120
Window3	29.4118%	29.4118%	35.2941%	35.2941%	**33.1092%**	33.4454%	0.0110	0.0120
Dataset	29.0598%	29.0598%	35.3276%	**29.3447%**	29.8006%	**29.1738%**	0.0140	0.0188
Pima	k–Means Min PBPO	DCA Min PBPO	k–Means Max PBPO	DCA Max PBPO	k–Means Avg PBPO	DCA Avg PBPO	k–Means Avg Time	DCA Avg Time
Window1	32.6797%	32.6797%	33.3333%	**32.6797%**	32.8976%	**32.6797%**	0.0103	0.0150
Window2	32.6797%	32.6797%	32.6797%	32.6797%	32.6797%	32.6797%	0.0107	0.1533
Window3	35.9477%	**34.6405%**	35.9477%	35.9477%	35.9477%	**35.5120%**	0.0053	0.0100
Window4	32.0261%	32.0261%	34.6405%	**33.3333%**	33.3333%	**32.8976%**	0.0053	0.0100
Window5	32.6923%	32.6923%	32.6923%	32.6923%	32.6923%	32.6923%	0.0053	0.0153
Dataset	33.9844%	**33.7240%**	34.0175%	**33.9844%**	34.0144%	**33.9583%**	0,0116	0.0170
Waveform	k–Means Min PBPO	DCA Min PBPO	k–Means Max PBPO	DCA Max PBPO	k–Means Avg PBPO	DCA Avg PBPO	k–Means Avg Time	DCA Avg Time
Window1	50.2503%	**46.3463%**	50.6507%	**48.8488%**	50.4171%	**47.2472%**	0.0830	0.1093
Window2	48.4484%	**47.1471%**	**48.9489%**	59.0591%	**48.6820%**	51.1512%	0.0833	0.1143
Window3	48.8488%	**22.8228%**	51.3514%	**50.6507%**	50.3837%	**39.9399%**	0.0880	0.1197
Window4	48.9489%	**48.3483%**	60.4605%	**50.5910%**	53.4535%	**49.9699%**	0.0987	0.1147
Window5	**46.9124%**	47.1116%	53.4535%	**50.8964%**	52.7888%	**49.8165%**	0.0830	0.1247
Dataset	49.8200%	**47.5400%**	60.7000%	**52.5800%**	51.0240%	**47.7160%**	0.1574	0.2535
Magic	k–Means Min PBPO	DCA Min PBPO	k–Means Max PBPO	DCA Max PBPO	k–Means Avg PBPO	DCA Avg PBPO	k–Means Avg Time	DCA Avg Time
Window1	**34.7570%**	35.0147%	35.2356%	**35.0147%**	35.0368%	**35.0147%**	0.2184	0.3368
Window2	35.1988%	**35.1252%**	35.6406%	**35.6143%**	35.4934%	**35.4579%**	0.2090	0.2994
Window3	**35.6038%**	35.7511%	35.8247%	**35.7511%**	35.7511%	35.7511%	0.2154	0.3714
Window4	34.3888%	**34.3520%**	34.4624%	**34.4256%**	34.4379%	**34.3765%**	0.2150	0.3216
Window5	36.0457%	36.0457%	36.0457%	36.0457%	36.0457%	36.0457%	0.2030	0.3278
Window6	34.6834%	**34.6097%**	34.6834%	34.6834%	34.6834%	**34.6588%**	0.1998	0.3146
Window7	35.6828%	**35.3891%**	35.6828%	**35.3891%**	35.6828%	**35.3891%**	0.2150	0.3964
Dataset	35,0894%	**35.0263%**	**35.0894%**	35.1052%	35.0894%	**35.0820%**	0.4714	0.6118

i) With an appropriate starting point, the independent local clustering strategy using DCA is very efficient in comparing with the global clustering. The Min Error varies from 0% to 2.5% over all windows.

ii) Clustering of stream data based on DCA is in general better than the one based on

Table 2.3: the adequation of global clustering and independent local clustering based on DCA–MSSC and k–Means

Iris	k–Means Min Error	DCA Min Error	k–Means Max Error	DCA Max Error	k–Means Avg Error	DCA Avg Error	k–Means Avg Time	DCA Avg Time
Window1	1.3333%	**0.0000%**	46.3333%	**17.3333%**	14.9333%	**7.5644%**	0.0015	0.0024
Window2	**0.0000%**	1.3333%	45.0000%	**42.6667%**	10.0000%	10.0000%	0.0008	0.0047
Wisconsin Breast	k–Means Min Error	DCA Min Error	k–Means Max Error	DCA Max Error	k–Means Avg Error	DCA Avg Error	k–Means Avg Time	DCA Avg Time
Window1	0.0000%	0.0000%	0.7353%	**0.0000%**	0.4412%	**0.0000%**	0.0094	0.0062
Window2	0.0000%	0.0000%	**0.0000%**	0.7353%	**0.0000%**	0.3676%	0.0062	0.0124
Window3	0.0000%	0.0000%	**0.0000%**	0.7353%	**0.0000%**	0.2206%	0.0062	0.0154
Window4	0.0000%	0.0000%	2.9412%	**0.0000%**	2.0588%	**0.0000%**	0.0062	0.0216
Window5	2.1583%	**0.0000%**	**2.1583%**	2.8777%	2.1583%	**0.8633%**	0.0064	0.0278
Ionosphere	k–Means Min Error	DCA Min Error	k–Means Max Error	DCA Max Error	k–Means Avg Error	DCA Avg Error	k–Means Avg Time	DCA Avg Time
Window1	2.5862%	**0.8621%**	47.4138%	**45.6900%**	16.3793%	**10.0860%**	0.0101	0.0118
Window2	0.0000%	0.0000%	48.2760%	**45.6900%**	9.2241%	**7.5862%**	0.0101	0.0237
Window3	1.6807%	**0.8403%**	**44.5380%**	47.0590%	**10.1681%**	12.1850%	0.0107	0.0353
Pima	k–Means Min Error	DCA Min Error	k–Means Max Error	DCA Max Error	k–Means Avg Error	DCA Avg Error	k–Means Avg Time	DCA Avg Time
Window1	1.9068%	**1.3072%**	12.4180%	**2.6144%**	5.4902%	**1.6994%**	0.0081	0.0082
Window2	0.6536%	0.6536%	3.9216%	**3.2680%**	1.6340%	**1.1765%**	0.0063	0.0152
Window3	5.2288%	**0.0000%**	12.4180%	**7.1896%**	10.4575%	**5.8170%**	0.0062	0.0222
Window4	0.0000%	0.0000%	18.654%	**0.0000%**	2.0915%	**0.0000%**	0.0058	0.0289
Window5	3.2051%	**2.5641%**	12.1800%	**3.8462%**	4.1027%	**3.0128%**	0.0067	0.0366
Waveform	k–Means Min Error	DCA Min Error	k–Means Max Error	DCA Max Error	k–Means Avg Error	DCA Avg Error	k–Means Avg Time	DCA Avg Time
Window1	1.4014%	**0.8008%**	**2.8028%**	3.0032%	1.9119%	**1.8622%**	0.0919	0.1404
Window2	1.0010%	**0.7007%**	**1.5015%**	4.0040%	**1.2412%**	2.0420%	0.0928	0.2797
Window3	**0.6006%**	1.0010%	**0.6006%**	2.2022%	**0.6006%**	1.6216%	0.0899	0.4180
Window4	**0.1001%**	0.3003%	**0.8008%**	5.1051%	**0.5205%**	2.7424%	0.0887	0.5520
Window5	1.5936%	**0.9961%**	**42.5300%**	48.3070%	**6.0259%**	6.4243%	0.0976	0.6879
Magic	k–Means Min Error	DCA Min Error	k–Means Max Error	DCA Max Error	k–Means Avg Error	DCA Avg Error	k–Means Avg Time	DCA Avg Time
Window1	8.1370%	**1.9514%**	10.2730%	**6.3697%**	9.5435%	**5.9278%**	0.2579	0.2963
Window2	**2.2176%**	2.4762%	7.4512%	**4.1243%**	3.4822%	**3.1842%**	0.2447	0.5807
Window3	1.9514%	**0.4050%**	2.1723%	**0.6259%**	2.1060%	**0.5817%**	0.2298	0.9000
Window4	0.9208%	**0.9205%**	**1.1419%**	1.5832%	**1.0309%**	1.4507%	0.2351	1.2713
Window5	**0.6627%**	0.7732%	1.0309%	**0.7732%**	**0.6996%**	0.7732%	0.2173	1.5880
Window6	1.8041%	**1.1046%**	1.8041%	**1.1782%**	1.8041%	**1.1267%**	0.2191	1.5880
Window7	3.4875%	**1.3216%**	3.5609%	**1.3216%**	3.5095%	**1.3216%**	0.2033	2.1687

k–Means. The Min Error of local clustering using k–Means varies from 0% to 8.1% over all windows.

2.4 Clustering data streams over sliding windows

As a common data mining task, clustering is widely studied to reveal similar features among data records. In general, clustering over data streams is dominated by the outdated historic

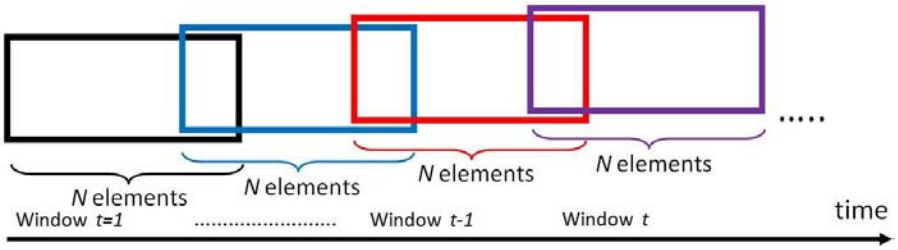


Figure 2.4: Analysis on sliding-windows

information of the stream, since the most recent N records are considered to be more critical and preferable in many applications. Hence, clustering data streams over sliding windows is a natural choice and becomes one of the most popular models. In the sliding window model, data elements arrive continually, and only the most recent elements are considered. In this context, at the period t - called window t, we consider N recent elements that contain some elements of window $t-1$.

Several subjects should be studied in clustering data streams over sliding window: the design of effective and scalable stream clustering algorithms, the analysis of characteristics of a cluster (e.g., the number of objects, the center and radius of the cluster) and of the evolving behaviors of different clusters as well as the evolution of the individual centers, etc. In this work, assuming that each window contains N records and k clusters (N and k are fixed), we attempt to investigate efficiency clustering algorithms over sliding windows.

The data in one window can formally be written as $(x_1, x_2, ..., x_N)$, where a single observation x_i has d-dimensional. In our sliding windows model, the evolution of data items can be presented as follows:

$$\begin{aligned} Data^{windowt} &= x_1, x_2, ..., x_M, x_{M+1}, x_{M+2}, ..., x_{N-1}, x_N \\ Data^{windowt+1} &= \qquad\qquad x_{M+1}, x_{M+2}, ..., x_{N-1}, x_N, x_{N+1}, ..., x_{N+M-1}, x_{N+M} \end{aligned}$$

This mean that M elements oldest will be remove and M new elements are inserted. Then the speed of evolution is defined as $(M/N)\%$. Since the new streams will differ from the current ones but slightly, it is important to study the initialization strategy of clustering algorithm performing on new windows. This is the main purpose of our work.

In out study, we investigate an efficient clustering algorithm in the nonconvex programming framework called DCA. Basing on an efficient and scalable DCA schema to solve the MSSC (Minimum Sum–of–Squares Clustering) problem: DCA–MSSC which is introduced in the previous section, we propose an initialization strategy for clustering on the new window and design a streaming clustering algorithm for data streams over sliding windows. To evaluate the efficiency of this strategy, we compare it with the same version of DCA using random initial points. We also study the effect of these strategies toward the speed of evolution of data, as well as the performance of DCA based approaches versus the standard k–Means algorithm.

2.4.1 A DCA-stream algorithm

For clustering data stream over sliding window we propose the following DCA based scheme, denoted by **DCA–stream**. The main idea is to start DCA in the window t from the optimal solution computed in the window $t-1$. In other words, the clustering structure of the current streams is taken as an initialization for the clustering structure of the new stream. This initialization will usually be good since the new stream will differ from the current ones but slightly.

DCA–stream:

Initialization:

At the first window, $t = 0$.

- Let $X^{(0,t)}$ be an initial point randomly chosen from the data of the first window.
- Apply DCA–MSSC from the initial point $X^{(0,t)}$ to get an optimal solution $X^{(t,*)}$.

Repeat

- Set $t \leftarrow t + 1$
- Set $X^{(0,t)} = X^{(t,*)}$
- Apply DCA–MSSC from the initial point $X^{(0,t)}$ to get an optimal solution $X^{(t+1,*)}$ at window $t + 1$.

Until All of windows are performed.

To study the performance of **DCA–stream** we consider another version, called **DCA-random**, which consist of applying DCA–MSSC at each window t from an initial points randomly chosen among the data set of this window.

DCA-random:

Set $t = 0$.

Repeat

- Let $X^{(0)}$ be an initial point randomly chosen from the data of the window t.
- Apply DCA–MSSC from $X^{(0)}$.
- Set $t \leftarrow t + 1$

Until All of windows are performed.

2.4.2 Numerical experiments

We execute experiments with 3 real-world datasets: *KDD99* from Zhu [web site], *KDD98* from Farnstrom et al. [2000] and *SEA* from Street and Kim [2001].

The data *KDD-CUP'99* Network Intrusion Detection corresponds to the important problem of automatic and real-time detection of cyber attacks. This is a challenging problem for dynamic stream clustering in its own right (Aggarwal et al. [2003]). This dataset contains totally 494.021 elements, and each element contains 42 attributes, including 34 continuous attributes, 7 categorical attributes and 1 class attribute. As in Aggarwal et al. [2003], all 34 continuous attributes will be used for clustering, and the number of clusters is 5.

KDD98 contains 95.412 records of information about people who have made charitable donations in response to direct mailing requests, and clustering can be used to group donors showing similar donation behavior. We use 56 fields which can be extracted from the total 481 fields of each record. The number of clusters is 10 (as in Aggarwal et al. [2003], Farnstrom et al. [2000]). This dataset is converted into a data stream by taking the data input order as the order of streaming and assuming that they flow-in with a uniform speed (as in Aggarwal et al. [2003], Farnstrom et al. [2000]).

SEA is proposed by Street and Kim (Street and Kim [2001]), with 60.000 elements, 3 attributes and 2 classes.

To evaluate the clustering quality, we are using the CH value for evaluation the results that is introduced by Calinski and Harabasz (Caliński and Harabasz [1974], Vendramin et al. [2009]). The CH value is expressed as follows:

$$CH(k) := \frac{[traceB/(k-1)]}{[traceW/(n-k)]} \qquad \text{(CH index)}$$

where

$$traceB := \sum_{i=1}^{k} |C_i| \ \|\overline{C_i} - \overline{x}\|^2; \quad traceW := \sum_{i=1}^{k} \sum_{j \in C_i} \|x_j - \overline{C_i}\|^2$$

with $|C_i|$ is the number of objects assigned to the cluster C_i $(i = 1, \ldots, k)$; $\overline{C_i}$ is the center of cluster i and $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ is the center of whole data set. The best clustering is achieved when CH is maximized.

For all data sets, the number of windows is 10, and all of data are normalized before perform clustering. The size of each window in the data sets SEA is $N = 5.000$, and in other data sets $N = 10.000$.

2.4.2.1 Comparision between DCA–stream and DCA-random

We perform clustering on each window by two algorithms: **DCA–stream** and **DCA-random** and execute experiment with three values of the speed of evolution is defined

as $(M/N)\%$. In the first window, the two algorithms start from the same solution randomly chosen from the data points.

Table 2.4: Number of iterations and running time in seconds of **DCA-Random** (1) & **DCA–stream** (2) on KDD99 dataset

KDD99	Speed 25%				Speed 50%				Speed 75%			
	No.Iter		Times		No.Iter		Times		No.Iter		Times	
Window	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
1	29	29	0,843	0,798	29	29	0,796	0,885	29	29	0,796	0,870
2	7	3	0,141	0,125	7	4	0,156	0,156	24	2	0,656	0,078
3	7	2	0,125	0,075	7	2	0,140	0,077	7	2	0,125	0,080
4	7	2	0,140	0,096	21	2	0,546	0,083	7	21	0,140	0,660
5	7	2	0,125	0,073	7	2	0,125	0,080	8	2	0,202	0,083
6	7	2	0,140	0,077	7	4	0,125	0,159	7	2	0,125	0,106
7	7	2	0,141	0,073	8	3	0,171	0,125	36	2	0,967	0,095
8	19	2	0,500	0,087	7	5	0,141	0,164	7	1	0,124	0,055
9	20	2	0,531	0,085	7	2	0,141	0,108	7	1	0,125	0,048
10	7	10	0,125	0,302	7	2	0,171	0,084	36	1	0,983	0,074
Avg.	11,7	**5,6**	0,2811	**0,1791**	10,7	**5,5**	0,2512	**0,1921**	16,8	**6,3**	0,4243	**0,2149**

From our numerical results (Fig. 2.5, 2.6, 2.7 and Table. 2.4, 2.5, 2.6), we observe that:

i) With an appropriate starting point in the first window, the **DCA–stream** algorithm is more efficient than **DCA-random** in all cases, on both the clustering quality (the value CH) and the rapidity (the number of iterations and average running time).

ii) The behavior of **DCA–stream** is quite stable: after the first one or two windows, the algorithm converges with a small number of iterations and in a short time.

2.4.2.2 Comparisom with k–Means stream

In this experiment we compare **DCA–stream** with the classical algorithm k–Means (Macqueen [1967]) with the same initialization strategy at each window. The initial points in the first window are the same for both **DCA–stream** and k–**Means stream**.

From numerical results (Fig. 2.8, 2.9, 2.10 and Table. 2.7), we see that:

i) With the same appropriate starting point in the first window, **DCA–stream** is better than the k–**Means stream** on the quality of clustering (the CH value).

ii) k–**Means stream** is slightly less expensive than **DCA–stream** on running time in 2/3 data sets, whereas both algorithms are fast.

In the effect of streaming algorithms toward the speed of evolution of data point of view, we are not surprising. In fact, for all experiments, in both **DCA–stream** and k–**Means stream**, the smaller speed of evolution is, the better algorithms will be (here we compare the number of iterations and running-time but not the CH value because the data sets change).

Table 2.5: Number of iterations and running time in seconds of **DCA-Random** (1) & **DCA–stream** (2) on KDD98 dataset

KDD98	Speed 25%				Speed 50%				Speed 75%			
	No.Iter		Times		No.Iter		Times		No.Iter		Times	
Window	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
1	50	50	4,131	3,939	50	50	3,851	4,095	50	50	4,418	4,061
2	16	22	1,106	1,831	25	22	1,733	1,905	52	23	4,060	1,953
3	39	4	3,166	0,361	7	7	0,419	0,594	6	7	0,363	0,629
4	15	6	1,106	0,526	37	6	2,609	0,500	8	7	0,493	0,606
5	24	3	1,664	0,319	8	4	0,497	0,339	33	5	2,357	0,442
6	15	4	1,048	0,354	22	6	1,613	0,519	15	5	1,059	0,451
7	45	3	3,418	0,304	27	4	1,936	0,340	6	5	0,340	0,448
8	6	2	0,358	0,197	18	4	1,250	0,369	45	5	3,369	0,430
9	12	3	0,775	0,270	6	4	0,344	0,377	37	5	2,780	0,443
10	18	4	1,341	0,398	7	5	0,408	0,407	9	4	0,574	0,364
Avg.	24	**10,1**	1,8113	**0,8499**	20,7	**11,2**	1,4660	**0,9445**	26,1	**11,6**	1,9813	**0,9827**

Table 2.6: Number of iterations and running time in seconds of **DCA-Random** (1) & **DCA–stream** (2) on SEA dataset

SEA	Speed 25%				Speed 50%				Speed 75%			
	No.Iter		Times		No.Iter		Times		No.Iter		Times	
Window	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
1	10	10	0,039	0,033	10	10	0,038	0,032	10	10	0,043	0,031
2	14	2	0,055	0,008	25	2	0,102	0,008	15	3	0,057	0,011
3	22	2	0,087	0,008	30	2	0,118	0,008	30	4	0,120	0,020
4	30	2	0,109	0,009	30	3	0,105	0,013	20	6	0,082	0,021
5	29	2	0,114	0,010	14	3	0,053	0,010	19	2	0,073	0,011
6	12	3	0,044	0,013	22	6	0,091	0,022	12	6	0,050	0,019
7	23	2	0,096	0,012	22	2	0,082	0,008	20	3	0,077	0,012
8	14	2	0,052	0,009	17	2	0,071	0,009	26	2	0,111	0,008
9	20	3	0,078	0,011	13	2	0,048	0,008	19	3	0,081	0,013
10	26	4	0,094	0,013	12	3	0,042	0,011	16	3	0,073	0,014
Avg.	20	**3,2**	0,0768	**0,0126**	19,5	**3,5**	0,0750	**0,0129**	18,7	**4,2**	0,0767	**0,0160**

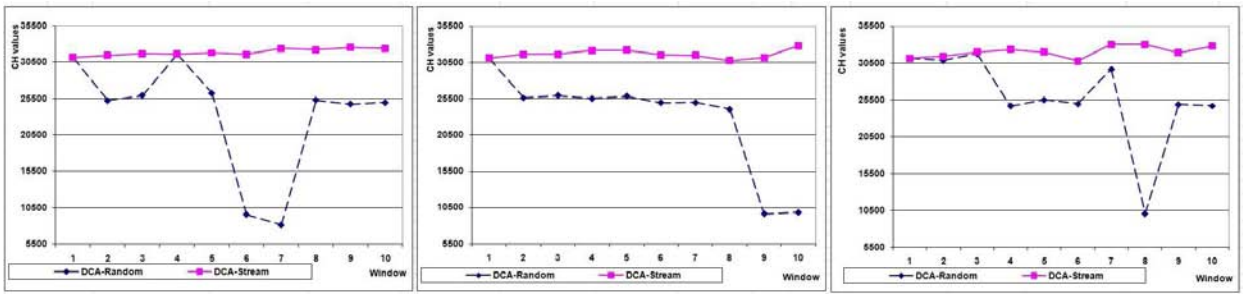


Figure 2.5: The CH values given by **DCA-random** & **DCA–stream** versus the speeds of evolution data: 25% (left) 50% (center) 75% (right) on KDD99 dataset

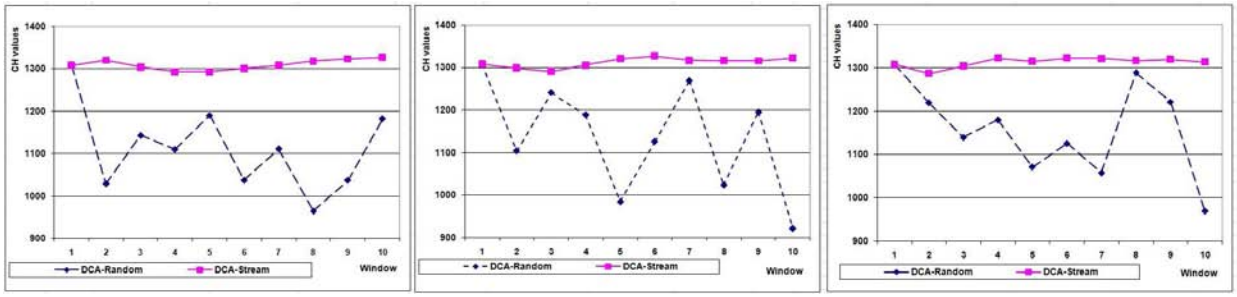


Figure 2.6: The CH values given by **DCA-random** & **DCA–stream** versus the speeds of evolution data: 25% (left) 50% (center) 75% (right) on KDD98 dataset

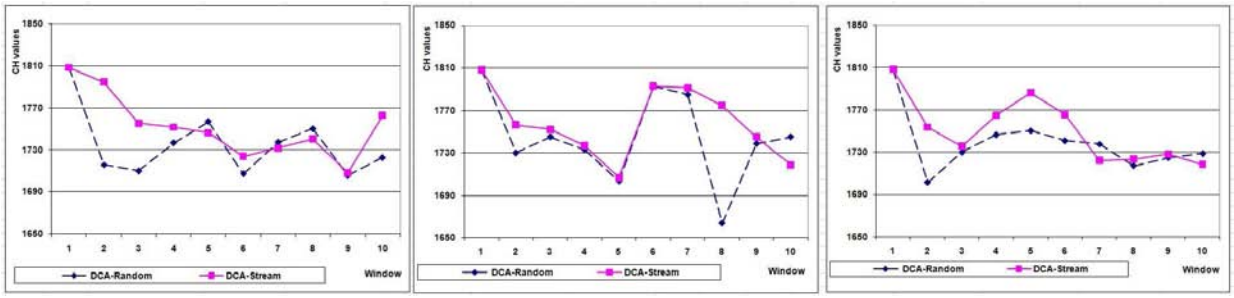


Figure 2.7: The CH values given by **DCA-random** & **DCA–stream** versus the speeds of evolution data: 25% (left) 50% (center) 75% (right) on SEA dataset

2.5 Conclusion

In this chapter we focus on the data streams clustering using two strategies. In the first part, we proposed sub–windows approach with an efficient DCA based clustering algorithm. The data is divided into sub–windows with the same size. Then, we perform clustering by DCA–MSSC algorithm in each sub–window. Mining data in sub–window allows reduce both physical memory and CPU time. Also, it may detect the clusters when data evolving with time. The numerical experiments show the adequation between independent local cluster-

Table 2.7: Average of number of iterations and running time over 10 windows of **DCA–stream** & k–**Means stream**

Data	Algorithm	No.Iteration			Times (s)		
		Speed 25%	Speed 50%	Speed75%	Speed 25%	Speed 50%	Speed 75%
KDD99	k–Means stream	2,5	2,8	3,1	0,0618	0,0710	0,0805
	DCA–stream	5,6	5,5	6,3	0,1791	0,1921	0,2149
KDD98	k–Means stream	4,0	4,9	6,2	0,2204	0,2588	0,3122
	DCA–stream	10,1	11,2	11,6	0,8499	0,9445	0,9827
SEA	k–Means stream	8,0	8,8	9,9	0,0187	0,0199	0,0243
	DCA–stream	3,2	3,5	4,2	0,0126	0,0129	0,0160

ing and global clustering strategies. They also prove that the independent local clustering strategy using DCA can be effectively investigated for clustering data stream and clustering on mass of data.

In the last part, we proposed DCA clustering algorithm in the context of data stream over sliding windows. We have improved the computational aspects of DCA clustering algorithm by investigating a good initialization strategy for performing clustering on new windows. Preliminary numerical experiments show the advantage of this strategy and the efficiency of **DCA–stream** algorithm for clustering over sliding windows. The performance of DCA suggests us to develop this approach for other tasks of mining data streams. On the other hand, a deeper study on other subjects of clustering data streams could be interesting and useful for designing efficient streaming algorithms.

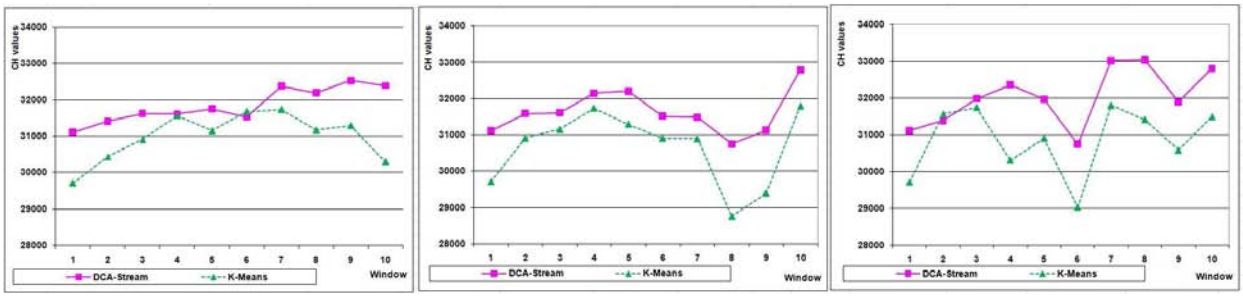


Figure 2.8: The CH values given by **DCA–stream** & k–**Means stream** versus the speeds of evolution data: 25% (left) 50% (center) 75% (right) on KDD99 dataset

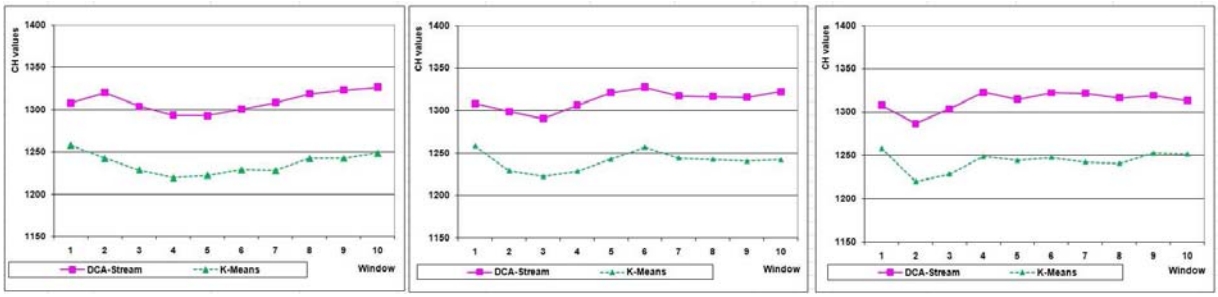


Figure 2.9: The CH values given by **DCA–stream** & k–**Means stream** versus the speeds of evolution data: 25% (left) 50% (center) 75% (right) on KDD98 dataset

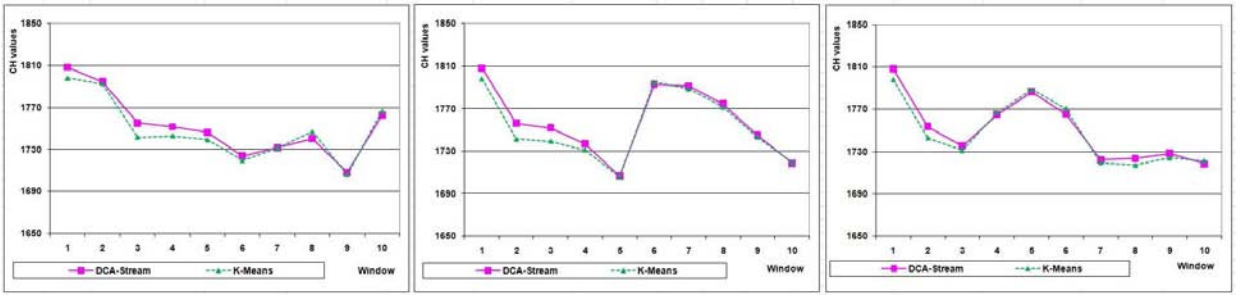


Figure 2.10: The CH values given by **DCA–stream** & k–**Means stream** versus the speeds of evolution data: 25% (left) 50% (center) 75% (right) on SEA dataset

Chapter 3

Clustering massive data sets [1]

Clustering massive data sets is a challenging research area in data mining. In the literature, there are several methods for solving problem clustering on massive data sets. But finding an efficient solution for this problem remains a challenge to researchers. In this chapter, we study an efficient nonconvex optimization method for clustering on massive data sets. Our approach consists of two phases and is based on DC (Difference of Convex functions) programming and DCA (DC Algorithms). In the first phase, the data is divided into subsets on which an efficient DCA for clustering is investigated. In the second phase, another DCA for weighted clustering on the set of centers obtained by the first phase is presented. The numerical results on real data sets show the efficiency of our method. Experimental results on both synthetic and real data sets are promising and they demonstrate the effectiveness of our approach.

3.1 Introduction

The massive data will be linked not only to the volume but also the complexity of processing. The techniques of clustering and learning must be adapted to such problems.

In the recent literatures, there has been research on clustering large or very large data sets, which can be broadly classified into 2 categories: partitioning methods and hierarchical methods.

We consider some partitioning methods. First, the method described by Bradley et al. [1998], which is improved by Farnstrom et al. [2000]. Bradley's algorithm used two compression steps to store information of data. The first step is primary compressing: the data points are in

1. A part of this chapter is published under the title:
Ta Minh Thuy, Hoai An Le Thi, and Lydia Boudjeloud-Assala. *An Efficient Clustering Method for Massive Dataset Based on DC Programming and DCA Approach*. 20th International Conference on Neural Information Processing (ICONIP2013), 3–7 November, Part II, Lecture Notes in Computer Science Volume 8227, pp. 538–545 (2013).

one cluster, which are unlikely to change to a different cluster, will be put into a discarded set. In the second compression, the remaining data points will be partitioned into a larger number of clusters. The objective of this phase is to reduce the space of memory by storing some representative clusters instead of the data points. The points satisfying a threshold criteria is removed, then, the memory becomes available. Note that, memory still stores a retained set including all retained points. Now, the new points are inserted and the process is repeated. This algorithm takes a cost for some operations because, it is related with several data compression steps and it stores the retained set in the memory.

Farnstorm et al.'s (Farnstrom et al. [2000]) presents a simple single pass k-Means method which is a special case of Bradley's algorithm. The difference subsists in the second compression, after performing clustering where all points in memory are removed. Each discard sets in this algorithm is compacted like a weighted point, and are joined with the new arrival points.

Similarly, by focusing on the representative information, Nittel et al.(Nittel and Leung [2004], Nittel et al. [2004]) introduced a *partial/merge* method which consists of two phases for clustering mass of data sets. In the first phase, data is divided into p subsets, then an algorithm clustering on each subset is performed to choose k centers x^l $(l = 1, .., k)$. This algorithm leads to a optimization problem, that minimize the associated MSSC clustering formulation. In their work, the authors focus on the k-Means approach. In the second phase, on the set of $s = k \cdot p$ center points x^{ij} $(i = 1, .., k; j = 1, .., p)$ and s weight values w_i $(i = 1, .., s)$ obtained from phase 1, the authors executed partition by a k-Means Weight based clustering algorithm. s center points redefined as a set of b^i $(i = 1, ..., s)$ (called by weight centers), w_i defined as the number of points that are assigned to the centers b^i. The clustering problem in this phase leads to another problem optimization: find k centers $(u^1, u^2, ..., u^k)$ of s points b^i such that the formula as follows is minimized:

$$E_{weight} = \sum_{\ell=1}^{k} \sum_{b^i \in C_l} w_i \parallel u^\ell - b^i \parallel^2 \longrightarrow \min \tag{3.1}$$

Another partitioning method for clustering is the k-medoids based algorithm: CLARANS (CLustering Algorithm based on RANdomized Search) (Ng and Han [2002]). CLARANS use a random searching to facilitate the large data set clustering. First, the algorithm take k arbitrary points from data points as centers, then consider a random neighbour with aim of search a solution minimize the sum of distances between all objects in a cluster from their respective cluster center. The advantage of this algorithm is to not dealt with data points, which is dedicate to handling polygonal objects. But, this algorithm uses random searching, so the quality of the results cannot be guaranteed in case of large–size. Moreover, CLARANS are designed with the assumption that the entire data set is in memory. It is not always the case for large data sets. In this paper, we also consider the case the memory can not read all data set.

For the hierarchical methods where the data set is decomposed into a tree–like structure, we can cite, for example, BIRCH (Balanced Iterative Reducing and Clustering using Hi-

erarchies) (Zhang et al. [1996]). BIRCH builds a hierarchical data structure (denoted by CF–tree: clustering feature tree) to incrementally cluster the incoming objects using the available memory. When considering an object, the algorithm will search the closest leaf entry. Starting from the root of CF-tree, it traverses the CF tree from top to down and finds the closest leaf node entry, the object will be added to this entry. In summary, BIRCH uses a CF-tree to summarize the entire data set. It is the first algorithm that can handle the outlier (Marjan Kuchaki Rafsanjani and Chukanlo [2012]). However, during processing, we must check the threshold condition T, that represents the number of children per non leaf node is smaller than B and the maximum number of points in the leaf node (L). Therefore, in some cases if the choice the values B, T, L is not appropriate, CF tree does not be represented well. Furthermore, BIRCH based on calculating the radius (or diameter) to control the cluster limit, so it is only suitable if the clusters are spherical.

CURE (Clustering Using REpresentatives) (Guha et al. [1998]) is also, a hierarchical clustering algorithm, which presents some different points. Firstly, it is not initialized by the individual centers, it starts by a fixed number of well-scattered points. Secondly, it shrinks the multiple representative points towards their center by a constant fraction. At each step of the algorithm, the two closest pair of representative points are merged and their representatives are re-calculated. Since CURE use a multiple representative, it can cluster with non spherical clusters. In the shrinking phase, CURE can easy detect the outlier points. However, CURE calculates the similarity of two clusters based on the closest of the representative points belonging to different clusters, without considering the internal correlation of the two clusters.

In this chapter, we use the same idea of Nittel et al. (Nittel and Leung [2004], Nittel et al. [2004]) to design a two phases algorithm. In the first phase, we use DCA–MSSC clustering algorithm instead of classical k-Means. In the second phase, we developed another DCA based clustering algorithm, which is an extension of the first phase introducing weighted parameters in the objective function. Finally, we get the final centers of clusters of the whole data. Then, we perform partition of all elements of the data set.

We study 2 options:

Option 1: The elements are assigned to final center corresponding to their center in the phase 1.

Option 2: We recompute the distance from elements to the final centers, and all elements are assigned to the closest center.

The aims of our study are:

1. The efficiency of clustering with two phases comparing with the clustering on the whole data set.
2. The efficiency of our method with two options above.

According to these problems, we perform experiments on both synthetic and real-world data

sets. In the next section, we will review the algorithm for the clustering problem on the subsets. Then, we study the clustering algorithm on the sets of weight centers.

3.2 DCA clustering algorithms

3.2.1 DCA for the first phase

In this phase, we applied a DCA clustering algorithm, **Algorithm DCA-MSSC**, in each subsets, we perform clustering the data set $\mathcal{A} := \{a^1, ..., a^q\}$ of q points in $\mathbb{R}^n$ into k homogeneous clusters such that members of the same cluster are similar and members of distinct clusters are dissimilar. This clustering problem corresponding optimization formulation is expressed as:

$$\min \quad \left\{ \sum_{i=1}^{q} \min_{\ell=1,\dots,k} \left\| x^\ell - a^i \right\|^2 : x^\ell \in \mathbb{R}^n, \ell = 1, \dots, k \right\}. \tag{3.2}$$

This algorithm has been introduced in the section 1.3, which is developed in the original paper Le Thi et al. [2007a].

3.2.2 DCA for the second phase

We developed a DC schema for clustering problem on the weighted centers set by using the same idea of DCA–MSSC and taking into account of the weight values.

As for previous problems, to simplify related computations in DCA for solving problem (3.1) we will work on the vector space $\mathbb{R}^{k\times n}$ of $(k \times n)$ real matrices. We have:

$$\begin{aligned} F(U) &= \frac{1}{2} \sum_{i=1}^{s} w_i \min_{\ell=1,..,k} \|U_\ell - b^i\|^2 \\ &= \sum_{i=1}^{s} \sum_{\ell=1}^{k} w_i \frac{1}{2} \|U_\ell - b^i\|^2 - \sum_{i=1}^{s} w_i \max_{j=1,..,k} \frac{1}{2} \sum_{\ell=1,\ell\neq j}^{k} \|U_\ell - b^i\|^2 \\ &= G(U) - H(U). \end{aligned} \tag{3.3}$$

where:

$$G(U) = \sum_{i=1}^{s} \sum_{\ell=1}^{k} G_{i\ell}(U), \quad G_{i\ell}(U) = \frac{1}{2}\, w_i \|U_\ell - b^i\|^2$$

$$H(U) = \sum_{i=1}^{s} H_i(U), \quad H_i(U) = \max_{j=1,..,k} H_{ij}(U) = \max_{j=1,..,k} \sum_{\ell=1,\ell\neq j}^{k} \frac{1}{2}\, w_i \|U_\ell - b^i\|^2$$

Since $G(U)$ and $H(U)$ are convex functions, problem (3.1) is a DC program.

Recall that m is the objects in the whole data set, w_i denotes the number of points assigned to the centers b^i, $s = k \cdot p$ with p being the number of subsets, so $\sum_{i=1}^{s} w_i = m$. It leads to:

$$G(U) = \frac{1}{2}\sum_{i=1}^{s}\sum_{\ell=1}^{k} w_i \|U_\ell - b^i\|^2 = \frac{m}{2}\|U\|^2 - \langle B, U\rangle + \frac{1}{2}\|A\|^2, \tag{3.4}$$

where $A_i = \sqrt{w_i} b^i \in \mathbb{R}^n$ (i=1,..., s); $B_\ell = \sum_{i=1}^{s} w_i b^i \in \mathbb{R}^n (\ell = 1, ..., k)$.

The DC program (3.3) then is minimized the difference of the simplest convex quadratic function $G(U)$ and the nonsmooth convex function one $H(U)$. This is quite similar with the DC program in the first phase, solving (3.3) by DCA amounts to computing the two sequences $\{U^{(p)}\}$ and $\{V^{(p)}\}$ in $\mathbb{R}^{k\times n}$ such that: $V^{(p)} \in \partial H(U^{(p)}), U^{(p+1)} \in \partial G^*(V^{(p)})$.

Similarly, we can calculate $V \in \partial H(U)$ by:

$$V = mU - \sum_{i=1}^{s} w_i B^{[i]} - \sum_{i=1}^{s} e_{j(i)}^{[k]} w_i (U_j - b^i). \tag{3.5}$$

where $B^{[i]} \in \mathbb{R}^{k\times n}$ is the matrix whose rows are all equal to b^i; $e_j^{[k]}$, $j = 1, .., k$ are the canonical basis of $\mathbb{R}^k$.

Since the function G is strictly convex quadratic and its conjugate G^* is differentiable, from (3.4) we have: $U = \nabla G^*(V)$.

So:

$$V = \nabla G(U) = mU - B$$

Finally:

$$U = (V + B)/m. \tag{3.6}$$

Now, we have the DC algorithm to solve problem (3.1) as follows:

Algorithm DCA Weight:

Initialization: Let $\epsilon > 0$ be given, $p := 0$. $U^{(0)}$: initial cluster centers.

Repeat:

Calculate $V^{(p)}$ from (3.5).

Calculate $U^{(p+1)}$ from (3.6).

Set $p = p + 1$.

Until: $|F^{(p+1)} - F^{(p)}| \le \epsilon(|F^{(p)}| + 1)$ or $\|U^{(p+1)} - U^{(p)}\| \le \epsilon(\|U^{(p)}\| + 1)$.

3.3 Clustering massive data set with the two phases DC Algorithm

In this section, we clearly present a two phase DCA based algorithm for clustering massive data set. In the first phase, the data is divided into p subsets. On each subset, we perform clustering by DCA–MSSC algorithm. Then, we get a centers set and a set of the weight values. In the second phase, we execute clustering by DCA-Weight algorithm on the centers set. Finally, we perform the partition of all points of data set. For this purpose, we study the two options:

First option we assign these points to final center corresponding to their center in the first phase.

Second option we calculate the distance from one point to all final centers. This point will belong to the closest center.

Description of two phase DCA based algorithm:

Phase 1: Divide the data set X into p subset.

Clustering each subset X_i by DCA–MSSC algorithm.

$X' \leftarrow i \cdot k$ centers obtained from clustering subset X_1 to X_p.

Phase 2: Clustering X' using DCA Weight algorithm.

Partitioning all points by first option (named **DCA1**)

or second option (named **DCA2**).

3.4 Experiments

3.4.1 First experiments

In the first experiment, we focus on the effect of clustering with two phases comparing with the clustering on the whole data set. We study three artificial data sets, which are generated by Gauss distribution using R package MixSim (Melnykov et al. [2012]). They are slightly overlapped and present some outliers. This data set include 5, 10, 15 million points described by 10 dimensions. The information about data sets is summarized in Table 3.1.

Table 3.1: Artificial data sets.

Data sets	Points	Attributes	Clusters	Subsets	Elements/Subset
5M10D10C	5.000.000	10	10	5	1.000.000
10M10D10C	10.000.000	10	10	10	1.000.000
15M10D5C	15.000.000	10	5	10	1.500.000

Firstly, the data is randomly divided following a Gauss distribution into different subsets having same size. Then, we execute the two phases clustering algorithm. We also perform clustering by DCA–MSSC algorithm on the whole data set with the assumption that we can read and store whole points of data set on computer memory (4MB RAM). Both of the algorithms have the same initial points, which are randomly chosen from given objects. The initial points in the phase 2 of the algorithm are k centers chosen from s weight centers b^i (i=1,.., s), that correspond to k largest value of s weight values w_i (i=1,.., s).

Since these data sets are artificial, we know the real label of the points, so we focus on comparing the percentage of well classified points (PWCO) and the running time execution. We report the average and standard deviation (denoted by SD) of these values from 5 different sets of initial seeds (Table 3.2 and Table 3.3).

Table 3.2: PWCO of clustering whole data set and clustering with two phases.

Data sets	Avg. PWCO ± SD		
Name	DCA–MSSC	DCA1	DCA2
5M10D10C	82.65% ± 10.17%	79.76% ± 7.63%	**86.55%** ± **7.17%**
10M10D10C	86.57% ± **6.85%**	78.43% ± 8.74%	**87.51%** ± 9.93%
15M10D5C	82.16% ± 14.25%	76.53% ± 14.26%	**85.37%** ± **11.55%**

Table 3.3: Time running of clustering whole data set and clustering with two phase.

Data sets	Avg. Time ± SD		
Name	DCA–MSSC	DCA1	DCA2
5M10D10C	1681,90 ± 22.28%	**1519,00** ± 17.21%	1624.36 ± **16.10%**
10M10D10C	3570,73 ± 23.81%	**3291,30** ± 10.07%	3540,87 ± **9.36%**
15M10D5C	1206,39 ± 20.40%	**1122,51** ± 13.29%	1389.68 ± **11.21%**

The numerical results presented in the Table 3.2 and Table 3.3 shown that DCA2 is the best algorithm. It not only gives better solutions in term of the average of PWCO (with all data sets) but also on the standard deviation (SD) values (with two data sets: 5M10D10C and 15M10D5C). DCA1 is not better than DCA–MSCC, but the execution time is faster, while DCA2 is not too time consuming, even we perform sequential clustering on all subsets. We note that the running time of two phase algorithm include time process and time regrouping. In case of using option 2, we must read again entire data. This time can significantly improve by using parallel processing: multi processer or multi–machines in the first phase.

3.4.2 Second experiments

In the second experiment, we compare the two phases clustering algorithm results with thus of the k–Means algorithm applied in the two phases. We study with the data set: 20M10D10C, which include 20 million points, 10 dimensions and 10 classes. This data set is very large, we can not read the whole points at the same time, so we divide it into 10 subsets, each subset consist of 2.000.000 points.

We execute the two phases algorithms based on DCA and k–Means. In our experiments, we applied clustering with two options presented before. DCA1 and DCA2, respectivly k–Means1 and k–Means2, are the associated results. The initial points in the first phase of the two algorithms based on DCA and k–Means are the same and randomly chosen from given objects.

Table 3.4: The PWCO values of DCA1, DCA2 vs. k–Means1, k–Means2.

Data sets	Avg.PWCO ± SD		Avg.PWCO±SD	
Name	DCA1	KM1	DCA2	KM2
20M10D10C	72.52% ± 8.20%	72.16% ± **5.00%**	**83.18%**± 8.62%	81.76% ± 6.41%

Table 3.5: The CPU Time values of DCA1, DCA2 vs. k–Means1, k–Means2.

Data sets	Avg.Time ± SD		Avg.Time±SD	
Name	DCA1	KM1	DCA2	KM2
20M10D10C	4847,80 ± 7.78%	**2742,10** ± 4.86%	5207,11 ± 7.23%	3080,57 ± **4.31%**

As we can see in Table 3.4 and Table 3.5: the results of option 2 are always better than the results of option 1 in both DCA based algorithm and k–Means based algorithm. The gain of k–Means2 and k–Means1 is 9.6%, while DCA2 and DCA1 is 10.66%. The two phases based DCA algorithm given PWCO values better than two phases based k–Means algorithm in both options. DCA2 is the best in term of PWCO values. Contrary, k–Means1 is the worst, but it is the faster algorithm.

3.4.3 Third experiments

In the last experiment, we tested with real–world data sets. These data sets are generally used as stream data sets with the assumption that we can not randomly access the data, so we take data into the subsets using their index.

We perform experiments on 7 data sets: *Forest Covertype (FCT)* taken from Blackard [1999], *KDD98* from Farnstrom et al. [2000], *KDD99*, *HyperPlane (HP)* from Zhu [web site], *SEA*

from Street and Kim [2001], Sensor_1 from Zhu et al. [2011], Sensor_2 from M. Rastgoo et al. [2012]. The information is summarized in Table 3.6.

Data set *Forest Covertype* for 30×30 meter cells is obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. It contains 581.012 instances and 54 attributes and 7 classes, and it has been used in several papers on data stream classification. The data can be downloaded from UCI (Newman and Merz) or from Blackard [1999] it is normalized.

The *KDD98* data set contains 95.412 records of information about people who have made charitable donations in response to direct mailing requests. The similar donation behavior will be grouping into the same cluster. This data set is converted into a data stream with 56 attributes, which are extracted from the total 481 fields of each record. The number of clusters is 10 (same as Farnstrom et al. [2000]).

KDD-CUP'99 Network Intrusion Detection corresponds to a challenging problem for dynamic stream clustering. The objective of this problem is to detect the kind of attack. Data set contains 494.021 points, 41 attributes with 24 types attack, including 34 continuous attributes, 7 categorical attributes. We only use 34 continuous attributes (as Aggarwal et al. [2003]) and redefinition type attacks fall into two main categories: normal and abnormal.

HyperPlane contains 100.000 instances and 10 attributes and 5 classes, the data set can be downloaded from Zhu [web site]; and *SEA* proposed by Street, W.N. and Kim, Y. (Street and Kim [2001]) with 60.000 elements, 3 attributes and 2 classes.

Sensor_1 (Zhu et al. [2011]) holds five information: rcd_minutes, temperature, humidity, light and voltage that selected from four regions of 54 sensors deployed in Intel Berkeley Research Lab. The processed stream has 1.169.260 points, each of which has 5 attributes and 4 clusters (four regions).

The *Sensor_2* (M. Rastgoo et al. [2012]) data set contains 2.219.803 elements, 5 attributes. This experiment aims to infer the illuminance state based on the measurements provided by each sensor. This data set consists of 2 clusters.

Table 3.6: Data sets

Data sets	Points	Att	Clusters	Subsets	Elements/Subset
SEA	60000	3	2	10	6.000
HP	100000	10	5	10	10.000
KDD98	95412	56	10	10	9.541
KDD99	494021	34	2	10	49.402
FCT	581012	54	7	10	58.101
SENSOR_1	1169260	5	4	10	116.926
SENSOR_2	2219803	5	2	10	221.980

All these data sets are normalized before experiments. Since the clustering problem unknown the real label of the data, in this experiment, we use 3 criteria to do the comparison: the

value of weight objective function (formula 3.1), the CH value (formula CH index) and CPU running time. This last value includes the time process and the time regrouping.

Table 3.7: The weight objective functions of two phase algorithms based on DCA and k–Means.

Data sets	Avg. Obj ± SD	
Name	DCA	k–Means
SEA	**176025 ± 6.41%**	195418 ± 8.29%
KDD98	**29205** ± 25.77%	32097 ± **21.37%**
HP	**40266± 1.74%**	43404 ± 1.92%
KDD99	**90054** ± 12.79%	105460 ± **5.58%**
FCT	**190057± 9.82%**	225873 ± 11.89%
Sensor_1	**54792** ± 31.20%	68986 ± **28.60%**
Sensor_2	**86837** ± 9.65%	96987 ± **6.66%**

Table 3.8: The CH values of DCA1, DCA2 vs. k–Means1, k–Means2.

Data sets	Avg.CH ± SD		Avg.CH ± SD	
Name	DCA1	KM1	DCA2	KM2
SEA	**7701 ± 7.79%**	6655 ± 14.43%	**8687± 8.27%**	7657± 11.21%
KDD98	**7083 ± 10.79%**	6963 ± 10.89%	**8287** ± 12.08%	7815 ± **11.38%**
HP	1510± **2.45%**	**1516**± 4.68%	**1770**± 2.66%	1755 ±**2.28%**
KDD99	**279700± 14.39%**	199994 ± 29.98%	**358484 ± 2.41%**	303059 ± 15.06%
FCT	**88739 ± 6.65%**	85390 ± 8.59%	**97823± 6.25%**	92091 ± 8.64%
Sensor_1	**656922 ± 23.94%**	594034 ± 30.28%	**812461± 24.75%**	724526 ± 34.49%
Sensor_2	**669221 ± 12.68%**	550953 ± 15.35%	**799818 ± 10.46%**	670419 ± 13.88%

Table 3.9: The CPU time of DCA1, DCA2 vs. k–Means1, k–Means2

Data sets	Avg.Time ± SD		Avg.Time ± SD	
Name	DCA1	KM1	DCA2	KM2
SEA	3,870 ± **0.47%**	**2,294** ± 1.31%	4,962 ± **5.82%**	**3,336** ± 7.61%
KDD98	165,999 ± **0.02%**	**37,743** ± 0.06%	174,965 ± **0.22%**	**46.500** ± 0.83%
HP	**8,742** ± 0.30%	14,201 ± **0.20%**	**12,064** ± 3.17%	17,465 ± **2.77%**
KDD99	41,383 ± 3.26%	**30,874 ± 2.00%**	64,073 ± 2.42%	**60,218 ± 2.19%**
FCT	295,961 ± 5.57%	**98,133 ± 3.09%**	335,945 ± 4.89%	**146,997 ± 2.09%**
Sensor_1	85,621 ± 6.82%	**53,305 ± 5.92%**	108,533 ± 5.75%	**95,569 ± 4,84%**
Sensor_2	113,530 ± 9.94%	**81,505 ± 8.84%**	155,846 ± 7.34%	**153,308 ± 5.81%**

From our experiments (Table 3.7, 3.8, 3.9), we observe that: the two phases DCA based clustering algorithm gives the weight objective functions, CH values and SD values better

than k–Means based algorithm on all experiments. DCA2 is the best in term of CH values, while k–Means1 is the worst. k–Means1 is less time consuming.

3.5 Conclusion

In this chapter, we have studied the DC programming and DCA for clustering on massive data sets. The classical k-Means and k-Means Weight models were reformulated using a DC programs.

To deal with massive data sets we divide the work on two phases: In the first phase, we use DCA–MSSC algorithm instead of classical k-Means. By using the same idea of DCA–MSSC and taking into account the weight values of the associated clusters, another DCA for clustering on the set of centers obtained by the first phase is studied in the second phase. For this phase, similarly to DCA–MSSC, we get a new DCA Weight algorithm where all computations are explicit and require only matrix–vector product, so our method is simple and inexpensive.

The experiments focus on two studies: the effect of clustering algorithm with two phases in comparison with the clustering on the whole data set, and the effect of clustering algorithm with two phases based on DCA Weight. We also studied the efficiency with the two options to define the partitions. The numerical results on both synthetic and real data sets show that DCA is an efficient approach for clustering in massive data sets.

Chapter 4

Clustering using weighted features

4.1 Solving minimum sum-of-squares clustering using weighted dissimilarity measures [1]

In this section, we study new efficient DCA based algorithms for MSSC (Minimum Sum-of-Squares Clustering) using weighted dissimilarity measures.

4.1.1 Introduction

Nowadays, the growth of technologies leads to exponential augmentation in recorded data in both dimensionality and sample size. In many applications such as e-commerce applications, computational biology, text classification, image analysis, etc. datasets are large volume and contain a large number of features. However, the large number of features can lead to some problems in classification task. Usually, features can be divided into three categories: relevant, redundant and irrelevant features. Relevant features are essential for classification process, redundant features add no new information to the classifier (i.e. information already carried by other features) while irrelevant features do not carry any useful information. The performance of classification algorithms can be significantly degraded if many irrelevant or redundant features are used.

Feature selection is one of the techniques to deal with irrelevant or redundant features.

1. This section is published under the titles:
[1]. Le Hoai Minh, Ta Minh Thuy. *DC programming and DCA for solving Minimum Sum–of–Squares Clustering using weighted dissimilarity measures*. Special Issue on Optimization and Machine Learning, Transactions on Computational Intelligence XIII, ISBN: 978–3–642–54454–5 (Print) 978–3–642–54455–2 (Online) (2014).
[2]. Le Hoai Minh, Ta Minh Thuy, Le Thi Hoai An and Pham Dinh Tao. *DC Programming and DCA for clustering using weighted dissimilarity measures*, Proceedings of the 5th NIPS Workshop on Optimization for Machine Learning, Lake Tahoe, Nevada, USA, 5 pages (2012).

Feature selection methods aim to select a subset of features that minimize redundancy while preserving or improving the classification rate of algorithm. Recently, feature weighting has attracted the attention of many researchers. Feature weighting can be seen as an extension of feature selection. In feature selection, a feature is assigned a binary decision variable (value 1 implies that the feature is selected while value 0 means that it will be removed). In feature weighting, each feature is assigned a continuous value, named a weight, in the interval $[0, 1]$. Relevant features correspond to a high weight value, whereas a weight value close to zero represent irrelevant features. On the contrary to feature selection, the main objective of feature weighting is to improve the quality of classification algorithm, but not to reduce the number of features.

Feature weighting has been applied successfully in many classification algorithms. Feature weighting in SVMs (Do et al. [2009]), in K-Means type clustering (Chan et al. [2004], Huang et al. [2005], Jing et al. [2007]), in Fuzzy classification (Frigui and Nasraoui [2004]), etc to name a few.

In this section, we deal with the MSSC (Minimum Sum of Squares Clustering) using weighted features that is based on the MSSC models in which a weight is added to each feature.

First of all, let us remember the two most used models of MSSC (Minimum Sum-of-Squares Clustering): the bilvel formulation (which has been indicated in Chapter 1) and the mixed integer programming formulation.

An instance of the feature weighting clustering problem consists of a data set $\mathcal{X} := \{x_1, x_2, \ldots, x_n\}$ of n entities in $\mathbb{R}^m$, a measured distance $d(z, x)$ and the number of clusters k $(2 \leq k \leq n)$. The well-known MSSC problem consists in partitioning the set $\mathcal{X}$ into k clusters in order to minimize the sum of squared distances from the entities to the centroid of their cluster.

The bilevel formulation MSSC which was first introduced by Vinod (Vinod [1969]) is defined as follows:

$$\min F_{BI}(Z) := \sum_{j=1}^{n} \min_{\ell=1,\ldots,k} d_E^2(z_l, x_j) \tag{4.1}$$

where the matrix of centers Z is a $(k \times m)$ - matrix whose l^{th} row is the center of cluster C_l and d_E is Euclidean norm, namely $d_E^2(z_l, x_j) = ||z_l - x_j||^2 = \sum_{i=1}^{m}(z_{li} - x_{ji})^2$.

The mixed integer formulation is given by:

$$\begin{cases} \min F_{MI}(W, Z) := \sum_{l=1}^{k} \sum_{j=1}^{n} w_{jl} d_E^2(z_l, x_j) \\ s.t : \sum_{l=1}^{k} w_{jl} = 1, \quad j = 1..n, \\ \quad\quad w_{jl} \in \{0, 1\}, \ j = 1..n, \quad l = 1..k. \end{cases} \tag{4.2}$$

Here W is a $n \times k$ matrix with

$$\begin{cases} w_{jl} & = 1 \text{ if } x_j \in C_l \\ & = 0 \text{ otherwise.} \end{cases} \quad \forall j = 1, \ldots, n, \ l = 1, \ldots, k.$$

The constraint of (4.2) ensures that each point x_j is assigned to one and only one cluster.

Let us now introduce the MSSC models with feature weighting. The dissimilarity measure between z_l and x_j is now defined by m weighted features, namely

$$d^2_{WF}(z_l, x_j) = \sum_{i=1}^{m} \lambda_{li}^{\beta}(z_{li} - x_{ji})^2 \tag{4.3}$$

where $\lambda_{li} \in [0, 1]$ defines the weight (degree of relevance) of i-th feature to the cluster C_l. Hence the bilevel programming formulation of MSSC using weighted features is given by

$$\begin{cases} \min F_1(Z, \Lambda) := \sum\limits_{j=1}^{n} \min\limits_{\ell=1,\ldots,k} \sum\limits_{i=1}^{m} \lambda_{li}^{\beta}(z_{li} - x_{ji})^2 \\ s.t : \sum\limits_{i=1}^{m} \lambda_{li} = 1, \ l = 1..k, \\ \quad \lambda_{li} \in [0, 1], \ l = 1..k, \ i = 1..m. \end{cases} \tag{4.4}$$

Similarly, we have the mixed integer formulation of MSSC using weighted dissimilarity measures:

$$\begin{cases} \min F_2(W, Z, \Lambda) := \sum\limits_{l=1}^{k} \sum\limits_{j=1}^{n} \sum\limits_{i=1}^{m} w_{jl} \lambda_{li}^{\beta}(z_{li} - x_{ji})^2 \\ s.t : \sum\limits_{l=1}^{k} w_{jl} = 1, \ j = 1..n, \\ \quad \sum\limits_{i=1}^{m} \lambda_{li} = 1, \ l = 1..k, \\ \quad w_{jl} \in \{0, 1\}, \ j = 1..n, \ l = 1..k, \\ \quad \lambda_{li} \in [0, 1], \ l = 1..k, \ i = 1..m. \end{cases} \tag{4.5}$$

where β is an exponent greater than 1.

As indicated before, the bilevel programming problem formulation of MSSC (4.1) is a continuous nonsmooth non-convex program. K-Means is the most popular and fast algorithm for solving (4.1). Unfortunately K-means algorithm can not guarantee the global optimality of computed solutions, and it is quite sensitive to initial solutions. Furthermore, K-means doesn't work well with clusters of different size and different density. Problem (4.2) is a mixed integer program where the objective function is nonconvex. It is NP-hard with possibly many local minima. Several exact methods have been developed for solving (4.2). However, these methods are intractable for large scale datasets.

While several heuristic and deterministic approaches have been investigated to MSSC, there is a few deterministic methods dealing with weighted MSSC models. In Chan et al. [2004],

the authors considered a K-means type algorithm, named **WF-KM**, to solve the problem (4.5). At first, **WF-KM** fixes Z, Λ and finds W to minimize $F(W, ., .)$. Then W, Λ are fixed for finding Z minimizing $F(., Z, .)$. Finally, Λ is obtained by minimizing $F(., ., \Lambda)$ with W and Z fixed. The process is repeated until no more improvement in the objective function can be made. In Jing et al. [2007], the authors proposed a variance of (4.5) by adding the entropy of dimensions, namely $\gamma \sum_{j=1}^{m} \lambda_{l,i} \log \lambda_{l,i}$, to objective function. By modifying the objective function, the algorithm can avoid the problem of identifying clusters by few dimensions in sparse data. In another work (Huang et al. [2005]), a simplified version of (4.5) was considered where the matrix of weights Λ becomes a vector $\bar{\Lambda}$. More precisely, $\bar{\Lambda}_j$ defines the relevance of i-th feature to all cluster C_l ($l = 1..k$). The proposed algorithms in Jing et al. [2007] and Huang et al. [2005] are similar to the **WF-KM** developed in Chan et al. [2004].

We investigate in this work DCA based algorithms for both formulations of MSSC using weighted dissimilarity measures: the bilevel programming problem (4.4) and the mixed integer programming program (4.5).

As mentioned in Chapter 1, DC programming and DCA have been extensively developed for hard clustering (Le Thi et al. [2007a], Le Thi et al. [2007b], Le Thi et al. [2014c], Le Hoai et al. [2013b]), fuzzy clustering (Le Thi et al. [2007c]). The starting point of our work is the DC approaches developed in Le Thi et al. [2007a] and Le Thi et al. [2014c]. The DCA–MSSC developed in Le Thi et al. [2007a] for solving (4.1) is very simple and inexpensive (all computation are explicit and require only matrix-vector product). The numerical results on several datasets proved that the proposed DCA is superior to K-means in terms of quality of solutions while the difference of computational time is negligible. Very recently in Le Thi et al. [2014c], the authors have proposed a DCA algorithm for mixed integer program formulation (4.2). Using an exact penalty technique, (4.2) was reformulated as a continuous optimization problem which was then recasted to a DC program.

Observing that the objective function of (4.1) and (4.4) (resp.(4.2) and (4.5)) are slightly different (the introduction of the weight Λ), we adapt the techniques investigated in Le Thi et al. [2007a] (resp. Le Thi et al. [2014c]) for solving problems (4.4) (resp. (4.5)). The main contribution is to study the effectiveness of feature weighting in MSSC.

4.1.2 Solving MSSC using weighted features by DCA

We will adapt the techniques developed in Le Thi et al. [2007a] and Le Thi et al. [2014c] for getting DC formulations and corresponding DCA for solving (4.4) and (4.5).

4.1.2.1 DCA for solving the bilevel problem (4.4)

In the problem (4.4) the variable Λ are a priori bounded. One can also find a constraint to bound the variable Z. Let $\alpha_i := \min_{j=1,\ldots,n} x_{j,i}$, $\gamma_i := \max_{j=1,\ldots,n} x_{j,i}$. Hence $z_l \in \mathcal{T}_l :=$

$\Pi_{i=1}^{m}[\alpha_i, \gamma_i]$ for all $l = 1, ..., k$. Finally, $Z \in \mathcal{T} := \Pi_{l=1}^{k}\mathcal{T}_l$. Let Δ_l be the $(m-1)$-simplex in $\mathbb{R}^m$, for each $l \in \{1, ..., k\}$, defined by:

$$\Delta_l := \left\{ \Lambda_l := (\lambda_{li})_l \in [0,1]^m : \sum_{i=1}^{m} \lambda_{li} = 1 \right\}$$

and $\mathcal{T} := \Pi_{l=1}^{k}\ \mathcal{T}_l, \Delta := \Pi_{l=1}^{k}\Delta_l$.

Then the problem (4.4) can be rewritten as:

$$\min \left\{ F_1(Z, \Lambda) : Z \in \mathcal{T}, \Lambda \in \Delta \right\}. \tag{4.6}$$

Denote $f_l(z_{li}, \lambda_{li}) = \sum_{i=1}^{m} \lambda_{li}^{\beta}(z_{li} - x_{ji})^2$. Then

$$F_1(Z, \Lambda) = \sum_{j=1}^{n} \min_{\ell=1,...,k} f_l(z_{li}, \lambda_{li}). \tag{4.7}$$

We see that f_l can be decomposed as follows

$$f_l(z_{li}, \lambda_{li}) = g_l(z_{li}, \lambda_{li}) - h_l(z_{li}, \lambda_{li})$$

where $g_l(z_{li}, \lambda_{li}) = \sum_{i=1}^{m}(\frac{\rho_1}{2}z_{li}^2 + \frac{\rho_1}{2}\lambda_{li}^2)$ and $h_l(z_{li}, \lambda_{li}) = \sum_{i=1}^{m}\left[(\frac{\rho_1}{2}z_{li}^2 + \frac{\rho_1}{2}\lambda_{li}^2) - \lambda_{li}^{\beta}(z_{li} - x_{ji})^2\right]$.
On another hand, one has

$$\min_{\ell=1,...,k} f_l = \min_{\ell=1,...,k}(g_l - h_l) = \sum_{l=1}^{k} g_l - \max_{\ell=1,...,k}\left\{ \sum_{p=1,p\neq l}^{k} g_p + h_l \right\} \tag{4.8}$$

By applying the above formula to the objective function of (4.7), we obtain:

$$\begin{aligned} F_1(Z, \Lambda) &= \sum_{j=1}^{n}\sum_{l=1}^{k} g_l - \sum_{j=1}^{n} \max_{\ell=1,...,k}\left\{ \sum_{p=1,p\neq l}^{k} g_p + h_l \right\} \\ &= G_1(Z, \Lambda) - H_1(Z, \Lambda) \end{aligned} \tag{4.9}$$

where

$$G_1(Z, \Lambda) = \sum_{j=1}^{n}\sum_{l=1}^{k} g_l$$

and

$$H_1(Z,\Lambda)=\sum_{j=1}^{n}\max_{\ell=1,\ldots,k}\Big\{\sum_{p=1,p\neq l}^{k}\sum_{i=1}^{m}\frac{\rho_1}{2}(z_{pi}^2+\lambda_{pi}^2)+\sum_{i=1}^{m}\Big[\frac{\rho_1}{2}(z_{li}^2+\lambda_{li}^2)-\lambda_{li}^{\beta}(z_{li}-x_{ji})^2\Big]\Big\}.$$

Clearly, $G_1(Z,\Lambda)$ is a convex function. On another hand, $H_1(Z,\Lambda)$ is also convex according to following Proposition.

Proposition 4.1 *There exists $\rho_1 > 0$ such that the function H_1 is a convex function on $\{Z \in \mathcal{T}, \Lambda \in \Delta\}$.*

Proof: We consider the function $f_1 : \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ defined by:

$$f_1(v,y)=y^{\beta}(v-a)^2. \tag{4.10}$$

The Hessian of f_1 is given by:

$$J(v,y)=\begin{pmatrix} 2y^{\beta} & 2\beta(v-a)y^{\beta-1} \\ 2\beta(v-a)y^{\beta-1} & \beta(\beta-1)(v-a)^2y^{\beta-2}\end{pmatrix}. \tag{4.11}$$

We have for the determinant of $J(v,y)$ (with λ is an eigenvalue):

$$det\ |J(v,y)|=(2y^{\beta}-\lambda)[\beta(\beta-1)(v-a)^2y^{\beta-2}-\lambda]-[2\beta(v-a)y^{\beta-1}]^2\ .$$

Hence:

$$\lambda_{1,2}=\frac{1}{2}\Big\{[2y^{\beta}+\beta(\beta-1)(v-a)^2y^{\beta-2}]\pm sqrt(\triangle)\Big\} \tag{4.12}$$

where $\lambda_{1,2}$ are eigenvalues and

$$\triangle=[2y^{\beta}-\beta(\beta-1)(v-a)^2y^{\beta-2}]^2+4[2\beta(v-a)y^{\beta-1}]^2.$$

Hence, the function:

$$h_1(v,y)=\frac{\rho_1}{2}\left(v^2+y^2\right)-y^{\beta}(v-a)^2 \tag{4.13}$$

is convex on $\{v\in[\alpha,\sigma], y\in[0,1]\}$ if:

$$\rho_1\geq\frac{1}{2}\Big(2+\beta(\beta-1)\gamma^2+\sqrt{4+\beta^2(\beta-1)^2\gamma^4+12\beta^2\gamma^2+4\beta\gamma^2}\Big) \tag{4.14}$$

where $\gamma=\sigma-\alpha$; and $\beta\geq 2$.

As a consequence, for $v\leftarrow z_{li}, y\leftarrow\lambda_{li}$, the function

$$h_{li}(z_{li},\lambda_{li})=\frac{\rho_1}{2}\left(z_{li}^2+\lambda_{li}^2\right)-\lambda_{li}^{\beta}(z_{li}-x_{ji})^2 \tag{4.15}$$

is convex on $\{z_{li} \in [\alpha_i, \gamma_i], \lambda_{li} \in [0,1]\}$

Hence, the function $H_1(Z,\Lambda)$ is convex on $\{Z \in \mathcal{T}, \Lambda \in \Delta\}$. ■

DCA applied to (4.9)

According to the generic DCA scheme, we have to compute, at each iteration, $(\bar{Z}^r, \bar{\Lambda}^r) \in \partial H_1(Z^r, \Lambda^r)$ and then solve the convex program:

$$\min\left\{\sum_{j=1}^{n}\sum_{l=1}^{k}\sum_{i=1}^{m}\left(\frac{\rho_1}{2}z_{li}^2+\frac{\rho_1}{2}\lambda_{li}^2\right)-\langle(Z,\Lambda),(,\bar{Z}^r,\bar{\Lambda}^r)\rangle : Z\in\mathcal{T},\Lambda\in\Delta\right\}. \tag{4.16}$$

We have

$$H_1(Z,\Lambda)=\sum_{j=1}^{n}H_j(Z,\Lambda) \tag{4.17}$$

where $H_j(Z,\Lambda)=\max\limits_{\ell=1,\ldots,k} H_{jl}(z_{li},\lambda_{li})$ and

$$H_{jl}(z_{li},\lambda_{li})=\sum_{p=1,p\neq l}^{k}\sum_{i=1}^{m}\frac{\rho_1}{2}(z_{pi}^2+\lambda_{pi}^2)\ +\sum_{i=1}^{m}\left[\frac{\rho_1}{2}(z_{li}^2+\lambda_{li}^2)-\lambda_{li}^{\beta}(z_{li}-x_{ji})^2\right]. \tag{4.18}$$

Applying the usual rules in the calculations of subgradients of convex function, we get

$$\partial H_1(Z,\Lambda)=\sum_{j=1}^{n}\partial H_j(Z,\Lambda) \tag{4.19}$$

where (co denotes convex hull)

$$\partial H_j(Z,\Lambda)=co\{\partial H_{jl} : H_{jl}=H_j\}. \tag{4.20}$$

H_{jl} is differentiable and

$$\begin{aligned}
&\frac{\nabla H_{jl}}{\nabla z_{ri}}=\begin{cases}\rho_1 z_{ri}-2\lambda_{ri}^{\beta}(z_{ri}-x_{ji}) \text{ if } r=\ell\\ \rho_1 z_{ri} \text{ if } r\neq\ell\end{cases}\quad \forall r=1..k,\\
&\frac{\nabla H_{jl}}{\nabla \lambda_{ri}}=\begin{cases}\rho_1 \lambda_{ri}-\beta\lambda_{ri}^{\beta-1}(z_{ri}-x_{ji})^2 \text{ if } r=\ell\\ \rho_1 z_{ri} \text{ if } r\neq\ell\end{cases}\quad \forall r=1..k.
\end{aligned} \tag{4.21}$$

On another hand, the solution of the subproblem (4.16) is explicitly computed as ($Proj_D$ stands for the orthogonal projection on D):

$$\begin{aligned}
&(Z^{r+1})_{li}=\text{Proj}_{[\alpha_i,\gamma_i]}\left(\tfrac{1}{n\rho_1}(\bar{Z}^r)_{li}\right)\ \ l=1,..,k,\ \ i=1,...m;\\
&(\Lambda^{r+1})_{l}=\text{Proj}_{\Delta_l}\left(\tfrac{1}{n\rho_1}(\bar{\Lambda}^r)_{l}\right)\ \ l=1,...k.
\end{aligned} \tag{4.22}$$

Note that the projection of points onto a rectangle is explicit while there exists many efficient methods for computing the projection of points onto a simplex (Júdice et al. [2008]).

The algorithm can be described as follows.

BIWF-DCA: DCA applied to (4.9)

- **Initialization:** Let $\epsilon > 0$ be sufficiently small and (Z^0, Λ^0) be given, $r = 0$.
- **Repeat**
  - Compute $(\bar{Z}^r, \bar{\Lambda}^r)$ via (4.17)-(4.21).
  - Compute (Z^{r+1}, Λ^{r+1}) via (4.22).
  - $r = r + 1$
- **Until** $\|(Z^{r+1}, \Lambda^{r+1}) - (Z^r, \Lambda^r)\| \leq \epsilon$ or $|F_1(Z^{r+1}, \Lambda^{r+1}) - F_1(Z^r, \Lambda^r)| \leq \epsilon$.

4.1.2.2 DCA for solving the mix-integer problem (4.5)

In this section, we deal with the mix-integer programming problem (4.5). Since $w_{jl} \in \{0, 1\}$ we can replace w_{jl} by w_{jl}^2 and rewrite the objective function of (4.5) by $F_2(W, Z, \Lambda) := \sum_{l=1}^{k} \sum_{j=1}^{n} \sum_{i=1}^{m} w_{jl}^2 \lambda_{li}^{\beta} (z_{li} - x_{ji})^2$. In the problem (4.5) the variables W and Λ are a priori bounded. By the same way as in Section 4.1.2.1, the variable Z can be bounded. Let $\mathcal{C}_j$ be the $(k-1)$-simplex in $\mathbb{R}^k$, for each $j \in \{1, ..., n\}$, defined by:

$$\mathcal{C}_j := \left\{ W_j := (w_{jl})_j \in [0, 1]^k : \sum_{l=1}^{k} w_{jl} = 1 \right\}$$

and $\mathcal{C} := \Pi_{j=1}^{n} \mathcal{C}_j, \mathcal{T} := \Pi_{l=1}^{k} \mathcal{T}_l, \Delta := \Pi_{l=1}^{k} \Delta_l$.

The problem (4.5) can be rewritten as:

$$\min \left\{ F_2(W, Z, \Lambda) : W \in \mathcal{C} \cap \{0, 1\}^{n \times k}, Z \in \mathcal{T}, \Lambda \in \Delta \right\}. \tag{4.23}$$

A continuous reformulation

Our reformulation technique is based on the following new results developed in Le Thi et al. [2012a]. We first show that $F_2(W, Z, \Lambda)$ is a DC function. Clearly, $F_2(W, Z, \Lambda)$ can be reformulated as:

$$F_2(W, Z, \Lambda) = G_2(W, Z, \Lambda) - H_2(W, Z, \Lambda) \tag{4.24}$$

where

$$\begin{aligned} G_2(W, Z, \Lambda) &:= \sum_{l=1}^{k} \sum_{j=1}^{n} \sum_{i=1}^{m} \frac{\rho_2}{2} \left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2 \right), \\ H_2(W, Z, \Lambda) &:= \sum_{l=1}^{k} \sum_{j=1}^{n} \sum_{i=1}^{m} \left[\frac{\rho_2}{2} \left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2 \right) - w_{jl}^2 \lambda_{li}^{\beta} (z_{li} - x_{ji})^2 \right]. \end{aligned} \tag{4.25}$$

It is easy to see that $G_2(W, Z, \Lambda)$ is a convex function. $H_2(W, Z, \Lambda)$ also a convex function by following proposition.

Proposition 4.2 *There exists $\rho_2 > 0$ such that the function $H_2(W, Z, \Lambda)$ is a convex function on $\{W \in \mathcal{C}, Z \in \mathcal{T}, \Lambda \in \Delta\}$.*

Proof: First, we consider the function $f_2 : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ defined by:

$$f_2(u, v, y) = u^2 y^\beta (v - a)^2. \tag{4.26}$$

The Hessian of f_2 is given by:

$$J(u, v, y) = \begin{pmatrix} 2y^\beta (v-a)^2 & 4uy^\beta (v-a) & 2\beta u y^{\beta-1}(v-a)^2 \\ 4uy^\beta (v-a) & 2u^2 y^\beta & 2\beta u^2 y^{\beta-1}(v-a) \\ 2u\beta y^{\beta-1}(v-a)^2 & 2\beta u^2 y^{\beta-1}(v-a) & \beta(\beta-1)u^2 y^{\beta-2}(v-a)^2 \end{pmatrix}.$$

The determinant $|J(u, v, y)|_1$ of $J(u, v, y)$ is defined by:

$$\begin{aligned} |J(u, v, \quad y)|_1 = \max \Big\{ & \quad 2y^\beta (v-a)^2 + 4uy^\beta (v-a) + 2\beta u y^{\beta-1}(v-a)^2; \\ & 4uy^\beta (v-a) + 2u^2 y^\beta + 2\beta u^2 y^{\beta-1}(v-a); \\ & 2u\beta y^{\beta-1}(v-a)^2 + 2\beta u^2 y^{\beta-1}(v-a) + \beta(\beta-1)u^2 y^{\beta-2}(v-a)^2 \Big\}. \end{aligned} \tag{4.27}$$

For all $(u, v, y) : u \in \{0, 1\}, v \in [\alpha, \sigma], y \in [0, 1], \beta \geq 2$, we have:

$$\begin{aligned} |J(u, v, y)|_1 < \rho_2 : &= \max\{2\gamma^2 + 4\gamma + 2\beta\gamma^2; 4\gamma + 2 + 2\beta\gamma; \\ &\qquad 2\beta\gamma^2 + 2\beta\gamma + \beta(\beta-1)\gamma^2\} \\ &= \max\left\{4\gamma + 2(\beta+1)\gamma^2; 2 + 2(\beta+2)\gamma; 2\beta\gamma + \beta(\beta+1)\gamma^2\right\} \end{aligned}$$

where $\gamma = \sigma - \alpha$.

As a consequence, with ρ_2 defined above, the function:

$$h_2(u, v, y) = \frac{\rho_2}{2}\left(u^2 + v^2 + y^2\right) - u^2 y^\beta (v - a)^2 \tag{4.28}$$

is convex on $\{u \in \{0, 1\}, v \in [\alpha, \sigma], y \in [0, 1]\}$.

Hence, for $u \leftarrow w_{jl}, v \leftarrow z_{li}, y \leftarrow \lambda_{li}$, the function:

$$h_{lij}(w_{jl}, z_{li}, \lambda_{li}) = \frac{\rho_2}{2}\left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right) - w_{jl}^2 \lambda_{li}^\beta (z_{li} - x_{ji})^2 \tag{4.29}$$

is convex on $\{w_{jl} \in [0, 1], z_{li} \in [\alpha_i, \gamma_i], \lambda_{li} \in [0, 1]\}$.

As a result, the function $H_2(W, Z, \Lambda)$ is convex on $\{W \in \mathcal{C}, Z \in \mathcal{T}, \Lambda \in \Delta\}$. ■

Then $F_2(W, Z, \Lambda)$ is a DC function with DC decomposition (4.24).

We will now reformulate (4.24) as a continuous optimization problem thanks to an exact penalty technique (Le Thi et al. [2012a]). For reader's convenience, we will give bellow a brief description of the theorem.

Theorem 1 *(Le Thi et al. [2012a]) Let K be a nonempty bounded polyhedral convex set, f be a DC function on K and p be a nonnegative concave function on K. Then there exists $t_0 \geq 0$ such that for all $t > t_0$ the following problems have the same optimal value and the same solution set:*

$$\begin{array}{rrcl} (P) & \gamma & = & \inf\{f(x) : x \in K, p(x) \leq 0\} \\ (P_t) & \gamma(t) & = & \inf\{f(x) + tp(x) : x \in K\}. \end{array}$$

Let us consider the function p defined on $\mathbb{R}^{n\times k}$ by:

$$p(W) := \sum_{j=1}^{n}\sum_{l=1}^{k} w_{jl}(1 - w_{jl}).$$

Clearly, p is finite concave on $\mathbb{R}^{n\times k}$, nonnegative on $\mathcal{C}$, and

$$\mathcal{C} \cap \{0,1\}^{n\times k} = \{W \in \mathcal{C} :\ p(W) = 0\} = \{W \in \mathcal{C} :\ p(W) \leq 0\}.$$

By using Theorem 1, we obtain the following problem which is equivalent to problem (4.5):

$$\min\left\{\bar{F}_2(W, Z, \Lambda) := F_2(W, Z, \Lambda) + tp(W) : W \in \mathcal{C}, Z \in \mathcal{T}, \Lambda \in \Delta\right\}, \tag{4.30}$$

where $t > t_0$ is called penalty parameter.

We will now develop DC programming and DCA for solving (4.30). Remark that, if F_2 is a DC function with DC components G_2 and H_2 then the function $\bar{F}_2(W, Z, \Lambda)$ is also DC:

$$\bar{F}_2(W, Z, \Lambda) \ := G_2(W, Z, \Lambda) - \bar{H}_2(W, Z, \Lambda) \tag{4.31}$$

where $\bar{H}_2(W, Z, \Lambda) = H_2(W, Z, \Lambda) - tp(W)$.

DCA applied to (4.31) For designing a DCA applied to (4.31), we first need to compute $(\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r) \in \partial\bar{H}_2(W^r, Z^r, \Lambda^r)$ and then solve the following convex program:

$$\min\left\{\frac{\rho_2}{2}\sum_{l=1}^{k}\sum_{j=1}^{n}\sum_{i=1}^{m}\left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right) - \langle(W, Z, \Lambda), (\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r)\rangle :\right.$$
$$\left. W \in \mathcal{C}, Z \in \mathcal{T}, \Lambda \in \Delta\right\}. \tag{4.32}$$

The function $\bar{H}_2$ is differentiable and its gradient at the point (W^r, Z^r, Λ^r) is given by:

$$\begin{aligned} \bar{W}^r &= \nabla_W\bar{H}_2(W, Z, \Lambda) = \left(m\rho_2 w_{jl} - \sum_{i=1}^{m} 2w_{jl}\lambda_{li}^{\beta}(z_{li} - x_{ji})^2 + t(2w_{jl} - 1)\right)_{j=1..n}^{l=1..k}, \\ \bar{Z}^r &= \nabla_Z\bar{H}_2(W, Z, \Lambda) = \left(n\rho_2 z_{li} - \sum_{j=1}^{n} 2w_{jl}^2\lambda_{li}^{\beta}(z_{li} - x_{ji})\right)_{l=1..k}^{i=1..m}, \\ \bar{\Lambda}^r &= \nabla_\Lambda\bar{H}_2(W, Z, \Lambda) = \left(n\rho_2\lambda_{li} - \sum_{j=1}^{n}\beta w_{jl}^2\lambda_{li}^{\beta-1}(z_{li} - x_{ji})^2\right)_{l=1..k}^{i=1..m}. \end{aligned} \tag{4.33}$$

Furthermore, the solution of the subproblem (4.45) is explicitly computed as:

$$\begin{aligned}
(W^{r+1})_j &= \text{Proj}_{\mathcal{C}_j}\left(\tfrac{1}{m\rho_2}(\bar{W}^r)_j\right) \quad j = 1, ...n;\\
(Z^{r+1})_{li} &= \text{Proj}_{[\alpha_i,\gamma_i]}\left(\tfrac{1}{n\rho_2}(\bar{Z}^r)_{li}\right) \quad l = 1, .., k, i = 1, ...m;\\
(\Lambda^{r+1})_l &= \text{Proj}_{\Delta_l}\left(\tfrac{1}{n\rho_2}(\bar{\Lambda}^r)_l\right) \quad l = 1, ...k.
\end{aligned} \tag{4.34}$$

Finally, DCA scheme applied to (4.31) can be described as follows:

MIWF-DCA: DCA applied to (4.31)
- **Initialization:** Let $\epsilon > 0$ be sufficiently small and (W^0, Z^0, Λ^0) be given, $r = 0$.
- **Repeat**
  - Compute $(\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r)$ via (4.33).
  - Compute $(W^{r+1}, Z^{r+1}, \Lambda^{r+1})$ via (4.34).
  - $r = r + 1$
- **Until** $\|(W^{r+1}, Z^{r+1}, \Lambda^{r+1}) - (W^r, Z^r, \Lambda^r)\| \leq \epsilon$ or $|\bar{F}_2(W^{r+1}, Z^{r+1}, \Lambda^{r+1}) - \bar{F}_2(W^r, Z^r, \Lambda^r)| \leq \epsilon$.

4.1.2.3 Finding a good starting points for DCAs

Finding a good starting point is an important question while designing DCA schemes. The research of such a point depends on the structure of the problem being considered. As proposed in Le Thi et al. [2007a], we use an alternative KMeans - DCA procedure for finding a starting point. The procedure is described as follows.

KM - DCA procedure
- **Initialization:** Choose Z^0. Let $max\_iter > 0$ be a given integer. Set $s = 0$.
- **Repeat**
  - Perform one iteration of DCA-KM from Z^s.
  - Perform one iteration of KMeans from the solution given by DCA-KM to obtain Z^{s+1}.
  - $s = s + 1$
- **Until** $s = max\_iter$

In our experiments, we observed that $max\_iter = 2$ is sufficient to find a good initial points.

4.1.3 Numerical experiments

4.1.3.1 Datasets

Numerical experiments were performed on 11 real-world datasets: *Stalog Shuttle, Wave form, Breast Cancer Wiscosin, Ecoli, Column_3C, Magic, Breast Tissue and Madelon* taken from UCI Machine Learning Repository (Newman and Merz), *SVM guide1* taken from LibSVM-Dataset Repository (Chang and Lin [2011]), *ItalyPowerDemand* taken from KENT-Dataset

Repository (Keogh et al. [2006]) and *Mamographic* taken from KEEL-Dataset Repository (Alcalá-Fdez et al. [2011]). The information about datasets is summarized in Table 4.1.

Table 4.1: Datasets

Dataset	No. Points	No. Attributes	No. Clusters
Stalog Shuttle	14500	9	7
Wave form	5000	40	3
Breast Cancer Wiscosin	683	9	2
Ecoli	336	7	8
Column_3C	310	6	3
Magic	19020	3	2
Breast Tissue	106	9	6
Madelon	600	500	2
Svmguide1	4000	4	2
Italy Power Demand	1029	24	2
Mamographic	830	5	2

4.1.3.2 Set up experiments and Parameters

The following criteria were used to compare the performances of algorithms: the percentage of well classified points (PWCO), Rand Index value and the CPU running time.

The Rand index (named after William M. Rand), simply measures the number of pairwise agreements. Let's denote, for every instance x_i, its initial class by $I_{ref}(x_i)$ and its cluster obtained from the clustering algorithm by $I_{class}(x_i)$. The Rand index is defined by:

$$RandI = \frac{a+d}{a+b+c+d} \tag{4.35}$$

where

$$\begin{array}{rl} a = & \mid \{i,j \mid I_{ref}(x_i) = I_{ref}(x_j) \quad \& \quad I_{class}(x_i) = I_{class}(x_j)\} \mid, \\ b = & \mid \{i,j \mid I_{ref}(x_i) = I_{ref}(x_j) \quad \& \quad I_{class}(x_i) \neq I_{class}(x_j)\} \mid, \\ c = & \mid \{i,j \mid I_{ref}(x_i) \neq I_{ref}(x_j) \quad \& \quad I_{class}(x_i) = I_{class}(x_j)\} \mid, \\ d = & \mid \{i,j \mid I_{ref}(x_i) \neq I_{ref}(x_j) \quad \& \quad I_{class}(x_i) \neq I_{class}(x_j)\} \mid . \end{array} \tag{4.36}$$

All algorithms clustering was implemented in the Visual C++ 2008, and performed on a PC Intel i5 CPU650, 3.2 GHz of 4Gb RAM.

We suggested using the following set of candidate values in our experiments $[1.1, \ldots, 5.0]$ for WF-KM and $[2.0, \ldots, 5.0]$ for BIWF-DCA, MIWF-DCA respective.

For every data instance, we perform each algorithm 10 times from 10 same random starting points and report the mean and the standard deviation of each criterion. Bold values in result tables are best value for each data instance.

Experiment 1

In the first experiment we are interested in the effect of feature weighting in classification task. For this purpose, we perform our two algorithms **BIWF-DCA**, **MIWF-DCA** and **KM-DCA** - the DCA for bilevel formulation of MSSC (Le Thi et al. [2007a]). We report in Table 4.2 the mean and standard deviation of PWCO. The comparative results of Rand Index(resp. CPU Time) are reported in Table 4.3 (resp. Table 4.4).

Table 4.2: Comparison of PWCO between **DCA-KM**, **BIWF-DCA** and **MIWF-DCA**

Data	PWCO		
	DCA-KM	BIWF-DCA	MIWF-DCA
Stalog Shuttle	43.987%±5.143%	62.884%±7.093%	**79.166%±0.000%**
Wave form	48.116%±15.855%	**64.514%±7.118%**	**64.514%±7.118%**
Breast Cancer Wiscosin	96.032%±0.044%	96.076%±0.471%	**96.354%±0.406%**
Ecoli	57.798%±8.311%	**61.012%±9.658%**	**61.012%±9.658%**
Column_3C	59.613%±5.652%	**72.032%±5.107%**	71.677%±4.921%
Magic	64.516%±0.580%	65.419%±2.581%	**65.991%±7.136%**
Breast Tissue	54.906%±4.401%	56.604%±5.236%	**57.547%±5.660%**
Madelon	54.433%±4.096%	55.867%±3.900%	**57.033%±2.845%**
Svmguide1	75.073%±0.013%	**87.355%±6.062%**	86.818%±6.427%
Italy Power Demand	51.448%±0.078%	**82.187%±15.543%**	**82.187%±15.543%**
Mamographic	68.687%±0.266%	68.831%±0.607%	**79.880%±0.258%**
Average	61.328%±4.040%	70.253%±5.761%	**72.925%±5.452%**

From the numerical results, we observe that:

- **BIWF-DCA** and **MIWF-DCA** give better PWCO than **KM-DCA** on all datasets. The gain **BIWF-DCA** (resp. **MIWF-DCA**) over **KM-DCA** is more than 10% for 5 (resp. 6) our of 11 datasets. The gain can go up to 30.74% (*Italy Power Demand* dataset) with **BIWF-DCA** and 35.18% (*Stalog Shuttle* dataset) with **MIWF-DCA**. The average of PWCO of **BIWF-DCA** is 70.25% and that of **MIWF-DCA** is 72.93% which a much more better than **KM-DCA** (61, 33%).
- The quality of our two algorithms **BIWF-DCA** and **MIWF-DCA** are comparable. However, **MIWF-DCA** furnishes better PWCO than **BIWF-DCA** with a big gain on *Stalog Shuttle* (16.29%) and *Mamographic* (11.05%).
- Concerning the Rand index criterion, except for *Stalog Shuttle* where **KM-DCA** is better than **MIWF-DCA**, **BIWF-DCA** and **MIWF-DCA** alway furnish better result.
- Undoubtedly **DCA-KM** is fastest algorithm out of three and **MIWF-DCA** is the most time consuming, especially *Magic* datasets where **DCA-KM** is somehow 44 and 116 times faster than **MIWF-DCA**. Except for *Ecoli*, **DCA-KM** is faster than **BIWF-DCA**, the gain varies from 1.2 times to 9, 8 times.

Table 4.3: Comparison of Rand index between **DCA-KM**, **BIWF-DCA** and **MIWF-DCA**

Data	Rand Index		
	DCA-KM	BIWF-DCA	MIWF-DCA
Stalog Shuttle	0.525±0.037	**0.617±0.072**	0.515±0.058
Wave form	0.686±0.038	**0.688±0.029**	**0.688±0.029**
Breast Cancer Wiscosin	0.924±0.001	0.925±0.009	**0.930±0.007**
Ecoli	0.821±0.027	**0.823±0.043**	**0.823±0.043**
Column_3C	0.669±0.009	**0.731±0.019**	0.726±0.017
Magic	0.542±0.003	0.549±0.015	**0.561±0.042**
Breast Tissue	0.811±0.007	0.813±0.020	**0.822±0.015**
Madelon	0.506±0.009	0.509±0.009	**0.510±0.010**
Svmguide1	0.626±0.001	**0.786±0.083**	0.779±0.088
Italy Power Demand	0.500±0.000	**0.755±0.141**	**0.755±0.141**
Mamographic	0.569±0.002	0.570±0.004	**0.678±0.003**
Average	0.653±0.012	0.706±0.040	**0.708±0.041**

Table 4.4: Comparison of CPU Time between **DCA-KM**, **BIWF-DCA** and **MIWF-DCA**

Data	Running time		
	DCA-KM	BIWF-DCA	MIWF-DCA
Stalog Shuttle	**0.984±0.008**	8.539±1.323	43.972±2.808
Wave form	**0.401±0.029**	0.689±0.021	1.441±0.036
Breast Cancer Wiscosin	**0.016±0.007**	0.053±0.072	0.506±0.038
Ecoli	0.030±0.008	**0.022±0.007**	0.045±0.008
Column_3C	**0.010±0.002**	0.089±0.006	0.242±0.017
Magic	**0.539±0.143**	4.944±1.093	61.779±2.087
Breast Tissue	**0.010±0.001**	0.088±0.009	0.206±0.026
Madelon	**0.404±0.048**	0.788±0.019	1.153±0.028
Svmguide1	**0.094±0.012**	0.490±0.081	0.268±0.019
Italy Power Demand	**0.039±0.020**	0.072±0.008	0.131±0.016
Mamographic	**0.013±0.008**	0.016±0.003	0.573±0.107
Average	**0.231±0.026**	1.435±0.240	10.029±0.472

From the above observes, we can conclude that using weighted dissimilarity measure allowed us to improve greatly the performance classifier in term of quality of classification.

Experiment 2

In the second experiment, we compare the performance of 3 algorithm for MSSC using weighted dissimilarity measure: our algorithms **BIWF-DCA**, **MIWF-DCA** and **WF-KM** (Chan et al. [2004]). We also reported the PWCO, Rand Index and CPU Time of each algorithm.

Table 4.5: Comparison of PWCO between **WF-KM**, **BIWF-DCA** and **MIWF-DCA**

Data	PWCO		
	WF-KM	BIWF-DCA	MIWF-DCA
Stalog Shuttle	55.737%±0.544%	62.884%±7.093%	**79.166%±0.000%**
Wave form	50.420%±3.410%	**64.514%±7.118%**	**64.514%±7.118%**
Breast Cancer Wiscosin	92.943%±4.774%	96.076%±0.471%	**96.354%±0.406%**
Ecoli	43.036%±0.583%	**61.012%±9.658%**	**61.012%±9.658%**
Column_3C	51.452%±7.795%	**72.032%±5.107%**	71.677%±4.921%
Magic	55.499%±0.000%	65.419%±2.581%	**65.991%±7.136%**
Breast Tissue	50.849%±5.524%	56.604%±5.236%	**57.547%±5.660%**
Madelon	54.167%±2.337%	55.867%±3.900%	**57.033%±2.845%**
Svmguide1	58.550%±0.150%	**87.355%±6.062%**	86.818%±6.427%
Italy Power Demand	59.038%±10.035%	**82.187%±15.543%**	**82.187%±15.543%**
Mamographic	52.205%±9.867%	68.831%±0.607%	**79.880%±0.258%**
Average	56.718%±4.093%	70.253%±5.761%	**72.925%±5.452%**

Table 4.6: Comparison of Rand index between **WF-KM**, **BIWF-DCA** and **MIWF-DCA**

Data	Rand Index		
	WF-KM	BIWF-DCA	MIWF-DCA
Stalog Shuttle	0.499±0.003	**0.617±0.072**	0.515±0.058
Wave form	0.624±0.024	**0.688±0.029**	**0.688±0.029**
Breast Cancer Wiscosin	0.873±0.070	0.925±0.009	**0.930±0.007**
Ecoli	0.296±0.023	**0.823±0.043**	**0.823±0.043**
Column_3C	0.648±0.021	**0.731±0.019**	0.726±0.017
Magic	0.506±0.000	0.549±0.015	**0.561±0.042**
Breast Tissue	0.783±0.023	0.813±0.020	**0.822±0.015**
Madelon	0.504±0.005	0.509±0.009	**0.510±0.010**
Svmguide1	0.517±0.006	**0.786±0.083**	0.779±0.088
Italy Power Demand	0.536±0.087	**0.755±0.141**	**0.755±0.141**
Mamographic	0.520±0.061	0.570±0.004	**0.678±0.003**
Average	0.573±0.029	0.706±0.040	**0.708±0.041**

Table 4.7: Comparison of CPU time between **WF-KM**, **BIWF-DCA** and **MIWF-DCA**

Data	Running time		
	WF-KM	BIWF-DCA	MIWF-DCA
Stalog Shuttle	**5.023±2.635**	8.539±1.323	43.972±2.808
Wave form	11.623±3.053	**0.689±0.021**	1.441±0.036
Breast Cancer Wiscosin	0.055±0.017	**0.053±0.072**	0.506±0.038
Ecoli	0.024±0.008	**0.022±0.007**	0.045±0.008
Column_3C	**0.056±0.020**	0.089±0.006	0.242±0.017
Magic	**3.901±0.719**	4.944±1.093	61.779±2.087
Breast Tissue	**0.040±0.014**	0.088±0.009	0.206±0.026
Madelon	4.245±1.321	**0.788±0.019**	1.153±0.028
Svmguide1	**0.044±0.006**	0.490±0.081	0.268±0.019
Italy Power Demand	0.362±0.155	**0.072±0.008**	0.131±0.016
Mamographic	0.017±0.003	**0.016±0.003**	0.573±0.107
Average	2.308±0.723	**1.435±0.240**	10.029±0.472

We observe that, in all datasets, our algorithms give better solutions than **WF-KM**. The gain can go up to 28.81% (*Svmguide1* dataset) with **BIWF-DCA** and 28, 27% with **MIWF-DCA**. **BIWF-DCA** is faster than **WF-KM** for 6 out of 11 datasets while **MIWF-DCA** is the slowest algorithm.

In Figure 4.1 and Figure 4.2, we summarize the results of Experiment 1 and Experiment 2 for all 4 algorithms.

4.1.4 Conclusion

We have studied the DC programming and DCA on two widely used models of MSSC using weighted features. Two optimization models were recast as a DC program, one of which is based on the reformulation technique and exact penalty in DC programming. It fortunately turns out that the corresponding DCA consists in computing, at each iteration, the projection of points onto a simplex and/or a rectangle, that all are given in the explicit form. From numerical experiments, we can say that the introduction of weighted feature allows to improve the performance of classification task. Furthermore, computational experiments show the superiority in term of quality of solution of our algorithms with respect to the standard algorithm in feature weighting.

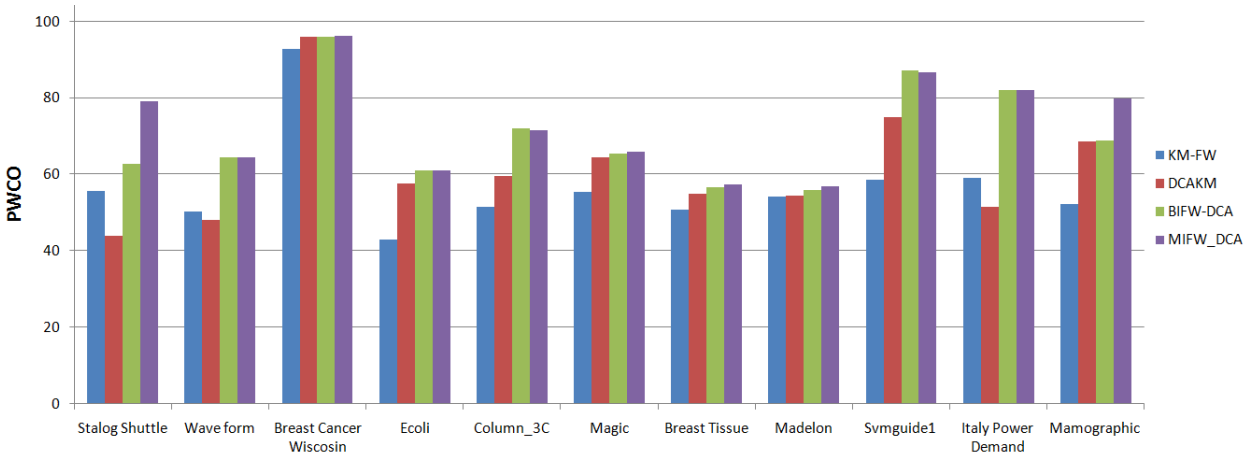


Figure 4.1: Comparative results PWCO of **FW-KM**, **DCA-KM**, **BIFW-DCA** and **MIFW-DCA**

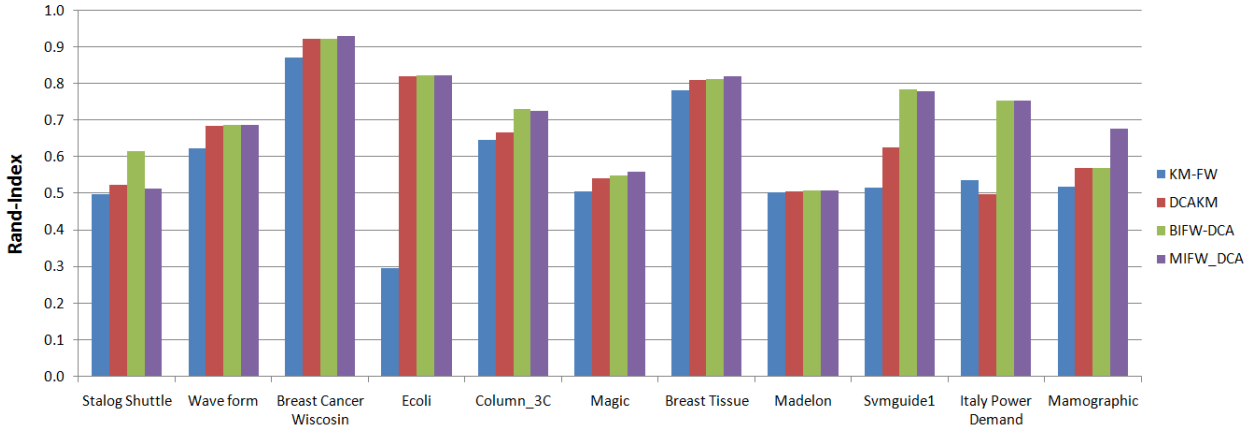


Figure 4.2: Comparative results Rand-Index of **FW-KM**, **DCA-KM**, **BIFW-DCA** and **MIFW-DCA**

4.2 Feature weighted fuzzy clustering and its application on image segmentation [1]

In this section, we study a weighted feature fuzzy clustering model and a DCA based algorithm to solve it. We then present an application of weighted feature fuzzy clustering for image segmentation. Firstly, we reformulate the weighted feature fuzzy clustering model as a DC program. Then, a DCA scheme is developed to solve the resulting problem. Experimen-

1. A part of this section is published under the title:
Hoai Minh Le, Bich Thuy Nguyen Thi, Minh Thuy Ta, Hoai An Le Thi. *Image Segmentation via Feature Weighted Fuzzy Clustering by a DCA Based Algorithm.* Advanced Computational Methods for Knowledge Engineering, Studies in Computational Intelligence. Volume 479, Springer, ISSN: 1860–949X (Print) 1860–9503 (Online), pp 53–63 (2013).

tal results on real color images have illustrated the effectiveness of the proposed algorithm and its superiority with respect to the standard weighted feature fuzzy clustering algorithm in quality of solutions.

4.2.1 Introduction

In general the clustering algorithms can be divided into two categories: hard clustering and soft (fuzzy) clustering. In hard clustering, the data elements are divided into distinct clusters, where each data belongs to exactly one cluster. In contrary, one data element is assigned a membership level with each cluster, in fuzzy clustering. Fuzzy C-Means (FCM) clustering, introduced by Bezdek in 1981 (Bezdek [1981]), is a most widely used fuzzy clustering method.

In the section 4.1, we have seen the advantages of using features weighting in hard clustering model. In this section, we discover the power of using features weighting in the fuzzy clustering model, and apply it in the image segmentation problem.

The problem FCM using features weighting is investigated in the research of Frigui and Nasraoui [2004]. It can be stated as follows. Let $\mathcal{X} := \{x_1, x_2, ..., x_n\}$ be a data set of n entities with m attributes and the known number of clusters k $(2 \leq k \leq n)$. Denote by Λ a $k \times m$ matrix defined as $\Lambda = (\lambda_{l,i})$ where $\lambda_{l,i}$ defines the relevance of i-th feature to the cluster C_l. $W = (w_{j,l}) \in \mathbb{R}^{n \times k}$ with $j = 1, \ldots, n$ and $l = 1, \ldots, k$ called the *fuzzy partition* matrix in which each element $w_{j,l}$ indicates the membership degree of each point x_j in the cluster C_l (the probability that a point x_j belongs to the cluster C_l).

We are to regrouping the set $\mathcal{X}$ into k clusters in order to minimize the sum of squared distances from the entities to the centroid of their cluster. The dissimilarity measure includes m weighted attributes. Then a straightforward formulation of the clustering using weighted dissimilarity measures is (μ, β are exponents greater than 1):

$$\begin{cases} \min F(W, Z, \Lambda) := \sum\limits_{l=1}^{k} \sum\limits_{j=1}^{n} \sum\limits_{i=1}^{m} w_{jl}^{\mu} \lambda_{li}^{\beta} (z_{li} - x_{ji})^2 \\ s.t : \sum\limits_{l=1}^{k} w_{jl} = 1, j = 1..n, \\ \quad \sum\limits_{i=1}^{m} \lambda_{li} = 1, l = 1..k, \\ \quad w_{jl} \in [0, 1], j = 1..n, l = 1..k, \\ \quad \lambda_{li} \in [0, 1], l = 1..k, i = 1..m. \end{cases} \tag{4.37}$$

Problem (4.37) is difficult due to the nonconvexity of the objective function. Moreover, in real applications this is a very large scale problem (high dimension and large data set, i.e. m and n are very large), that is why global optimization approaches such as Branch & Bound, Cutting plane algorithms etc. cannot be used. In Frigui and Nasraoui [2004], the authors proposed a FCM type algorithm, called **SCAD** (Simultaneous Clustering and Attribute Discrimination), to solve the problem (4.37). At first, **SCAD** fixes Z, Λ and finds

W to minimize $F(W,.,.)$. Then W, Λ are fixed for finding Z minimizing $F(.,Z,.)$. Finally, Λ is obtained by minimizing $F(.,.,\Lambda)$ with W and Z fixed. The process is repeated until no more improvement in the objective function can be made.

In this section, we investigate DC Programming and DCA for solving the problem (4.37).

4.2.2 A DC formulation of the problem (4.37)

In a similar way as the previous section, we find the bounds of variables W, Λ and Z. Let $\alpha_i := \min_{j=1,\ldots,n} x_{j,i}$, $\gamma_i := \max_{j=1,\ldots,n} x_{j,i}$. Hence $z_l \in \mathcal{T}_l := \Pi_{i=1}^{m} [\alpha_i, \gamma_i]$ for all $l = 1, ..., k$, and $Z \in \mathcal{T} := \Pi_{l=1}^{k} \mathcal{T}_l$.

Let Δ_l (resp. $\mathcal{C}_j$) be the $(m-1)$-simplex in $\mathbb{R}^m$(resp. $(k-1)$-simplex in $\mathbb{R}^k$), for each $l \in \{1, ..., k\}$ (resp. for each $j \in \{1, ..., n\}$), defined by:

$$\Delta_l := \left\{ \Lambda_l := (\lambda_{l,i})_l \in [0,1]^m : \sum_{i=1}^{m} \lambda_{l,i} = 1 \right\}; \quad \mathcal{C}_j := \left\{ W_j := (w_{j,l})_j \in [0,1]^k : \sum_{l=1}^{k} w_{j,l} = 1 \right\},$$

and $\mathcal{C} := \Pi_{j=1}^{n} \mathcal{C}_j, \mathcal{T} := \Pi_{l=1}^{k} \mathcal{T}_l, \Delta := \Pi_{l=1}^{k} \Delta_l$.

The problem (4.37) can be rewritten as:

$$\min \{F(W, Z, \Lambda) : (W, Z, \Lambda) \in (\mathcal{C} \times \mathcal{T} \times \Delta)\}. \tag{4.38}$$

Our DC decomposition of F is based on the following result.

Proposition 4.3 *There exists $\rho > 0$ such that the function*

$$h(u, v, y) := \frac{\rho}{2}\left(u^2 + v^2 + y^2\right) - u^{\mu} y^{\beta} (v - a)^2$$

is convex on $(u, v, y) \in [0, 1] \times [\alpha, \gamma] \times [0, 1]$.

Proof: The proof of this proposition is similar to the one of 4.2. Hence we state below the main arguments without details.

Let $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be the function defined by:

$$f(u, v, y) = u^{\mu} y^{\beta} (v - a)^2. \tag{4.39}$$

Then the Hessian of f is computed as:

$$\begin{pmatrix} \mu(\mu-1)u^{\mu-2}y^{\beta}(v-a)^2 & 2\mu u^{\mu-1}y^{\beta}(v-a) & \mu u^{\mu-1}\beta y^{\beta-1}(v-a)^2 \\ 2\mu u^{\mu-1}y^{\beta}(v-a) & 2u^{\mu}y^{\beta} & 2\beta u^{\mu}y^{\beta-1}(v-a) \\ \beta y^{\beta-1}\mu u^{\mu-1}(v-a)^2 & 2\beta u^{\mu}y^{\beta-1}(v-a) & \beta(\beta-1)u^{\mu}y^{\beta-2}(v-a)^2 \end{pmatrix}. \tag{4.40}$$

Hence, with

$$\rho := \max\{\mu(\mu-1)\delta^2 + 2\mu\delta + \beta\mu\delta^2; 2\mu\delta + 2 + 2\beta\delta; \beta\mu\delta^2 + 2\beta\delta + \beta(\beta-1)\delta^2\} \tag{4.41}$$

where $\delta = \gamma - \alpha$, $\mu \geq 2, \beta \geq 2$, the function h is convex. ■

As a result, the function $H(W, Z, \Lambda)$ defined by:

$$H(W, Z, \Lambda) := \sum_{l=1}^{k}\sum_{j=1}^{n}\sum_{i=1}^{m}\left[\frac{\rho}{2}\left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right) - w_{jl}^{\mu}\lambda_{li}^{\beta}(z_{li} - x_{ji})^2\right] \tag{4.42}$$

is convex on $(\mathcal{C} \times \mathcal{T} \times \Delta)$.

Finally, we can express F as follows:

$$F(W, Z, \Lambda) := G(W, Z, \Lambda) - H(W, Z, \Lambda), \tag{4.43}$$

where

$$G(W, Z, \Lambda) := \frac{\rho}{2}\sum_{l=1}^{k}\sum_{j=1}^{n}\sum_{i=1}^{m}\left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right);$$

and $H(W, Z, \Lambda)$ as (4.42) are clearly convex functions. Therefore, we get the following DC formulation of (4.37):

$$\min\left\{F(W, Z, \Lambda) := G(W, Z, \Lambda) - H(W, Z, \Lambda) : (W, Z, \Lambda) \in (\mathcal{C} \times \mathcal{T} \times \Delta)\right\}. \tag{4.44}$$

4.2.3 DCA applied to (4.44)

For designing a DCA scheme applied to (4.44), we first need to compute $(\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r) \in \partial H(W^r, Z^r, \Lambda^r)$ and then solve the convex program

$$\min\left\{\frac{\rho}{2}\sum_{l=1}^{k}\sum_{j=1}^{n}\sum_{i=1}^{m}\left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right) - \langle(W, Z, \Lambda), (\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r)\rangle : (W, Z, \Lambda) \in (\mathcal{C} \times \mathcal{T} \times \Delta)\right\}. \tag{4.45}$$

The function H is differentiable and its gradient at the point (W^r, Z^r, Λ^r) is given by:

$$\begin{aligned}
\bar{W}^r &= \nabla_W H(W, Z, \Lambda) = \left(m\rho w_{jl} - \sum_{i=1}^{m}\mu w_{jl}^{\mu-1}\lambda_{li}^{\beta}(z_{li} - x_{ji})^2\right)_{j=1..n}^{l=1..k},\\
\bar{Z}^r &= \nabla_Z H(W, Z, \Lambda) = \left(n\rho z_{li} - \sum_{j=1}^{n} 2w_{jl}^{\mu}\lambda_{li}^{\beta}(z_{li} - x_{ji})\right)_{l=1..k}^{i=1..m},\\
\bar{\Lambda}^r &= \nabla_\Lambda H(W, Z, \Lambda) = \left(n\rho\lambda_{li} - \sum_{j=1}^{n}\beta w_{jl}^{\mu}\lambda_{li}^{\beta-1}(z_{li} - x_{ji})^2\right)_{l=1..k}.
\end{aligned} \tag{4.46}$$

The solution of the subproblem (4.45) is explicitly computed as (Proj stands for the projection)

$$\begin{aligned}
(W^{r+1})_j &= \text{Proj}_{\mathcal{C}_j}\left(\tfrac{1}{m\rho}(\bar{W}^r)_j\right) \;\; j = 1, ...n;\\
(Z^{r+1})_{li} &= \text{Proj}_{[\alpha_i,\gamma_i]}\left(\tfrac{1}{n\rho}(\bar{Z}^r)_{li}\right) \;\; l = 1, .., k, i = 1, ...m;\\
(\Lambda^{r+1})_l &= \text{Proj}_{\Delta_l}\left(\tfrac{1}{n\rho}(\bar{\Lambda}^r)_l\right) \;\; l = 1, ...k.
\end{aligned} \tag{4.47}$$

Finally, DCA applied to (4.44) can be described as follows.

DCA–SI: DCA applied to (4.44)
- **Initialization:** Choose W^0, Z^0 and Λ^0. Let $\epsilon > 0$ be sufficiently small, $r = 0$.
- **Repeat**
  - Compute $(\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r)$ via (4.46).
  - Compute $(W^{r+1}, Z^{r+1}, \Lambda^{r+1})$ via (4.47).
  - $r = r + 1$
- **Until** $\|(W^{r+1}, Z^{r+1}, \Lambda^{r+1}) - (W^r, Z^r, \Lambda^r)\| \leq \epsilon$ or $|F(W^{r+1}, Z^{r+1}, \Lambda^{r+1}) - F(W^r, Z^r, \Lambda^r)| \leq \epsilon$.

4.2.4 Finding a good starting point of DCA

In a similar way to the initial procedure introduced in (Le Thi et al. [2008a]), we propose an alternative SCAD - DCA–SI procedure for (4.44) which is described as follows.

SCAD - DCA–SI procedure
- **Initialization:** Choose randomly W^0, Z^0 and Λ^0. Let $max\_iter > 0$ be a given integer. Set $s = 0$.
- **Repeat**
  - Perform one iteration of **SCAD** from (W^s, Z^s, Λ^s).
  - Perform one iteration of **DCA–SI** from the solution given by **SCAD** to obtain $(W^{s+1}, Z^{s+1},$ $\Lambda^{s+1})$.
  - $s = s + 1$.
- **Until** $s = max\_iter$.

In our experiments, we use $max\_iter = 2$.

4.2.5 Application on image segmentation

Image segmentation is an important processing step in many image, video and computer vision applications. It is a critical step towards content analysis and image understanding. The aim of image segmentation is to partition an image into a set of non-overlapped, consistent regions with respect to some characteristics such as colors/gray values or textures. Image

segmentation is an important research field and many segmentation methods have been proposed in the literature. For a more complete review on image segmentation methods, the reader is referred to Haralick and Shapiro [1985], Pal and Pal [1993], Skarbek and Koschan [1994], Verge Llahi [2005] and the references therein. We can classify image segmentation methods into four categories (Skarbek and Koschan [1994], Verge Llahi [2005]): methods based on pixels, on areas, on contours and on physical model for image formation.

Pixel based methods, which consist in regrouping in different regions the pixels contained in an image, are the simplest approach for image segmentation. There are three main classes of techniques in pixel based methods:

- Histogram-based technique: firstly, a histogram is computed from all of the pixels in the image. Then, image pixels are classified as belonging to one of those classes thus formed by using the peaks and valleys in the histogram.
- Clustering techniques: pixels are grouped, using a hard clustering method, by means of their color values/textures.
- Fuzzy clustering techniques: instead of using hard clustering, fuzzy clustering is used for pixel classification task. A popular choice is the FCM algorithm.

We apply now our DCA based algorithm for the weighted feature fuzzy clustering model to segmentation of color images via fuzzy clustering of pixels.

Protocol testing

We compare the efficiency of our method with three others methods: **SCAD** (Frigui and Nasraoui [2004]), **DCAFCM** (Le Thi et al. [2007c]) and **FCM** (Bezdek [1981]).

The parameters of each algorithm as chosen as follows: $\mu \in [1.1, \ldots, 4.0]$ for both FCM and DCAFCM, $\mu, \beta \in [1.1, \ldots, 4.0]$ for SCAD, and $\mu, \beta \in [2.0, \ldots, 4.0]$ for DCA–SI. We stop DCA based algorithms (DCA–SI and DCAFCM) with the tolerance $\epsilon = 10^{-4}$.

As the same way in Frigui and Nasraoui [2004], we map each pixel to an 8–dimensional feature vector consisting of three colors, three texture features and the two positions of pixels. The three color features are L^*a^*b coordinates of the color image. The three texture features (polarity, anisotropy and contrast (cf. Belongie et al. [1998], Frigui and Nasraoui [2004]) are computed as follows. First, the image $I(x, y)$ is convolved with Gaussian smoothing kernels $G_\delta(x, y)$ of several scales δ: $M_\delta(x, y) = G_\delta(x, y) \otimes (\Delta I(x, y))(\Delta I(x, y))^t$.

- The polarity is defined by $p = |E_+ - E_-|/(E_+ - E_-)$, where E_+ and E_- represent, respectively, the number of gradient vectors in the matrix Gauss kernels $G_\delta(x, y)$ of scale δ at the pixel (x, y) on the positive and negative sides of the dominant orientation. For each pixel, an optimal scale value is selected such that it corresponds to the value where polarity stabilizes with respect to scale.
- The anisotropy is computed by $a = 1 - \lambda_2/\lambda_1$, where λ_1, λ_2 are the eigenvalues of $M_\delta(x, y)$ at the selected scale.
- The texture contrast is defined as $c = 2(\sqrt{\lambda_1 + \lambda_2})^3$.

Experiments

In this section, we execute image segmentation on 7 images taken from *Berkeley segmentation dataset (BSD)* (Martin et al. [2001]). These images include the real label of pixels.

To evaluate the segmentation results obtained by four algorithms we compare the results with the real labels, then compute the PWCO index (percentage of well classified objects). The results of all experiments are shown in the table 4.8 and the graphic is presented in the figure 4.3.

Let us consider in detail three examples: image 113044, image 12003 and image 134052.

In the figure 4.5 (Image 113044), we would like to clustering the original image into two clusters: "horses" and "grasses". Comparing the methods using feature weights (DCA–SI and SCAD) with the methods without feature weights (DCAFCM and FCM) we see that the first approach is clearly better. Indeed, DCA–SI and SCAD detect clearly two part in images: "horses" and "grasses", while DCAFCM and row FCM detect incorrectly some pixel "grasses" (approximate 6.5% with DCAFCM and 12.5% with FCM). Comparing the effectiveness of DCA based algorithms with other (SCAD and FCM), we observe that DCA–SI is better than SCAD with the accuracy 97.49% versus 96.98% and DCAFCM is better than FCM with the accuracy 93.55% versus 87.50%.

Figure 4.6 (Image 12003) is a picture of a starfish. This image includes 3 clusters: the starfish, the coral (on the upper left and on the right side) and the moss. The clusters of this image is not separate, the coral seems transparent with the moss background, so segmentation task of this image is not easy. We see that all of methods can not detect well the coral class, while the starfish is separated from two remain classes. The best accuracy in this case is 90.41% (by DCA–SI) and the worst is 51.42% (by FCM).

The leopard image, figure 4.7 (Image 134052) consists of 2 clusters: leopard and background. DCAFCM and FCM detect some background pixels belonging to leopard and a part of body leopard as background. As in other examples, the methods using features weighting are better. The PWCO of DCA–SI as well as SCAD is: 96.61%, while the one of DCAFCM and FCM a is 80.61%.

Table 4.8: The results of PWCO(%) of 4 methods

Image	Size	No. Classes	DCA–SI	DCAFCM	SCAD	FCM
113044	321×481	2	**97.49**	93.55	96.98	87.50
124084	321×481	3	**93.92**	75.19	82.86	74.35
113016	321×481	2	**97.06**	86.54	96.76	92.68
12003	321×481	3	**90.41**	64.56	71.05	51.42
134052	321×481	2	96.60	80.61	**96.61**	80.61
35070	321×481	3	**83.06**	75.70	80.44	65.69
157032	321×481	6	**76.42**	68.71	68.12	42.08

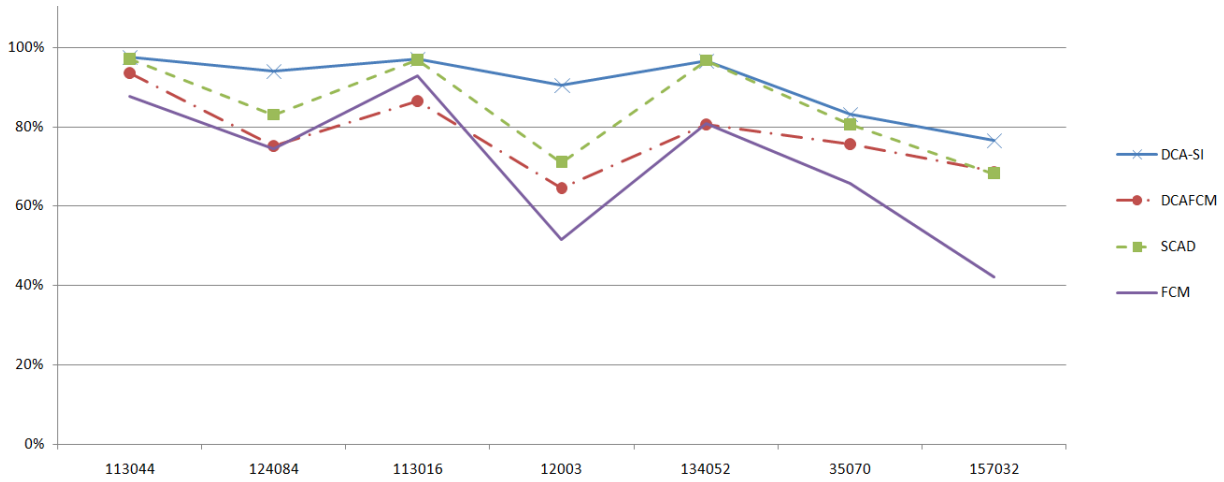


Figure 4.3: Accuracy.

Table 4.9: CPU Time running in seconds

Image	**Size**	**No. Classes**	**DCA–SI**	**DCAFCM**	**SCAD**	**FCM**
113044	321×481	2	78.35	24.10	42.32	9.22
124084	321×481	3	94.31	32.63	79.63	13.64
113016	321×481	2	86.36	21.90	44.13	9.08
12003	321×481	3	108.37	32.74	253.22	14.04
134052	321×481	2	19.49	16.73	38.42	12.43
35070	321×481	3	98.64	32.85	112.82	13.51
157032	321×481	6	164.35	59.20	623.07	27.27

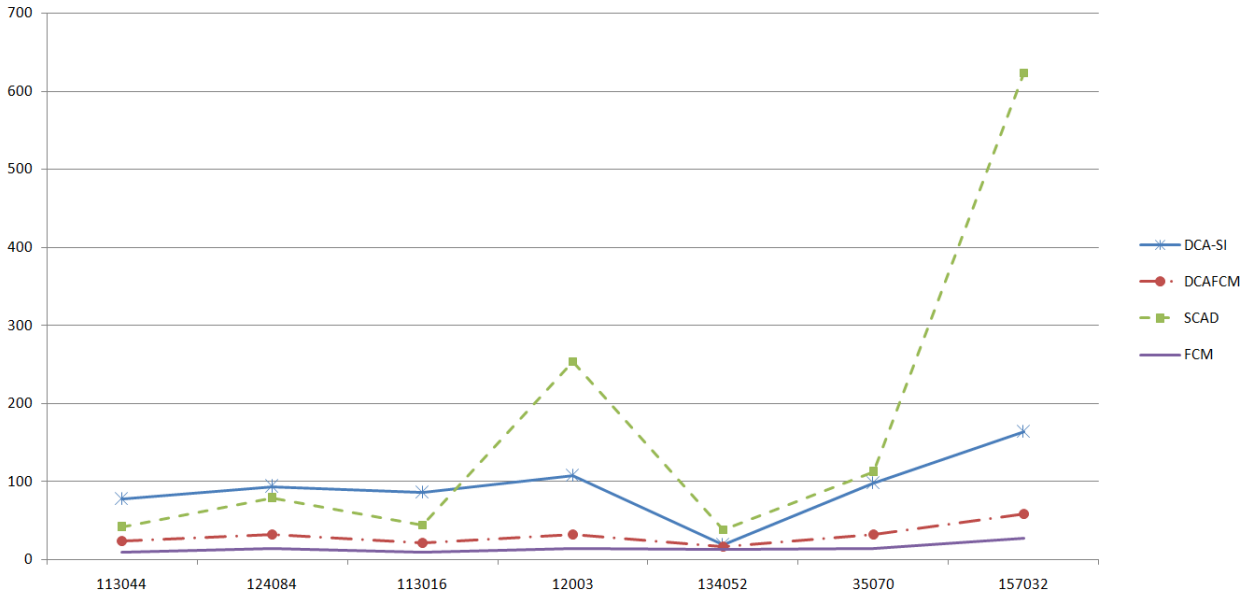


Figure 4.4: CPU Time running in seconds.

The table 4.8 and table 4.9 (respect. graphic 4.3 and graphic 4.4) show the PWCO values and time running of all executions on all images. The first two columns are the index and size of images, which are defined in repository of *Berkeley segmentation dataset* (Martin et al. [2001]), the last four columns are the PWCO values and CPU time running for each algorithm on the corresponding image.

From these tables, we observe that:

i) Comparing the fuzzy clustering algorithms with and without features weighting, say DCA–SI and SCAD versus DCAFCM and FCM:

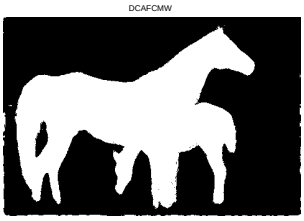
   DCA–SI give better PWCO than DCAFCM and FCM on all data sets; the gain of DCA–SI go up to 25.9% (in comparing with DCAFCM) and 39% (in comparing with FCM) in the starfish (Image 12003);

   SCAD is only slightly worse than DCAFCM in a case of Image 157032 (68.12% with 68.71%) and better than DCAFCM in other cases; SCAD is better than FCM on all cases.

ii) DCA–SI furnishes better PWCO than SCAD with a big gain on Image 12003 (19.4%) and Image 124084 (11.1%). The PWCO values of DCA–SI are greater than the PWCO values of SCAD on 6/7 data sets, while the algorithm SCAD is slightly better than DCA–SI in a case of Image 134052 (96.61% versus 96.60%).

iii) FCM is fastest algorithm out of four and SCAD is the most time consuming.

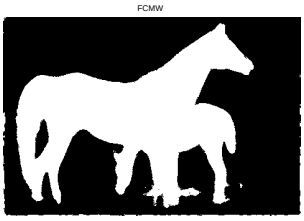
(a) Original image



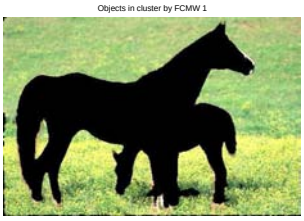
(a) DCA–SI

(b) Class 1

(c) Class 2

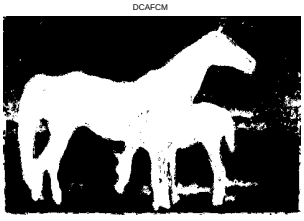


(a) SCAD



(b) Class 1

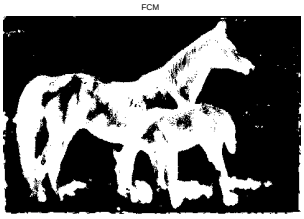
(c) Class 2



(a) DCAFCM

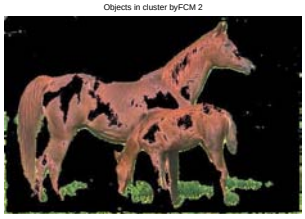
(b) Class 1

(c) Class 2



(a) FCM

(b) Class 1



(c) Class 2

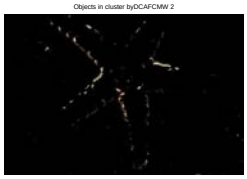
Figure 4.5: Image 113044

(a) Original image

(a) DCA–SI

(b) Class 1

(c) Class 2

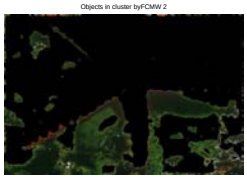


(d) Class 3

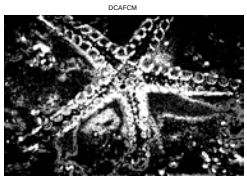
(a) SCAD

(b) Class 1

(c) Class 2



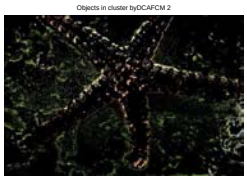
(d) Class 3



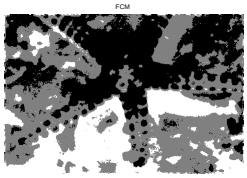
(a) DCAFCM

(b) Class 1

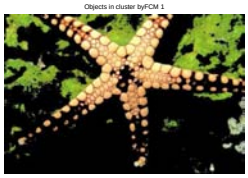
(c) Class 2



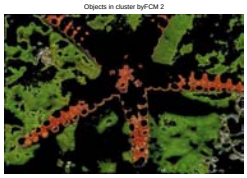
(d) Class 3



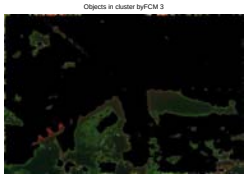
(a) FCM



(b) Class 1



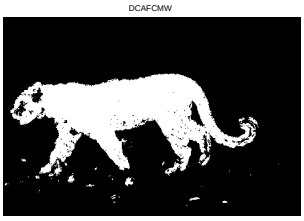
(c) Class 2



(d) Class 3

Figure 4.6: Image 12003

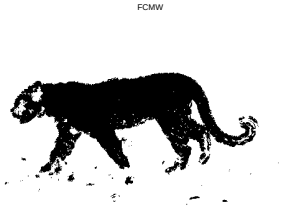
(a) Original image



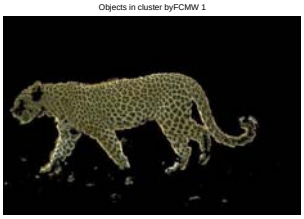
(a) DCA–SI

(b) Class 1

(c) Class 2



(a) SCAD



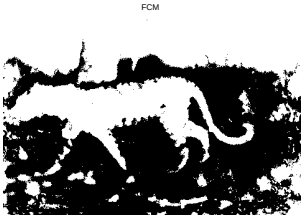
(b) Class 1

(c) Class 2

(a) DCAFCM

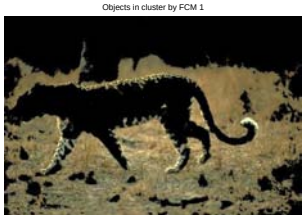
(b) Class 1

(c) Class 2



(a) FCM

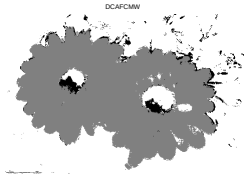
(b) Class 1



(c) Class 2

Figure 4.7: Image 134052

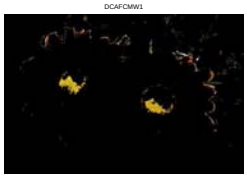
(a) Original image



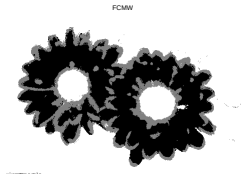
(a) DCA–SI

(b) Class 1

(c) Class 2



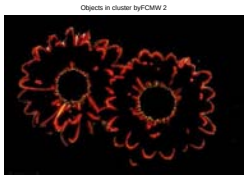
(d) Class 3



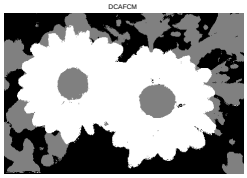
(a) SCAD

(b) Class 1

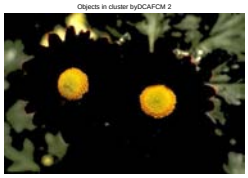
(c) Class 2



(d) Class 3

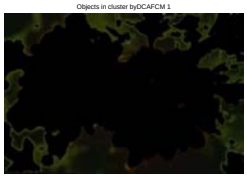


(a) DCAFCM

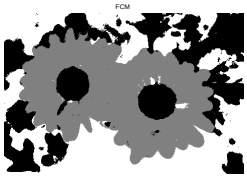


(b) Class 1

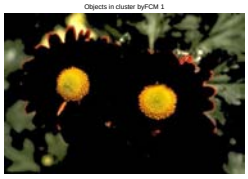
(c) Class 2



(d) Class 3

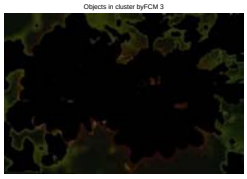


(a) FCM



(b) Class 1

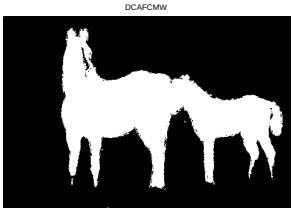
(c) Class 2



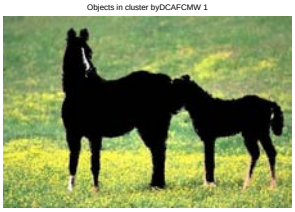
(d) Class 3

Figure 4.8: Image 124084

(a) Original image

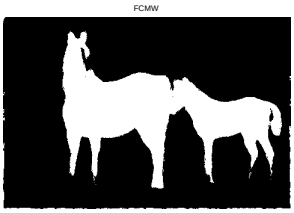


(a) DCA–SI

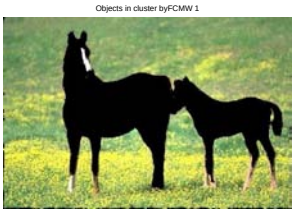


(b) Class 1

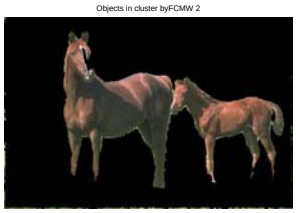
(c) Class 2



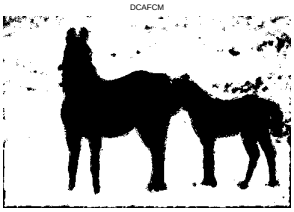
(a) SCAD



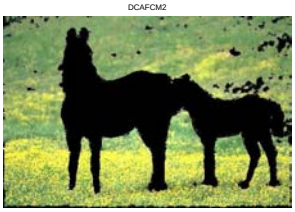
(b) Class 1



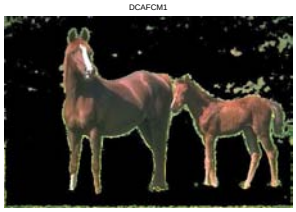
(c) Class 2



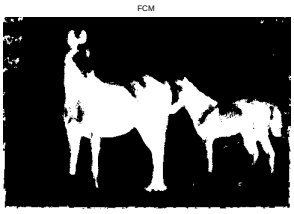
(a) DCAFCM



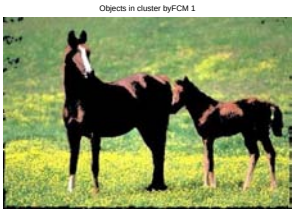
(b) Class 1



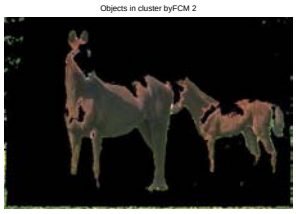
(c) Class 2



(a) FCM



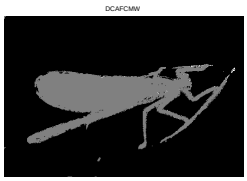
(b) Class 1



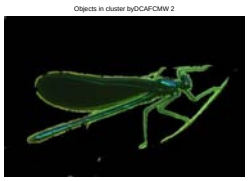
(c) Class 2

Figure 4.9: Image 113016

(a) Original image

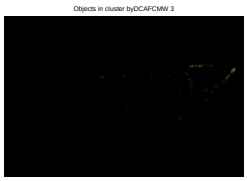


(a) DCA–SI

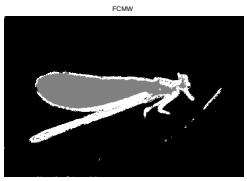


(b) Class 1

(c) Class 2



(d) Class 3

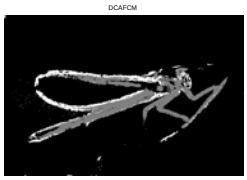


(a) SCAD

(b) Class 1

(c) Class 2

(d) Class 3

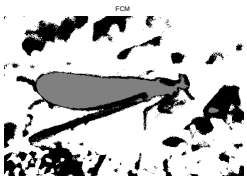


(a) DCAFCM

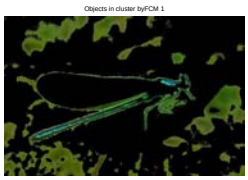
(b) Class 1

(c) Class 2

(d) Class 3

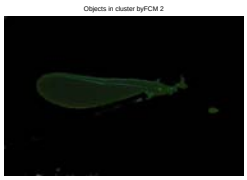


(a) FCM



(b) Class 1

(c) Class 2



(d) Class 3

Figure 4.10: Image 35070

(a) Origin image

(a) DCA–SI (b) Class 1 (c) Class 2 (d) Class 3 (e) Class 4 (f) Class 5 (g) Class 6

(a) SCAD (b) Class 1 (c) Class 2 (d) Class 3 (e) Class 4 (f) Class 5 (g) Class 6

(a) DCAFCM (b) Class 1 (c) Class 2 (d) Class 3 (e) Class 4 (f) Class 5 (g) Class 6

(a) FCM (b) Class 1 (c) Class 2 (d) Class 3 (e) Class 4 (f) Class 5 (g) Class 6

Figure 4.11: Image 157032

4.2.6 Conclusion.

In this section, we have studied DC programming and DCA for weighted feature fuzzy clustering problem. The optimization model has been formulated as a DC program. Then, a DC algorithm is developed to solve the resulting problem. The DC decomposition has a nice feature that is, at each iteration, all operations can be explicitly calculated.

As an application, we have applied our method to image segmentation problem. In our experiments, we have compared 4 algorithms: DCA–SI, DCAFCM, SCAD and FCM. The results show the efficiency of weighted feature measures, which allows to improve the performance of segmentation task. Furthermore, computational experiments show the superiority of DCA–SI with respect to the other algorithms. We are convinced that our approach is promising for weighted features fuzzy clustering.

Chapter 5

An elitist evolutionary approach for clustering tasks [1]

This chapter proposes an elitist evolutionary approach for clustering data sets, without prior knowledge of the clusters number. The proposed method is based on the clusters number optimization and in the same time, propose the potential clusters seeds. This method can be used directly as a clustering algorithm or as an initialization method of k–Means algorithm without prior knowledge of the clusters number. In the proposed approach, elitist population is composed of the individuals with potential clusters seeds, we introduce a new mutation strategy according to the neighborhood search and new evaluation criteria. This strategy allow us to find the global optimal solution or near-optimal solution for clustering tasks, precisely finding the optimal clusters seeds without prior knowledge of the clusters number. The diversity of population can be maintained by evolutionary algorithms subpopulation. The difference between evolutionary algorithms is implemented by the parameters and we select only the best concurrent solution which represent the elite solution. The experimental results show that our algorithm performs well on multi-class data sets, parameters-sensitive and large-size data sets.

5.1 Introduction

Clustering is a challenging research area in data mining. A common form of clustering is partitioning the data set into homogeneous clusters such that members of the same cluster are similar and members of distinct clusters are dissimilar. Determining the optimal clusters

1. This chapter is published under the titles:
[1]. Lydia Boudjeloud–Assala, Ta Minh Thuy. *Determine optimal number of clusters with an elitist evolutionary approach*, Advances in Knowledge Discovery and Data Mining (PAKDD2014), Lecture Notes in Computer Science, Volume 8444, pp 324–335 (2014).
[2]. Lydia Boudjeloud–Assala, Ta Minh Thuy. *A clustering algorithm based on elitist evolutionary approach.* Submitted to Journal of Classification.

number is one of the most difficult issues in clustering data. In this work, we deal with the clustering problem without prior knowledge on the appropriate clusters number and we try to propose the global optimal or near-optimal clusters seeds.

Clustering algorithms can be broadly classified into two groups: hierarchical and partition (Jain [2010]). Hierarchical algorithms recursively find nested clusters either in a divisive or agglomerative method. In contrast, partition algorithms find all the clusters simultaneously as a partition of the data and do not impose a hierarchical structure.

Common formulation of the clustering problem is assuming S is the given data set include n data points: $S = \{x_1, x_2, ..., x_n\}$ where x_i $(i = 1, ..., n)$ is a real vector $d - dimensions$ and an integer k. The goal of clustering is determine a set of k clusters $C_1, C_2, ..., C_k$ such that, the points belong to the same cluster are similar, while the points belong to different clusters are dissimilar in the sense of the given metric. The problem of finding an optimal solution to the partition of N data into k clusters is $NP - complete$, and heuristic methods are widely effective on $NP - complete$ global optimization problems and they can provide good sub–optimal solutions in reasonable time.

We propose a clustering algorithm that can detect compact, hyperspherical and hyperellipsoidal clusters that are well separated using an Euclidian and the Mahalanobis distance respectively.

Some recent researches have shown that the problem of search efficient initialization methods for k–Means clustering algorithm is a great challenge. Numerous initialization methods have been proposed to address this problem. Celebi et al. (Celebi et al. [2013]) present an overview of these methods with an emphasis on their computational efficiency. In their study, they investigate some of the most popular initialization methods developed for the k–Means algorithm. They describe and compare initialization methods that can be used to initialize other partition clustering algorithms such as Fuzzy c-means and its variants, and they conclude that most of these methods can be used independently of k–Means as stand alone clustering algorithms.

Some others methods are proposed, based on metaheuristics such as simulated annealing (Babu and Murty [1994]) and genetic algorithms (Babu and Murty [1993]). These algorithms start from a random initial configuration (population) and use k–Means to evaluate their solutions in each iteration (generation). There are three main disadvantages associated with these methods. Firstly, they involve numerous parameters that are difficult to tune (Jain et al. [1999]). Secondly, due to the large search space, they often require a large number of iterations, which renders them computationally prohibitive for all most of data sets. Finally, they use k–Means to evaluate their solutions, which is very inadvisable according to the objective to use these methods independently of k–Means.

This chapter presents a method that proposes, in the same time, the initial clusters seeds, with the optimal cluster number, without a prior knowledge of clusters number. We don't use k–Means or any other clustering algorithm to evaluate our solution, so, our method can be used as stand alone clustering algorithms. Generally metaheuristics approaches involve

numerous parameters that are difficult to tune, to deal with this problem, we propose an Elitist Evolutionary Approach that involve numerous evolutionary algorithms (EAs). The difference between them is implemented by the parameters and we select only the best concurrent solution. This proposition can also address to the problem of exploration the large search space. The initial populations of numerous evolutionary algorithms (EAs) are substantially different, so, we can deal with the large data sets in few numbers of iterations.

5.2 State of the art

In this section, we review some of the commonly used initialization methods and elitist methods.

5.2.1 Initialization methods

Celebi et al. (Celebi et al. [2013]) investigate some of the most popular initialization methods developed for the k–Means algorithm. Their motivation is threefold. Firstly, a large number of initialization methods have been proposed in the literature and thus a systematic study that reviews and compares these methods is desirable. Secondly, these initialization methods can be used to initialize other partition clustering algorithms such as Fuzzy c-means and its variants and Expectation Maximization (EM). Finally, most of these initialization methods can be used independently of k–Means as stand alone clustering algorithms. They review some of the commonly used initialization methods with an emphasis on their time complexity (with respect to the number of data points: n). They describe firstly, a linear time-complexity initialization methods and they compare with quadratic-complexity initialization methods. They conclude that the super linear methods often have more elaborate designs when compared to linear ones. An interesting feature of the super linear methods is that they are often deterministic, which can be considered as an advantage especially when dealing with large data sets. In contrast, linear methods are often non-deterministic and/or order-sensitive. A frequently cited advantage of the more elaborate methods is that they often lead to faster k–Means convergence, i.e. require fewer iterations, and as a result the time gained during the clustering phase can offset the time lost during the initialization phase. This may be true when a standard implementation of k–Means is used. However, convergence speed may not be as important when a fast k–Means variant is used as such methods often require significantly less time compared to a standard k–Means implementation.

Some other initialization methods such as the binary-splitting method (Linde et al. [1980]) takes the mean of data as the first center. In iteration t, each of the existing 2^{t-1} centers is split into two new centers by subtracting and adding a fixed perturbation vector. These 2^t new centers are then refined using k–Means. There are two main disadvantages associated with this method. First, there is no guidance on the selection of a proper value for the vector, which determines the direction of the split (Huang and Harris [1993]). Second, the

method is computationally demanding since after each iteration k–Means has to be run for the entire data set.

Some other methods based on metaheuristics such as simulated annealing (Babu and Murty [1994]) and genetic algorithms (Babu and Murty [1993]). These algorithms start from a random initial configuration (population) and use k–Means to evaluate their solutions in each iteration (generation). There are three main disadvantages associated with these methods. Firstly, they involve numerous parameters that are difficult to tune (initial temperature, cooling schedule, population size, crossover, mutation probability, etc.) (Jain et al. [1999]). Secondly, due to the large search space, they often require a large number of iterations, which renders them computationally prohibitive for all but the smallest data sets. Finally, their objective is to use them independently of k–Means, but they use k–Means to evaluate their solutions, which is very inadvisable. Interestingly, with the recent developments in combinatorial optimization algorithms, it is now feasible to obtain globally minimum Sum of Squared Error (SSE) clusterings for small data sets without resorting to metaheuristics (Aloise et al. [2012]).

We introduce a new multi evolutionary algorithm that runs together independently on k–Means evaluation. To deal with the numerous parameters that are difficult to tune, we propose to involve together numerous evolutionary algorithms. The difference between them is the different tunes of parameters and they concur to find the elite solution.

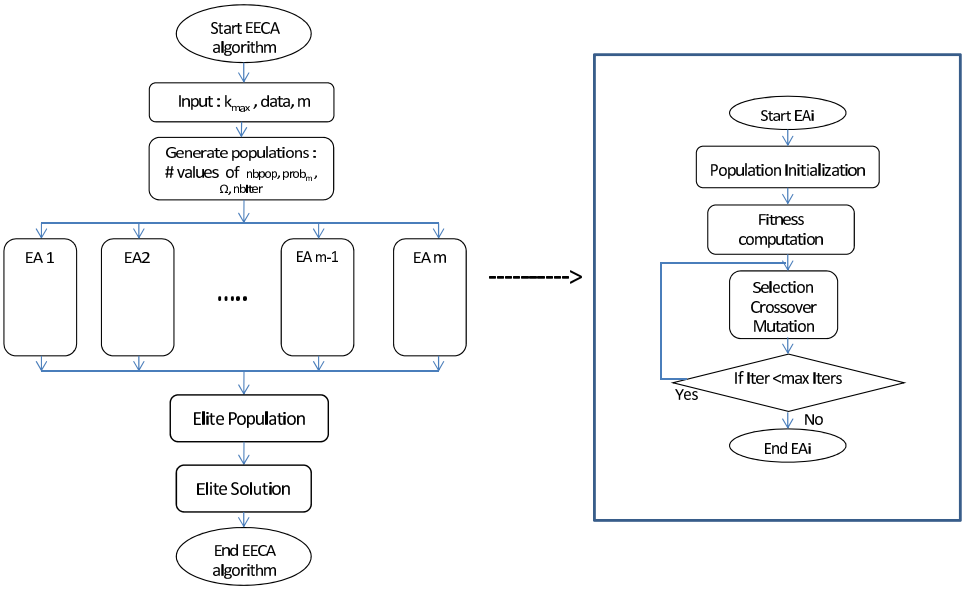


Figure 5.1: EECA schema

5.2.2 Elitist methods

One of very important tasks during the application of genetic operators is to prevent promising individuals eliminate from the population. To ensure that the best chromosome is preserved, elitist method copy the best individual found so far into the new population. Different EAs variants achieve this goal of preserving the best solution in different ways. However, elitist strategies tend to make the search more exploitative rather than explorative and may not work for problems in which one is required to find multiple optimal solutions (Sarma and De Jong [1997]).

Elitist methods are widely applied on different domains, Leung and Liang (Leung and Liang [2003], Liang and Leung [2011]) introduce a new technique called adaptive elitist population search method for allowing uni-modal function optimization methods to be extended to efficiently locate all optima of multi-modal problems. The developed method has been shown to be very efficient and effective in finding multiple solutions of the benchmark multi-modal optimization problems. The proposed technique is based on the concept of adaptively adjusting the population size according to the individual dissimilarity and the novel elitist genetic operators.

Das et al. (Das et al. [2008]) proposed a method based on a modified version of classical Particle Swarm Optimization algorithm, known as the Multi-Elitist Particle Swarm Optimization model. The proposed algorithm has been shown to meet or beat the other state of the art clustering algorithms in a statistically meaningful way over several benchmark data sets. The unique disadvantage of this algorithm is to choosing the best suited parameters to find optimal solution.

Chen and Lu (Chen and Lu [2008]) present an investigation on local-search heuristic method called Extremal Optimization (EO) with its application in numerical multi-objective optimization and proposes a new elitist multi-objective algorithm, called Multi-Objective Extremal Optimization (MOEO). The simulation results indicate that the proposed approach is highly competitive with the state-of-the-art multi-objective evolutionary algorithms.

Qasem and Shamsuddin (Qasem and Shamsuddin [2011]) developed an Mimetic Elitist Pareto Differential Evolution algorithm, in order to deal with the hybrid learning problem (unsupervised and supervised learning), they use the multi-elitist approach to help the learning algorithm to get out of local minimum, therefore improving the accuracy of the proposed learning model.

Gou et al. (Gou et al. [2013]) apply Multi Elitist approach on quantum clustering problems. They use these methods to don't getting stuck in local extremes. They used the mechanism of cluster center updating with a property of k–Means clustering, which could influence the clustering results. This is one of disadvantages of this method, with adding some parameters which the method is based on.

According to the elitist strategy, that work for problems in which one is required to find multiple optimal solutions, we introduce a new approach where multi evolutionary algorithms

run together in same time to compare their proposed solutions, and we select only the best one which is the optimal or nearest optimal solution and considered as the elite solution.

5.3 Proposed elitist evolutionary clustering algorithm - EECA

Elitist Evolutionary Clustering Algorithm (EECA) (figure 5.1) combines different techniques that consists of evolutionary algorithms with elitist approach and local search approach. This EECA allows us to determine the optimal cluster number as well as finding cluster seeds, to obtain clustering solution. It can also be used as an initialization method for k–Means algorithm or other clustering algorithms. Elitist population is composed of the individuals with high affinity, which is considered to play dominant roles in the evolutionary process. It can help to find the global optimal solution or near-optimal solution for most tested tasks. The diversity of population can be well maintained by general evolutionary subpopulation. These different clustering task are implemented by the new proposed mutation strategy, crossover operators and fitness function.

5.3.1 Evolutionary algorithm

In this section, we try to explain succinctly the first part of Elitist Evolutionary Clustering Algorithm (EECA) which is the collection of evolutionary algorithms as we can see in right of the figure 5.1. We describe evolutionary components of one evolutionary algorithm before details on the difference between them.

5.3.1.1 Gene representation

We consider an evolutionary individual (chromosome) which is a combination of k_{max} potential optimal seeds, with k_{max} is an input parameter of algorithm. Each gene in evolutionary individual is an integer number, which take values from $\{1, 2, .., n\}$ indicates the data point identification, these points are chosen as initial cluster centers. The gene with value 0 means that no point is selected. The number of gene in the optimal solution different from 0 is the optimal clusters number. Each data point is a vector $d - dimensions$ containing the d real values. We want to diversify the population, for this, each gene is selected by random uniform distribution over the set of position data $\{0, 1, 2, ..., n\}$. Associating a frequency rate at each data point, each new evolutionary individual will be composed by the data points that have frequencies equal or near zero (equal 0 when this point has not been used previously).

5.3.1.2 Population initialization

The population initialization is created by nb_{pop} evolutionary individuals with nb_{pop} given before. This population represents the evolutionary subpopulation of our elitist evolutionary clustering algorithm (EECA). Each evolutionary individual corresponds to a specific clustering solution.

We impose to have an initial population without redundant evolutionary individuals. Study individual is applied on each evolutionary individual which is created, we then verify, that is no identical evolutionary individual, in the population which have same gene. At this step, the first population is ready. Once the population is evaluated and sorted according to the fitness function we operate the evolutionary operator described bellow.

5.3.1.3 Evolutionary operators

The classical or natural operators are the mostly used; even if they are used frequently they are not appropriate. Therefore some researchers (Radcliffe [1991]) propose to adapted the operator to the proposed problem.

The crossover operation produces new offspring evolutionary individuals from parent individuals. Two new evolutionary individuals are created by exchanging genes from two parent chromosomes. This exchange may start from one or several positions in the chromosomes called cut point. We can use two types of crossover, a randomly determined cut point or an optimized cut point. In the latter case, we determine the best point before the cut, this implies an evaluation of each possible cut for the individual.

For our case, we choose the cut point that obtain the child better than the parent or the best child resulted by repeated iteratively crossover.

The mutation makes gene inversion in the evolutionary individual. This evolutionary operator is used to avoid the degeneration of the population in order to prevent a too fast convergence of the evolutionary algorithm. If implemented appropriately, the operator can make the algorithm able to leave from local optimum. The mutation is usually applied with a small probability ($prob_m$, algorithm parameter). The mutated gene is replaced by a new gene, which is chosen according to neighborhood search (section 5.3.2), we should verify if a new gene value is not in neighborhood of others genes composed the evolutionary individual according to our clustering problem. If we do not find this new gene, we change the this value by 0. It is evident that we study the composition of each creation of new individual to have a different genes in a specific evolutionary individual.

5.3.2 Adaptive neighborhood search

The mutation is performed to avoid the degeneration of the population in order to prevent a too fast convergence of the evolutionary algorithm. If implemented appropriately, the

operator can make the algorithm able to leave from local optimum. The mutation operator is generally performed to optimize the local search with a fixed probability $prob_m$. When we operate the mutation we should verify firstly, if all genes are different, secondly, according to obtained different potential cluster seeds, where clusters must be separate, we should verify if a new mutated gene is not in neighborhood of others genes composed the evolutionary individual. We operate mutation by, choosing randomly the gene to be muted, and changing the gene value by new value which is not on the neighborhood of other genes composed the evolutionary individual, and if we do not find this new gene, we change the value by 0. In order to obtain the neighborhood gene, we search the set of data points that are in the neighborhood of this point, this set represents then the cluster of the ones close enough to the seed. To determine automatically this cluster containing the close enough points to the seed, we determine automatically the cluster limit. For this, we choose the threshold Ω (algorithm parameter) by computing the distance between all the data points and the cluster seed and ordering them from the closest object to the farthest. We then try to find an abrupt increasing of distance that will indicate the cluster limit.

We choose to use the peak detection method presented by Palshikar (Palshikar [2009]). This method depends on the parameter Ω to detect abrupt of different distance. By authors, $\Omega = \{1, 2, 3\}$ is sufficient to find a good solution. Other cluster limit detection might be used, but this one is fast and gives the algorithm a complexity of $O(n \times d \times (p + g))$, where n is the number of data points, d the number of dimensions, p the population size and g the number of generations.

5.3.3 Adaptive elitist-population search

Generally, the goal of the adaptive elitist-population search method is to adaptively adjust the population size according to the features of the objective to achieve. Firstly, each single elitist individual searches one solution; and secondly, all the individuals in the population search different solutions in parallel.

We define elite population as a set of individuals with the best fitness on different solutions of evolutionary populations. Then we propose the elitist evolutionary operators that can maintain and even improve the diversity of the population through adaptively adjusting the population size and performing different evolutionary algorithms.

A major advantage of using EAs over traditional learning algorithms is the ability to escape from local minimum, its robustness and its ability to adapt itself to a changing potential solution environment.

Our evolutionary algorithm depends on 4 parameters: nb_{pop}, nb_{iter}, $prob_m$ and Ω. We perform as much as possible many evolutionary algorithms according different values of these parameters and then we select the better solution of each of them in the elite population. The best solution represents a single elite individual, without focusing on setting parameters.

5.3.4 Fitness computation

Our objective is to find clusters that are compact and separated between them, for this we combine three functions. The first objective is to minimize the overlapping between clusters, this criteria is defined by:

$$OP = \sum_{i \neq j}^{k} Card(C_i \cap C_j)$$

The second objective is to verify that whole data are contained in the different clusters. We define this criteria by:

$$\sum_{i \neq j}^{k} Card(C_i \cup C_j) = n$$

Finally, to optimize the clusters number we use Calinski and Harabasz index ($CH - index$) (Caliński and Harabasz [1974], Vendramin et al. [2009]), which represent, a ratio of the sum of between-cluster and the sum of within-cluster. The best clustering is achieved when $CH - index$ is maximized, and is expressed as follows:

$$CH(k) := \frac{[traceB/(k-1)]}{[traceW/(n-k)]}$$

where n is the number of points data, k is clusters number

$$traceB := \sum_{i=1}^{k} |C_i| \ \|\overline{C_i} - \overline{x}\|^2$$

$$traceW := \sum_{i=1}^{k} \sum_{j \in C_i} \|x_j - \overline{C_i}\|^2$$

with $|C_i|$ is the number of objects assigned to the cluster C_i $(i = 1, \ldots, k)$; $\overline{C_i}$ is a center of class C_i and $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ is the global center of all data points.

Several measures to optimize the clusters number have been proposed. Davies-Bouldin index Davies and Bouldin [1979] considered as a ratio of the intra cluster scatter, to the inter cluster separation, a lower value will mean that the clustering is a good partition. Another index is Dunn's index Dunn [1974], which aims to identify dense and well-separated clusters. It is calculated by the minimal the ratio between the inter cluster distance and maximal intra cluster distance. One index is the most frequently using in the literature is the sum of squared error (SSE) index Kudová [2007] Sharma and Rai [2012]. It is sum of minimal distance

objects and centers. Milligan and Cooper Milligan and Cooper [1985] presented a survey and comparison of 30 internal validity indexes for clustering algorithms and out-perform that $CH - index$ is one of the best solutions. This is why we focus only on $CH - index$, and use it to evaluate and find optimal number of clusters.

5.4 Experiments

In our experiments, we perform an Elitist Algorithm with varying different parameters values in each Evolutionary Algorithm. We vary Evolutionary Algorithm parameters nb_{pop}, $prob_m$, nb_{iter} and Ω as follows:

- $nb_{pop} \in \{50, 100, 150\}$
- $prob_m \in \{0.1, 0.2, 0.3\}$
- $nb_{iter} \in \{200, 300, 500, 1000\}$
- $\Omega \in \{1, 2, 3\}$

For each data set we perform the elitist algorithm with 12 EAs and $k_{max} = 10$. All algorithms have been implemented in Visual C++ 2008, run on a PC Intel i5CPU650, 3.2 GHz of 4GB RAM.

5.4.1 Fitness function evaluation

In this series of experiments, we evaluate our fitness function, which focus on minimizing OP in each EAi and sort the elite population by $CH(k)$ ($EECA$ in the table 5.2). To do this evaluation we compare the results with the obtained results on maximizing $CH(k)$ in each EAi and in the elite population ($EACH$ in the table 5.2).

To evaluate the performance of the proposed method, we proceed several experiments on data sets from University of California at Irvine (UCI) machine learning benchmark repository (Newman and Merz). Data sets information are summarized in Table 5.1. The synthetic data set is composed by five clusters with Gaussian distribution in two dimensions.

The results of finding optimal k are illustrated in table 5.2.

As we can see in the table 5.2, we find exactly the same number of clusters as in real data sets using overlapping function combined with $CH(k)$ index. When only $CH(k)$ index is used, we always find $k = 2$, (except two data sets: Ecoli and Synthetic data sets). This result can be explained by the formula of $CH(k)$ index, when we try to maximize only this index, we converge to small number of clusters. To find the data set partitions, we use a radius limit detection method based on hyperspherical cluster forms. For Synthetic data set which presents prefect hyperspherical clusters, both cases perform perfectly. But in other data sets it seems important to consider the overlapping fitness in the first, and then to select the better solution according to the $CH(k)$ index. To confirm our results, we visualize some data sets using scatter plot methods (Carr et al. [1986]) which represent all 2D projection

Table 5.1: Datasets description

Dataset	No. Points	No. Attributes	No. Clusters
Iris	150	4	3
Vehicule	846	18	4
Haberman	306	3	2
Synthetic	500	2	5
Wine	178	13	3
Blood Tranfusion	748	4	2
Seed	210	7	3
Ecoli	336	7	8

Table 5.2: Results description

Dataset	k-real	*EACH*	*EECA*
Iris	3	2	3
Vehicule	4	2	4
Haberman	2	2	2
Synthetic	5	5	5
Wine	3	2	3
Blood Tranfusion	2	2	2
Seed	3	2	3
Ecoli	8	7	8

of the data set. The figure 5.2 represents the projection of Iris data set, and the figure 5.3 represents the projection of Haberman data set. As we can see in the figures, the color points (different forms) represent the real clusters, and in red (square form) we can see the clusters seeds that are detected by our method. We can note that each of them corresponds to the real clusters, and can be considered as the center of the different clusters or as initial seed for any clustering algorithm.

These results and the visualizations showed the effectiveness of our methods on data sets that have compactness clusters structures. As we can see in the figure 5.4, that represent a synthetic data set, composed by five clusters, with Gaussian distribution. We can easily adapt our method with changing and adapting distance measures to extract other cluster structures. We can also improve our method with allowing an overlapping degree between different extracted clusters. This approach allows us to apply our method on more different data sets structures that can be added on the benchmark. In the next experiment, we will introduce another fitness function according to the overlapping data or hyperellipsoidal structure.

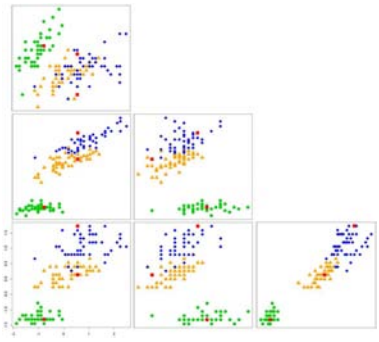


Figure 5.2: Scatter plot visualization of Iris data set with detected seeds

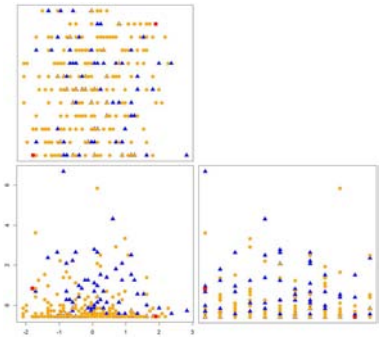


Figure 5.3: Haberman data set with detected seeds

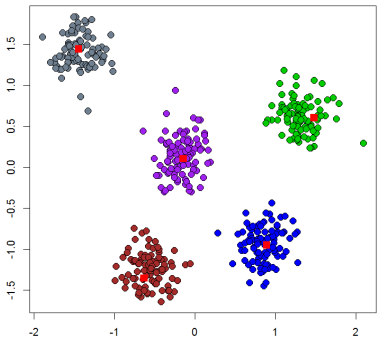


Figure 5.4: Synthetic data set with detected seeds

5.4.2 Finding optimal k

In this series of experiments, we test if our methods find automatically the number of clusters on different data sets and the computational time. The data sets are from University of California at Irvine (UCI) machine learning benchmark repository (Newman and Merz) (table 5.3) and some other high dimensional data sets are from the Kent Ridge Bio-medical data sets Repository (Jinyan and Huiqing) and the InfoSel++ library (Jacek et al.) (table 5.4). We indicate for each data set the size, the real number of cluster as we known.

Table 5.3: Multi-class data sets

Data sets	No. Points	No. Attributes	No. Clusters	optimal k	Time (s)
Synthetic	500	2	5	5	11
Iris	150	4	3	3	3
Vehicule	846	18	4	4	26
Glass	214	9	6	6	4
Blood Transfusion	748	4	2	2	30
Ecoli	336	7	8	8	7
Breast Tissue	106	9	6	6	2

As we can see in tables 5.3 and 5.4 the algorithm is pretty fast on moderate size data sets and it takes just a few minutes on the large ones. These results are encouraging, the proposed method can be used and optimized to explore relatively large data sets. We can apply, for example, pretreatment data sets as feature selection to improve the execution time for high dimensional data sets.

Table 5.4: High dimensional data sets

Data sets	No. Points	No. Attributes	No. Clusters	optimal k	Time (s)
Leukemia 2	34	7129	2	2	81
Leukemia 3	72	7129	3	3	182
Leukemia 7	327	12558	7	7	790
Embryonal Tumors	60	7129	2	2	133
Colon Tumor	62	2000	2	2	41

5.4.3 Accuracy

In order to evaluate the algorithm accuracy, we will be using Rand index and PWCO (percent well classified objects) as correctness evaluation. We compare our approach with k–Means (Macqueen [1967]) and k–Medoids (Kaufman and Rousseeuw [1987]) clustering algorithms on different data sets. For k–Means and k–Medoids clustering algorithms, we fixed k (number of clusters) as the known real number of data set clusters, and then we apply respective algorithms. For our approach (EECA), we run with 12 EAs and $k_{max} = 10$ to find the optimal number of clusters and potential clusters seeds. The Rand index and PWCO are computed according to affected points to respective clusters seeds.

Rand index is defined as :

$$Rand = \frac{a+b}{a+b+c+d}$$

With given a data set S of n elements and two partitions $X = \{X_1, \ldots, X_r\}$ which is a partition of S into r subsets and a partition $Y = \{Y_1, \ldots, Y_s\}$ of S into s subsets. a represents the number of pairs of elements in S that are in the same set in X and in the same set in Y, b represents the number of pairs of elements in S that are in different sets in X and in different sets in Y, c represents the number of pairs of elements in S that are in the same set in X and in different sets in Y and d represents the number of pairs of elements in S that are in different sets in X and in the same set in Y.

The other index is PWCO, which is defined as:

$$PWCO = \frac{1}{n}\sum_{i=1}^{k} Correct(w_i, c_i).$$

where w_i is the partition of cluster i, c_i is the real partition of this class, $Correct(w_i, c_i)$ represents the number of correct points in cluster i when compare its partition cluster with real known class result, n is the total number of points, and k is the clusters number.

5.4.3.1 Test performance on multi-class data sets

Due to the random initialization of k–Means and k–Medoids clustering algorithms, the clustering results may be influenced by its initialization. The table 5.5 presents the evaluation

indexes of different clustering algorithms on multi-class data sets, and it is shown that our approach is competitive with other classical clustering algorithms. The principal advantage of our approach is that we don't need to know the number of clusters and in the same time, find the potential clusters seeds that concur perfectly with those founded with k–Means and k–Medoids clustering algorithms, knowing that we had in input knowledge of the clusters number. We note that, the initial points of k–Means and k–Medoids are randomly chosen from given points.

As we can see in the table 5.5 were the data sets represents multi-class patterns between 2 and 8 clusters, EECA find a greater of equal the PWCO and Rand index values in 4 data sets, and EECA find PWCO and Rand index results are very close for the two last data sets.

5.4.3.2 Test performance on high dimensional data sets

In order to test the efficiency of EECA on high dimensional data sets clustering, we choose five data sets from Kent Ridge Bio-medical data sets Repository (Jinyan and Huiqing) and/or from InfoSel++ library (Jacek et al.). As we can see in the table 5.6, in Embrional Tumors data set, our method is better both k–Means and k–Medoids algorithms. In other cases we are as well as one of the two other clustering algorithms. In Leukemia 7, EECA is better than k–Medoids, same as with Lukemia 2, where k–Means presents the best. In Leukemia 3, EECA is better than k–Means, where k–Medoids presents a better PWCO.

5.4.3.3 Test performance on overlapped data sets

We generate artificial data sets to validate proposed algorithm for multi-class overlapped data sets clustering performance. Three data sets are generated by MixSim software (Melnykov et al. [2012]) based on R language. The MixSim package has a parameter $BarOmega$ control the overlapping degree. Each data set consists of 500 points on 2 dimensions with 5 clusters: MS-001 (figure 5.5), MS-005 (figure 5.7), MS-008 (figure 5.9), with $BarOmega = \{0.01, 0.05, 0.08\}$ respectively. The figures 5.5, 5.7 and 5.9 shown the different generated data sets and how clusters in the different data set are overlapped.

We then apply our approach EECA, to find automatically the optimal number of clusters and the potential cluster seeds, with running 12 EAs and $k_{max} = 10$.

To respect the overlapping degree of the data sets, we replace the overlapping function by the new formula:

$$\sum_{i \neq j}^{k} Card(C_i \cap C_j) = \alpha * n$$

With α represent percentage of data points that we accept to be in the intersection of some

Table 5.5: Performance on multi-class data sets

Data set	Algorithm	PWCO (%)	Rand index
Synthetic	EECA	**100**	**1.00**
	k–Medoids	**100**	**1.00**
	k–Means	**100**	**1.00**
Iris	EECA	**90.67**	**0.89**
	k–Medoids	89.33	0.88
	k–Means	89.33	0.88
Vehicule	EECA	36.76	0.50
	k–Medoids	34.40	**0.65**
	k–Means	**37.35**	**0.65**
Glass	EECA	**49.53**	0.67
	k–Medoids	42.99	0.66
	k–Means	44.86	**0.68**
Blood Transfusion	EECA	**76.87**	**0.64**
	K-Medoids	57.35	0.51
	k–Means	75.27	0.63
Ecoli	EECA	59.23	0.79
	k–Medoids	**66.67**	**0.83**
	k–Means	60.42	0.81
Breast Tissue	EECA	40.57	0.63
	k–Medoids	**51.89**	**0.79**
	k–Means	41.51	**0.79**

Table 5.6: Performance on high dimensional data sets

Data set	Algorithm	PWCO (%)	Rand index
Colom Tumor	EECA	51.61	0.49
	k–Medoids	53.23	0.49
	k–Means	**54.84**	**0.50**
Embryonal Tumors	EECA	**65.00**	**0.54**
	k–Medoids	46.67	0.49
	k–Means	45.00	0.50
Leukemia 2	EECA	61.76	0.51
	k–Medoids	61.76	0.51
	k–Means	**70.59**	**0.57**
Leukemia 3	EECA	43.06	0.43
	k–Medoids	**51.39**	**0.55**
	k–Means	41.67	0.53
Leukemia 7	EECA	55.05	0.73
	k–Medoids	53.21	0.76
	k–Means	**61.16**	**0.83**

clusters. Without forgetting to verify that whole data are contained in the different clusters:

$$\sum_{i \neq j}^{k} Card(C_i \cup C_j) = n$$

The figures 5.6, 5.8 and 5.10, represent the projected data sets with in red (square shape) potential seeds that EECA found. As we can see in these figures, the EECA detect correctly the different seeds. Indeed the optimal number of cluster is 5 and correspond on the real number of clusters, in the projection, the optimal founded seeds have a correct positions for all clusters. EECA find the optimal seeds with the value of $\alpha = 10\%$. We can conclude that our approach can be applied on the overlapping data sets. As we can see in figure 5.11 and figure 5.12, where represent the founded seeds in red (square shape), applying k–Means clustering algorithm with respectively $k = 4$ and $k = 5$. EECA results are better than or as well as k–Means results, without forgetting that k–Means needs to fixed the k input parameter. The projected seeds in the figure 5.12 are obtained with fixing k to 5 but their positions are not already optimal unlike EECA where find optimal seeds without fixing k.

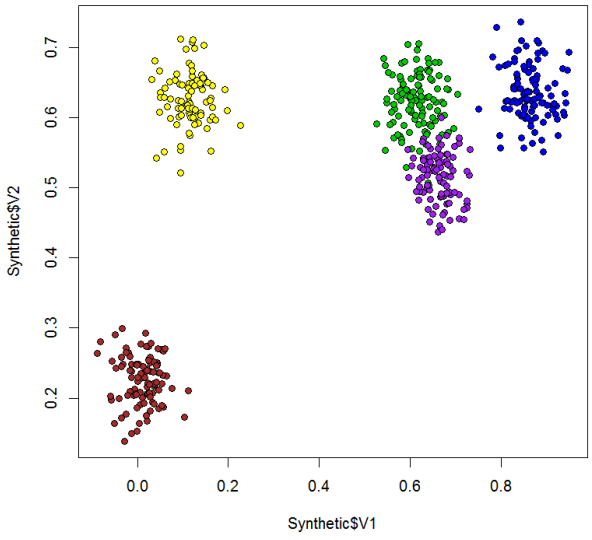


Figure 5.5: MS-001 Data set

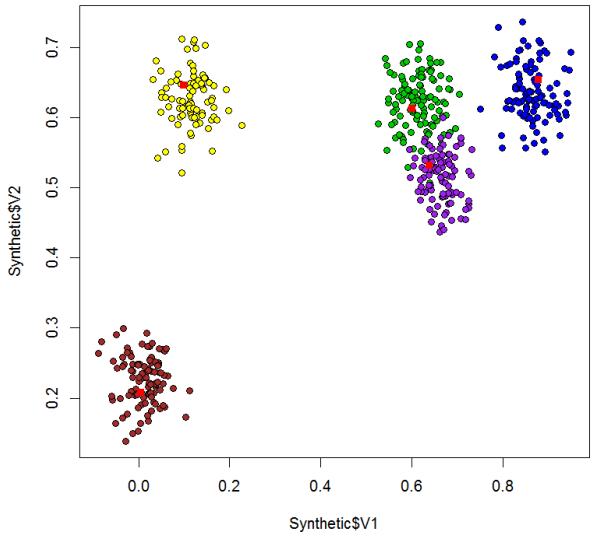


Figure 5.6: EECA cluster seeds on MS-001

5.4.4 Initialization methods performance

In order to evaluate how our approach is benefit as initialization methods, we test the performance of k–Means and k–Medoids clustering algorithms with fixing initial points as the seeds founded by EECA. We fist apply EECA, we found the optimal number of cluster

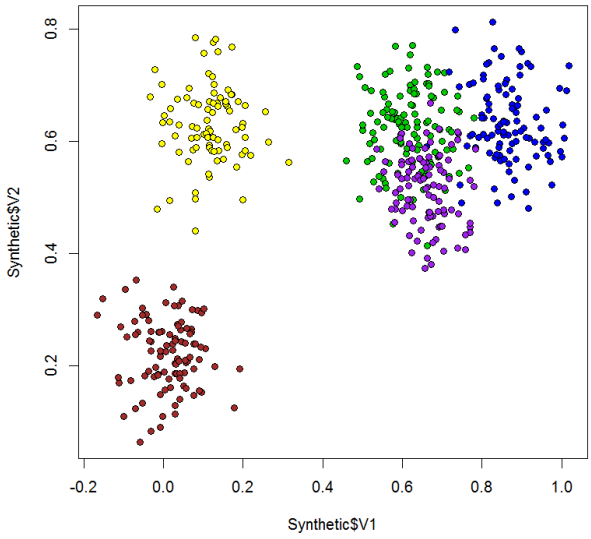


Figure 5.7: MS-005 Data set

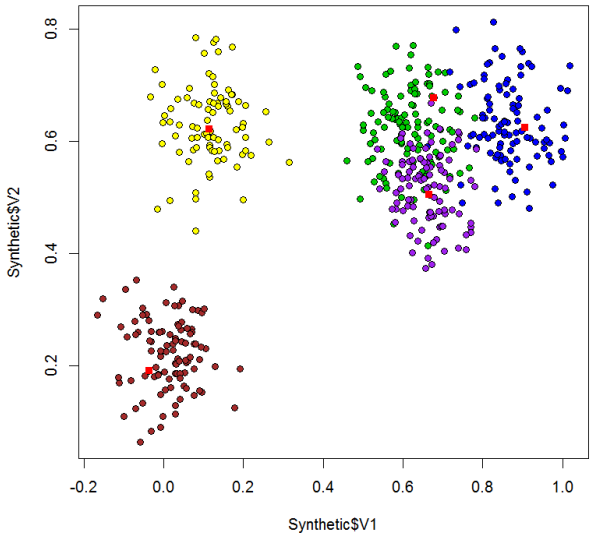


Figure 5.8: EECA cluster seeds on MS-005

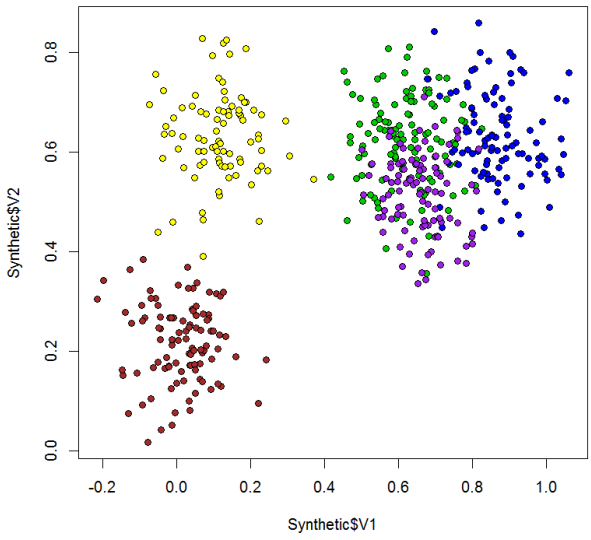


Figure 5.9: MS-008

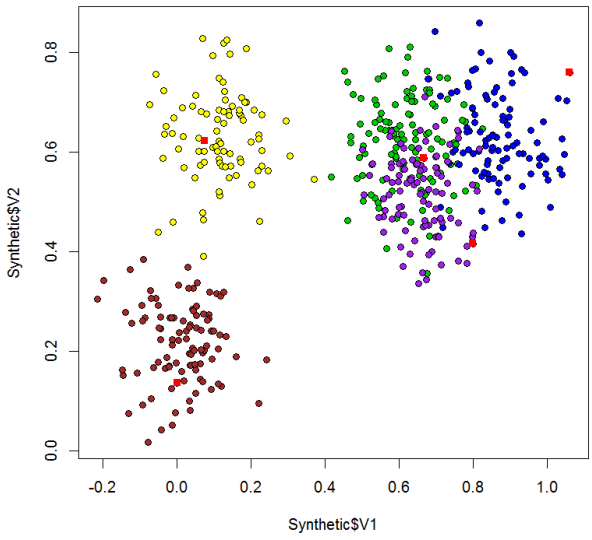


Figure 5.10: EECA clusters seeds on MS-008

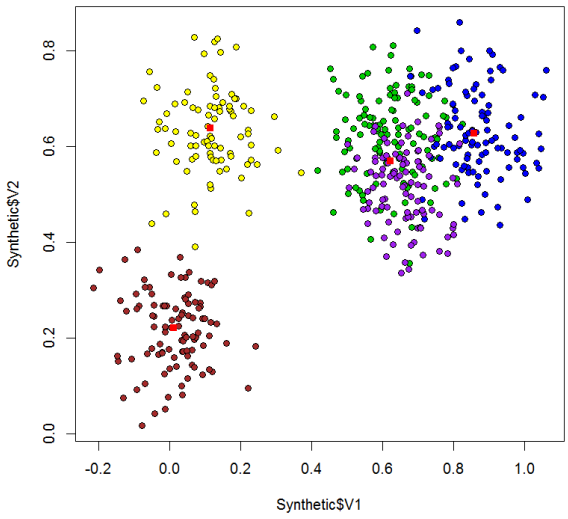


Figure 5.11: k–Means clusters seeds on MS-008 Data set with k=4

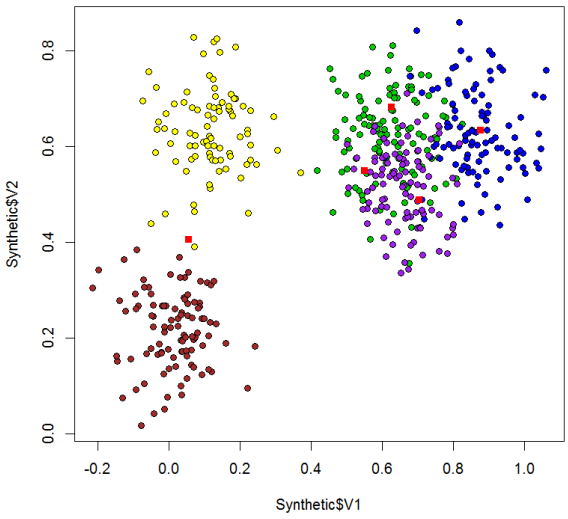


Figure 5.12: k–Means clusters seeds on MS-008 Data set with k=5

and the potential cluster seeds. We then fixed k and apply k-Means and k-Medoids with parameters the identification of initial seed points.

The table 5.7 shown in the third column (column *Algorithm Alone PWCO*), the different results of stand alone clustering algorithms with fixing only k for k-Means and k-Medoids algorithms. The initial points are choosing randomly.

In the last colon (column *With optimal initial point*), the results shown the accuracy where applying algorithms with fixed parameters founded with EECA (k and initial seeds).

As we can see in the table 5.7, with using EECA initial seeds the classical algorithms improve their performances for the majority of data sets. These results proofs that our approach is as effective as classical algorithms and even better in some cases. It is important to note that our approach find automatically the optimal number of clusters, and thus overcomes the associated problems.

5.4.5 Finding hyperellipsoidal clusters

In order to evaluate how our approach is benefit to other cluster patterns, we test the performance of EECA on generated artificial data sets to validate proposed algorithm for multi-class hyperellipsoidal pattern data sets (figure 5.13). This data is generated by C language over Julia's code (Julia and Joshua).

As we can see in the figure 5.14, EECA finds the optimal seeds corresponding to the dense clusters. Indeed the optimal number of cluster is 4 and EECA find 7 centers. In the pro-

Table 5.7: Algorithms correctness comparison

Datasets	Algorithms	Algorithm Alone PWCO(%)	With optimal initial point(%)
Synthetic	EECA	**100**	-
	k–Medoids	**100**	**100**
	k–Means	**100**	**100**
Iris	EECA	**90.67**	-
	k–Medoids	89.33	**90.67**
	k–Means	89.33	89.33
Vehicule	EECA	36.76	-
	k–Medoids	34.40	34.40
	k–Means	37.35	**40.31**
Glass	EECA	**49.53**	-
	k–Medoids	42.99	42.99
	k–Means	44.86	**49.53**
Blood Transfusion	EECA	**76.87**	-
	K-Medoids	57.35	58.16
	k–Means	75.27	68.32
Ecoli	EECA	59.23	-
	k–Medoids	66.67	66.67
	k–Means	60.42	**73.81**
Breast Tissue	EECA	40.57	-
	k–Medoids	**51.89**	**51.89**
	k–Means	41.51	50.00
Colom Tumor	EECA	51.61	-
	k–Medoids	53.23	53.23
	k–Means	**54.84**	51.61
Embryonal Tumors	EECA	**65.00**	-
	k–Medoids	46.67	46.67
	k–Means	45.00	45.00
Leukemia 2	EECA	61.76	-
	k–Medoids	61.76	61.76
	k–Means	**70.59**	**70.59**
Leukemia 3	EECA	43.06	-
	k–Medoids	51.39	51.39
	k–Means	41.67	**62.50**
Leukemia 7	EECA	55.05	-
	k–Medoids	53.21	53.21
	k–Means	61.16	**66.06**

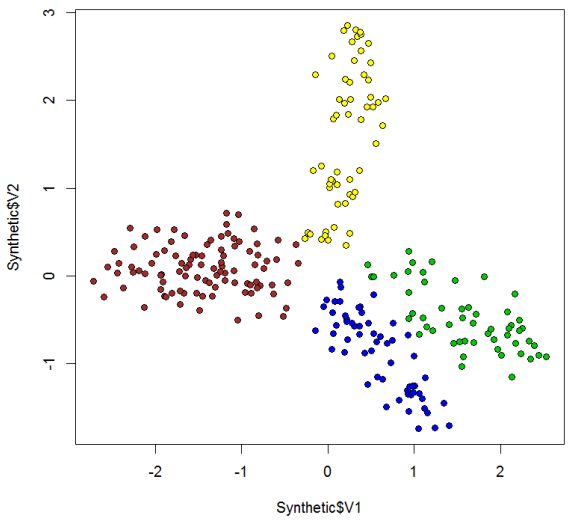


Figure 5.13: Projection of hyperellipsoidal clusters

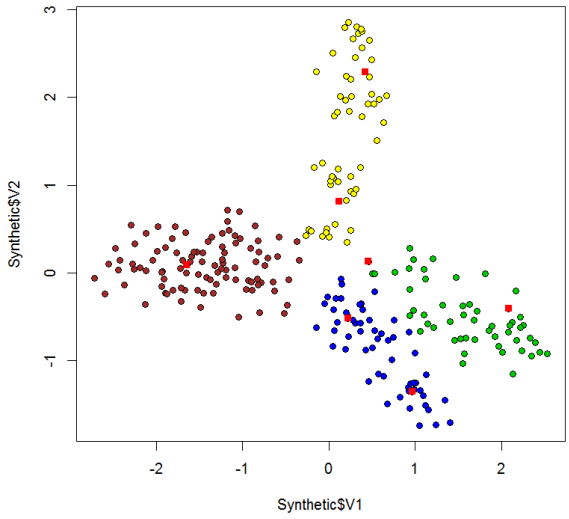


Figure 5.14: EECA optimal cluster seeds on hyperellipsoidal clusters

jection, the optimal founded seeds have correct positions for all dense clusters: two dense regions in the yellow, blue and green clusters, but only one ellipsoidal dense cluster (in brown). We also conclude that our approach can be applied on the different patterns of cluster that present dense patterns, it just need to adapt the distance measure, in this case we use the Mahalanobis distance (Mahalanobis [1936]).

5.5 Conclusion

This chapter proposes a new method that finds in the same time the initial clusters seeds with the optimal cluster number without a prior knowledge of cluster number in reasonable execution time. We don't use any clustering algorithm to evaluate our solution, so, our method can be used as a stand alone clustering algorithm dealing with hyperspherical and hyperellipsoidal clusters. We propose a new mutation strategy using neighborhood search and we use an automatic radius limit detection method in this strategy. We also introduce a new combined fitness functions to evaluate our solution.

To deal with the problem of involving numerous parameters, we propose an Elitist Evolutionary approach that involves numerous evolutionary algorithms (EAs). The difference between them is implemented by the parameters and we select only the best concurrent solution. This proposition can also address to the problem of exploration the large search space. The initial populations of numerous EAs are substantially different, so, we can deal with the large data sets in few numbers of iterations.

We provide several experiments on data sets with different sizes, patterns and overlapped structures. Our results are promising, and the proposed clustering methodology produces high clustering success. These results demonstrate our approach is as effective as classical algorithms, even better in some cases. It is important to note that our approach find automatically the optimal clusters number, and thus overcome the related problems.

In the further work, we intend to test our approach on different subspaces, we think that the optimal clustering can be different according the subspace data projection. Another possibility would be to explore our method on multi-view clustering or on subspaces clustering. We can also investigate the possibility to apply our approach to data stream, with assuming the possibility to treat the data on sub-time windows.

Conclusion and Future works

In this thesis, we have focused on four classes of problems in data clustering: clustering data streams, clustering massive data sets, weighted hard and fuzzy clustering and finally the clustering without a prior knowledge of the clusters number.

We developed DC programming and DCA for solving some classes of problems cited before. Our methods are also, based on elitist evolutionary approaches.

We proposed firstly, to use the clustering algorithm DCA-MCSS to deal with data streams using two windows models: fix-windows and sliding windows. We presented a local clustering strategy of clustering data stream by sub windows problem, and DCA-Stream for clustering data stream over sliding windows problem.

For the problem of clustering massive data sets, we proposed DCA based algorithms with two phases. In the first phase, the data is divided into subsets, on which we applied DCA-MSSC algorithm for clustering. In the second phase, we developed a DCA scheme for the problem clustering on the weighted centers set obtained in first phase.

The relevant attributes are useful for clustering process, thus we have investigated DC programming and DCA on two models: bilevel and mixed integer program using attribute weights. Two models were recast as DC programs, one of them is based on the reformulation technique and exact penalty in DC programming. Then, we proposed appropriate DC decompositions and corresponding DCA. We also presented an extended model of weighted attributes based on DCA, which is weighted feature fuzzy clustering model. We applied our approach on the image segmentation problem.

The final issue addressed in this thesis is the clustering without a prior knowledge of the clusters number. We proposed an elitist evolutionary approach, where we applied several evolutionary algorithms (EAs) at the same time, to find the optimal combination of initial clusters seed and in the same time the optimal clusters number.

We tested our algorithms on both synthetic data set and real–world data set on different parameter settings. The experiment results illustrated the efficiency of our proposed algorithms and its superiority with respect to standard algorithms.

Concerning the future works, we plan to develop new models for maintaining the information structure and/or summarizing statistics of stream, to deal with data stream.

We will study the use of attribute weights in the context of clustering data stream, clustering massive data sets or features selection problems. The studies of the DC decomposition as well as the strategies of initial points in the DC algorithms are still open issues.

The partition of data in clusters as well as the number of clusters may change over time. At two different times, one cluster can be split, merge, delete or a new cluster can appear, one data point can move from this cluster to another cluster. Some of the proposed methods such as evolutionary elitist approach can be improved to propose some solutions for these problems.

We can also investigate the possibility of combining the DCA approach with elitist evolutionary approach to determine the number of groups and clustering tasks and/or mining data stream with sub-windows strategy.

We may consider parallel processing of our approach in experiments: perform parallel clustering in each sub–window; in the first stage of clustering massive data sets problem; or execute parallel each evolutionary algorithm.

In the future, we intend to apply the DC Programming and DCA in other domains such as outlier detection, intrusion detection system, also classification problems, ... We believe that DCA is an innovative approach in data mining, as well as in nonconvex programming, non smooth and/or large-scale problems.

References

Ajith Abraham, Nadia Nedjah, and Luiza de Macedo Mourelle. Evolutionary computation: from genetic algorithms to genetic programming. In *Genetic Systems Programming*, pages 1–20. 2006.

Charu C. Aggarwal and Philip S. Yu. A framework for clustering uncertain data streams. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 150–159, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-1-4244-1836-7.

Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Very large Database*, pages 81–92, 2003.

Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 852–863. VLDB Endowment, 2004. ISBN 0-12-088469-0.

Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, and Salvador García. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

Daniel Aloise, Pierre Hansen, and Leo Liberti. An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematica Programming*, 131(1-2):195–220, 2012.

A. Auslender. *Optimisation: méthodes numériques*. Maîtrise de mathématiques et applications fondamentales. Masson, 1976. ISBN 978-2225429002.

Brain Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 234–243, New York, NY, USA, 2003. ACM. ISBN 1-58113-670-6.

Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA, 2002. ACM. ISBN 1-58113-507-6.

G. Phanendra Babu and M. Narasimha Murty. A near-optimal initial seed value selection in k-means algorithm using a genetic algorithm. *Pattern Recognition Letters*, 14(10):763–769, October 1993. ISSN 0167-8655.

G.Phanendra Babu and M.Narasimha Murty. Simulated annealing for selecting optimal initial seeds in the k-means algorithm. *Indian Journal of Pure & Applied Mathematics*, 25(1-2):85–94, 1994. ISSN 0019-5588.

Serge Belongie, Chad Carson, Hayit Greenspan, and Jitendra Malik. Color- and texture-based image segmentation using em and its application to content-based image retrieval. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 675–682, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 81-7319-221-9.

Jurgen Beringer and Eyke Hullermeier. Online clustering of parallel data streams. *Data & Knowledge Engineering*, 58(2):180 – 204, 2006. ISSN 0169-023X.

James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. ISBN 0306406713.

J. Blackard. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131–151, December 1999. ISSN 01681699.

Tobias Blickle. *Theory of Evolutionary Algorithms and Application to System Synthesis*. PhD thesis, Swiss Federal Institute of Technology, Zurich, November 1996.

Paul S. Bradley and Olvi L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, pages 82–90, 1998.

Paul S. Bradley, Usama M. Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. In *Proceeding Fourth International Conference Knowledge Discovery and Data Mining, KDD-1998*, pages 9–15. American Association for Artificial Intelligence Press, August 1998.

T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-Simulation and Computation*, 3(1):1–27, 1974.

Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *In 2006 SIAM Conference on Data Mining*, pages 328–339, 2006.

Daniel B. Carr, Richard J. Littlefield, and Wesley L. Nichloson. Scatterplot matrix techniques for large n. In *Proceedings of the Seventeenth Symposium on the Interface of Computer Sciences and Statistics on Computer Science and Statistics*, pages 297–306, New York, NY, USA, 1986. Elsevier North-Holland, Inc. ISBN 0-444-70018-8.

M. Emre Celebi, Hassan A. Kingravi, and Patricio A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

A. Chambolle, R. A. De Vore, Nam-Yong Lee, and B. J. Lucier. Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage. *IEEETrans. Image Process.*, 7(3):319–335, March 1998. ISSN 1057-7149.

Elaine Y. Chan, Wai-Ki Ching, Michael K. Ng, and Joshua Zhexue Huang. An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern Recognition*, 37 (5):943–952, 2004.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM TIST*, 2(3):27, 2011.

Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 378–, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0409-4.

Min-Rong Chen and Yong-Zai Lu. A novel elitist multiobjective optimization algorithm: Multiobjective extremal optimization. *European Journal of Operational Research*, 188(3): 637–651, 2008.

O. Cheng Soon and H. A. Le Thi. Learning sparse classifiers with difference of convex functions algorithms. *Optimization Methods and Software*, 28(4):830–854, 2013.

Xuan Hong Dang, Vincent C. S. Lee, Wee Keong Ng, Arridhana Ciptadi, and Kok-Leong Ong. An em-based algorithm for clustering data streams in sliding windows. In *DASFAA*, pages 230–235, 2009a.

Xuan Hong Dang, Vincent C. S. Lee, Wee Keong Ng, and Kok-Leong Ong. Incremental and adaptive clustering stream data over sliding window. In Sourav S. Bhowmick, Josef Küng, and Roland Wagner, editors, *DEXA*, volume 5690 of *Lecture Notes in Computer Science*, pages 660–674. Springer, 2009b. ISBN 978-3-642-03572-2.

Swagatam Das, Ajith Abraham, and Amit Konar. Automatic kernel clustering with a multielitist particle swarm optimization algorithm. *Pattern Recognition Letters*, 29(5):688–699, 2008.

David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 1(2):224–227, February 1979. ISSN 0162-8828.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977.

Huyen Do, Alexandros Kalousis, and Melanie Hilario. Feature weighting using margin and radius based error bound optimization in svms. In *ECML/PKDD (1)*, pages 315–329, 2009.

J. C. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104, 1974.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

Fredrik Farnstrom, James Lewis, and Charles Elkan. Scalability for clustering algorithms revisited. *ACM SIGKDD Explorations Newsletter*, 2:51–57, June 2000. ISSN 1931-0145.

Alhussein Fawzi, Mike Davies, and Pascal Frossard. Dictionary learning for fast classification based on soft-thresholding. *CoRR*, abs/1402.1973, 2014.

David B. Fogel. The advantages of evolutionary computation. In *Biocomputing and Emergent Computation: Proceedings of BCEC97*, pages 1–11. World Scientific Press, 1997. ISBN 981-02-3262-4.

Hichem Frigui and Olfa Nasraoui. Unsupervised learning of prototypes and attribute weights. *Pattern Recognition*, 37(3):567–581, 2004.

Gilles Gasso, Alain Rakotomamonjy, and Stéphane Canu. Recovering sparse signals with a certain family of nonconvex penalties and DC programming. *IEEE Transactions on Signal Processing*, 57:4686–4698, 2009.

Shuiping Gou, Xiong Zhuang, Yangyang Li, Cong Xu, and Licheng Jiao. Multi-elitist immune clonal quantum clustering algorithm. *Neurocomputing*, 101:275–289, 2013.

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: An efficient clustering algorithm for large databases. *SIGMOD Rec.*, 27(2):73–84, June 1998. ISSN 0163-5808.

Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *41st Annual Symposium on Foundations of Computer Science, FOCS, 12-14 November 2000, Redondo Beach, California, USA*, pages 359–366. IEEE Computer Society, 2000. ISBN 0-7695-0850-2.

Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, March 2003. ISSN 1041-4347.

Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques, 1985.

Reiner Horst and Hoang Tuy. *Global optimization - deterministic approache*. Springer, 1996. ISBN 978-3-540-61038-0.

C M Huang and R W Harris. A comparison of several vector quantization codebook generation approaches. *IEEE Transactions on Image Processing*, 2(1):108–112, 1993. ISSN 1057-7149.

Joshua Zhexue Huang, Michael K. Ng, Hongqiang Rong, and Zichen Li. Automated variable weighting in k-means type clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(5): 657–668, 2005.

Biesiada Jacek, Blachnik Marcin, Duch Wlodzislaw, and Kachel Adam. Infosel++ library. http://kzi.polsl.pl/~jbiesiada/Infosel/files/datasets.html. Accessed on March 2014.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999. ISSN 0360-0300.

Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8): 651–666, 2010.

Cheqing Jin, JeffreyXu Yu, Aoying Zhou, and Feng Cao. Efficient clustering of uncertain data streams. *Knowledge and Information Systems*, pages 1–31, 2013. ISSN 0219-1377.

Liping Jing, Michael K. Ng, and Joshua Zhexue Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1026–1041, 2007. ISSN 1041-4347.

L. Jinyan and L. Huiqing. Kent ridge bio-medical data set repository. http://datam.i2r.a-star.edu.sg/datasets/krbd/index.html. Accessed on March 2014.

Joaquim Júdice, Marcos Raydan, Silvério S. Rosa, and Sandra Augusta Santos. On the solution of the symmetric eigenvalue complementarity problem by the spectral projected gradient algorithm. *Numerical Algorithms*, 47(4):391–407, 2008.

Handl Julia and Knowles Joshua. Cluster generators for large high-dimensional data sets with large numbers of clusters. http://personalpages.manchester.ac.uk/mbs/Julia.Handl/generators.html. Accessed on March 2014.

L. Kaufman and P. Rousseeuw. *Clustering by Means of Medoids*. Reports of the Faculty of Mathematics and Informatics. Faculty of Mathematics and Informatics, 1987.

E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering homepage [www.cs.ucr.edu/ẽamonn/time_series_data/], 2006. accessed Feb. 2008.

E. Kokiopoulou, D. Kressner, N. Paragios, and P. Frossard. Optimal image alignment with random projections of manifolds: algorithm and geometric analysis. In *Proceedings of EUSIPCO*, 2009.

Petra Kudová. Clustering genetic algorithm. In *Database and Expert Systems Applications, (DEXA'07) Workshops*, pages 138–142, 2007.

P.J. Laurent. *Approximation et optimisation*, volume 1 of *Collection Enseignement des sciences*. Université Scientifique et Médicale de Grenoble, 1972.

M. Le Hoai and M. T. Ta. Dc programming and dca for solving minimum sum-of-squares clustering using weighted dissimilarity measures. *Transaction Computational Collective Intelligence*, 13:113–131, 2014.

M. Le Hoai, H. A. Le Thi, T. Pham Dinh, and Pascal Bouvry. A deterministic optimization approach for generating highly nonlinear balanced boolean functions in cryptography. In HansGeorg Bock, Ekaterina Kostina, HoangXuan Phu, and Rolf Rannacher, editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 381–391. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-79408-0.

M. Le Hoai, H. A. Le Thi, T. Pham Dinh, and Pascal Bouvry. A combined dca: Ga for constructing highly nonlinear balanced boolean functions in cryptography. *Journal of Global Optimization*, 47(4):597–613, 2010. ISSN 0925-5001.

M. Le Hoai, Adnan Yassine, and Riadh Moussi. Dca for solving the scheduling of lifting vehicle in an automated port container terminal. *Computational Management Science*, 9 (2):273–286, 2012. ISSN 1619-697X.

M. Le Hoai, H. A. Le Thi, and M. C. Nguyen. Dca based algorithms for feature selection in semi-supervised support vector machines. In *Machine Learning and Data Mining in Pattern Recognition*, volume 7988 of *Lecture Notes in Computer Science*, pages 528–542. 2013a. ISBN 978-3-642-39711-0.

M. Le Hoai, H. A. Le Thi, T. Pham Dinh, and V. N. Huynh. Block clustering based on difference of convex functions (dc) programming and dc algorithms. *Neural Computation*, 25(10):2776–2807, 2013b.

M. Le Hoai, T. B. T. Nguyen, M. T. Ta, and H. A. Le Thi. Image segmentation via feature weighted fuzzy clustering by a dca based algorithm. In *Advanced Computational Methods for Knowledge Engineering*, volume 479 of *Studies in Computational Intelligence*, pages 53–63. 2013c. ISBN 978-3-319-00292-7.

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, Université de Rouen, 1994.

H. A. Le Thi. *Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algoritmes et Applications.* Habilitation à diriger des recherches, Université de Rouen, 1997.

H. A. Le Thi. Dc programming and dca in machine learning. Technical report, University of Lorraine, 2012. Submitted.

H. A. Le Thi and M. Moeini. Long-short portfolio optimization under cardinality constraints by difference of convex functions algorithm. *Journal of Optimization Theory and Applications*, 161(1):199–224, 2014. ISSN 0022-3239.

H. A. Le Thi and T. Pham Dinh. Solving a class of linearly constrained indefinite quadratic problems by dc algorithms. *Journal of Global Optimization*, 11(3):253–285, 1997.

H. A. Le Thi and T. Pham Dinh. Dc programming approach for solving the multidimensional scaling problem. *Nonconvex Optimizations and Its Applications: Special Issue From Local to Global Optimization*, pages 231–276, 2001.

H. A. Le Thi and T. Pham Dinh. Dc programming: Theory, algorithms and applications. In *The State of the Proceedings of The First International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02)*, Valbonne-Sophia Antipolis, France, October, 2002.

H. A. Le Thi and T. Pham Dinh. Large-scale molecular optimization from distance matrices by a d.c. optimization approach. *SIAM Journal on Optimization*, 14(1):77–114, 2003.

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 Operation Research*, 133(1-4):23–46, 2005.

H. A. Le Thi and T. Pham Dinh. Dc programming in communication systems: challenging problems and methods. *Vietnam Journal of Computer Science*, 1(1):15–28, 2014. ISSN 2196-8888.

H. A. Le Thi and D. Q. Tran. Solving continuous min max problem for single period portfolio selection with discrete constraints by DCA. *Optimization*, 61(8):1025–1038, 2012.

H. A. Le Thi and D. Q. Tran. Optimizing a multi-stage production/inventory system by dc programming based approaches. *Computational Optimization and Applications*, 57(2): 441–468, 2014. ISSN 0926-6003.

H. A. Le Thi, T. Pham Dinh, and M. Le Dung. Exact penalty in d.c. programming. *Vietnam Journal of Mathematics*, 27(2):169–178, 1999.

H. A. Le Thi, T. Pham Dinh, and V.T. Nguyen. Combination between local and global methods for solving an optimization problem over the efficient set. *European Journal of Operational Research*, 142:257–270, 2002.

H. A. Le Thi, M. T. Belghiti, and T. Pham Dinh. A new efficient algorithm based on dc programming and dca for clustering. *Journal of Global Optimization*, 37(4):593–608, 2007a.

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*, 183(3):1067–1085, 2007b.

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. *Adv. Data Analysis and Classification*, 1(2):85–104, 2007c.

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. *European Journal of Operational Research*, 183(3):1001 – 1012, 2007d. ISSN 0377-2217.

H. A. Le Thi, M. Le Hoai, T. P. Nguyen, and T. Pham Dinh. Noisy image segmentation by a robust clustering algorithm based on dc programming and dca. In *ICDM*, pages 72–86, 2008a.

H. A. Le Thi, M. Le Hoai, V. V. Nguyen, and T. Pham Dinh. A dc programming approach for feature selection in support vector machines learning. *Adv. Data Analysis and Classification*, 2(3):259–278, 2008b.

H. A. Le Thi, M. Le Hoai, V. V. Nguyen, and T. Pham Dinh. Combined feature selection and classification using dca. In *IEEE International Conference on Research, Innovation and Vision for the future in Computing & Communications Technologies, Ho Chi Minh (RIVF 2008)*, pages 233–239, July 2008c.

H. A. Le Thi, Q. T. Nguyen, K. Phan Tran, and T. Pham Dinh. Energy minimization-based cross-layer design in wireless networks. In *Proceeding of the High Performance Computing & Simulation Conference (HPCS 2008), Nicosia, Cyprus*, pages 283–289, June 3 - 6, 2008d.

H. A. Le Thi, V. V. Nguyen, and O. Samir. Gene selection for cancer classification using dca. In *Advanced Data Mining and Applications*, volume 5139 of *Lecture Notes in Computer Science*, pages 62–72. 2008e. ISBN 978-3-540-88191-9.

H. A. Le Thi, M. Moeini, and T. Pham Dinh. Portfolio selection under downside risk measures and cardinality constraints based on dc programming and dca. *Computational Management Science*, 6(4):459–475, 2009a. ISSN 1619-697X.

H. A. Le Thi, M. Moeini, and T. Pham Dinh. DC programming approach for portfolio optimization under step increasing transaction costs. *Optimization journal*, 58(3):267–289, 2009b.

H. A. Le Thi, Q. T. Nguyen, H. T. Nguyen, and T. Pham Dinh. Solving the earliness tardiness scheduling problem by dc programming and dca. *Mathematica Balkanica,*, pages 271–288, 2009c.

H. A. Le Thi, Q. T. Nguyen, H. T. Nguyen, and T. Pham Dinh. A time-indexed formulation of earliness tardiness scheduling via dc programming and dca. In *Computer Science and Information Technology, 2009. IMCSIT '09. International Multiconference on*, pages 779–784, Oct 2009d.

H. A. Le Thi, T. Pham Dinh, and S. Bouallagui. Cryptanalysis of an identification scheme based on the perceptron problem using a hybridization of deterministic optimization and genetic algorithm. In *Proceedings of the 2009 International Conference on Security and Management, World Congress in Computer Science Computer Engineering, and Applied Computing, Las Vegas, USA*, pages 117–123, July 13-16 2009e.

H. A. Le Thi, T. Pham Dinh, and V. N. Huynh. Exact penalty techniques in dc programming. *Journal of Global Optimization*, pages 1–27, 2011.

H. A. Le Thi, T. Pham Dinh, and V.N. Huynh. Exact penalty and error bounds in dc programming. *J. Global Optimization*, 52(3):509–535, 2012a.

H. A. Le Thi, T. Pham Dinh, and D. Q. Tran. A DC programming approach for a class of bilevel programming problems and its application in portfolio selection. *Numerical Algebra, Control and Optimization (NACO)*, 2(1):167–185, 2012b.

H. A. Le Thi, M. T. Le, and T. B. T. Nguyen. A novel approach to automated cell counting based on a difference of convex functions algorithm (dca). In *Computational Collective Intelligence. Technologies and Applications*, volume 8083 of *Lecture Notes in Computer Science*, pages 336–345. 2013a. ISBN 978-3-642-40494-8.

H. A. Le Thi, M. Le Hoai, T. Pham Dinh, and V.N. Huynh. Binary classification via spherical separator by dc programming and dca. *J. Global Optimization*, 56(4):1393–1407, 2013b.

H. A. Le Thi, T. B. T. Nguyen, and M. Le Hoai. Sparse signal recovery by difference of convex functions algorithms. In *Intelligent Information and Database Systems*, volume 7803 of *Lecture Notes in Computer Science*, pages 387–397. 2013c. ISBN 978-3-642-36542-3.

H. A. Le Thi, D. Q. Tran, and H. A. Kondo. A difference of convex functions algorithm for optimal scheduling and real-time assignment of preventive maintenance jobs on parallel processors. *Journal of Industrial and Management Optimization (JIMO)*, pages 1–20, 2013d. ISSN 0925-5001.

H. A. Le Thi, X. T. Vo, and T. Pham Dinh. Robust feature selection for svms under uncertain data. In *Advances in Data Mining. Applications and Theoretical Aspects*, volume 7987 of *Lecture Notes in Computer Science*, pages 151–165. 2013e. ISBN 978-3-642-39735-6.

H. A. Le Thi, P. Damel, P. Nadège, and T. P. Nguyen. The confrontation of two clustering methods in portfolio management: Ward's method versus dca method. In *Advanced Computational Methods for Knowledge Engineering*, volume 282 of *Advances in Intelligent Systems and Computing*, pages 87–98. 2014a. ISBN 978-3-319-06568-7.

H. A. Le Thi, A. V. Le, X. T. Vo, and Z. Ahmed. A filter based feature selection approach in msvm using dca and its application in network intrusion detection. In *Intelligent Information and Database Systems*, volume 8398 of *Lecture Notes in Computer Science*, pages 403–413. 2014b. ISBN 978-3-319-05457-5.

H. A. Le Thi, M. Le Hoai, and T. Pham Dinh. New and efficient dca based algorithms for minimum sum-of-squares clustering. *Pattern Recognition*, 47(1):388–401, 2014c.

H.A. Le Thi and M. Moeini. Portfolio selection under buy-in threshold constraints using dc programming and dca. In *Service Systems and Service Management, 2006 International Conference on*, volume 1, pages 296–300, Oct 2006.

H. A. Le Thi (web site). Dc programming and dca. http://lita.sciences.univ-metz.fr/~lethi. Accessed on March 2014.

Kwong-Sak Leung and Yong Liang. Adaptive elitist-population based genetic algorithm for multimodal function optimization. In *Genetic and Evolutionary Computation - GECCO'2003*, pages 1160–1171, 2003.

Yong Liang and Kwong-Sak Leung. Genetic algorithm with adaptive elitist-population strategies for multimodal function optimization. *Applied Soft Computing*, 11(2):2017–2034, 2011.

Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, 1980.

G. Lemaitre M. Rastgoo, X. Rafael, F. Miralles, and P. Casale. Pruning adaboost for continuous sensors mining applications. In *Ubiquitous Data Mining Works, 20th European Conference in Artificial Intelligence - ECAI*, pages 53–57, Montpellier, France, 27-31 August 2012.

J. Macqueen. Some methods for classification and analysis of multivariate observations. In *5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

P. C. Mahalanobis. On the generalised distance in statistics. *Proceedings of the National Institute of Sciences, India*, 2(1):49–55, 1936.

Thiao Mamadou, T. Pham Dinh, and H. A. Le Thi. A dc programming approach for sparse eigenvalue problem. In *Internationale Conference on Machine learninh ICML 2010*, pages 1063–1070, 2010.

Zahra Asghari Varzaneh Marjan Kuchaki Rafsanjani and Nasibeh Emami Chukanlo. A survey of hierarchical clustering algorithms. *The Journal of Mathematics and Computer Science*, 5(3):229–240, 2012.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra. Mixsim: An r package for simulating data to study performance of clustering algorithms. *Journal of Statistical Software*, 51(12):1–25, 11 2012. ISSN 1548-7660.

Glenn Milligan and Martha Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, June 1985. ISSN 0033-3123.

Mehryar Mohri and Andres Muñoz Medina. Learning theory and algorithms for revenue optimization in second-price auctions with reserve. In *Proceedings of the 31st International Conference on Machine Learning, Beijing, China. JMLR: W&CP*, volume 32, 2014.

M. Mokhtar, A. Shuib, and D. Mohamad. Mathematical programming models for portfolio optimization problem: A review. *International Journal of Social, Human Science and Engineering*, 8(2):76 – 83, 2014. ISSN 1307-6892.

B. Ndiaye, T. Pham Dinh, and H. A. Le Thi. Single straddle carrier routing problem in port container terminals: Mathematical model and solving approaches. In *International Journal of Intelligent Information and Database Systems IJIIDS*, volume 14, pages 21–31. 2008. ISBN 978-3-540-87476-8.

C.L. Blake D.J. Newman and C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. Accessed on March 2014.

R.T. Ng and Jiawei Han. Clarans: a method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 14(5):1003–1016, Sep 2002. ISSN 1041-4347.

C. N. Nguyen, H. A. Le Thi, and T. Pham Dinh. A branch and bound algorithm based on dc programming and dca for strategic capacity planning in supply chain design for a new market opportunity. In Karl-Heinz Waldmann and UlrikeM. Stocker, editors, *Operations Research Proceedings 2006*, volume 2006 of *Operations Research Proceedings*, pages 515–520. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-69994-1.

D. M. Nguyen, H. A. Le Thi, and T. Pham Dinh. A cross-entropy method for value-at-risk constrained optimization. In *Intelligent Information and Database Systems*, volume 6592 of *Lecture Notes in Computer Science*, pages 442–451. 2011. ISBN 978-3-642-20041-0.

Q. T. Nguyen and H. A. Le Thi. Solving an inventory routing problem in supply chain by dc programming and dca. In *Intelligent Information and Database Systems*, volume 6592 of *Lecture Notes in Computer Science*, pages 432–441. 2011. ISBN 978-3-642-20041-0.

Silvia Nittel and Kelvin T. Leung. Parallelizing clustering of geoscientific data sets using data streams. In *16th International Conference on Scientific and Statistical Database Management (SSDBM 2004)*, pages 73–84, 2004. ISBN 0-7695-2146-0.

Silvia Nittel, Kelvin T. Leung, and Amy Braverman. Scaling clustering algorithms for massive data sets using data streams. In *20th International Conference on Data Engineering - ICDE*, page 830, 2004.

Liadan O'Callaghan, Adam Meyerson, Rajeev Motwani, Nina Mishra, and Sudipto Guha. Streaming-data algorithms for high-quality clustering. In *ICDE*, pages 685–694, 2002.

N.R. Pal and S.K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.

G. K. Palshikar. Simple Algorithms for Peak Detection in Time-Series. In *Proceeding 1st International Conference Advanced Data Analysis, Business Analytics and Intelligence*, 2009.

Wei Pan, Xiaotong Shen, and Binghui Liu. Cluster analysis: unsupervised learning via supervised learning with a non-convex penalty. *Journal of Machine Learning Research*, 14 (1):1865–1889, 2013.

V. N. Pham, H. A. Le Thi, and T. Pham Dinh. A dc programming framework for portfolio selection by minimizing the transaction costs. In *Advanced Computational Methods for Knowledge Engineering*, volume 479 of *Studies in Computational Intelligence*, pages 31–40. 2013. ISBN 978-3-319-00292-7.

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.

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.

T. Pham Dinh. *Algorithms for solving a class of non convex optimization problems. Methods of subgradients*, volume 129 of *North-Holland Mathematics Studies*. Elsevier Science Publishers, 1986.

T. Pham Dinh and S. E. Bernoussi. Duality in d. c. (difference of convex functions) optimization. Subgradient methods. Trends in mathematical optimization, 4th French-German Conference, Irsee/FRG 1986, ISNM 84, 277-293, 1988.

T. Pham Dinh and H. A. Le Thi. Convex analysis approach to d.c. programming: theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355, 1997. ISSN 0251-4184.

T. Pham Dinh and H. A. Le Thi. Recent advances in dc programming and dca. *Transactions on Computational Collective Intelligence*, 8342:1–37, 2014.

T. Pham Dinh and H.A. Le Thi. Dc optimization algorithms for solving the trust region subproblem. *SIAM Journal of Optimization*, 8(2):476–505, 1998.

T. Pham Dinh, C. N. Nguyen, and H. A. Le Thi. Dc programming and dca for globally solving the value-at-risk. *Computational Management Science*, 6(4):477–501, 2009. ISSN 1619-697X.

T. Pham Dinh, V. N. Pham, and H. A. Le Thi. Dc programming and dca for portfolio optimization with linear and fixed transaction costs. In *Intelligent Information and Database Systems*, volume 8398 of *Lecture Notes in Computer Science*, pages 392–402. 2014. ISBN 978-3-319-05457-5.

D. N. Phan, M. C. Nguyen, and H. A. Le Thi. A dc programming approach for sparse linear discriminant analysis. In *Advanced Computational Methods for Knowledge Engineering*, volume 282 of *Advances in Intelligent Systems and Computing*, pages 65–74. 2014. ISBN 978-3-319-06568-7.

Sultan Noman Qasem and Siti Mariyam Shamsuddin. Memetic elitist pareto differential evolution algorithm based radial basis function networks for classification problems. *Applied Soft Computing*, 11(8):5565–5581, 2011.

N. J. Radcliffe. Equivalence class analysis and presentation of strong rules. *In Knowledge Discovery in Database*, 11:229–248, 1991.

Jiadong Ren, Ruiqing Ma, and Jiadong Ren. Density-based data streams clustering over sliding windows. In *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery - Volume 5*, FSKD'09, pages 248–252, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-4545-5.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

J. Sarma and K. De Jong. Selection: generation gap methods. In *Handbook on Evolutionary Computation*, pages C2.7:1–C2.7:5. Institute of Physics Publishing and Oxford University Press, Bristol and New York, 1997.

Christoph Schnörr. Signal and image approximation with level-set constraints. *Computing*, 81(2-3):137–160, 2007.

S. Sharma and S. Rai. Genetic k-means algorithm - implementation and analysis. *International Journal of Recent Technology and Engineering (IJRTE)*, 1(2):117–120, June 2012.

Alzennyr Gomes Da Silva. European workshop on data stream analysis. In *Analyzing the Evolution of Web Usage Data*, volume 36, pages 77– 79, Caserta, Italy, 14-16, March 2007.

Wladyslaw Skarbek and Andreas Koschan. Colour image segmentation: A survey, 1994.

W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, USA, 2001. ACM. ISBN 1-58113-391-X.

A. S. Ta, H. A. Le Thi, D. Khadraoui, and T. Pham Dinh. Solving qos routing problems by dca. In *Intelligent Information and Database Systems*, volume 5991 of *Lecture Notes in Computer Science*, pages 460–470. 2010a. ISBN 978-3-642-12100-5.

A. S. Ta, H. A. Le Thi, D. Khadraoui, and T. Pham Dinh. Solving multicast qos routing problem in the context v2i communication services using dca. In *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference, Yamagata, Japan*, pages 471–476, Aug 2010b.

A. S. Ta, H. A. Le Thi, D. Khadraoui, and T. Pham Dinh. Solving partitioning-hub location-routing problem using dca. *Journal of Industrial and Management Optimization*, 8(1): 87–102, 2012a.

A. S. Ta, T. Pham Dinh, H. A. Le Thi, and D. Khadraoui. Solving many to many multicast qos routing problem using dca and proximal decomposition technique. In *International Conference on Computing, Networking and Communications (ICNC2012), Hawaii, American*, pages 809–814, Jan 2012b.

Liang Tang, Chang jie Tang, Lei Duan, Chuan Li, Ye xi Jiang, Chun qiu Zeng, and Jun Zhu. Movstream: An efficient algorithm for monitoring clusters evolving in data streams. In *Granular Computing, 2008. GrC 2008. IEEE International Conference on*, pages 582–587, Aug 2008.

Xilan Tian, Gilles Gasso, and Stéphane Canu. A multiple kernel framework for inductive semi-supervised {SVM} learning. *Neurocomputing*, 90(0):46 – 58, 2012. ISSN 0925-2312. Advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011).

J. F. Toland. Direct calculation of the information matrix via the EM algorithm. *Journal of Mathematical Analysis and Applications*, 66:399–415, 1978.

Li Tu and Yixin Chen. Stream data clustering based on grid density and attraction. *ACM Trans. Knowl. Discov. Data*, 3(3):12:1–12:27, July 2009. ISSN 1556-4681.

J. B. H. Urruty. Generalized differentiability duality and optimization for problem dealing with differences of convex functions. *Lecture Notes in Economics and Mathematical Systems, volume*, 256:260–277, 1986.

Lucas Vendramin, Ricardo J. G. B. Campello, and Eduardo R. Hruschka. On the comparison of relative clustering validity criteria. In *Proceedings of the SIAM International Conference on Data Mining, (SDM 2009)*, pages 733–744, 2009.

J. Verge Llahi. *Color Constancy and Image Segmentation Techniques for Applications to Mobile Robotics*. PhD thesis, Universitat Politècnica de Catalunya, 2005.

H. D. Vinod. Integer programming and the theory of grouping. *Journal of the American Statistical Association*, 64(326):506–519, June 1969. ISSN 0162-1459 (print), 1537-274X (electronic).

Nikola Vucic, Shuying Shi, and Martin Schubert. Dc programming approach for resource allocation in wireless networks. In *WiOpt*, pages 380–386, 2010.

Junhui Wang, Xiaotong Shen, and Wei Pan. On efficient large margin semisupervised learning: Method and theory. *Journal of Machine Learning Research*, 10:719–742, 2009.

Kuaini Wang, Ping Zhong, and Yaohong Zhao. Training robust support vector regression via d. c. program. *Journal of Information & Computational Science*, 7(12):2385–2394, 2010.

Stefan Weber, Thomas Schüle, Attila Kuba, and Christoph Schnörr. Binary tomography with deblurring. In Ralf Reulke, Ulrich Eckardt, Boris Flach, Uwe Knauer, and Konrad Polthier, editors, *Combinatorial Image Analysis*, volume 4040 of *Lecture Notes in Computer Science*, pages 375–388. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-35153-5.

Darrell Whitley. An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and Software Technology*, 43(14):817 – 831, 2001. ISSN 0950-5849.

Changzhi Wu, Chaojie Li, and Qiang Long. A dc programming approach for sensor network localization with uncertainties in anchor positions. *Journal of Industrial and Management Optimization*, 10(3):817 – 826, 2014.

Liming Yang and Laisheng Wang. A class of semi-supervised support vector machines by dc programming. *Adv. Data Analysis and Classification*, 7(4):417–433, 2013.

Yiming Ying, Kaizhu Huang, and Colin Campbell. Enhanced protein fold recognition through a novel data integration approach. *BMC Bioinformatics*, 10:267, 2009.

Alan L. Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

Chen Zhang, Cheqing Jin, and Aoying Zhou. Efficiently clustering probabilistic data streams. In *APWeb/WAIM*, pages 273–284, 2009.

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.

Yiqing Zhong and El Houssaine Aghezzaf. A dc programming approach to solve the single-vehicle inventory routing problem. In *Proceedings of the international conference CIE39*, 2009.

Yiqing Zhong and El Houssaine Aghezzaf. Combining dc-programming and steepest-descent to solve the single-vehicle inventory routing problem. *Computers & Industrial Engineering*, 61(2):313–321, 2011.

Aoying Zhou, Feng Cao, Weining Qian, and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. *Knowl. Inf. Syst.*, 15(2):181–214, 2008.

Xingquan Zhu, Wei Ding, Philip S. Yu, and Chengqi Zhang. One-class learning and concept summarization for data streams. *Knowledge Information Systems*, 28(3):523–553, 2011.

X. Zhu (web site). Stream data mining repository. http://cse.fau.edu/xqzhu/stream.html. Accessed on March 2014.