Contribution au développement des méthodes avancées de la programmation DC et DCA pour certaines classes de problèmes d'optimisation non-convexes. Applications en apprentissage automatique

Hoai Minh Le

UNIVERSITÉ DE LORRAINE

Contribution au développement des méthodes avancées de la programmation DC et DCA pour certaines classes de problèmes d'optimisation non-convexes. Applications en apprentissage automatique

THÈSE

présentée et soutenue publiquement le 8 mars 2022

pour l'obtention d'une

Habilitation à Diriger des Recherches de l'Université de Lorraine

(mention informatique)

par

Hoai Minh Le

Composition du jury

Rapporteurs :	Emilio Carrizosa	PR, Université de Séville
	Jin-Kao Hao	PR, Université d'Angers
	Yaroslav Sergeyev	PR, Université de Calabre
Examinateurs :	Anne Boyer	PR, Université de Lorraine
	Charles Soussen	PR, Centrale Supelec
Marraine scientifique :	Hoai An Le Thi	PR, Université de Lorraine
Invité :	Tao Pham Dinh	PR, INSA-Rouen

Remerciements

En premier lieu, je tiens à remercier Madame le Professeur Hoai An Le Thi qui a accepté d'être ma marraine scientifique et à lui exprimer toute ma gratitude pour m'avoir accompagné, conseillé et soutenu dans ce parcours de préparation de l'habilitation.

Je souhaite exprimer mes plus profonds remerciements à Monsieur Emilio Carrizosa, Professeur à l'Université de Séville, Monsieur Jin-Kao Hao, Professeur à l'Université d'Angers, Monsieur Yaroslav Sergeyev, Professeur à l'Université de Calabre de m'avoir fait l'honneur d'accepter la charge de rapporteur de mon mémoire de HDR.

J'adresse mes plus sincères remerciements à Madame le Professeur Anne Boyer à l'Université de Lorraine, Monsieur le Professeur Charles Soussen à Central Supelec d'avoir participé à juger mon travail.

Je remercie plus particulièrement Monsieur le Professeur Tao Pham Dinh à l'INSA de Rouen pour ses conseils, son suivi de mes recherches, les conseils pertinents et formateurs, les encouragements qu'il a menées pour me suggérer les voies de recherche.

Le travail décrit dans ce mémoire d'HDR n'est pas le fruit de mon seul travail, mais celui de nombreuses collaborations. Je remercie Duy Nhat Phan, Bach Tran, Hoang Duc Hau Luu pour les échanges et réalisations.

Ce mémoire a été préparé au Laboratoire de Génie Informatique, de Production et de Maintenance (LGIPM), Université de Lorraine. Je voudrais remercier chaleureusement les services administratifs, techniques et supports qui m'ont permis de mener mes travaux dans de bonnes conditions. Je tiens, plus largement, à exprimer ma reconnaissance à toutes celles et à tous ceux qui ont contribué, directement ou indirectement, au bon déroulement de mes travaux.

Enfin, je remercie ma mère, ma femme et mes enfants qui ont su écouter mes incertitudes et qui m'ont accompagné, soutenu, encouragé tout au long de ces années.

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

Sommaire

Abbreviations and Notations

Throughout the dissertation, we use uppercase letters to denote matrices, and lowercase letters for vectors or scalars. Vectors are also regarded as matrices with one column. Some of the abbreviations and notations used in the dissertation are summarized as follows.

DC	Difference of Convex functions
DCA	DC Algorithm
SDCA	Stochastic DCA
ADCA	Accelerated DCA
ADCA-Like	Accelerated DCA-Like
$\mathbb{R}$	set of real numbers
$\mathbb{R}^n$	set of real column vectors of size n
$\overline{\mathbb{R}}$	the set of extended real numbers, $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$
$\|\cdot\|_0$	zero-norm, $\|x\|_0 = \|\{i = 1, ..., n : x_i \neq 0\}\|$, $x \in \mathbb{R}^n$
$\|\cdot\|_p$	ℓ_p-norm ($0 < p < \infty$), $\|x\|_p = (\sum_{i=1}^n \|x_i\|^p)^{1/p}$, $x \in \mathbb{R}^n$
$\|\cdot\|$	Euclidean norm (or ℓ_2-norm), $\|x\| = (\sum_{i=1}^n \|x_i\|^2)^{1/2}$, $x \in \mathbb{R}^n$
$\|\cdot\|_\infty$	ℓ_∞-norm, $\|x\|_\infty = \max_{i=1,...,n} \|x_i\|$, $x \in \mathbb{R}^n$
$\langle\cdot,\cdot\rangle$	scalar product, $\langle x, y\rangle = \sum_{i=1}^n x_i.y_i$, $x, y \in \mathbb{R}^n$
$\chi_C(\cdot)$	indicator function of a set C, $\chi_C(x) = 0$ if $x \in C$, $+\infty$ otherwise
co$\{C\}$	convex hull of a set of points C
$\text{Proj}_C(x)$	projection of a vector x onto a set C
dom f	effective domain of a function f
$\nabla f(x)$	gradient of a function f at x
$\partial f(x)$	subdifferential of a function f at x
$W_{i,:}$	the i-th row of the matrix W
$W_{:,i}$	the i-th column of the matrix W
I_a	a-by-a identity matrix
$\|S\|$	cardinality of set S

Notice individuelle

1 Curriculum vitæ

Etat civil et coordonnées

Nom : Le
Prénom : Hoai Minh
Date et lieu de naissance : 14 septembre 1980, à Nghe An, Vietnam
Nationalité : Française & Vietnamienne
Situation de famille : marié

Coordonnées professionnelles
Département Informatique & Applications,
Laboratoire de Génie Informatique, de Production et de Maintenance,
Université de Lorraine,
Campus Technopole : 3 rue Augustin Fresnel, BP 45112, 57073 Metz Cedex 03
Téléphone : +33 (0)3 72 74 79 54
Adresse électronique : minh.le@univ-lorraine.fr

1.1 Titres universitaires

- 2007 : Doctorat de l'Université de Paul Verlaine - Metz, spécialité Informatique
Directrice de thèse : Prof. Hoai An Le Thi.
Titre : Modélisation et Optimisation non convexe basées sur la programmation DC et DCA pour la résolution de certaines classes des problèmes en Fouille de Données et Cryptologie.
Date de souteance : 24 octobre 2007.

Composition du jury : Prof. Tao Pham Dinh, président - Prof. Jean Pierre Crouzeix, rapporteur, Prof. Abdel Lisser, rapporteur - Prof. Hoai An Le Thi - Prof. Noëlle Carbonell, examinatrice - Prof. Hans-Herman Bock - examinateur, Prof. Pascal Bouvry - examinateur, Prof. Franciszek Seredynski, examinateur.
- 2004 : Diplôme d'Etudes Approfondies (DEA) "Sciences de l'Ingénieur" à l'Institut National des Sciences Appliquées de Rouen (INSA-Rouen).
- 2004 : Diplôme d'ingénieur à l'INSA-Rouen, département Architecture des Systèmes d'Information (ASI).

1.2 Expérience professionnelle

- De septembre 2017 à ce jour

Maître de conférences à l'Université de Lorraine, affecté à l'UFR Mathématiques, Informatique, Mécanique (MIM) et au département Informatique & Applications, Laboratoire de Génie Informatique, de Production et de Maintenance (LGIPM).

- De juillet 2016 à août 2017

Chercheur contractuel au Laboratoire d'Informatique Théorique et Appliqué (LITA), à l'Université de Lorraine. J'ai été membre du comité de pilotage du projet "Smart Marketing", en collaboration avec RTE (Réseaux de transport d'électricité) dont le but est de développer les techniques avancées en apprentissage automatique et fouille de données pour découvrir des informations pertinentes des clients de RTE qui seront ensuite utilisées dans différents services de RTE.

- De juin 2015 à mai 2016

Chercheur contractuel à Luxembourg Centre for Systems Biomedicine (LCSB), à l'Université de Luxembourg. Pendant cette période, j'ai travaillé dans le cadre du projet international intitulé "Multi-scale Molecular Systems Biology : Reconstruction and Model Optimization", financé par "US Department of Energy".

- De septembre 2010 à mai 2015

Chercheur contratuel à LITA, à l'Université Paul-Verlaine Metz (devenu l'Université de Lorraine en 2012). J'ai participé à plusieurs projets, contrats industriels dont un projet FEDER (Fonds Européen de Développement Régional) intitulé "Innovations techniques d'optimisation pour le traitement Massif de Données (InnoMaD)".

- De février 2009 à juin 2010

Stage post-doctoral effectué au Laboratoire de Mathématiques Appliquées du Havre (LMAH), à l'Université du Havre. J'ai travaillé dans le cardre du projet "Carrier Laser Tracking System (CALAS)" dont le but est de développer un système de planification des cavaliers pour le port à conteneurs du Havre.

- De décembre 2007 à décembre 2008

Stage post-doctorat à LITA, à l'Université Paul Verlaine - Metz. Durant ce stage, j'ai participé au développement de plusieurs méthodes d'optimisation pour l'apprentissage automatique.

2 Activités d'enseignement

Les thématiques que j'enseigne se focalisent principalement sur la recherche opérationnelle, la modélisation et la résolution des problèmes industriels. Le tableau ci-dessous résume les enseignements que j'ai effectués au 1er juillet 2021. L'ensemble représente 970 heures TD équivalent.

Formation	**Module**	**Type**	**Année**	**Volume horaire (TD équivalent)**
Master 2	Modèles et Algorithmes pour la Logistique et le Transport	CM, TD, TP	2018-2019 2019-2020 2020-2021	82 82 82
Master 2	Management et optimisation de la production	CM, TD	2018-2019 2019-2020 2020-2021	19 19 19
Master 2	Tools supporting decision making	CM, TD	2017-2018	30
Master 2	Scheduling and applications	CM, TD	2017-2018	30
Master 2	Outils d'Aide à la décision	CM, TD, TP	2017-2018	37

Master 2	Ordonnancement et applications	CM, TD	2017-2018	40
Master 2	Modélisation et optimisation industrielle	TD	2012-2013 2011-2012 2010-2011 2006-2007 2005-2006	24 15 15 15 15
Master 2	Encadrement de projets de synthèse de Master 2 Informatique	Projet	2011-2012	12
Master 1	Modèles et Outils pour l'Optimisation	CM, TD, TP	2018-2019 2019-2020 2020-2021	51 51 51
Master 1	Gestion de Production	CM, TD, TP	2018-2019 2019-2020 2020-2021	38 38 53
Master 1	Encadrement de projets d'initiation à la recherche de Master 1 Informatique	Projet	2011-2012	4
Master 1	Outils pour l'aide à la décision	TD	2011-2012 2010-2011 2006-2007 2005-2006	15 15 15 15
Licence 3	Recherche opérationnelle	TD	2011-2012 2010-2011	40 40
Licence 1	Méthodologie	TD	2006-2005 2004-2005	15 15
				Total : 970 h

3 Activités de recherche

3.1 Thèmes de recherche

Mes activités de recherche sont centrées autour de la modélisation et l'optimisation (convexe et non-convexe) et ses applications. Passionné par la recherche appliquée, je ne limite pas mes recherches à des études théoriques et cherche toujours à m'orienter vers les applications industrielles. Parmi les domaines d'application de l'optimisation, je porte un intérêt particulier à des méthodes de fouille de données et d'apprentissage. La double compétence en informatique et mathématique appliquée a rendu dynamique et fructueuse l'orientation de mes recherches vers les applications concrètes issues des domaines importants d'application industrielle. Au travers de nombreux projets et des contrats industriels, le développement de logiciels à l'usage industriel constitue une partie importante de mes activités.

En ce qui concerne l'optimisation, mes recherches s'appuient principalement sur la programmation DC (Difference of convex functions) et DCA (DC Algorithm). Cette démarche est motivée par leur versatilité, flexibilité, robustesse et performance comparés à des méthodes existantes, leur adaptation aux structures des problèmes traités et leur capacité de résoudre des problèmes industriels de très grande dimension. DCA est reconnu par les chercheurs en optimisation comme un

des rares algorithmes de la programmation non-convexe étant capables de traiter des problèmes (différentiables ou non) de très grande taille.

Mes thèmes de recherche principaux sont :

- Optimisation convexe & non-convexe
  - Approches déterministes : optimisation globale, optimisation DC, optimisation combinatoire.
  - Approches méta-heuristiques.
- Fouille de données & Apprentissage par des techniques d'optimisation : méthodes et applications
  - Classification supervisée, non-supervisée et semi-supervisée.
  - Sélection de variables, sélection de groupe de variables et pondération de variables.
  - Domaine d'application : analyse prédictive, système de recommandation, traitement d'image.
- Modélisation et méthodes numériques pour des problèmes industriels : la gestion d'un port à conteneurs, la flexibilisation de l'énergie, la navigation robotique, l'approvisionnement quotidien d'une chaine de supermarchés/supérettes, le système de recommandation.

3.2 Publications

- 16 articles dans les revues internationales avec comité de lecture
- 2 articles dans les revues nationales avec comité de lecture
- 4 chapitres de livre
- 16 articles dans des actes de colloque international avec comité de lecture dont 1 dans ICML (International Conference on Machine learning), 1 dans IJCAI (International Joint Conference in Artificial Intelligence), 1 dans NIPS (Neural Information Processing Systems), 2 dans IEEE et 7 dans LNAI/LNCS.
- 3 directions d'ouvrage

Articles dans les revues internationales avec comité de lecture

[RI01] H.A. Le Thi, H.M. Le, D.N Phan, B. Tran, Novel DCA Based Algorithms for a special class of nonconvex problems with Application in Machine learning, Applied Mathematics and Computation, 409, 125904, November 2021.

[RI02] H.A. Le Thi, H.M. Le, D.N Phan, B. Tran, Stochastic DCA for minimizing a large sum of DC functions with application to multi-class logistic regression, Neural Networks, 132 :220-231, 2020.

[RI03] L. Heirendt, ..., H.M. Le, ..., R.M.T. Fleming, Creation and analysis of biochemical constraint-based models using the COBRA Toolbox v. 3.0, Nature protocols, 14(3) : 639-702, 2019.

[RI04] H.M. Le, H.A. Le Thi, M.C. Nguyen, Sparse semi-supervised support vector machines by DC programming and DCA, Neurocomputing, 153 :62-72, 2015.

[RI05] H.A. Le Thi, T. Pham Dinh, H.M. Le, X.T. Vo, DC approximation approaches for sparse optimization, European Journal of Operational Research, 244(1) :26-46, 2015.

[RI06] H.A. Le Thi, H.M. Le and T. Pham Dinh, Feature Selection in machine learning : an exact penalty approach using a Different of Convex function Algorithm, Machine Learning, 101(1) :163-186, 2015.

[RI07] H.A. Le Thi, H.M. Le, T. Pham Dinh, F. Lauer, A DC programming algorithm for switched linear regression, IEEE Transactions On Automatic Control, 59(8) :2277-2282, 2014.

[RI08] H.A. Le Thi, H.M. Le, T. Pham Dinh, New and efficient DCA based algorithms for Minimum Sum-of-Squares, Pattern Recognition, 47(1) :388-40, 2014.

[RI09] H.M. Le, H.A. Le Thi, T. Pham Dinh, V.N. Huynh, Block Clustering based on DC programming and DCA, Neural Computation, 25(10) :2776-2807, 2013.

[RI10] H.A. Le Thi, H.M. Le, T. Pham Dinh, V.N. Huynh, Binary classification via spherical separator by DC programming and DCA, Journal of Global Optimization, 56(4) : 1393-1407, 2013.

[RI11] H.M. Le, A. Yassine, R. Moussi, DCA for solving the scheduling of lifting vehicle in an automated port container terminal, Computational Management Science, 9(2) :273-286, 2012.

[RI12] H.M. Le, A. Yassine and Moussi Riadh, Scheduling of lifting vehicle and Quay Crane in automated port container terminals, Int. J. Intelligent Information and Database Systems, 6(6) :516-531, 2012.

[RI13] H.M. Le, 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 :597-613, 2010.

[RI14] H.A. Le Thi, H.M. Le, V.V. Nguyen, T. Pham Dinh, A DC Programming approach for Feature Selection in Support Vector Machines learning, Journal of Advances in Data Analysis and Classification, 2(3) :259-278, 2008.

[RI15] H.A. Le Thi, H.M. Le and T. Pham Dinh, Optimization based DC programming and DCA for Hierarchical Clustering, European Journal of Operational Research, 183 :067-1085, 2007.

[RI16] H.A. Le Thi, H.M. Le and T. Pham Dinh, Fuzzy clustering based on nonconvex optimisation approaches using difference of convex (DC) functions algorithms, Journal of Advances in Data Analysis and Classification, 1 :85-104, 2007.

Articles dans les revues nationales avec comité de lecture

[RN01] H.A. Le Thi, H.M. Le, T.P. Nguyen and T. Pham Dinh, Segmentation d'images par la classification floue, Revue de Nouvelles Technologies d'Information, RNTI-C-2, 221-239, 2008.

[RN02] H.A. Le Thi, H.M. Le and T. Pham Dinh, Une nouvelle approche basée sur la programmation DC et DCA pour la classification floue, Revue de Nouvelles Technologies d'Information, RNTI, 2 :703-714, 2007.

Chapitres de livre

[CL01] H.A. Le Thi, T. Pham Dinh, H.P.H. Luu, H.M. Le, Deterministic and stochastic DCA for DC programming, Handbook of Engineering Statistics 2nd edition, In press, Springer.

[CL02] H.M. Le, M.T. Ta, DC Programming and DCA for Solving Minimum Sum-of-Squares Clustering Using Weighted Dissimilarity Measures, Transactions on Computational Intelligence XIII, 113-131, 2014.

[CL03] H.M. Le, T.B.T. Nguyen, M.T. Ta, H.A. Le Thi, Image Segmentation via Feature Weighted Fuzzy Clustering by a DCA based algorithm, Advanced Computational Methods for Knowledge Engineering, Studies in Computational Intelligence, 53-63, Springer-Verlag, 2013.

[CL04] H.M. Le, H.A. Le Thi, T. Pham Dinh, Pascal Bouvry, A deterministic Optimization approach for generating highly nonlinear balanced boolean functions in Cryptography, in Modeling, Simulation and Optimization of Complex Processes, 381-392, Springer, 2008.

Articles dans des actes de colloque international avec comité de lecture

[CI01] V.A. Nguyen, H.A. Le Thi, H.M. Le, A DCA based algorithm for Feature Selection in Model-Based Clustering, Lecture Notes in Artificial Intelligence (LNAI) 12033, 404-415, 2020.

[CI02] H.M. Le, X.T. Vo, Reweighted l1 Algorithm for Robust Principal Component Analysis, Advances in Intelligent Systems and Computing 1121, 133-142, 2020.

[CI03] G. Da Silva, H.M. Le, H.A. Le Thi, V. Lefieux, B. Tran, Customer Clustering of French Transmission System Operator (RTE) Based on Their Electricity Consumption, Advances in Intelligent Systems and Computing 991, 893-905, 2019.

[CI04] D.N. Phan, H.M. Le, H.A. Le Thi, Accelerated Difference of Convex functions Algorithm and its Application to Sparse Binary Logistic Regression, 27th International Joint Conference on Artificial Intelligence and 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 2018), 1396-1375, 2018.

[CI05] H.A. Le Thi, H.M. Le, D.N. Phan, B. Tran, Stochastic DCA for the Large-sum of Non-convex Functions Problem and its Application to Group Variable Selection in Classification, Internationale Conference on Machine learning ICML, 3394-3403, 2017.

[CI06] H.A. Le Thi, H.M. Le, D.N. Phan, B. Tran, Stochastic DCA for Sparse Multiclass Logistic Regression, Advances in Intelligent Systems and Computing 629, 1-12, 2017.

[CI07] H.M. Le, H.A. Le Thi and M.C. Nguyen, DCA based algorithms for feature selection in Semi-Supervised Support Vector Machines, Lecture Notes in Computer Science (LNCS) 7988, 528-542, 2013.

[CI08] H.A. Le Thi, T.B.T. Nguyen, H.M. Le, Sparse Signal Recovery by Difference of Convex Functions, Lecture Notes in Computer Science (LNCS) 7803, 387-397, 2012.

[CI09] H.M. Le, H.A. Le Thi, T. Pham Dinh, Gaussian Kernel Minimum Sum-of-Squares Clustering and solution method based on DCA, Lecture Notes in Artificial Intelligence (LNAI) 719, 331-340, 2012.

[CI10] H.M. Le, M.T. Ta, H.A. Le Thi, T. Pham Dinh, DC Programming and DCA for clustering using weighted dissimilarity measures, Proceeding of 5th NIPS Workshop on Optimization for Machine Learning (OPT2012/NIPS 2012), 2012.

[CI11] H.M. Le, H.A. Le Thi, T. Pham Dinh, V.N. Huynh, An efficient DCA for Spherical Separation, Lecture Notes in Artificial Intelligence 6592, Intelligent Information and Database Systems, 421-431, 2011.

[CI12] H.A. Le Thi, H.M. Le, T. Pham Dinh and Pascal Bouvry, Solving the Perceptron Problem by deterministic optimization approach based on DC programming and DCA, Proceeding of the 7th IEEE International Conference on Industrial Informatics INDIN 2009, 222-226, 2009.

[CI13] H.A. Le Thi, H.M. Le, V.V. Nguyen, T. Pham Dinh, Combined Feature Selection and Classification using DCA, Proceedings of RIVF'08, IEEE International conference on Research, Innovation and Vision for the future in Computing and Communications Technologies, Ho Chi Minh city, July 13-17, 232-239, 2008.

[CI14] H.A. Le Thi, H.M. Le, Nguyen Trong Phuc, T. Pham Dinh, Noisy image segmentation by a robust clustering algorithm based on DC programming and DCA, Lecture Notes in Computer Science (LNCS) 5077, Advances in Data Mining : Medical Applications, E-Commerce, Marketing, and Theoretical Aspects, 71-86, 2008.

[CI15] H.A. Le Thi, H.M. Le, T. Pham Dinh, Une approche en programmation DC pour la classification floue, Proceedings de la 13èmes Rencontres de la Société Francophone de Classification, 140-144, 2006.

[CI16] H.A. Le Thi, H.M. Le and T. Pham Dinh, Hierarchical Clustering based on Mathematical Optimization, Lecture Notes in Artificial Intelligence (LNAI) 3918, Advances on Knowledge Discovery and Data Mining, 160-173, 2006.

Directions d'ouvrage

[DO01] H.A. Le Thi, T. Pham Dinh, H.M. Le, (2022, Eds.), Modelling, Computation and Optimization in Information Systems and Management Sciences, Lecture Notes in Networks and Systems, Volume 363, 404 pages, Springer 2022.

[DO02] H.A. Le Thi, H.M. Le, T. Pham Dinh (2020, Eds.), Optimization of Complex Systems : Theory, Models, Algorithms and Applications, Advances in Intelligent Systems and Computing, AISC, Volume 991, 1152 pages, Springer 2020.

[DO03] H.A. Le Thi, H.M. Le, T. Pham Dinh, N.T. Nguyen (2020, Eds.), Advanced Computational Methods for Knowledge Engineering, Advances in Intelligent Systems and Computing, AISC, Volume 1121, 426 pages, Springer 2020.

4 Encadrement d'activités de recherche

4.1 Co-encadrement doctoral

1. Hoang Phuc Hau Luu
   - Sujet : Techniques avancées d'apprentissage automatique basées sur la programmation DC et DCA. Applications à la maintenance prédictive.
   - Directrice de thèse : Hoai An Le Thi.
   - Taux d'encadrement : 50%.
   - Date de début : 01 october 2019.
   - Financement : 50% financés par la région Grand Est, 50% sur fond propre.
   - Résumé : L'objectif principal de cette thèse est de développer les techniques avancées d'apprentissage automatique pour affronter les défis du Big Data et de l'analyse de séries temporelles.

     En amont, nous développerons des algorithmes stochastiques DCA pour résoudre les problèmes d'optimisation non-convexes de très grande taille. En effet, il est bien connu que les problèmes d'optimisation issus des applications de Big Data sont souvent de nature très grande taille. Pour faire face à ces problèmes, les méthodes d'optimisation

stochastiques semblent être plus adaptées que les méthodes déterministes car elles sont moins couteuses en termes de temps de calcul. Cependant, dans la littérature, il existe très peu de méthodes stochastiques pour des problèmes d'optimisation non-convexe. D'une part, nous développerons stochastique DCA pour déterminer les paramètres optimaux des réseaux de neurones profonds qui sont en général des problèmes d'optimisation non-convexes de très grande taille. D'autre part, nous concevrons des techniques performantes d'apprentissage pour les données de séries temporelles telles que la réduction de la dimensionnalité, la sélection de variables, la transformation de données de séries temporelles ainsi que des méthodes d'optimisation avancées afin de résoudre les problèmes d'optimisation résultant de ces techniques.

En aval, nous appliquerons les méthodes développées en amont aux problèmes réels en Big Data et analyse de séries temporelles, en particulier l'apprentissage profond et la prédiction des séries temporelles. Nous nous intéressons spécialement à la maintenance prédictive, un élément essentiel de l'industrie 4.0, qui consiste à prédire si une machine va tomber en panne dans le futur, en analysant des données historiques et l'état actuel de la machine. Notre but est de mettre en place un système d'apprentissage automatique, pour que les algorithmes apprennent et prédisent si l'équipement tombera en panne dans un avenir proche. Notre ambition est de relever les défis du DL en général et du DL pour la maintenance prédictive en particulier : quelle architecture du réseau neural profond (quels modèles du DL) construire ? Quelles méthodes d'optimisation utiliser pour obtenir des paramètres optimaux de ce réseau ? Comment affronter le défi du Big data ? Quelles sont les techniques bien adaptées aux séries temporelles et au cas où des données de défaillance ne sont pas disponibles ?

— Résultat obtenu : nous avons développé deux nouvelles variantes de DCA, à savoir deux versions de DCA stochastique en ligne. Les résultats obtenus font l'objet d'un chapitre de livre et d'un article soumis :

  — H.A. Le Thi, T. Pham Dinh, H.P.H. Luu, H.M. Le, Deterministic and stochastic DCA for DC programming, Handbook of Engineering Statistics, In press, Springer.

  — H.A. Le Thi, H.P.H. Luu, H.M. Le, T. Pham Dinh, Stochastic DCA with Variance Reduction, article soumis.

2. Viet Anh Nguyen

   — Sujet : Contributions à l'apprentissage statistique et l'apprentissage profond : nouveaux modèles et méthodes d'optimisation avancées.

   — Directrice de thèse : Hoai An Le Thi.

   — Taux d'encadrement : 50%.

   — Date de début : : 01 octobre 2018. Le doctorant a été contraint d'arrêter la thèse à cause de son état de santé en novembre 2019.

   — Financement : contrat doctoral de l'Université de Lorraine.

   — Résumé : Les modèles de mélange gaussiens (MGM) sont parmi les approches statistiques les plus connues pour clustering basé sur un modèle, un sujet important de l'apprentissage non supervisé. Bien que les problèmes de clustering MGM de faible dimension puissent être résolus efficacement par l'algorithme EM, un problème de grande dimension est un scénario différent. Le problème d'optimisation devient mal conditionné à cause de la sur-paramétrisation et de la singularité des matrices de

covariance. D'un autre côté, l'apprentissage profond est un cadre d'apprentissage automatique dont les modèles ont la capacité de représenter l'abstraction multi-niveau des données. Pour clustering en particulier, l'architecture profonde pourrait attraper des caractéristiques cachées et des modèles pour lesquels les approches classiques ne pourraient pas traiter. Par conséquence, un modèle combiné des réseaux neuronaux profonds et des MGM est très prometteur pour améliorer le résultat du clustering. Les méthodes d'optimisation pour les MGM et les réseaux neuronaux profonds sont non convexes et posent de nombreux problèmes, par exemple minima locaux, points de selle, régions de plateau et problèmes mal conditionnés. Le thèse vise à proposer de nouveaux modèles combinant l'apprentissage statistique et l'apprentissage profond et des méthodes d'optimisation avancées pour ces modèles. Deux sujets seront considérés :

  - Clustering profond : Modèles et méthodes de Deep Gaussian Mixture pour Big data.
  - Apprentissage profond pour la détection d'anomalies.

- Résultat obtenu : sous notre direction, le doctorant a travaillé sur un problème très difficile en apprentissage automatique, à savoir le modèle de mélange gaussien parcimonieux (Gaussian Mixture Model with sparse regularization). Il s'agit d'un problème d'optimisation non-convexe de très grande dimension pour lequel nous avons développé plusieurs algorithmes performants. Malheureusement, le doctorant a été contraint d'arrêter définitivement la préparation de sa thèse à cause de son état de santé en novembre 2019. Les résultats obtenus font l'objet d'un article dans un volume de la série Lecture Notes in Artificial Intelligence et deux articles soumis :
  - V.A. Nguyen, H.A. Le Thi, H.M. Le, A DCA based algorithm for Feature Selection in Model-Based Clustering, Lecture Notes in Artificial Intelligence (LNAI), Vol. 12033, pp. 404-415, 2020.
  - H.A. Le Thi, H.M. Le, V.A. Nguyen, DCA-Like for GMM Clsutering with Sparse Regularization, article soumis.
  - H.A. Le Thi, H.M. Le, V.A. Nguyen, Novel DCA based algorithm for joint GMM clustering with t-SNE, article soumis.

4.2 Encadrement de stage Master 2

1. Mathis Saillot
   - Sujet : Object detection and application on TutleBot3.
   - Master 2 en informatique Optimisation et Algorithmes (OPAL), UFR MIM, Université de Lorraine.
   - Taux d'encadrement : 50%.
   - Date : 06/04/2021 - 02/09/2021.
2. Cong Duc Hoang
   - Sujet : Reinforcement learning for time series prediction
   - Master 2 en mathématique, Université d'Orléans
   - Taux d'encadrement : 100%
   - Date : 01/04/2019 - 26/08/2019

3. Thi Minh Thi Nguyen
   - Sujet : Learning to rank
   - Master 2 en mathématique, UFR Mathématique de Rennes, Université de Rennes 1
   - Taux d'encadrement : 100%
   - Date : 03/04/2018 - 29/06/2018

5 Participation aux projets et contrats industriels

1. Contrat industriel "**Robot navigation**"
   - Etablissement : LGIPM, Université de Lorraine.
   - Responsable du projet : Hoai An Le Thi.
   - Partenariat industriel : Naval Group.
   - Date : septembre 2020 - juin 2021.
   - Description : le but du projet est de développer une méthode d'apprentissage pour la navigation automatique d'un robot terrestre. Notre méthode a été testée avec succès sur un robot dans les conditions réelles. Les détails du projet ne peuvent pas être présentés dans ce document en raison de la confidentialité.
   - Mon rôle : responsable du développement de logiciel.
2. Projet industriel "**Flexibilité de l'énergie**"
   - Etablissement : LGIPM, Université de Lorraine.
   - Responsable du projet : Hoai An Le Thi.
   - Partenariat industriel : Norske Skog Golbey (NSG), une papeterie du groupe norvégien Norske Skog ASA qui figure parmi les leaders mondiaux de la production de papiers de publication.
   - Date : janvier 2018 - décembre 2018.
   - Description : le prix électrique varie en fonction de l'offre et demande. Une entreprise peut aussi être un producteur d'électricité en vendant l'électricité qu'elle avait achetée. Il s'agit du mécanisme appelé flexibilité de la consommation. Nous avons développé un démonstrateur pour la flexibilisation de l'énergie de NSG. Notre démonstrateur fournit un plan de production intelligent permettant à maximiser le gain de la revente de l'électricité non-utilisée tout en respectant toutes les contraintes de l'entreprise. Notre démonstrateur a été présenté au salon Global Industrie 27-30 mars 2018 à Paris et a suscité l'attention de nombreux industriels.
   - Mon rôle : responsable du développement de logiciel.
3. Contrat industriel "**Smart marketing**"
   - Etablissement : LITA (LGIPM depuis 01/01/2018), Université de Lorraine.
   - Responsable du projet : Hoai An Le Thi.
   - Partenariat industriel : RTE (Réseaux de transport d'électricité).
   - Date : juin 2016 - mars 2019.

- Description : le projet consiste à développer un logiciel pour analyser les courbes de charge des clients de RTE qui sont les plus grands industriels en France. Une profonde analyse de courbe de charge permet à RTE de comprendre mieux le comportement des clients, identifier les éventuels problèmes, etc. afin de mieux répondre aux besoins de ses clients. Ce projet s'inscrit dans une démarche d'innovation de RTE.
- Mon rôle : j'ai été membre du comité de pilotage du projet et responsable du développement de logiciel. J'ai participé au développement de méthodes d'apprentissage automatique pour la prédiction de la consommation d'électricité et la segmentation de clients.

4. Contrat industriel "**Toul'embal**"
   - Etablissement : LITA (LGIPM depuis 01/01/2018), Université de Lorraine.
   - Responsable du projet : Hoai An Le Thi.
   - Partenariat industriel : Toul'embal - une entreprise, située à Toul, spécialiste de l'adhésif de communication et des produits d'emballage.
   - Date : janvier 2014 - avril 2019.
   - Description : le but du projet est dedévelopper un logiciel de vente pour Toul'embal et la maintenance du logiciel. Le logiciel est conçu pour proposer automatiquement des produits qui sont susceptibles de correspondre aux besoins de clients. Ceci permet de diminuer le temps de préparation des vendeurs ainsi qu'améliorer la pertinence des leurs propositions au client. Le système comporte plusieurs modules pour lesquels nous avons développé plusieurs méthodes efficaces en apprentissage automatique et fouille de données.
   - Mon rôle : j'ai été responsable du pôle logiciel et j'ai également participé au développement de méthodes apprentissage automatique pour système de recommandation.
5. Projet de recherche "**Multiscale Molecular Systems Biology : Reconstruction and Model**"
   - Etablissement : LCSB (Luxembourg Centre for Systems Biomedicine), University of Luxembourg.
   - Date : juin 2015 - mai 2016.
   - Description : le projet consiste à développer les outils pour la modélisation et l'optimisation en Biochimie.
   - Mon rôle : développer les méthodes d'optimisation pour résoudre certains problèmes relevant des systèmes biologiques.
6. Projet de recherche "**InnoMaD (Innovations techniques d'optimisation pour le traitement Massif de Données)**"
   - Etablissement : LITA, Université Paul-Verlaine Metz.
   - Responsable du projet : Hoai An Le Thi.
   - Date : septembre 2010 - août 2012.
   - Description : le projet consiste à créer une nouvelle génération d'outils et une plateforme de logiciels destinés au traitement de grande masse de données et leurs applications industrielles. Il s'agit des innovations techniques d'optimisation non-convexe développées au sein du LITA qui sont reconnues internationalement et ont été appliquées avec succès, par des chercheurs et praticiens dans le monde, aux nombreux problèmes industriels.

— Mon rôle : j'ai été responsable du pôle logiciel et j'ai participé au développement de nouvelles méthodes d'optimisation pour l'apprentissage automatique.

7. Projet de recherche "**DCA-SI (Logiciels d'optimisation et aide à la décision DCA pour des Systèmes d'Information et des Systèmes Industriels)**"
   — Etablissement : LITA, Université Paul-Verlaine Metz.
   — Responsable du projet : Hoai An Le Thi.
   — Date : septembre 2010 - décembre 2011.
   — Description : l'objectif du projet est de développer des logiciels des nouvelles méthodes d'optimisation basées sur DCA et les approches globales pour la résolution de plusieurs classes de problèmes d'optimisation non-convexe, un domaine extrêmement important d'aide à la décision, pour l'aide au pilotage des systèmes industriels complexes. Cette plateforme est composée de plusieurs modules, chacun correspondant à un modèle générique qui pourrait être utilisé dans différents systèmes d'information et systèmes industriels de divers domaines.
   — Mon rôle : j'ai été responsable du pôle logiciel, et j'ai participé à l'étude du marché pour identifier les industries intéressées par nos produits.
8. Projet industriel "**CALAS (Carrier Laser Tracking System)**"
   — Etablissement : LMAH (Laboratoire de Mathématiques Appliquées du Havre), Université du Havre.
   — Partenariat industriel : le port du Havre.
   — Responsable : Adnan Yassine.
   — Date : février 2009 - juin 2010.
   — Description : le projet vise à mettre en place un plateforme de logiciels pour la gestion du port de port du Havre. Ces systèmes d'exécution sont couplés avec des moyens de suivi et de traçabilité par positionnement laser particulièrement performant. L'objectif de ce type de solution technologique (logicielle et matérielle) est de permettre d'améliorer la productivité des terminaux, d'assurer la traçabilité des conteneurs et d'optimiser les déplacements des engins et vecteurs de transport. Pour répondre à cet objectif, la solution technologique vise à permettre la modélisation des plateformes portuaires, à valider l'implantation des moyens et zones de stockage, d'optimiser les déplacements des vecteurs de transport, de valider la stratégie des déplacements et localisation de conteneurs, de suivre à temps réel ces activités, de les visualiser et générer les éléments de guidage et d'affichage nécessaires aux conducteurs de cavaliers
   — Mon rôle : j'ai développé les méthodes d'optimisation déterministes pour la planification des cavaliers.

6 Activités d'administration, d'animation de la recherche

6.1 Organisation de conférences internationales

Je joue un rôle important dans l'organisation des conférences internationales suivantes :

- **World Congress on Global Optimization (WCGO 2019)**, Metz, 6-8 juillet 2019
  — `https://wcgo2019.event.univ-lorraine.fr/`

  - WCGO 2019 est la sixième édition de la série conférence internationale World Congress on Global Optimization de l'International Society of Global Optimization (iSoGO).
  - WCGO 2019 a été la plus grande édition de la série WCGO. WCGO 2019 réunissait, autour de 6 conférences plénières, 180 chercheurs venant de 40 pays.
  - 112 papiers sélectionnés, parmi 250 papiers soumis, sont publiés dans un volume de la série *Advances in Intelligent Systems and Computing* de Springer. Deux numéros spéciaux dans deux journaux *Journal of Global Optimization* et *Optimization Letters* étaient dédiés à WCGO 2019.
  - Mon rôle : j'ai été Pubilicity chair et membre du comité de programmes. J'ai été en charge de la création du site web de la conférence. Je gérais les soumissions des articles et les inscriptions. J'ai été également coéditeur des actes de la conférence.
- **International Conference on Computer Science, Applied Mathematics and Applications (ICCSAMA 2019)**, Hanoi, Vietnam, 19-20 décembre 2019
  - `http://www.lita.univ-lorraine.fr/~iccsama2019/`
  - ICCSAMA 2019 est la sixième édition de la série conférence International Conference on Computer Science, Applied Mathematics and Applications.
  - Le programme scientifique de ICCSAMA 2019 était composé de 4 conférences plénières, 4 conférences invitées et 37 articles sélectionnés.
  - Plus de 100 scientifiques ont participé à ICCSAMA 2019.
  - Les actes de conférence ont été publié dans un volume de la série *Advances in Intelligent Systems and Computing* de Springer.
  - Mon rôle : j'ai été co-président du comité d'organisation et membre du comité de programmes. J'ai été également coéditeur des actes de la conférence.
- **Modelling, Computation and Optimization in Information Systems and Management Sciences (MCO 2021)**, Hanoi, Vietnam, 13-14 décembre 2021
  - `https://mco2021.event.univ-lorraine.fr/`
  - MCO 2021 est la quatrième édition de la série conférence Modelling, Computation and Optimization in Information Systems and Management Sciences.
  - Le programme scientifique de MCO 2021 était composé de 3 conférences plénières, 37 articles sélectionnés.
  - Les actes de conférence ont été publié dans un volume de la série *Lecture Notes in Networks and Systems* de Springer.
  - Mon rôle : j'ai été co-président du comité d'organisation et membre du comité de programmes. Je suis également coéditeur des actes de la conférence.

J'ai été également membre du comité d'organisation de conférences internationales suivantes :

- **Non-convex Programming : Local and Global Approaches (NCP'07)**, 17-21 décembre 2007, Rouen, France. Le programme scientifique était composé de 12 conférences plénières et 54 sessions. Trois numéros spéciaux dans les revues internationales étaient dédiés à NCP'07.
- **Modelling, Computation and Optimization in Information Systems and Management Sciences (MCO'08)**, 8-10 septembre, 2008, Metz, France : cette conférence réunit, autour de 5 conférences plénières, 120 chercheurs. 65 papiers ont été sélectionnés, parmi 160 papiers soumis, pour être publiés dans un volume de la série *Communication*

in Computer ans Information Sciences de Springer et un numéro spécial du *Journal of Computational Optimization and Applications*.
`http://www.lita.univ-lorraine.fr/~mco08/`

- **International Conference on Computer Science, Applied Mathematics and Applications (ICCSAMA 2015)**,11-13 mai, 2015, Metz, France. ICCSAMA 2015 est la troisième édition de la série conférence internationale ICCSAMA avec plus de 50 participants. Les actes de la conférence ont été publiés dans un volume de la série *Advances in Intelligent Systems and Computing* de Springer.
`http://www.lita.univ-lorraine.fr/iccsama2015/ICCSAMA/`
- **Modelling, Computation and Optimization in Information Systems and Management Sciences (MCO 2015)**, 11-13 mai, 2015, Metz, France. MCO 2015 est la troisième édition de la série conférence internationale MCO. MCO a rassemblé 5 conférences plénières invitées, 130 participants. Les actes de la conférence ont été publiés dans un volume la série *Advances in Intelligent Systems and Computing* de Springer. Deux numéros spéciaux après la conférence étaient dédiés à MCO 2015.
`http://www.lita.univ-lorraine.fr/iccsama2015/MCO/`

6.2 Activités éditoriales

Je suis relecteur pour de nombreux journaux internationnaux :

— Advances in Data Analysis and Classification (ADAC)

— Computers & Operations Research (COR)

— International Journal of Intelligent Information and Database Systems (IJIIDS)

— Journal of Global Optimization (JOGO)

— Mathematical Programming Series B (MPB)

— Optimization Methods and Softwares (OMS)

Je suis également relecteur pour les conférences internationales suivantes : MCO'08, ACIIDS de 2011 à 2014, ICCSAMA & MCO 2015, WCGO 2019, ICCSAMA 2019, IEEE KSE 2020.

6.3 Jurys de thèse

J'ai participé au jury de thèse de :

- Bach Tran
  - Sujet : Algorithmes avancés de DCA pour certaines classes de problèmes en apprentissage automatique du Big Data.
  - Thèse soutenue en novembre 2019 à LGIPM, Université de Lorraine.
- Phuong Anh Nguyen
  - Sujet : Sécurité des réseaux par des méthodes d'optimisation et d'apprentissage.
  - Thèse soutenue en avril 2020 à LGIPM, Université de Lorraine.

Introduction générale

La programmation DC (Difference of Convex functions) et DCA (DC Algorithm), constituant l'épine dorsale de l'optimisation non-convexe, ont été introduits en 1985 par Tao PHAM DINH et intensivement développés par Hoai An LE THI & Tao PHAM DINH depuis 1994 pour devenir maintenant classiques et de plus en plus utilisés par des chercheurs et praticiens de par le monde, dans divers domaines des sciences appliquées. Leur popularité vient de leur robustesse et performance comparées aux méthodes existantes, de leur adaptation aux structures des problèmes traités et de leur capacité de résoudre des problèmes industriels de grande dimension. Ces outils théoriques et algorithmiques ont une histoire riche et réussie de trente-cinq ans de développement, et la recherche de ces dernières années est de plus en plus explorée aux nouvelles tendances dans le développement de DCA : concevoir de nouvelles variantes de DCA pour améliorer le DCA standard, pour faire face à la scalabilité et aux problèmes d'optimisation non convexes au-delà de la programmation DC. Les travaux présentés dans ce mémoire suivent ces tendances, nous abordons les méthodes avancées basées sur DCA pour différentes classes des problèmes d'optimisation non convexes et leurs applications à nombreuses thématiques de l'apprentissage automatique dans l'optique de relever les défis de big data.

Un programme DC standard s'écrit sous la forme :

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

où g et h sont des fonctions convexes semi-continues inférieurement et propres sur $\mathbb{R}^n$. La fonction f est dite une fonction DC alors que g et h sont les composantes DC de f. Ce modèle standard contient également la minimisation d'une fonction DC sur un ensemble convexe $\mathcal{C} \subset \mathbb{R}^n$, car grâce à l'utilisation de la fonction indicatrice $\chi_{\mathcal{C}}$ de $\mathcal{C}$ (définie par $\chi_{\mathcal{C}}(x) := 0$ si $x \in \mathcal{C}, +\infty$ sinon) on a :

$$\inf\{f(x) := g(x) - h(x) : \ x \in \mathcal{C}\} = \inf\{(g(x) + \chi_{\mathcal{C}}(x)) - h(x) : \ x \in \mathbb{R}^n\}.$$

La philosophie de DCA consiste en approximation d'un programme DC par une suite des problèmes convexes. A chaque itération de DCA on remplace la deuxième composante DC h par sa minorant affine et minimise ensuite le sous-problème convexe résultant. Le schéma de DCA standard prend la forme :

On constate ainsi que DCA fonctionne sur les composantes DC g et h (qui sont convexes) et non sur la fonction f elle-même. Et puisqu'une fonction DC admet une infinité de décompositions DC, il y a autant de DCA que des décompositions DC pour la résolution d'un programme DC. Le choix d'une décomposition DC appropriée est crucial car il conditionne les qualités essentielles (rapidité, robustesse, ...) du DCA résultant. Théoriquement, la détermination d'une décomposition DC "optimale" reste une question ouverte. En pratique on cherche des décompositions DC bien adaptées à la structure spécifique du problème traité afin que les calculs de deux suites $\{x^k\}$ et $\{y^k\}$ soient moins coûteux en temps de calcul (idéalement - elles sont déterminées de manière explicite).

Algorithm Schéma DCA standard

Initialisation : Choisir un point initial $x^0 \in \mathbb{R}^n$, $k \leftarrow 0$.
Répéter
1. Déterminer $y^k \in \partial h(x^k)$.
2. Déterminer $x^{k+1} \in \operatorname{argmin}\{g(x) - \langle y^k, x\rangle : x \in \mathbb{R}^n\}$. (P_k)
3. $k \leftarrow k + 1$.

jusqu'à la convergence de $\{x^k\}$.

Un aperçu de 30 ans de développement de DCA a été présenté dans le numéro spécial du journal *Mathematical Programming* dédié à la programmation DC et DCA à l'occasion de 30ème anniversaire de DCA [144]. Pour l'instance, il est pertinent de mentionner les propriétés suivantes de DCA :

- *la flexibilité* : la programmation DC et DCA constituent une philosophie sur l'extension de l'analyse/la programmation convexe - à l'analyse/la programmation non convexe non différentiable. Comme conséquence, *DCA n'est pas "un" algorithme* (comme les algorithmes de gradient ou de Newton, etc) mais *une infinité d'algorithmes* définis par diverses formulations DC équivalentes et une infinité de décompositions DC pour chaque programme DC formulé ;
- *l'universalité* : avec des décompositions DC appropriées et reformulations DC convenablement équivalentes, DCA permet de retrouver les méthodes standard en programmation convexe et non convexe (voir [144], Section 1.2), et de nombreux algorithmes d'optimisation non convexes récemment développés peuvent être vus comme une version spéciale des algorithmes basés sur DCA (voir [144], Section 3.3). L'universalité de DCA permet aux chercheurs d'étudier la programmation non convexe par un point de vue unifié via la programmation DC et DCA, d'autant plus que la flexibilité de DCA peut les aider à développer de meilleurs algorithmes ;
- *le succès* : de nombreux algorithmes basés sur DCA ont résolu avec succès des problèmes non convexes, non différentiables apparaissant dans divers domaines d'application, en particulier dans l'apprentissage automatique, le système de communication, la biologie, la finance, le traitement d'image, le transport-logistique, la robotique, etc (voir Section 4 de [144] et références inclues pour les travaux jusqu'à 2018).

Les travaux récents en programmation DC et DCA visent à aborder les questions ouvertes importantes suivantes :

i) l'amélioration de la vitesses de convergence de DCA ;
ii) le développement des solveurs performants pour la résolution de sous-problèmes convexes ;
iii) la recherche des bonnes décompositions DC par les techniques de régularisation et décomposition, et la recherche des bonnes approximations convexes de la fonction objectif DC ;
iv) le développement des schémas DCA adaptés aux problèmes avec big data ;
v) l'utilisation de DCA sans mettre en évidence une décomposition DC de la fonction objectif ;
vi) l'extension de DCA à la résolution des programmes non convexes au-delà de la programmation DC.

Les algorithmes récemment développés pour *la programmation DC* ne sont rien d'autre que des variantes de DCA obtenues en utilisant différentes reformulations DC du problème original, en

choisissant d'autres décompositions DC et en introduisant des techniques d'accélération pour que les DCAs "sur mesure" soient aussi efficaces que possibles. Il va sans dire que v) et vi) sont les deux grandes challenges. Les schémas DCAs pour la résolution de certaines classes de problèmes non convexe non différentiable au-delà des programmes DC ont vu le jour depuis plusieurs années via des techniques de reformulation ou d'approximation (voir [144] section 3.2), et nombreux travaux récents poursuivent cette direction de recherche pour la résolution d'autres classes des problèmes difficiles.

Les travaux présentés dans ce mémoire sont parmi les premières études dans la littérature sur des méthodes DCA avancées (sauf ceux pour les trois modèles de clustering dans les chapitres 7, 8 et 9 qui sont en fait des schémas DCA standard), ils visent à répondre aux questions ouvertes mentionnées ci-dessus.

Le mémoire est divisé en trois parties contenant de douze chapitres. Avant de présenter ces parties principales, nous donnons dans le chapitre 1 une brève introduction de l'état de l'art de la programmation DC et DCA.

Dans la première partie nous décrivons les trois approches DCA avancées pour adresser les cinq premières questions i) – v).

La première approche concernant *l'Accélération de DCA* (brièvement nommé ADCA) est présentée dans le chapitre 2. Pour améliorer la vitesse de convergence de DCA (question i)) nous introduisons la technique d'accélération de Nesterov (initialement proposée pour la programmation convexe) avant l'étape 1 du schéma DCA standard. Plus précisément, au lieu de calculer $y^k \in \partial h(x^k)$ comme DCA standard, nous cherchons un point v^k "plus prometteur" que x^k et prenons $y^k \in \partial h(v^k)$. v^k est dit "plus prometteur" que x^k si

$$f(v^k) \leq \max_{t=\max(0,k-q),\ldots,k} f(x^t), \tag{1}$$

pour un q fixé au départ. v^k est le point d'extrapolation de x^{k-1} et x^k via le formulation de Nesterov. Si ce point d'extrapolation n'est pas "plus prometteur" que x^k, on maintient $v^k = x^k$. Il est clair que si $q = 0$ alors la suite $\{f(x^k)\}$ est décroissante comme dans DCA standard. Par contre, avec un $q > 0$, ADCA peut faire augmenter la fonction objectif $f(x^k)$ et par conséquent échapper à un éventuel mauvais minimum local. Théoriquement, une valeur élevée de q augmente les chances d'utiliser les points d'extrapolation dans ADCA. Comme notre technique intervient au calcul de sous gradient de h (en quelques sortes, il s'agit "d'accélérer" le sous-gradient), elle nous semble plus avantageuse qu'une autre approche d'accélération DCA proposée dans [10] qui consiste à accélérer x^{k+1} par une recherche linéaire de direction de descente de type d'Amijo [77].

Nous démontrons la convergence de ADCA, et estimons le taux de convergence de ADCA sous les conditions de Lojasiewics. Nous déployons ensuite ce schéma ADCA à la résolution une classe spéciale des problèmes souvent rencontrés en pratique, qui est la minimisation de la somme d'une fonction différentiable à dérivé Lipschitzien (possiblement non-convexe) et une fonction DC. Pour étudier l'efficacité de ADCA, nous l'appliquons au problème de sélection de variable en régression logistique (ou la régression logistique parcimonieuse bi-classes). Les résultats numériques montrent que ADCA permet de réduire considérablement le temps de calcul de DCA standard tout en donnant une meilleure solution. Sur certains jeux de données, le gain de temps de ADCA par rapport à DCA standard peut aller jusqu'à 12 fois. ADCA montre également sa supériorité par rapport aux méthodes existantes comme la méthode du gradient proximal combiné avec la technique d'accélération.

La deuxième approche, nommée DCA-Like, est présentée dans le chapitre 3. Elle vise à répondre aux deux questions ouvertes iii) et v) pour la résolution de deux classes des problèmes

de structure spéciale. Une version d'accélération de DCA-Like, appelé ADCA-Like, est également introduite pour aborder la question i).

DCA-Like est une approche permettant d'utiliser la philosophie de DCA sans mettre en évidence une décomposition DC de la fonction objectif. De plus, DCA-Like propose une bonne majorante convexe de la fonction DC adaptée à chaque itération. Ce sont ses deux avantages considérables par rapport au DCA standard. C'est la structure spéciale des problèmes considérés qui nous a motivé et permit de construire les schémas DCA-Like.

Le premier problème consiste à minimiser la somme d'une fonction f non-convexe différentiable à dérivé Lipschitzien et d'une somme des fonctions composites $\sum_{i=1}^{m} h_i(g_i(x_i))$, où $g_i : \mathbb{R}^{n_i} \to \mathbb{R}$ sont convexes continues, x_i (pour $i = 1, \ldots, m$) sont les sous-vecteurs de x avec $\sum_{i=1}^{m} n_i = n$, et $h_i : \mathbb{R} \to \mathbb{R}$ sont concaves croissantes. Le deuxième problème est également considéré dans le chapitre 2 : la minimisation de la somme d'une fonction différentiable à dérivé Lipschitzien et une fonction DC. Bien que dans nombreux de cas le premier problème devient un cas spécial du deuxième, la structure des fonctions composites mérite un traitement spécifique.

DCA-Like est "similaire" à DCA dans le sens où ils approximent itérativement le programme DC original par une séquence des programmes convexes. Cependant, DCA-Like est "différent" à DCA via

i) la manière de décomposer $f = g_\rho - h_\rho$, avec un paramètre ρ, dans lequel h_ρ n'est pas forcément convexe (c'est-à-dire on n'a peut-être pas une décomposition DC de f), et ρ est mise à jour de sorte que la valeur de la fonction majorante convexe de f à la solution courant x^k puisse être aussi proche que possible de $f(x^k)$;

ii) la manière de détermination de minorant affine de h_ρ : ceci peut ne pas être une borne inférieure de h_ρ sur tout l'espace mais plutôt seulement à la solution courante x^k.

Nous montrons que, heureusement, malgré ces modifications, la séquence bornée $\{x^k\}$ générée par DCA-Like converge toujours vers un point critique du programme DC considéré, sous certaines conditions. De plus, nous démontrons que, sous l'hypothèse de Kurdyka-Łojasiewicz, le taux de convergence de DCA-Like est au moins sous-linéaire $\mathcal{O}(1/k^\alpha)$ avec $\alpha > 1$.

Nous appliquons ensuite des schémas DCA-Like et ADCA-Like aux deux thématiques importantes en apprentissage automatique : le problème t-SNE (t-distributed Stochastic Neighbor Embedding) et la régression logistique parcimonieuse multi-classes (sélection des groupes variables en régression logistique multi-classes). Les résultats numériques montrent la supériorité des algorithmes basés sur DCA contre des meilleures méthodes existantes, et l'avantage de DCA-Like par rapport à DCA standard, ainsi que l'atout de ADCA-Like contre DCA-Like.

La troisième approche, appelée DCA Stochastique (SDCA en bref), est présentée dans le chapitre 4. Elle adresse la question iv) - les DCAs adaptés aux problèmes avec big data. Nous considérons le problème de minimisation d'une somme d'un grand nombre des fonctions DC de la forme

$$\min_{x \in \mathbb{R}^d} \left\{ F(x) := \frac{1}{n} \sum_{i=1}^{n} F_i(x) \right\},$$

où F_i sont les fonctions DC, c'est-à-dire $F_i(x) = g_i(x) - h_i(x)$ avec g_i et h_i étant convexes, continues inférieurement et propres, et n est un très grand nombre. Ce problème apparait dans plusieurs contextes, en particulier dans l'optimisation stochastique et l'apprentissage automatique. En effet, la structure à grande somme apparait naturellement dans la minimisation des risques (ou encore la minimisation de la moyenne-variance) en programmation stochastique. De plus, à l'ère du big data, la structure à grande somme est l'une des formes les plus populaires rencontrées dans la pratique de modélisation de données.

Ce problème a une double difficulté due à la non convexité de F_i et la grande valeur de n. Néanmoins, la structure de somme dans F bénéficie d'un avantage : on peut travailler sur F_i au lieu de la fonction F. Puisque tous les F_i sont des fonctions DC, F l'est également, et par la suite c'est un programme DC standard auquel DCA peut être appliqué. Néanmoins, bien que DCA soit une approche capable de résoudre des problèmes de grande taille en optimisation non convexe, la structure à grande somme est difficile à gérer par des algorithmes déterministes. En fait, dans DCA standard on doit calculer le sous-différentiel de toutes les fonctions h_i à chaque itération. Mettre DCA dans un "cadre" stochastique nous semble plus avantageux.

Nous proposons une version stochastique de DCA, nommée SDCA, qui nécessite le calcul d'un faible nombre de sous-différentiel de h_i à chaque itération. Cela permet de réduire considérablement le temps de calcul de DCA. Cette approche peut être considérée comme la combinaison de DCA avec la technique de réduction de variante en programmation stochastique. Une version inexacte de SDCA est également proposée. Dans cette version, nous calculons seulement une solution approchée du sous-problème convexe à chaque itération au lieu de le résoudre exactement. Nous démontrons que SDCA et inexact SDCA convergent vers un point critique avec probabilité 1. Pour étudier l'efficacité de nos algorithmes, nous les déployons à la régression logistique parcimonieuse multi-classes. Les résultats numériques montrent que SDCA est beaucoup moins coûteux en temps de calcul que DCA tout en fournissant une solution de même qualité. Ils montrent également la supériorité de nos algorithmes par rapport aux méthodes existantes dont la plus populaire stochastique gradient descente.

Dans la deuxième partie de ce mémoire nous étudions une thématique difficile et extrêmement importante - l'optimisation parcimonieuse (sparse optimization en anglais). Cette thématique attire une attention particulière de nombreux chercheurs au cours de la dernière décennie, de par son rôle significatif dans divers domaines, en particulier l'apprentissage automatique. L'apprentissage avec parcimonie est une tâche cruciale pour relever les défis du big data.

L'optimisation parcimonieuse fait référence à un problème d'optimisation dont la fonction objectif ou les contraintes contiennent la norme zéro (la norme zéro d'un vecteur x est le nombre de ses composants non nulles). Due à la discontinuité de la norme zéro, il ne s'agit pas d'un problème DC auquel on peut appliquer DCA. Cette étude adresse donc à la question vi) ci-dessus : comment utiliser DCA pour la résolution des programmes non-convexes au-delà de la programmation DC ?

L'optimisation parcimonieuse a été démontrée NP-difficile à cause de la discontinuité de la norme zéro. Dans la littérature, il existe trois approches pour l'optimisation parcimonieuse. La plus ancienne approche est l'approximation convexe. Elle consiste à remplacer la norme zéro par des fonctions convexes. La norme ℓ_1 est incontestablement l'approximation convexe la plus populaire de la norme zéro. Dans le premier temps, les approximations convexes sont très utilisées car les problèmes approchés résultants sont convexes. Néanmoins, la convexité ne favorise pas la parcimonie. Dans la deuxième approche, la norme zéro est remplacée par les approximations non convexes qui jouissent meilleures parcimonies par rapport aux approximations convexes. Cependant les problèmes résultant sont non convexes et donc plus difficile à résoudre. Plusieurs approximations non-convexes ont été proposées dans la littérature comme les fonctions concave exponentielle, SCAD, capped-ℓ_1, etc. Nous donnerons dans le chapitre 5 la liste des fonctions d'approximation connues et montrerons que toutes ces fonctions sont DC, et on peut ainsi investir DCA à la résolution des problèmes approchés résultant. La troisième approche consiste à reformuler de manière équivalente le problème d'optimisation parcimonieuse sous la forme d'un problème d'optimisation non-convexe auquel on peut investir DCA. Nos travaux en optimisation parcimonieuse concernent la deuxième et la troisième approches. Parmi les études dans la deuxième approche, nos contributions (publiées dans [147]) sont les plus significatives car nous

proposons une fonction d'approximation DC qui contient toutes les approximations non convexes dans la littérature, et considérons un modèle d'optimisation unifié incombant tous les problèmes non convexes résultant des approximations non convexes connues. C'est ainsi que nos DCAs pour ce modèle général couvrent tous les algorithmes existants dans cette deuxième approche comme des cas particuliers. Quant à la troisième approche (reformulation via la pénalité exacte), notre algorithme développé dans [130] est le seul algorithme déterministe pour le problème reformulé.

Cette deuxième partie est composée de deux chapitres. Le chapitre 5 dédie à la description des deux approches développées (l'approximation DC et la reformulation via la pénalité exacte). Plusieurs résultats théoriques importants ont été prouvés, en particulier la cohérence entre les solutions des problèmes original et approché. Nous démontrons qu'une solution du problème résultant de l'approximation DC appartient à un epsilon voisinage de la solution optimale du problème original. De plus, nous démontrons que, dans le cas de l'approximation capped-ℓ_1 ou SCAD, le problème résultant est équivalent au problème original avec des paramètres appropriés. Une analyse profonde et une comparaison de toute les approximations non-convexes dans la littérature ont été réalisées. Cette étude est utile aux utilisateurs dans leur choix de la fonction d'approximation et de bons paramètres pour chaque approximation. Nous développons ensuite trois schémas de DCA pour la résolution du problème approché, qui couvrent tous les algorithmes existants comme des cas particuliers.

Concernant l'approche de reformulation via la pénalité exacte, nous montrons que le problème initial peut être formulé comme un problème d'optimisation en variable mixte $0-1$ qui est en fait équivalent à un programme DC grâce aux résultats de la pénalité exacte. Nous investirons ensuite DCA à la résolution du problème reformulé. Il est intéressant de mentionner que cette technique de reformulation permet de voir que le problème approché via la norme ℓ_1 est une relaxation linéaire de notre formulation en variables $0-1$. Ceci justifie pourquoi la norme ℓ_1 ne favorise pas la parcimonie comme la norme zéro.

Dans le chapitre 6, nous déployons nos deux approches proposées à la résolution de trois problèmes d'apprentissage automatique, tout en exploitant leur structure spécifique : Sélection des variables dans Support Vector Machine (sparse SVM), Sélection des variables dans semie-supervisé SVM (sparse S3VM), et la restauration du signal parcimonieuse (signal recovery in compressed sensing). Il est à noter que les problèmes de sélection des variables (resp. des groupes variables) dans bi-classes (resp. multi-classes) logistic regression étudiés dans la première partie concernent également l'optimisation parcimonieuse. Les résultats numériques montrent la performance de nos algorithmes et leur supériorité par rapport aux méthodes standard.

La troisième partie (contenant cinq chapitres) concerne un sujet fondamental de l'apprentissage et fouille de données : le clustering (ou la classification non supervisée, ou encore la classification automatique). Etant donné un ensemble des points et une mesure de similarité. Le clustering consiste à faire une répartition de ces points en K clusters de telle sorte que les points dans le même cluster soient similaires. Le clustering peut être formulé comme un problème d'optimisation mathématique, et l'une des approches les plus prometteuses pour le clustering est basée sur les méthodes d'optimisation. La difficulté majeure de clustering réside dans la non-convexité du modèle d'optimisation associé d'une part, et la taille très grande de ce modèle d'autre part, vu la dimension et le volume de masse de données considérée. Avec les différentes mesures de distance, la plupart des modèles d'optimisation de clustering sont de la forme DC ou peuvent être transformés en une programmation DC par les techniques de reformulation. Dès lors DCA peut être développé pour la résolution de ces problèmes, en particulier pour les problèmes de très grande dimension.

Nous considérons nombreux modèles de clustering dur où chaque objet appartient à une et une seule classe (cluster) et investirons plusieurs algorithmes basés sur DCA standard et/ou

DCA-Like à ces modèles.

Le modèle le plus populaire pour le clustering dur est incontestablement MSSC (Minimum of Sum of Squares Clustering). MSSC s'écrit sous la forme

$$\min \quad \left\{ \sum_{i=1}^{N} \min_{j=1,\ldots,K} \|a_i - x_j\|^2 : x_j \in \mathbb{R}^d, j = 1,\ldots,K \right\}. \tag{2}$$

où a_i, $i = 1,\ldots,N$ sont les n points à répartir dans les K clusters dont les centres sont x_j, $j = 1,\ldots,K$.

Le premier DCA pour MSSC a été développé dans [123]. Avec une décomposition DC appropriée, le schéma DCA proposé dans [123] est explicit dans tous les deux étapes de calcul y^k et x^{k+1}. Ce DCA nécessite tout simplement le produit de matrice-vecteur à chaque itération, il est donc très rapide et par conséquent, capable de résoudre des instances de grande taille. Néanmoins, il est connu que le modèle MSSC n'adapte pas bien au cas où les clusters ne sont pas bien "séparés". Ceci nous a motivé à considérer les deux autres formulations de MSSC : dans la première, nous utilisons les variables binaires pour déterminer si les points a_i appartiennent (ou non) au cluster C_j, et le problème correspondant est la minimisation d'une fonction DC en variables mixtes $0-1$ sous les contraintes linéaires. Ce problème est ensuite reformulé comme un programme DC via la technique de pénalité exacte. Dans la deuxième formulation, un noyau est introduit au modèle MSSC pour donner la naissance au modèle "Kernel MSSC" (KMSSC en bref) et le problème résultant du noyau Gaussien est bien un programme DC. Nous investissons les deux schémas DCA pour la résolutions de ces deux programmes DC. Il s'en suit que, similaire au premier DCA pour MSSC, les deux schémas DCA sont simples, rapides, et capables de résoudre des instances de grande taille. Ils nécessitent, à chaque itération, le calcul de la projection d'un point sur un simplexe et/ou un rectangle qui est déterminé de manière explicite. Nous proposons également une procédure de recherche d'un bon point initial pour DCA via un algorithme VNS (Variable Neighborhood Search). Les résultats numériques montrent que nos deux approches sont mieux que DCA appliqué au modèle MSSC dans [123] en termes de qualité de clustering, alors qu'en général ils sont plus coûteux en temps de calcul, en particulier dans les problèmes de grande dimension. Ces travaux font l'objet du chapitre 7.

Dans le chapitre 8 nous introduisons une pondération (un poids) à chaque variable du modèle MSSC et/ou au modèle de variables mixtes $0-1$. Le but est d'améliorer la qualité de clustering et déterminer des variables pertinentes jouant un rôle important dans le processus de clustering. Habituellement, les variables peuvent être divisées 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 pertinentes ne fournissent aucune information utile. Chaque variable est affectée d'une valeur continue dans l'intervalle $[0, 1]$, nommée un poids, et les variables pertinentes auront un poids élevé. Contrairement à la sélection de variables en classification, l'objectif principal de la pondération des variables est d'améliorer la qualité de l'algorithme de classification, mais pas de réduire le nombre des variables.

Nous développons deux schémas de DCA pour la résolution de ces deux modèles. Il s'avère heureusement que le DCA correspondant consiste à calculer, à chaque itération, la projection d'un point sur un simplexe et/ou un rectangle, qui est déterminée de manière explicite. A partir des expériences numériques nous pouvons dire que l'introduction du poids aux variables permet d'améliorer les performances de la tâche de classification. De plus, les résultats numériques montrent la supériorité en terme de qualité de solution de nos algorithmes par rapport à un algorithme standard en pondération des variables.

Le chapitre 9 concerne le clustering par blocs. Etant donné un ensemble des points représentés sous la forme d'un tableau dont chaque ligne correspond à un point et chaque colonne représente une variable. Le clustering par blocs consiste à procéder simultanément une partition des lignes et une partition des colonnes de ce tableau. Il s'agit d'un problème NP-difficile. Même pour un problème de taille petite, le nombre de partitions possibles peut être énormément grand. A notre connaissance, dans la littérature, la plupart des méthodes existantes sont méta-heuristiques. Nous formulons le clustering par blocs comme un problème d'optimisation avec variables binaires et ensuite reformulons ce dernier sous forme d'un programme DC grâce à la technique de pénalité exacte. Avec une décomposition DC appropriée, le DCA correspondant est simple et peu coûteux. Les résultats numériques montrent que notre méthode DCA fournit toujours des meilleurs solutions par rapport aux méthodes standard (EM, "two-mode K-means", "two-mode Fuzzy").

Le chapitre 10 est consacré au modèle de mélange gaussien (GMM - Gaussian Mixture Model) pour clustering. Si le MSSC est le modèle déterministe le plus utilisé en classification automatique, il en est de même pour le modèle GMM dans le cadre d'apprentissage statistique.

Le problème classique du clustering est de considérer qu'un échantillon de données provienne d'un nombre de groupes inconnus a priori qu'il faut retrouver. Si en plus, on considère que les lois que suivent les individus sont normales, alors on se place dans le cadre des modèles de mélanges gaussiens. L'algorithme EM (Expectation-Maximisation) est sans doute le plus célèbre pour le clustering via le GMM. Il a été montré dans [144] qu'avec une décomposition appropriée EM est une version de DCA.

Un problème rencontré lors de la mise en oeuvre des modèles de mélange concerne la taille du vecteur de paramètres à estimer. Dans le cas d'un mélange gaussien de K composantes de dimension d, le paramètre est de dimension $(K \times (1 + d + d^2)) - 1$. Une solution couramment employée est de sélection des variables qui apportent le plus d'information à l'analyse et d'éliminer celles qui ne présentent que peu d'intérêt. Cette technique, très employée dans des problèmes de discrimination l'est moins dans les problèmes de classification. Une raison peut être que le problème d'optimisation résultant est très difficile, car la norme zéro doit être intégrée pour la sélection de variables.

Une méthode alternative est de traiter "le sur-paramétrage" - on contraint le modèle initial de manière à n'estimer qu'un nombre plus restreint de paramètres (modèles dits parcimonieux).

Par ailleurs, comme pour tous les algorithmes de clustering, la détermination correcte du nombre de clusters (la valeur de K dans GMM) est un défi (la sélection du modèle dans GMM).

Dans ce travail, nous abordons toutes les trois problématiques fondamentales dans le clustering via modèles de mélange gaussien (GMM) qui sont la sélection du modèle, la sélection des variables et le sur-paramétrage. Bien que ces trois problématiques sont liées entre eux, la plupart des travaux existants les abordent séparément. Pour la première fois dans la littérature, nous présentons une formulation d'optimisation unifiée qui prend en compte ces trois problématiques en introduisant simultanément les trois régularisations via la norme zéro. Nous sommes donc face à l'optimisation parcimonieuse étudiée dans la partie 2 de ce mémoire. Nous choisissons l'approche d'approximation non convexe pour traiter la norme zéro et développons dans le premier temps DCA standard pour la résolution du problème résultant. Il s'avère que ce schéma DCA est éventuellement coûteux pour les problèmes de grande taille. Cela nous a motivé d'investir un schéma de DCA-Like, et par la suite un algorithme en deux étapes basés sur DCA-Like afin d'améliorer le DCA-Like.

Le dernier chapitre de cette troisième partie est centré sur le clustering des séries temporelles par une technique de deep clustering. Cette étude fait l'objet d'un grand contrat de recherche avec RTE qui a pour but de regrouper des clients industriels de RTE en fonction de leurs

consommations électriques.

Nous investissons des méthodes performantes pour aborder les trois problématiques cruciales dans le clustering des séries temporelles, qui sont

i) une (des) mesure(s) de similarité appropriée(s),

ii) des techniques de réduction de dimension (pour confronter aux données de grande taille),

iii) des algorithmes de clustering efficaces et capables de traiter les jeux de données de grande dimension.

Pour i), nous utilisons la distance DTW (Dynamic Time Warping) dans l'espace original de données de séries temporelles. De nombreux travaux ont démontré que la distance DWT est bien adaptée aux séries temporelles. Cependant, un inconvénient majeur de la distance DWT est le temps de calcul. L'algorithme de calcul de la distance DWT est un algorithme récursif. Etant donnée la taille importante des données de notre problème, il n'est pas envisageable d'appliquer une méthode de clustering utilisant directement la distance DTW. Nous nous proposons donc d'employer t-SNE (t-distributed stochastic neighbor embedding) pour transformer les données de séries temporelles de grande dimension dans un autre espace de dimension beaucoup plus petite (une technique de réduction de dimension). t-SNE est relativement nouveau, il est basé sur une idée complètement différentes que les autres méthodes classiques comme l'analyse en composantes principales, la factorisation par matrices non négatives ("Non-negative matrix factorization" en anglais) ou sélection de variables. Nous utilisons ensuite DCA pour le problème t-SNE (que nous avons présenté dans le chapitre 3 section 3.4) avec la transformation DTW- Euclidienne. Enfin, nous appliquons le schéma DCA pour le modèle MSSC [123] pour réaliser le clustering dans le nouvel espace. Avant cela, pour déterminer le nombre de clusters nous utilisons un autre schéma DCA pour clustering via la maximisation de modularité [138].

Cette approche peut être considérée comme une méthode de deep clustering. Elle est une combinaison de plusieurs outils et techniques efficaces : la conception d'une méthode basée sur la distance DWT bien adaptée aux séries temporelles et des meilleurs algorithmes de clustering sur la distance Euclidienne pour confronter au Big data est originale. Pour la première fois dans la littérature la distance DWT est considérée dans le modèle de transformation t-SNE, et le schéma DCA résultant est particulièrement efficace. Les expériences numériques sur les données réelles de RTE ont montré que nos résultats de clustering sont cohérents et très profitables pour RTE.

Chapitre 1

DC programming and DCA : the state of the art

In this chapter, we present the key idea and some basic properties of DC programming and DCA. The materials of this chapter are extracted from the state-of-the-arts papers on DCA [142, 144, 189, 190]. The readers are referred to the paper [144] for an extensive overview of more than thirty years of development of DCA.

1.1 Elementary concepts of convex analysis

We first recall some basic notions and results in convex analysis (refer to the references [34, 189, 206] for more details).

We are working with the space $X = \mathbb{R}^n$ which is equipped with the canonical inner product $\langle \cdot, \cdot \rangle$ and the corresponding Euclidean norm $\| \cdot \|$, thus the dual space Y of X can be identified with X itself. We follow [206] for definitions of usual tools of convex analysis where functions could take the infinite values $\pm\infty$. We use in the sequel the convention $+\infty - (+\infty) = +\infty$.

A subset C of X is said to be *convex* if $(1-\lambda)x + \lambda y \in C, \forall x, y \in C$ and $\lambda \in [0,1]$.

Let $f : S \to \mathbb{R} \cup \{\pm\infty\}$ be a function whose values are in $\overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$ and whose domain is $S \subset X$. The *effective domain* of f, denoted by $\mathrm{dom} f$, is defined as

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

The function f is called *proper* if $\mathrm{dom} f \neq \emptyset$ and $f(x) > -\infty$ for all $x \in S$.

The *epigraph* of f, denoted by $\mathrm{epi}\, f$, is defined as

$$\mathrm{epi}\, f = \{(x,t) : x \in S,\ t \in \mathbb{R}, f(x) \leq t\}.$$

The function f is convex if S is convex and

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda) f(x) + \lambda f(y), \quad \forall x, y \in S,\ \forall \lambda \in [0,1].$$

f is *strictly convex* if

$$f((1-\lambda)x + \lambda y) < (1-\lambda) f(x) + \lambda f(y), \quad \forall x, y \in S, 0 < \lambda < 1.$$

The function f is said to be *lower semi-continuous* at a point $x \in S$ if

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

Denote by $\Gamma_0(X)$ the set of all proper lower semi-continuous convex functions on X.

Let ρ be a nonnegative number and S be a convex subset of X. One says that a function $f : S \to \mathbb{R} \cup \{+\infty\}$ is ρ–convex if

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

for all $x, y \in S$ and $\lambda \in (0,1)$. It amounts to say that $f - (\rho/2)\|\cdot\|^2$ is convex on S. The modulus of strong convexity of f on S, denoted by $\rho(f,S)$ or $\rho(f)$ if $S = X$, is given by

$$\rho(f,S) = \sup\{\rho \geq 0 : f - (\rho/2)\|\cdot\|^2 \text{ is convex on } S\}.$$

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

A real-valued function f defined on a set $C \in \mathbb{R}^n$ is said to be Lipschitz on C, if there exists a nonnegative scalar L such as

$$|f(x) - f(y| \leq L\|x-y\|, \qquad \forall x, y \in C.$$

Also f is said to be locally Lipschitz relative to C at some $x \in C$ if for some $\epsilon > 0$, f is Lipschitz on $B(x,\epsilon) \cap C$.

A vector $y \in Y$ is said to be a *subgradient* of a convex function f at a point $x^0 \in \operatorname{dom} f$ if

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

The set of all subgradients of f at x^0 is called the *subdifferential* of f at x^0 and is denoted by $\partial f(x^0)$. If $\partial f(x)$ is not empty, f is said to be *subdifferentiable* at x.
For $\varepsilon \geq 0$, a vector $y \in Y$ is said to be an ε*–subgradient* of a convex function f at a point $x_i^0 n\operatorname{dom} f$ if

$$f(x) \geq (f(x^0) - \varepsilon) + \langle x - x^0, y\rangle, \qquad \forall x \in X.$$

The set of all ε–subgradients of f at x^0 is called the ε*–subdifferential* of f at x^0 and is denoted by $\partial_\varepsilon f(x^0)$.
For $\epsilon \geq 0$, a point x_ϵ is called an ϵ-solution of the problem $\inf\{f(x) : x \in \mathbb{R}^d\}$ if

$$f(x_\epsilon) \leq f(x) + \epsilon \ \forall x \in \mathbb{R}^d.$$

Proposition 1.1 *Let f be a proper convex function. Then*

1. *$\partial_\varepsilon f(x)$ is a closed convex set, for any $x \in X$ and $\varepsilon \geq 0$.*
2. $\operatorname{ri}(\operatorname{dom} f) \subset \operatorname{dom} \partial f \subset \operatorname{dom} f$
   where $\operatorname{ri}(\operatorname{dom} f)$ *is the* relative interior *of* $\operatorname{dom} f$.
3. *If f has a unique subgradient at x, then f is differentiable at x, and $\partial f(x) = \{\nabla f(x)\}$.*
4. *$x_0 \in \operatorname{argmin}\{f(x) : x \in X\}$ if and only if $0 \in \partial f(x_0)$.*

Conjugates of convex functions

The *conjugate* of a function $f : X \to \overline{\mathbb{R}}$ is the function $f^* : X \to \overline{\mathbb{R}}$, defined by

$$f^*(y) = \sup_{x \in X}\{\langle x, y\rangle - f(x)\}.$$

Proposition 1.2 *Let $f \in \Gamma_0(X)$. Then we have*

1. *$f^* \in \Gamma_0(X)$ and $f^{**} = f$.*
2. *$y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y)$.*
3. *$f(x) + f^*(y) \geq \langle x, y\rangle$, for any $x, y \in X$.*
   Equality holds if and only if $y \in \partial f(x)$.

Polyhedral convex functions

A *polyhedral convex* set is a closed convex set of the form

$$C = \{x \in X : \langle x, b_i\rangle \leq \beta_i,\ \forall i = 1, \ldots, m\}.$$

A convex set C is *locally polyhedral* if, for every $x \in C$, there exists a polyhedral convex neighbourhood of x relative to C. A convex function is said to be *locally polyhedral convex* if its epigraph is locally polyhedral convex.

A function $f \in \Gamma_0(X)$ is said to be *polyhedral convex* if

$$f(x) = \max\{\langle a_i, x\rangle - \alpha_i : i = 1, \ldots, k\} + \chi_C(x), \quad \forall x \in X \tag{1.1}$$

where $a_i \in X$, $\alpha_i \in \mathbb{R}$ for all $i = 1, \ldots, k$ and C is a nonempty polyhedral convex set. It is clear that $\mathrm{dom}\, f = C$.

Proposition 1.3 *Let f be a polyhedral convex function, and $x \in \mathrm{dom} f$. Then we have*

i) f is subdifferentiable at x, and $\partial f(x)$ is a polyhedral convex set. In particular, if f is defined by (1.1) with $C = X$ then

$$\partial f(x) = \mathrm{co}\{a_i : i \in I(x)\}$$

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

ii) The conjugate f^* is a polyhedral convex function. Moreover, if $C = X$ then

$$\mathrm{dom} f^* = \mathrm{co}\{a_i : i = 1, \ldots, k\},$$
$$f^*(y) = \inf\left\{\sum_{i=1}^k \lambda_i \alpha_i \Big| \sum_{i=1}^k \lambda_i a_i = y, \sum_{i=1}^k \lambda_i = 1, \lambda_i \geq 0, \forall i = 1, \ldots, k\right\}.$$

In particular,

$$f^*(a_i) = \alpha_i, \qquad \forall i = 1, \ldots, k.$$

Kurdyka-Lojasiewicz (KL) property

Let $\eta \in (0, +\infty]$. Denote by $\mathcal{M}_\eta$ the class of continuous concave functions $\psi : [0, \eta) \to [0, +\infty)$ satisfying

- $\psi(0) = 0$, and ψ is continuously differentiable on $(0, \eta)$.
- $\psi'(t) > 0$ for all $t \in (0, \eta)$.

Definition 1.1 *A lower semicontinuous function σ satisfies the KL property [13] at $\mathbf{u}^* \in dom\, \partial^L \sigma$ if there exists $\eta > 0$, a neighborhood $\mathcal{V}$ of $\mathbf{u}^*$, and $\psi \in \mathcal{M}_\eta$ such that for all $u \in \mathcal{V} \cap \{u : \sigma(\mathbf{u}^*) < \sigma(\mathbf{u}) < \sigma(\mathbf{u}^*) + \eta\}$, one has*

$$\psi'(\sigma(\mathbf{u}) - \sigma(\mathbf{u}^*))\, dist(0, \partial^L \sigma(\mathbf{u})) \geq 1.$$

The class of functions σ satisfying the KL property at all points in dom $\partial^L\sigma$ is very ample, for example, semi-algebraic, subanalytic, and log-exp functions. In particular, these classes of functions satisfy the KL property with $\psi(s) = cs^{1-\theta}$, for some $\theta \in [0, 1)$ and $c > 0$.

Subanalytic set and function

Definition 1.2 *(i) A subset C of $\mathbb{R}^n$ is said to be semianalytic if each point of $\mathbb{R}^n$ admits a neighborhood V such that $C \cap V$ is of the following form :*

$$C \cap V = \bigcup_{i=1}^{p} \bigcap_{j=1}^{q} \{x \in V : \ f_{ij}(x) = 0, \, g_{ij}(x) > 0\},$$

where f_{ij}, $g_{ij} : V \to \mathbb{R}$ $(1 \le i \le p, \, 1 \le j \le q$) are real-analytic functions.

(ii) A subset C of $\mathbb{R}^n$ is called subanalytic if each point of $\mathbb{R}^n$ admits a neighborhood V such that

$$C \cap V = \{x \in \mathbb{R}^n : \ \exists y \in \mathbb{R}^m, \, (x, y) \in B\},$$

where B is a bounded semianalytic subset of $\mathbb{R}^n \times \mathbb{R}^m$ with $m \ge 1$.

(iii) A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is said to be subanalytic if its graph gph f *is a subanalytic subset of $\mathbb{R}^n \times \mathbb{R}$.*

It is obvious that the class of subanalytic sets (resp. functions) contains all analytic sets (resp. functions). Some key properties of subanalytic sets and subanalytic functions :

— Subanalytic sets are closed under locally finite union and intersection (A collection of sets $\mathcal{C}$ is locally finite if any compact set intersects only finitely many sets in $\mathcal{C}$). The complement of a subanalytic set is subanalytic.

— The closure, the interior, the boundary of a subanalytic set are subanalytic.

— A closed $C \subseteq \mathbb{R}^n$ is subanalytic *iff* its indicator function χ_C, defined by $\chi_C(x) = 0$ if $x \in C$ and $+\infty$ otherwise, is subanalytic.

— Given a subanalytic set C, the distance function $d_C(x) := \inf_{z \in C} \|x - z\|$ is a subanalytic function.

— Let $f, g : X \to \mathbb{R}$ be continuous subanalytic functions, where $X \subseteq \mathbb{R}^n$ is a subanalytic set. Then the sum $f + g$ is subanalytic if f maps bounded sets on bounded sets, or if both functions f, g are bounded from below.

— Let $X \subseteq \mathbb{R}^n$, $T \subseteq \mathbb{R}^m$ be subanalytic sets, where T is compact. If $f : X \times T \to \mathbb{R}$ is a continuous subanalytic function, then $g(x) := \min_{t \in T} f(x, t)$ is continuous subanalytic.

Proposition 1.4 ([190]) *If $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a lower semicontinuous subanalytic strongly convex function then its conjugate f^* is a $C^{1,1}$ (the class of functions whose derivative is Lipschitz) subanalytic convex function.*

1.2 DC function

Definition 1.3 *A function $f : X \to \mathbb{R} \cup \{\pm\infty\}$ is called DC function if*

$$f = g - h$$

with g and h belonging to $\Gamma_o(X)$. One says that $g - h$ is a DC decomposition (or DC representation) of f, and g, h are its convex DC components. If g and h are finite on X, then $f = g - h$ is said to be a finite DC function on X.

Remark 1.1 *A DC function has infinitely many DC decompositions*

The set of DC functions (resp. finite DC functions) on X is denoted by $\mathcal{DC}(X)$ (resp. $\mathcal{DC}_f(X)$). It is worth noting the richness of $\mathcal{DC}(X)$ and $\mathcal{DC}_f(X)$: they contain almost realistic objective functions and are closed under all the operations usually considered in optimization.

(i) $\mathcal{DC}_f(X)$ is a subspace containing the class of lower-$\mathcal{C}^2$ functions (f is said to be lower-$\mathcal{C}^2$ if f is locally a supremum of a family of $\mathcal{C}^2$ functions). In particular, $\mathcal{DC}_f(X)$ contains the space $\mathcal{C}^{1,1}(X)$ of functions whose gradient is locally Lipschitzian on X.

(ii) Under some caution we can say that $\mathcal{DC}(X)$ is the subspace generated by the convex cone $\Gamma_o(X) :: \mathcal{DC}(X) = \Gamma_o(X) - \Gamma_o(X)$. This relation marks the passage from convex optimization to nonconvex optimization and also indicates that $\mathcal{DC}(X)$ constitutes a minimal realistic extension of $\Gamma_o(X)$.

(iii) $\mathcal{DC}_f(X)$ is closed under all the operations usually considered in optimization. In particular, a linear combination of $f_i \in \mathcal{DC}_f(X)$ belongs to $\mathcal{DC}_f(X)$, a finite supremum of DC functions is DC.

Proposition 1.5 ([103]) *Every nonnegative DC function $f = g - h$ ($g, h \in \Gamma_0(X)$) admits a nonnegative DC decomposition, i.e., $f = g_1 - h_1$ with g_1, h_1 being in $\Gamma_0(X)$ and nonnegative.*

The function h_1 and g_1 can be defined as [103] :

$$g_1 := g - (\langle b, \cdot\rangle - h^*(b)),\ h_1 := h - (\langle b, \cdot\rangle - h^*(b))$$

with $b \in \mathrm{dom} h^*$.

More generally, every $f \in \mathcal{DC}_f(X)$ admits a nonngative DC decomposition. If follows that the product of two DC function is a DC function [189].

1.3 DC Programming

Standard DC program

A standard DC program takes the form

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

where $f \in \mathcal{DC}(X)$ and $g, h \in \Gamma_0(X)$.

A DC program with a convex constraint set C (a nonempty closed convex set in $\mathbb{R}^n$) can be rewritten in the standard form (P) by using the indicator function on C, defined by $\chi_C(x) = 0$ if $x \in C$, $+\infty$ otherwise :

$$\inf\{f(x) := g(x) - h(x) : x \in C\} = \inf\{\chi_C(x) + g(x) - h(x) : x \in \mathbb{R}^n\}.$$

As mentioned previously, the vector space of DC functions, $\mathcal{DC}(X)$ 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 ([189, 142] and references therein), but not too large in order to leverage the powerful arsenal of the latter.

DC duality associates a primal DC program (P) with its dual, which is also a DC program. Indeed, using the definition of conjugate functions, we can rewrite the problem (P) as

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

where

$$\beta(y) = \inf\{g(x) - (\langle x, y\rangle - h^\star(y)) : x \in X\}.$$

It is clear that $\beta(y) = h^\star(y) - g^\star(y)$ if $y \in dom\, h^\star$, $+\infty$ otherwise. Finally, the dual problem of (P) is stated as

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

We can observe that the dual problem (D) is also a DC program and there is a perfect symmetry between primal and dual programs (P) and (D) : the dual program to (D) is exactly (P).

Note that the finiteness of α implies that

$$\operatorname{dom} g \subset \operatorname{dom} h \quad \text{and} \quad \operatorname{dom} h^\star \subset \operatorname{dom} g^\star. \tag{1.2}$$

Polyhedral DC program

In the DC problem (P), if one of the DC components g and h is a polyhedral convex function, (P) is called a *polyhedral DC program*. Polyhedral DC program is an important class of DC optimization which is often encountered in practice and has interesting properties.

Consider the case where h is a polyhedral convex function defined as

$$h(x) = \max\{\langle a_i, x\rangle - \alpha_i : i = 1, \ldots, k\}.$$

By Proposition 1.3, the dual problem (D) of (P) can be written as

$$\begin{aligned} \alpha &= \inf\{h^\star(y) - g^\star(y) : y \in X\} \\ &= \inf\{h^\star(y) - g^\star(y) : y \in \operatorname{co}\{a_i : i = 1, \ldots, k\}\} \\ &= \inf\{\alpha_i - g^\star(a_i) : i = 1, \ldots, k\}. \end{aligned}$$

In the case g is polyhedral convex and h is not polyhedral, the dual problem (D) also has the similar formulation as above since $g^\star$ is polyhedral.

Remark 1.2 *Polyhedral DC programming, which plays a central role in nonconvex optimization and global optimization has interesting properties (from both a theoretical and an algorithmic point of view) on local optimality conditions and the finiteness of DCA's convergence [190].*

General DC program

A general DC program takes the form

$$\begin{aligned} \inf \quad & \{f_0(x) := g_0(x) - h_0(x) : \\ & f_i(x) := g_i(x) - h_i(x) \leq 0, i = 1, \ldots, m \\ & x \in C\} \end{aligned} \tag{1.3}$$

where C is a nonempty convex set in $\mathbb{R}^n$, $g_i, h_i \in \Gamma_0(X), i = 0, \ldots, m$ and its feasible set $E = \{x \in C, f_i(x) \leq 0, i = 1, \ldots, m\}$ is supposed to be nonempty.

This class of nonconvex programs is the most general in DC Programming and, a fortiori, more difficult to treat than that standard DC programs (P) because of the nonconvexity of the constraints. Its interests is due to the fact that this class appears, increasingly, in many models of nonconvex variational approaches.

1.3.1 Optimality conditions for DC optimization

A point x^* is said to be a *local minimizer* of $g-h$ if $x^* \in \operatorname{dom} g \cap \operatorname{dom} h$ (so, $(g-h)(x^*)$ is finite) and there is a neighborhood U of x^* such that

$$g(x) - h(x) \geq g(x^*) - h(x^*), \qquad \forall x \in U. \tag{1.4}$$

Under the convention $+\infty - (+\infty) = +\infty$, the above condition is equivalent to

$$g(x) - h(x) \geq g(x^*) - h(x^*), \qquad \forall x \in U \bigcup \operatorname{dom} g. \tag{1.5}$$

A point x^* is said to be a *critical point* of $g-h$ if it verifies the generalized Kuhn–Tucker condition

$$\partial g(x^*) \cap \partial h(x^*) \neq \emptyset. \tag{1.6}$$

Let $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of problems (P) and (D) respectively, and let

$$\mathcal{P}_\ell = \{x^* \in X : \partial h(x^*) \subset \partial g(x^*)\}, \quad \mathcal{D}_\ell = \{y^* \in X : \partial g^\star(y^*) \subset \partial h^\star(y^*)\}.$$

Theorem 1.1 ([189]) i) *Global optimality condition : $x \in \mathcal{P}$ if and only if*

$$\partial_\varepsilon h(x) \subset \partial_\varepsilon g(x), \ \forall \varepsilon > 0.$$

ii) *Dually, $y \in \mathcal{D}$ if and only if $\partial_\varepsilon g^\star(y) \subset \partial_\varepsilon h^\star(y) \quad \forall \varepsilon > 0$.*

iii) *Transportation of global minimizers : $\cup\{\partial h(x) : x \in \mathcal{P}\} \subset \mathcal{D} \subset \operatorname{dom} h^\star$.*
The first inclusion becomes equality if $g^\star$ is subdifferentiable in $\mathcal{D}$. In this case, $\mathcal{D} \subset (\operatorname{dom} \partial g^\star \cap \operatorname{dom} \partial h^\star)$.

iv) *Dually, $\cup\{\partial g^\star(y) : y \in \mathcal{D}\} \subset \mathcal{P} \subset \operatorname{dom} g$.*
The first inclusion becomes equality if h is subdifferentiable in $\mathcal{P}$. In this case $\mathcal{P} \subset (\operatorname{dom} \partial g \cap \operatorname{dom} \partial h)$.

v) *Necessary local optimality : if x^* is a local minimizer of $g-h$, then $x^* \in \mathcal{P}_\ell$.*

vi) *Sufficient local optimality : Let x^* be a critical point of $g-h$ and $y^* \in \partial g(x^*) \cap \partial h(x^*)$. Let U be a neighborhood of x^* such that $(U \cap \operatorname{dom} g) \subset \operatorname{dom} \partial h$. If for any $x \in U \cap \operatorname{dom} g$, there is $y \in \partial h(x)$ such that $h^\star(y) - g^\star(y) \geq h^\star(y^*) - g^\star(y^*)$, then x^* is a local minimizer of $g-h$. More precisely,*

$$g(x) - h(x) \geq g(x^*) - h(x^*), \qquad \forall x \in U \cap \operatorname{dom} g.$$

vii) *Transportation of local minimizers : Let $x^* \in \operatorname{dom} \partial h$ be a local minimizer of $g-h$. Let $y^* \in \partial h(x^*)$ and a neighborhood U of x^* such that $g(x)-h(x) \geq g(x^*)-h(x^*), \ \forall x \in U \cap \operatorname{dom} g$. If*

$$y^* \in \operatorname{int}(\operatorname{dom} g^\star) \quad \text{and} \quad \partial g^\star(y^*) \subset U$$

then y^* is a local minimizer of $h^\star - g^\star$.

Remark 1.3 *a) It is clear that the global optimality condition i) is not verifiable in practice.*

b) There is no gap between the primal problem (P) and its dual (D). Globally/locally solving the primal problem (P) implies globally/locally solving the dual problem (D) and vice–versa. Thus, it is useful if one of them is easier to solve than the other.

c) *The necessary local optimality condition* $\partial h^\star(x^*) \subset \partial g^\star(x^*)$ *is also sufficient for many important classes programs, for example, if* h *is polyhedral convex, or when* f *is locally convex at* x^**, i.e. there exists a convex neighborhood* U *of* x^* *such that* f *is finite and convex on* U *[142]. We know that a polyhedral convex function is almost everywhere differentiable, that is to say, it is differentiable everywhere except on a set of measure zero. Thus, if* h *is a polyhedral convex function, then a critical point of* $g - h$ *is almost always a local solution to* (P).

d) *If* f *is actually convex on* X*, we call (P) a* "false" DC program. *In addition, if* $\mathrm{ri}(\mathrm{dom}\, g) \cap \mathrm{ri}(\mathrm{dom}\, h) \neq \emptyset$ *and* $x^0 \in \mathrm{dom}\, g$ *such that* g *is continuous at* x^0*, then* $0 \in \partial f(x^0) \Leftrightarrow \partial h(x^0) \subset \partial g(x^0)$ *[142]. Thus, in this case, the local optimality is also sufficient for the global optimality. Consequently, if in addition* h *is differentiable, a critical point is also a global solution.*

1.3.2 Exact penalty in DC programming

Penalty techniques are powerful tools to transform a difficult (resp. nonconvex) constraints in easier (resp. convex) constraints. Of course, ensuring the equivalence between the original problem and the penalized problem, i.e. exact penalty, is very important. This concern is naturally much more complex in nonconvex programming (than in convex one). Exact penalty techniques in DC programming have been widely investigated [146, 145, 126]. They permit to recast several classes of difficult nonconvex programs into suitable DC programs to be tackled by the efficient DCA.

Exact penalty in concave programming

In [146, 145], the authors have established exact penalty for nonconvex programs having concave objective functions and bounded polyhedral convex constraint sets with additional concave constraint functions. They considered the two following nonconvex programs

$$\alpha = \min\{f(x) : x \in K, g(x) = 0\} \tag{1.7}$$

and

$$\alpha = \min\{f(x) : x \in K, g(x) \leq 0\} \tag{1.8}$$

where K is a nonempty bounded polyhedral convex subset of $\mathbb{R}^n$ and f, g are finite concave functions on K.

In [146], the exact penalty results have been established under the assumption that g is nonnegative on K. Later, Le Thi et al. [145] extended the exact penalty results for a more general case where the assumption of the nonnegativity of g is not needed.

The problem (1.7) can be equivalently rewritten as

$$\alpha = \min\{f(x) : x \in K, g(x) \leq 0, g(x) \geq 0\} \tag{1.9}$$

And its penalized problem is given by

$$\alpha(\tau) = \min\{f(x) + \tau g(x) : x \in K, g(x) \geq 0\} \tag{1.10}$$

Denote by $V(K)$ the vertex set of K and by $\mathcal{P}$ and $\mathcal{P}_\tau$ the optimal solution sets of (1.9) and (1.10), respectively. Furthermore, one uses the convention $\min_\emptyset g(x) = +\infty$.

Theorem 1.2 ([145]) *Let* K *be a nonempty bounded polyhedral convex set in* $\mathbb{R}^n$ *and let* f*,* g *be finite concave functions continuous relative to* K*. Suppose that the feasible set of* (1.9) *is*

nonempty. Then there exists $\tau_0 \geq 0$ *such that for all* $\tau \geq \tau_0$*, the problems* (1.9) *and* (1.10) *have the same optimal value and the same optimal solution set. Furthermore, we can take* $\tau_0 = \frac{f(x_0)-\alpha(0)}{m}$ *with* $m = \min\{g(x) : x \in V(K), g(x) \geq 0\}$ *and any* $x_0 \in K$*,* $g(x_0) = 0$*.*

On the other hand, the problem (1.8) is equivalent to

$$\alpha = \min\{f(x) : (x,t) \in K \times [0,\beta], g(x) + t = 0,\} \tag{1.11}$$

where $\beta \geq \max\{-g(x) : x \in K\}$. The equivalence between (1.8) and (1.11) is in the sense that : if x is a solution of (1.8) then $(x, -g(x)$ is a solution of (1.11) and conversely, if (x,t) is a solution of (1.11) then x is a solution of (1.8).

In virtue of Theorem 1.2, there exists $\tau_0 \geq 0$ such that for all $\tau \geq \tau_0$, the problems (1.11) is equivalent to

$$\alpha = \min\{f(x) + \tau(t + g(x)) : (x,t) \in K \times [0,\beta], g(x) + t \geq 0,\}. \tag{1.12}$$

The above result permits to recast several classes of difficult in nonconvex programming and combinatorial optimization including linear/quadratic integer programming, complementarity problems, bilevel programming, multiple objective programming and optimization over the efficient set into suitable DC programs and open the door to an effective investigation of DCA for solving them. These works allowed building a first bridge between DC programming and operations research, founded on which DCA based algorithms were developed for the above mentioned classes of problems.

Let us consider now a very important class of problems, the linearly constrained quadratic programming with binary variables, which is written as

$$\alpha = \min\left\{f(x) := \frac{1}{2}\langle x, Qx\rangle + \langle q, x\rangle : Ax \leq b, x \in \{0,1\}^n\right\} \qquad (BQP)$$

where Q is an $(n \times n)$ symmetric matrix, $q, x \in \mathbb{R}^n$, A is an $(m \times n)$ matrix and $b \in \mathbb{R}^m$.

In [141], Le Thi and Pham Dinh have reformulated (BQP) as a (continuous) DC program by the following way. First, (BQP) is transformed equivalently in the form

$$\alpha = \min\left\{f(x) := \frac{1}{2}\langle x, (Q - \bar{\lambda}I)x\rangle + \langle q + \frac{\bar{\lambda}}{2}e, x\rangle : x \in K, p(x) \leq 0\right\},$$

where $\bar{\lambda} \geq 0$ is such that the matrix $Q - \bar{\lambda}I$ is negative semidefinite, $K := \{Ax \leq b, x \in [0,1]^n\}$ and $p := \frac{1}{2}[e^T x - x^T x]$. It is clear that p is a concave function with nonnegative values on K. Hence, from the above exact penalty results in concave programming, it follows that there exists $\tau_0 \geq 0$ such that for all $\tau > \tau_0$, (BQP) is equivalent to the (strictly) concave quadratic minimization program :

$$\alpha = \min\left\{\frac{1}{2}\langle x, (Q - \lambda I)x\rangle + \langle q + \frac{\lambda}{2}e, x\rangle : x \in K\right\},$$

with $\lambda = \bar{\lambda} + \tau$.

Remark 1.4 *The linear constrained quadratic zero-one programming problem is equivalently reformulated as a (continuous) concave quadratic program for which DCA can be applied. It is important to note that DCA works on a continuous domain but, with a suitable penalty parameter, it gives an integer solution. Indeed, if at iteration* r*, DCA gives a integer solution* $x^r \in \{0,1\}^n$ *than* $x^k \in \{0,1\}^n$ *for all* $k \geq r$*.*

Following these results, several large-scale problems that are modeled as 0–1 (or mixed 0–1) linear/quadratic programming problems, were successfully solved by DCA and combined DCA—global methods, including the strategic capacity planning in supply chain the strategic supply chain design problem from qualified partner set, the single-vehicle cyclic inventory routing problem, the logistics network design and planning problem, the optimization of traffic signals in networks considering rerouting, the nonlinear UAV task assignment problem under uncertainty, the quality of service (QoS) routing problems, the minimum m-dominating set problem, etc.

Exact penalty in DC Programming via error bounds

One considers the following optimization problem

$$\alpha := \inf\{f(x) : (x) \in C, h(x) \leq 0\} \tag{1.13}$$

where f, h are real-valued functions defined on C. Denote by $\mathcal{P}$ is optimal solution set of (1.13). Define $S := \{x \in C : h(x) \leq 0\}$.

Let $g : C \to \mathbb{R}$ be a nonnegative function such that S can be expressed by

$$S := \{x \in C : g(x) \leq 0\}$$

Such a function g must verify

$$g(x) = 0 \text{ if and only if } x \in S.$$

Exact penalty in mathematical programming usually deals with

$$g(x) := [h^+(x)]^t.$$

For $\tau \geq 0$, let us define the penalized problem

$$\alpha(\tau) := \inf\{f(x) + \tau g(x) : (x) \in C\} \tag{1.14}$$

whose the optimal solution set is denoted by $\mathcal{P}_\tau$.

Proposition 1.6 ([145]) *Let f be a Lipschitz function on C with constant L and let g be a nonnegative finite function on C such that $S := \{x \in C : h(x) \leq 0\} = \{x \in C : g(x) \leq 0\}$. If S is nonempty and there exists some $\ell > 0$ such that*

$$d(x, S) \leq \ell g(x) \quad \forall x \in C,$$

then one has :

(i) $\alpha(\tau) = \alpha$ and $\mathcal{P} \subset \mathcal{P}_\tau$ for all $\tau \geq L\ell$

(ii) $\mathcal{P} = \mathcal{P}_\tau$ for all $\tau > L\ell$.

Exact penalty techniques in Mixed Integer DC Programming

A Mixed Interger DC Program is of the form

$$\alpha := \inf\{f(x, y) = g(x, y) - h(x, y) : (x, y) \in K, x \in [l, u] \cap \mathbb{Z}^n\} \tag{1.15}$$

where K is a bounded polyhedral convex set in $\mathbb{R}^{n+p}$: $K := \{(x, y) \in \mathbb{R}^{n+p} : C(x, y) := Ax + By \leq b\}$ with $C \in \mathbb{R}^{m\times(n+p)}$, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times p}$, $[l, u] := \prod_{i=1}^{n} [l_i, u_i] \subset \mathbb{R}^n$, with $l_i, u_i \in \mathbb{Z}$,

$l_i < u_i$ for $i = 1, ..., n$. This class of problems encompasses most combinatorial optimization problems, in particular usual mixed integer linear/quadratic programming problems. In [190], Pham Dinh and Le Thi proposed five following functions to penalize the interger variable x :

1) $p_1(x, y) := \sum_{j=1}^{n} \sin^2(\pi x_j)$.

2) $p_2(x, y) := d_2^2(x, [l, u] \cap \mathbb{Z}^n) = \min\{\|x - z\|_2^2 : \ z \in [l, u] \cap \mathbb{Z}^n\} = \sum_{j=1}^{n} \min\{(x_j - z_j)^2 : \ z_j \in [l_j, u_j] \cap \mathbb{Z}\}$ is piecewise convex.

3) $p_3(x, y) := \sum_{j=1}^{n} |\sin \pi x_j|$ is piecewise concave.

4) $p_4(x, y) := d_1(x, [l, u] \cap \mathbb{Z}^n) = \min\{\|x - z\|_1 : z \in [l, u] \cap \mathbb{Z}^n\}$
$= \sum_{j=1}^{n} \min\{|x_j - z_j| : \ z_j \in [l_j, u_j] \cap \mathbb{Z}\}$ is piecewise concave.

5) $p_5(x, y) = p_5(x) := \sum_{j=1}^{n} p_5^j(x_j)$, with $p_5^j(x_j) := \max\{[x_j - (l_j + k)][(l_j + k + 1) - x_j] : k = 0, ..., (u_j - l_j) - 1\}$ is piecewise concave.

It is clear that all five above penalty functions are DC functions. Using one of these penalty functions, the problem (1.15) can be reformulate as a continuous optimization problem

$$\alpha := \inf\{f(x, y) = g(x, y) - h(x, y) + \tau p(x, y) : (x, y) \in K, x \in [l, u]\} \tag{1.16}$$

Pham Dinh and Le Thi [190] have proved the exact penalty for the last three penalty functions. As for the first two penalty functions, there is no proof for exact penalty to date.

1.4 DC Algorithm

1.4.1 DCA for solving standard DC programs : standard DCA

DCA consists in the construction of the two sequences $\{x^k\}$ and $\{y^k\}$ (candidates to primal and dual solutions, respectively) which are easy to calculate and satisfy the following properties :

i) The sequences $(g - h)(x^k)$ and $(h^\star - g^\star)(y^k)$ are decreasing.

ii) Their corresponding limits x^∞ and y^∞ either satisfy the local optimality condition $(x^\infty, y^\infty) \in \mathcal{P}_\ell \times \mathcal{D}_\ell$ or are critical points of $g - h$ and $h^\star - g^\star$, respectively.

Based on the local optimality conditions (Theorem 1.1), DCA constructs the sequences $\{x^k\}$ and $\{y^k\}$, from a given initial point $x^0 \in \operatorname{dom} g$, by setting

$$y^k \in \partial h(x^k); x^{k+1} \in \partial g^\star(y^k).$$

The interpretation of DCA is simple. At iteration k, one replaces the second component h in the primal program (P) by its affine minorant

$$h_k(x) = h(x^k) + \langle x - x^k, y^k \rangle, \tag{1.17}$$

where $y^k \in \partial h(x^k)$. Then the primal program (P) becomes the following convex program

$$(P_k) \qquad \inf\{g(x) - h_k(x) : x \in X\} = \inf\{g(x) - \langle x, y^k \rangle : x \in X\}$$

The solution of (P_k) is nothing but $x^{k+1} \in \partial g^\star(y^k)$. Dually, a solution x^{k+1} of (P_k) is then used to define the dual convex program (D_{k+1}) obtained from (D) by replacing the second DC

component $g^\star$ by its affine minorization $g_k^\star(y) = g^\star(y^k) + \langle y - y^k, x^{k+1}\rangle$. The convex problem (D_{k+1}) is written as

$$(D_{k+1}) \qquad \inf\{h^\star(y) - g_k^\star(y) : y \in X\} = \inf\{h^\star(y) - \langle y, x^{k+1}\rangle : y \in Y\}.$$

The solution set of (D_{k+1}) is exactly $\partial h(x^{k+1})$.

The standard DCA for solving the standard DC program (P) is described in Algorithm 1.1.

Algorithm 1.1 Standard DCA for solving a standard DC program

1: **Initialization :** Choose an initial point x^0, $k \leftarrow 0$.
2: **repeat**
3: Compute $y^k \in \partial h(x^k)$.
4: Compute $x^{k+1} \in \arg\min\left\{g(x) - \langle y^k, x\rangle\right\}$.
5: $k \leftarrow k + 1$.
6: **until** Stopping criterion.

Convergence properties of DCA

Complete convergence properties are given in [189]. However, it is worthwhile to mention the following properties :

Theorem 1.3 ([189]) *Suppose that the sequences $\{x^k\}$ and $\{y^k\}$ are generated by DCA. Then we have*

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

- *$g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ if and only if $\{x^k, x^{k+1}\} \subset \partial g^\star(y^k) \cap \partial h^\star(y^k)$ and $[\rho(h) + \rho(g)]\|x^{k+1} - x^k\| = 0$.*
- *$h^\star(y^{k+1}) - g^\star(y^{k+1}) = h^\star(y^k) - g^\star(y^k)$ if and only if $\{y^k, y^{k+1}\} \subset \partial g(x^k) \cap \partial h(x^k)$ and $[\rho(h^\star) + \rho(g^\star)]\|y^{k+1} - y^k\| = 0$.*

DCA terminates at the k-th iteration if either of the above equalities holds.

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

iii) *If the optimal value α is finite and the 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. $h^\star - g^\star$).*

iv) *DCA has a linear convergence for general DC program.*

v) *In polyhedral DC programs, the sequences $\{x^k\}$ and $\{y^k\}$ contain finitely many elements and DCA has a finite convergence.*

vi) *If DCA converges to a point x^* that admits a convex neighborhood in which the objective function f is finite and convex (i.e. the function f is locally convex at x^*) and if the second DC component h is differentiable at x^*, then x^* is a local minimizer to the problem (P).*

Remark 1.5

- *DCA works with the convex DC components g and h but not with the DC function f itself. Thus DCA can enjoy all the fundamental properties and results of convex optimization.*

- *The most important among the key properties of DCA is the flexibility : there are as many DCA as there are DC decompositions. Since a DC function f has infinitely many DC decompositions which have crucial impacts on the qualities (speed of convergence, robustness, efficiency, globality of computed solutions,...) of DCA, one can explore and exploit the effects of DC decomposition to design efficient DCA schemes. For a given DC program, the choice of "good" DC decompositions is still open. Of course, it should strongly depend on the very specific structure of the considered problem.*
- *In practice, DCA quite often converges to global optimal solutions. The globality of DCA may be assessed either when the optimal values are known a priori, or through global optimization techniques such as Branch-and-Bound or cutting plane.*

Convergence of DCA with subanalytic data

The general convergence of DCA above mentioned states that every convergent subsequence of the sequence $\{x^k\}$ (resp. $\{y^k\}$) converges to a generalized KKT point of (P) (resp. (D)). From a theoretical point of view, convergence rate analysis of DCA is an open issue. In [190], the authors have studied the convergence of the whole sequences $\{x^k\}$ and $\{y^k\}$ as well as their convergence rate, in case the objective functions and the constraints are subanalytic.

Theorem 1.4 ([190]) *Let us consider DC program (P) with $\alpha \in \mathbb{R}$. Suppose that the sequences $\{x^k\}$, $\{y^k\}$ are defined by the DCA and contained in the two convex sets C and D, respectively.*

(i) Suppose that the DC function $f := g - h$ is subanalytic such that dom f *is closed; $f\,|_{\mathrm{dom}\, f}$ is continuous and that g or h is differentiable on* dom h *or* dom g*, respectively with locally Lipschitz derivative. Assume that $\rho(g, C) + \rho(h, C) > 0$, where $\rho(g, C)$, $\rho(h, C)$ are the strong convexity modulus of g, h, respectively. If either the sequence $\{x^k\}$ or $\{y^k\}$ is bounded then $\{x^k\}$ and $\{y^k\}$ are convergent to critical points of (P_{dc}) and (D_{dc}), respectively.*

(ii) Similarly, for the dual problem, suppose that $h^* - g^*$ is subanalytic such that dom $(h^* - g^*)$ *is closed; $(h^* - g^*)\,|_{\mathrm{dom}\,(h^* - g^*)}$ is continuous and that g^* or h^* is differentiable on* dom g^* *or* dom h^**, respectively with locally Lipschitz derivative. If $\rho(g^*, D) + \rho(h^*, D) > 0$ and either the sequence $\{x^k\}$ or $\{y^k\}$ is bounded then $\{x^k\}$ and $\{y^k\}$ are convergent to critical points of (P_{dc}) and (D_{dc}), respectively.*

The convergence rates of the sequence $\{x^k\}$ generated by DCA is given in Theorem 1.5.

Theorem 1.5 ([190]) *Suppose that the assumptions of Theorem 1.4 (i) are satisfied. Let x^∞ be the limit point of $\{x^k\}$ with a Lojasiewicz exponent $\theta \in [0, 1)$. Then there exists constants $\tau_1, \tau_2 > 0$ such that*

$$\|x^k - x^\infty\| \leq \sum_{j=k}^{\infty} \|x^j - x^{j+1}\| \leq \tau_1 \|x^k - x^{k-1}\| + \tau_2 \|x^k - x^{k-1}\|^{\frac{1-\theta}{\theta}}, \quad k = 1, 2, ... \tag{1.18}$$

As a result, one has

- *If $\theta \in (1/2, 1)$ then $\|x^k - x^\infty\| \leq c k^{\frac{1-\theta}{1-2\theta}}$ for some $c > 0$.*
- *If $\theta \in (0, 1/2]$ then $\|x^k - x^\infty\| \leq c q^k$ for some $c > 0; q \in (0, 1)$.*
- *If $\theta = 0$ then $\{x^k\}$ is convergent in a finite number of steps.*

1.4.2 Links between DCA and standard convex/nonconvex programming approaches

As mentioned above, thanks to its flexibility, DCA recovers most standard methods in convex and nonconvex optimization with an appropriate DC decomposition and/or suitable DC reformulations.

Le Thi and Pham Dinh [144] have analyzed the common points and the difference between DCA and Majorization-Minimization (MM). Recall that MM method for solving $\min\{f(x) : x \in \mathcal{X} \subset \mathbb{R}^n\}$ consists on computing iteratively $x^{k+1} \in \arg\min\{\vartheta(x, x^k) : x \in \mathcal{X}\}$ where ϑ is a majorization of f such that $f(x) \leq \vartheta(x, y), \forall x, y \in \mathcal{X}$ and $f(x) = \vartheta(x, x), \forall x \in \mathcal{X}$. Note that, when ϑ is convex, the MM is known under the name Successive Convex Approximation (SCA). We can see that, DCA and MM are both a philosophy but not a simply an algorithm, they give a way to construct algorithms. However, MM does not specify the way to construct the surrogate function $\vartheta(x, x)$ which is a challenging task. Meanwhile, DCA gives the simplest and the most closed convex surrogate function of objective funcion f. MM works directly with the nonconvex f which can be a source of difficulty. As for DCA, by working with g and h, (the DC components of f) which are convex, DCA enjoys all the fundamental convex analysis. Furthermore, while the MM method majorizes iteratively the whole function f, DCA approximates only one part of f. Thus, it is likely that the majorization in DCA is better than in MM. Regardless of its generality, most related works using the MM/SCA method in the literature can be recovered as versions of DCA. Le Thi and Pham Dinh [144] have showed that SCA using commonly used convex surrogate functions in the literature (e.g. linear upper bounds, quadratic upper bounds and proximal upper bounds) are DCA versions.

Furthermore, in [144], Le Thi and Pham Dinh have proved that

— proximal point algorithm and Goldstein-Levitin-Polyak projection algorithm for convex programming are DCA versions ;

— concave-convex procedure (CCCP) is an instance of DCA in smooth optimization ;

— EM algorithm for exponential families is a special case of DCA ;

— Iterative shrinkage-thresholding algorithm (ISTA) is a DCA version.

Beside the above-mentioned algorithms, several other ones are also recognized as special cases of DCA, such as Single block SCA, Jacobi SCA Algorithm, NOVA (iNner cOnVer Approximation algorithm), SUM (Successive upper-bound minimization), FBSA (Forward-Backward Splitting Algorithm), GIST (General Iterative Shrinkage and Thresholding), etc. For a complete list and a deeper discussion, readers are referred to [144].

1.4.3 DCA for solving General DC Programs

Two approaches were proposed for the general DC program (1.3) [125, 190]. Both of them consist in reformulating (1.3) as a standard DC program then apply standard DCA to solve the latter. Both approaches can be viewed as a sequence of standard DCAs with updated penalty (resp. relaxation) parameters, which marks the passage from standard DCAs to general DCAs [144].

The first approach is based on penalty technique. Let the functions p and p^+ be defined by

$$p(x) := \max\{f_i(x) : i = 1, ..., m\};\ I(x) := \{i \in \{1, ..., m\} :\ f_i(x) = p(x)\}$$
$$p^+(x) := \max\{0, p(x)\}.$$

It is easy to see that $p(x)$ and $p^+(x)$ are DC functions with the following DC decompositions

$$p(x) = \max_{i=1,\ldots,m} \left\{ g_i(x) + \sum_{j=1, j\neq i}^{m} h_j(x) \right\} - \sum_{j=1}^{m} h_j(x) \tag{1.19}$$

$$p^+(x) = \max_{i=1,\ldots,m} \left\{ \sum_{j=1}^{m} h_j(x), g_i(x) + \sum_{j=1, j\neq i}^{m} h_j(x) \right\} - \sum_{j=1}^{m} h_j(x) \tag{1.20}$$

where g_i, h_i are the DC components of f_i, $i = 1, \ldots, m$.

Thus, the general DC program (1.3) is reformulated as

$$\alpha = \inf\{f_0(x) := g_0(x) - h_0(x) \ : \ x \in C, \ p^+(x) \leq 0\} \tag{1.21}$$

and its penalized problem is a standard DC program

$$\alpha(\tau) = \inf\{\varphi_\tau(x) := f(x) + \ \tau p^+(x) : \ x \in C\} \qquad (P_\tau) \tag{1.22}$$

where $\tau > 0$ is a penalty parameter.

Let DC decomposition of p^+ be given by

$$p^+(x) = p_1(x) - p_2(x), \tag{1.23}$$

Then, φ_τ is a DC function with the below DC decomposition

$$\varphi_\tau(x) = g_\tau(x) - h_\tau(x), \tag{1.24}$$

where,

$$g_\tau(x) := g_0(x) + \tau p_1(x); \quad h_\tau(x) := h_0(x) + \tau p_2(x). \tag{1.25}$$

Exact penalty (relative the constraint $p^+(x) \leq 0$) for (1.21) means that there is $\tau_0 \geq 0$ such that for all $\tau > \tau_0$ the general DC program (1.3) is equivalent to the penalized problem (1.22) in the sense that they have the same global solution set and $\alpha(\tau) = \alpha$. Hence, the solution of the general DC program (1.3) can be obtained by solving the standard DCA program (1.22) by standard DCA. However, in practice, the penalty parameter τ_0 is generally unknown. To overcome this issue, Pham Dinh and Le Thi [190] proposed a DCA scheme with updated penalty parameter. Instead of solving the problem (P_τ) with a fixed penalty parameter τ, one applies DCA to the sequence of (P_{τ_k}) with an increasing sequence of penalty parameters $\{\tau_k\}$ given by a updating rule from the current iteration x^k such that x^{k+1} is the next iteration of DCA applied to (P_{τ_k}) from x^k.

One considers the sequence of penalized DC programs (1.22)

$$\inf\{\varphi_k(x) := f_0(x) + \beta_k p^+(x) : x \in C\} \tag{1.26}$$

with penalty parameters $\tau = \beta_k$ and $\varphi_k(x) = \varphi_\tau(x)$. DCA with updated penalty parameter is described in Algorithm 1.2.

Note that the update rule of penalty parameter is to ensure that if the sequence $\{\beta_k\}$ is unbounded then $\|x^{k+1} - x^k\| \to 0$ and $r_k > 0$.

To state the converge properties of Algorithm 1.2, we need the following assumptions.

Assumption 1.1 *(i) $f_i's$ $(i = 0, ..., m)$ are locally Lipschitz functions at every point of C.*

Algorithm 1.2 DCA with updated penalty parameter for solving general DC program (1.3)

1: **Initialization :** Choose an initial point $x^0 \in C$; $\delta > 0$; an initial penalty parameter $\beta_0 > 0$, $k \leftarrow 0$, STOP = false.
2: **repeat**
3: Compute $y^k \in \partial h_k(x^k)$.
4: Compute $x^{k+1} \in \partial(g_k + \chi_C)^*(y^k)$,i.e., x^{k+1} is a solution of the convex program

$$\min\{g_k(x) - \langle x, y^k\rangle : \ x \in C\}. \tag{1.27}$$

5: Stopping test : if $x^{k+1} = x^k$ and $p(x^k) \leq 0$ then STOP=true.
6: Update the penalty parameter : Compute $r_k := \min\{p(x^k), p(x^{k+1})\}$ and set

$$\beta_{k+1} = \begin{cases} \beta_k & \text{if either} \quad \beta_k \geq \|x^{k+1} - x^k\|^{-1} \ \text{ or } \ r_k \leq 0, \\ \beta_k + \delta & \text{if} \quad \beta_k < \|x^{k+1} - x^k\|^{-1} \ \text{ and } \ r_k > 0, \end{cases}$$

7: $k \leftarrow k + 1$.
8: **until** STOP = true.

(ii) Either g_k or h_k is differentiable on C, and $\rho(g_0, C) + \rho(h_0, C) + \rho(p_1, C) + \rho(p_2, C) > 0$.

(iii) The (extended) Mangasarian-Fromowitz constraint qualification (EMFCQ) is satisfied at any $x \in \mathbb{R}^n$ with $p(x) \geq 0$.

Theorem 1.6 *Suppose that $C \subseteq \mathbb{R}^n$ is a nonempty closed convex set and f_i, $i = 1, ..., m$ are DC functions on C. Suppose further that the Assumption 1.1 is verified.*

Let $\delta > 0$, $\beta_1 > 0$ be given. Let $\{x^k\}$ be a sequence generated by Algorithm 1.2. Then Algorithm 1.2 either stops, after finitely many iterations, at a KKT point x^k for problem (1.3) *or generates an infinite sequence $\{x^k\}$ of iterates such that $\lim_{k\to\infty} \|x^{k+1} - x^k\| = 0$ and every limit point x^∞ of the sequence $\{x^k\}$ is a KKT point of problem* (1.3).

In the second approach for solving the general DC program (1.3), one applies the main idea of DCA (that is the linearization of concave part of the DC structure) to the DC constraints. Thus, the generalized DCA for general DC program (1.3) consists in solving a sequence of convex programs of the form :

$$\begin{aligned} \inf\{ \quad & g_0(x) - \langle y_0^k, x\rangle : \\ & x \in C, \ g_i(x) - h_i(x^k) - \langle y_i^k, x - x^k\rangle \leq 0, \quad i = 1, ..., m\} \end{aligned} \tag{1.28}$$

where, $x^k \in \mathbb{R}^n$ is the current iterate, $y_i^k \in \partial h_i(x^k)$ for $i = 0, ..., m$. This linearization introduces an inner convex approximation of the feasible set of (1.3) which is quite often poor and can lead to infeasibility of convex subproblem (1.28). To avoid this issue, one considers the following problem instead of (1.28)

$$\begin{aligned} \inf\{ \quad & g_0(x) - \langle y_0^k, x\rangle + \beta_k t : \\ & x \in C, g_i(x) - [h_i(x^k) + \langle y_i^k, x - x^k\rangle] \leq t, \quad i = 1, ..., m; \\ & t \geq 0\} \end{aligned} \tag{1.29}$$

where $\beta_k > 0$ is a penalty parameter. The second DCA scheme for solving the general DC problem (1.3) is presented in Algorithm 1.3.

The following lemma is needed to investigate the convergence of Algorithm 1.3.

Algorithm 1.3 DCA for solving general DC program (1.3)

1: **Initialization :** Choose an initial point $x^0 \in C$; $\delta_1, \delta_2 > 0$; an initial penalty parameter $\beta_0 > 0$, $k \leftarrow 0$, STOP = false.
2: **repeat**
3: Compute $y_i^k \in \partial h_i(x^k)$, $i = 0, ..., m$.
4: Compute (x^{k+1}, t^{k+1}) as the solution of (1.29), and the associated Lagrange multipliers $(\lambda^{k+1}, \mu^{k+1})$.
5: Stopping test : $x^{k+1} = x^k$ and $t^{k+1} = 0$ then STOP=true.
6: Update the penalty parameter : Compute $r_k := \min\{\|x^{k+1} - x^k\|^{-1}, \|\lambda^{k+1}\|_1 + \delta_1\}$, where $\|\lambda^{k+1}\|_1 = \sum_{i=1}^m |\lambda_i^{k+1}|$, and set

$$\beta_{k+1} = \begin{cases} \beta_k & \text{if} \quad \beta_k \geq r_k, \\ \beta_k + \delta_2 & \text{if} \quad \beta_k < r_k. \end{cases}$$

7: $k \leftarrow k + 1$.
8: **until** STOP = true.

Lemma 1.1 *The sequence (x^k, t^k) generated by Algorithm 1.3 satisfies the following inequality*

$$\varphi_k(x^k) - \varphi_k(x^{k+1}) \geq \frac{\rho}{2}\|x^{k+1} - x^k\|^2, \ \text{for all } k = 1, 2, ... \tag{1.30}$$

where, $\rho := \rho(g_0, C) + \rho(h_0, C) + \min\{\rho(g_i, C) : i = 1, ...m\}$.

The convergence of Algorithm 1.3 is stated in the Theorem 1.7.

Theorem 1.7 *Suppose that $C \subseteq \mathbb{R}^n$ is a nonempty closed convex set and f_i, $i = 1, ..., m$, are DC functions on C such that (i) and (iii) in Assumption 1.1 are verified. Suppose further that for each $i = 0, ..., m$, either g_i or h_i is differentiable on C and that*

$$\rho := \rho(g_0, C) + \rho(h_0, C) + \min\{\rho(g_i, C) : i = 1, ...m\} > 0.$$

Let $\delta_1, \delta_2 > 0$, $\beta_1 > 0$ be given. Let $\{x^k\}$ be a sequence generated by Algorithm 1.3. Then Algorithm 1.3 either stops, after finitely many iterations, at a KKT point x^k for problem (1.3) or generates an infinite sequence $\{x^k\}$ of iterates such that $\lim_{k\to\infty} \|x^{k+1} - x^k\| = 0$ and every limit point x^∞ of the sequence $\{x^k\}$ is a KKT point of problem (1.3).

1.5 Open issues and new trends in DC Programming and DCA development

Despite the bright successes of DC Programming and DCA for modeling and solving numerous nonconvex non-smooth optimization problems, one can still exploit the full power and freedom offered by these tools to further improve the proposed DCAs . Indeed, DCA is rather a philosophy than an algorithm and the design of an efficient DCA for a considered problem is not a simple procedure. Several aspects should be studied while developing DCA [144].

(i) *Finding an good DC decomposition.* A DC function has infinitely many DC decompositions, thus there is not only one DCA for a considered problem. The choice of a "good" DC decomposition is still an open issue. This should greatly depend on the very specific structure

of the problem being considered. From the computational point of view, the complexity of DCA depends strongly on the solution methods for convex sub-problems. Thus, designing fast and scalable solvers for convex sub-problems is an important issue of DCA. The ideal DC decomposition corresponds to an explicit DCA, i.e. the solution of convex sub-problem at each iteration can be explicitly computed.

(ii) *Finding a good starting point.* A good starting point for DCA can be obtained by using or combining with other approaches such as heuristic algorithms or local search methods. Another way consists is finding a convex minorant of the objective function and solving the resulting convex program whose solution is used to initialize DCA [144].

(iii) *Improve the convergence speed of DCA.* One can use the Nesterov acceleration technique or Armijo type linesearch to boost the convergence speed of DCA.

(iv) *Globalizing DCA.* Several strategies can be investigated to obtain the global solution : DCA multi-start, combine DCA with global approaches such as Brand and Bound or Cut methods, finding convex minorants of DC functions for computing lower bounds of optimal values.

The new trend in development of DC programming and DCA consists in developing advanced variants of DCA in order to improve the standard DCA and to deal with very large-scale optimization problems. Recently, several variants of DCA have been proposed : Approximate DCA [230], DCA with successive DC decomposition [144], Stochastic DCA [191], Online DCA [105, 124, 106], Boosted DCA [10, 11], etc.

Première partie

Advanced techniques in DC programming and DCA

In this part, we present several advanced variants of DCA in order to improve the standard DCA and to deal with very large-scale optimization problems.

In Chapter 2, we introduce Accelerated DCA (ADCA) with the aim to improve the convergence of standard DCA. Firstly, we develop ADCA for solving the standard DCA by incorporating the Nesterov's acceleration technique into standard DCA (Section 2.2). The acceleration step in ADCA which consists in using an extrapolated point from the current iterate x^k and the previous one, aims to find a point z^k which is better than x^k for the computation of x^{k+1}. We prove that ADCA converges to a critical point of the standard DC program. Furthermore, we study the convergence rate of ADCA under Kurdyka-Lojasiewicz (KL) assumption. In Section 2.3, we investigate ADCA for a special class of optimization problems, namely the sum of two nonconvex function minimization problem

$$\min_{x\in\mathbb{R}^n} f(x) + r(x),$$

where f is a differentiable function with L-Lipschitz continuous gradient (possibly nonconvex) and r is a DC function. Exploiting the fact that $f(x)$ is differentiable with L-gradient, we propose, an efficient DC decomposition for which the corresponding ADCA scheme is inexpensive. As application, we address the sparse binary logistic regression problem (Section 2.4). It turns out that ADCA for solving the latter is very inexpensive : it only required the soft thresholding operator which is explicitly computed.

In Chapter 3, we develop the second variant of DCA, named DCA-Like. DCA-Like is "like" DCA in the sense that they iteratively approximate the DC program by a sequence of convex ones. However, DCA-Like is "unlike" DCA in the way to approximate the objective function for which we possibly do not have a DC decomposition. While standard DCA works with a convex majorization of the objective function on the whole space via an available DC decomposition, DCA-Like seeks a better convex approximation at the current solution through a decomposition which is not necessarily DC. Thus, DCA-Like works even when one can not highlight a DC decomposition. We investigate DCA-Like for two special classes of optimization problems. In Section 3.2, we consider the minimization of the sum of a nonconvex, differentiable function with L-Lipschitz continuous gradient and composite functions, i.e.,

$$\min_{x\in X} f(x) + \sum_{i=1}^{m} h_i(g_i(x_i)),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is nonconvex differentiable with L-Lipschitz continuous gradient, $g_i : \mathbb{R}^{n_i} \to \mathbb{R}$ are continuous convex (possibly nonsmooth) where x_i (for $i = 1, \ldots, m$) are sub-vectors of x with $\sum_{i=1}^{m} n_i = n$, $h_i : \mathbb{R} \to \mathbb{R}$ are concave increasing and $X \subset \mathbb{R}^n$ is closed convex set. Exploiting the special structure of the considered problem, we develop DCA-Like to solve it. Furthermore, we improve DCA-Like by incorporating the Nesterov's acceleration technique into it, to obtain the so named Accelerated DCA-Like (ADCA-Like). We prove that the whole bounded sequence $\{x^k\}$ generated by DCA-Like converges to a critical point of considered problem under some conditions. Furthermore, under Kurdyka-Łojasiewicz assumption, the convergence rate of DCA-Like is proved to be at least sublinear $\mathcal{O}(1/k^\alpha)$ with $\alpha > 1$.

Following the same direction, in Section 3.3, we develop DCA-Like and DCA-Like for the minimization of the sum of two nonconvex function minimization problem (that we had considered in Section 2.3).

The efficiency of DCA-Like and ADCA-Like is illustrated in three applications in machine learning : the t-distributed Stochastic Embedding (t-SNE) problem (Section 3.4), the group variables selection multi-class logistic regression (Section 3.5) and the sparse binary logistic regression (Section 3.6).

In Chapter 4, we present another variant of DCA, named Stochastic DCA (SDCA), for solving the large sum of DC functions minimization which takes the form :

$$\min_{x\in\mathbb{R}^d}\left\{F(x) := \frac{1}{n}\sum_{i=1}^{n} F_i(x)\right\},$$

where F_i are DC functions, i.e., $F_i(x) = g_i(x) - h_i(x)$ with g_i and h_i being lower semi-continuous proper convex functions, and n is a very large integer number. The above problem has a double difficulties due to the nonconvexity of F_i and the large value of n. By taking the advantage of the sum structure, we develop SDCA to solve the large sum of DC functions minimization problem. The basic idea of stochastic DCA is to update, at each iteration, the minorant of only some randomly chosen h_i while keeping the minorant of the other h_i. We prove that the SDCA's convergence is guaranteed with probability one. We further propose an inexact stochastic DCA version in which both subgradient of H and optimal solution of the resulting convex program are approximately computed. In Section 4.3, we apply SDCA to the group variables selection in multi-class logistic regression. By using a suitable DC decomposition of the objective function, we design a SDCA scheme in which all computations are explicit and inexpensive.

Chapitre 2

Accelerated DCA for standard DC program[1]

Abstract: In this chapter, we present a variant of standard DCA with the aim to accelerate the DCA's convergence. The proposed algorithm, named Accelerated DCA (ADCA), consists in incorporating the Nesterov's acceleration technique into DCA. We first investigate ADCA for solving the standard DC program and rigorously study its convergence properties and the convergence rate. Secondly, we develop ADCA for a special case of the standard DC program whose the objective function is the sum of a differentiable function with L-Lipschitz continuous gradient (possibly nonconvex) and a DC function. We exploit the special structure of the problem to propose an efficient DC decomposition for which the corresponding ADCA scheme is inexpensive. As an application, we consider the sparse binary logistic regression problem. Numerical experiments on several benchmark datasets illustrate the efficiency of our algorithm and its superiority over well-known methods.

2.1 Introduction

Nowadays, especially with the Big Data explosion, it has been becoming more and more important to develop advanced optimization methods able to handle very large-scale problems. Motivated by the success of DCA on several nonconvex/nonsmooth programs to which it was proved to be more robust and more efficient than related standard methods, we aim to investigate a variant of DCA in order to improve its performance.

Our contribution : Firstly, we introduce Accelerated DCA (ADCA) for solving the standard DC program. In ADCA, we incorporate the Nesterov's acceleration technique into standard DCA in order to improve its performance. The idea of ADCA is different to the usual line search acceleration using Armijo type rule which can be computationally costly. The acceleration step in ADCA which consists in using an extrapolated point from the current iterate x^k and the previous one, aims to find a point z^k which is better than x^k for the computation of x^{k+1}. We then provide a rigorous argument to prove the interesting convergence properties of ADCA as well as the convergence rate under the Lojasiewicz assumption.

1. The results presented in this chapter were published in :

- D.N. Phan, H.M. Le, H.A. Le Thi, Accelerated Difference of Convex functions Algorithm and its Application to Sparse Binary Logistic Regression, 27th International Joint Conference on Artificial Intelligence and 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 2018), 1369-1375, 2018.

Secondly, we investigate ADCA for a special case of the standard DC program, namely the sum of two nonconvex function minimization problem

$$\min_{x\in\mathbb{R}^n} f(x) + r(x), \tag{2.1}$$

where f is a differentiable function with L-Lipschitz continuous gradient (possibly nonconvex) and r is a DC function. The problem (2.1) covers several nonconvex and nonsmooth problems arising from various fields such as machine learning, computational biology, signal processing, etc.

The problem (2.1) has been attracting attention of many researchers. Proximal gradient (PG) methods, which are also known as different names ISTA, fixed point iteration, forward-backward splitting (see e.g. [111, 21, 57, 93]), have been extensively developed for the convex case of (2.1), i.e., both f and r are convex. In [170], the author introduced the first accelerated proximal gradient (APG) method for solving (2.1) with f convex and $r = 0$. Later, Beck and Teboulle [20, 21] extended it for the case where both f and r are convex. These algorithms exhibit a global convergence rate $\mathcal{O}(1/k^2)$, where k is the iteration counter. Recently, several extensions of APG method for the convex case of (2.1) have been proposed for the nonconvex cases. For instance, Li and Lin [154] have extended the method of [20] for f differentiable with L-Lipschitz gradient and r nonconvex. In [90], the authors presented inexact versions of PG and APG for solving (2.1) with f smooth and r nonsmooth (possibly nonconvex). Another inexact APG for the nonconvex case of (2.1) which only requires one proximal step at each iteration, were proposed in [248]. However, the aforementioned algorithms have to compute the proximal map of nonconvex functions r which do not has closed form in many cases. Usually, this computation can be very expensive or impossible.

In this work, by exploiting the properties of f and r, we propose an efficient DC decomposition for which the corresponding DCA and ADCA are inexpensive.

Finally, as an application, we consider the sparse binary logistic regression and carefully perform numerical experiments of all proposed algorithms.

The remainder of the chapter is organized as follows. In Section 2.2 we introduce ADCA for the standard DC program and study its convergence properties. ADCA for solving the problem (2.1) is presented in Section 2.3. The numerical experiments on the sparse binary logistic regression problem are reported in Section 2.4 and Section 2.5 concludes the chapter.

2.2 Accelerated DCA for the standard DC program

Let us consider the standard DC program

$$(P) \qquad \alpha = \inf\{F(x) = G(x) - H(x) : x \in X\} \tag{2.2}$$

where $G, H \in \Gamma_0(X)$.

Recall that the standard DCA (Algorithm 1.1) for solving the standard DC program (P) consists in the construction of the two sequences $\{x^k\}$ and $\{y^k\}$ such that $y^k \in \partial H(x^k)$ and x^{k+1} is an optimal solution of the convex sub-problem $\min\big\{G(x) - \langle y^k, x\rangle\big\}$.

One of the ways to improve standard DCA is to incorporate acceleration techniques. Aragon Artacho et al. [10] introduced Boosted DCA which accelerates standard DCA by a line search using an Armijo type rule. In Boosted DCA, the point x^k generated by DCA is used to define the search direction which was proposed by Fukushima-Mine [77]. The first version of Boosted DCA [10] was proposed only to smooth DC functions. Later, it was extended for some classes of non-smooth DC functions [11].

In our work, we introduced a different way to accelerate the standard DCA. The so called Accelerated DCA (ADCA in short) consists in incorporating the Nesterov's acceleration technique [170] (which was proposed for convex programs) into standard DCA. More precisely, the acceleration step in ADCA aims to find a point z^k which is an extrapolated point of the current iterate x^k and the previous iterate x^{k-1} via Nesterov's acceleration formulation

$$z^k = x^k + \frac{t_k - 1}{t_{k+1}} \left(x^k - x^{k-1} \right),$$

where $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$. If z^k is better than one of last $q+1$ iterates $\{x^{k-q}, \ldots, x^{k-1}, x^k\}$ in terms of objective function, i.e., $F(z^k) \leq \max\limits_{t=\max(0,k-q),\ldots,k} F(x^t)$ then z^k will be used instead of x^k to compute y^k in the DCA scheme. This condition allows the objective function $F(x)$ to increase and consequently to escape from a potential bad local minimum [89, 239]. Theoretically, a large-value of q increases the chance of using the extrapolated points z^k in ADCA. Note that if $q = 0$ then $F(z^k) \leq F(x^k)$ and ADCA is a monotone algorithm like DCA. ADCA is described in Algorithm 2.1.

Algorithm 2.1 ADCA for solving the standard DC program

1: **Initialization :** Choose an initial point x^0, $z^0 = x^0$, $q \in \mathbb{N}$, $t_0 = (1+\sqrt{5})/2$, and $k \leftarrow 0$.
2: **repeat**
3: If $F(z^k) \leq \max_{t=\max(0,k-q),\ldots,k} F(x^t)$ then set $v^k = z^k$, otherwise set $v^k = x^k$.
4: Compute $y^k \in \partial H(v^k)$.
5: Compute $x^{k+1} = arg\min_{x \in X} \left\{ G(x) - \langle y^k, x \rangle \right\}$.
6: Compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$ and $z^{k+1} = x^{k+1} + \frac{t_k - 1}{t_{k+1}} \left(x^{k+1} - x^k \right)$ if $k \geq 1$
7: $k \leftarrow k + 1$.
8: **until** Stopping criterion.

Convergence of ADCA. The first result provides the behavior of the limit points of the sequence $\{x^{\phi(k)}\}$ generated by ADCA, where $\phi(k) = \arg\min_{t=k+1,\ldots,k+1+q} \|x^t - v^{t-1}\|^2$.

Theorem 2.1 *Let $\mu(g)$ and $\mu(h)$ be the convex modulus of g and h, respectively. If $\alpha = \inf_{x \in \mathbb{R}^n} f(x) > -\infty$ and $\min\{\mu(g), \mu(h)\} > 0$, then for any subsequence $\{x^{\phi(k_j)}\}$ of $\{x^{\phi(k)}\}$, converging to x^* such that $\{y^{\phi(k_j)-1}\}$ is bounded, the limit point x^* is a critical point of* (2.2).

To prove Theorem 2.1, we will use the following lemma. Denote by $\{\Gamma^k\}$ the sequence defined as

$$\Gamma^k \quad = \max_{t=\max(0,k-q),\ldots,k} F(x^t).$$

Lemma 2.1 *Let $\{x^k\}$ and $\{v^k\}$ be sequences generated by ADCA. The following statements hold.*
(i) For any $k = 0, 1, ...,$

$$\Gamma^k - \Gamma^{k+1+q} \geq \frac{\mu(G) + \mu(H)}{2} \|x^{\phi(k)} - v^{\phi(k)-1}\|^2. \tag{2.3}$$

As a result, by choosing $q = 0$, we get the monotone property of $\{F(x^k)\}$, i.e., $F(x^k) - F(x^{k+1}) \geq \frac{\mu(G)+\mu(H)}{2} \|x^{k+1} - v^k\|^2$.

(ii) If $\alpha = \inf_{x \in \mathbb{R}^n} F(x) > -\infty$ and $\min\{\mu(G), \mu(H)\} > 0$, then $\sum_{k=0}^{+\infty} \|x^{\phi(k)} - v^{\phi(k)-1}\|^2 < +\infty$, and therefore $\lim_{k \to +\infty} \|x^{\phi(k)} - v^{\phi(k)-1}\| = 0$.

Proof 2.1 *First let us justify (i) by noting from the μ-convexity of G and $y^k \in \partial G(x^{k+1})$ that*

$$G(v^k) \geq G(x^{k+1}) + \langle y^k, v^k - x^{k+1}\rangle + \frac{\mu(G)}{2}\|v^k - x^{k+1}\|^2.$$

It follows from the μ-convexity of H and $y^k \in \partial H(v^k)$ that

$$H(x^{k+1}) \geq H(v^k) + \langle y^k, x^{k+1} - v^k\rangle + \frac{\mu(H)}{2}\|x^{k+1} - v^k\|^2.$$

Summing two above inequalities, we have

$$F(v^k) - F(x^{k+1}) \geq \frac{\mu(G)+\mu(H)}{2}\|x^{k+1} - v^k\|^2. \tag{2.4}$$

Observe that $F(v^k) \leq \max_{t=\max(0,k-q),...,k} F(x^t) = \Gamma^k$. It follows from this and (2.4) that

$$F(x^{k+1}) \leq \Gamma^k - \frac{\mu(G)+\mu(H)}{2}\|x^{k+1} - v^k\|^2. \tag{2.5}$$

This implies that $F(x^{k+1}) \leq \Gamma^k$. We prove by induction that for all $t = 0, ..., q$

$$F(x^{k+1+t}) \leq \Gamma^k - \frac{\mu(G)+\mu(H)}{2}\|x^{k+1+t} - v^{k+t}\|^2. \tag{2.6}$$

Indeed, it follows from (2.5) that the claim holds for $t = 0$. We suppose that it also holds for $t = 0, ..., p-1$ with $1 \leq p \leq q$. Thus, we have

$$\begin{array}{rl} F(x^{k+1+p}) & \leq \Gamma^{k+p} - \frac{\mu(G)+\mu(H)}{2}\|x^{k+1+p} - v^{k+p}\|^2 \\ & \leq \max(\Gamma^k, F(x^{k+1}), ..., F(x^{k+p})) - \frac{\mu(G)+\mu(H)}{2}\|x^{k+1+p} - v^{k+p}\|^2 \\ & \leq \Gamma^k - \frac{\mu(G)+\mu(H)}{2}\|x^{k+1+p} - v^{k+p}\|^2, \end{array}$$

where last inequality follows from $F(x^{k+1+t}) \leq \Gamma^k$ for $t = 0, ..., p-1$. Therefore, we obtain

$$\begin{array}{rl} \Gamma^{k+q+1} = \max_{t=k+1,...,k+q+1} F(x^t) & \leq \Gamma^k - \min\limits_{t=k+1,...,k+1+q} \frac{\mu(G)+\mu(H)}{2}\|x^t - v^{t-1}\|^2 \\ & = \Gamma^k - \frac{\mu(G)+\mu(H)}{2}\|x^{\phi(k)} - v^{\phi(k)-1}\|^2. \end{array}$$

Next let us prove (ii) by noting that $\Gamma^k \geq \alpha$ for all k. Summing (2.3) over $k = 0, ..., N$ we get

$$\begin{array}{rl} \frac{\mu(G)+\mu(H)}{2}\sum_{k=0}^N \|x^{\phi(k)} - v^{\phi(k)-1}\|^2 & \leq \sum_{t=0}^q (\Gamma^t - \Gamma^{N+t+1}) \\ & \leq (q+1)(\max_{t=0,...,q} F(x^t) - \alpha). \end{array}$$

Since $\min\{\mu(G), \mu(H)\} > 0$, we have

$$\sum_{k=0}^N \|x^{\phi(k)} - v^{\phi(k)-1}\|^2 \leq \frac{2(q+1)(\max\limits_{t=0,...,q} F(x^t) - \alpha)}{\mu(G)+\mu(H)}.$$

Passing to the limit over the sequence $\{N\}_{N\in\mathbb{N}}$, we obtain

$$\sum_{k=0}^{+\infty} \|x^{\phi(k)} - v^{\phi(k)-1}\|^2 < +\infty, \tag{2.7}$$

and therefore $\lim_{k\to+\infty} \|x^{\phi(k)} - v^{\phi(k)-1}\| = 0$. ■

Proof 2.2 (Proof of Theorem 2.1) *Let $\{x^{\phi(k_j)}\}$ be a subsequence of $\{x^{\phi(k)}\}$ that converges to x^*. It follows from (ii) of Lemma 2.1 that $\lim_{j\to+\infty} v^{\phi(k_j)-1} = x^*$. Without loss of generality, we can suppose that the sequence $\{y^{\phi(k_j)-1}\}$ converges to y^*. By the closed property of the subdifferential mapping ∂H, we have $y^* \in \partial H(x^*)$. We note that*

$$x^{\phi(k_j)} \in \arg\min\{G(x) - \langle y^{\phi(k_j)-1}, x\rangle\}.$$

This implies that $y^{\phi(k_j)-1} \in \partial G(x^{\phi(k_j)})$. By the closedness of ∂G, we obtain $y^* \in \partial G(x^*)$. Therefore, $y^* \in \partial G(x^*) \cap \partial H(x^*)$. It follows from this that x^* is a critical point of the DC program (2.2). ■

Recall that a lower semicontinuous function f has the Lojasiewicz property [12] if for any limiting-critical point x^*, that is $0 \in \partial^L F(x^*)$, there exist $C, \epsilon > 0$ and $\theta \in [0, 1)$ such that

$$|F(x) - F(x^*)|^\theta \leq C\|\hat{x}\|, \ \forall x \in B(x^*, \epsilon), \ \forall \hat{x} \in \partial^L F(x).$$

Here $\partial^L F(x)$ denotes the limiting-subdifferential of F at x. The class of functions having the Lojasiewicz property is very ample, for example, semi-algebraic, subanalytic, and log-exp functions.

Lemma 2.2 *Consider the settings of Theorem 2.1. Let $\{x^k\}$ be sequence generated by ADCA with $q = 0$. Denote by Ω the set of limit points of $\{x^k\}$. Suppose further that $\{x^k\}$ and $\{y^k\}$ are bounded, and f is lower semicontinuous. The following statements hold.*

(i) Ω is a compact set and $\lim_{k\to\infty} F(x^k) = F(x^*)$ for some $x^* \in \Omega$. Thus, F has the same value on Ω, which is denoted by F^*.

(ii) If F has the Lojasiewicz property and H is differentiable, then there exist $C, \epsilon > 0$ and $\theta \in [0, 1)$ such that $\forall x \in \{x \in \mathbb{R}^n : dist(x, \Omega) \leq \epsilon\}$, one has

$$|F(x) - F^*|^\theta \leq C\|\hat{x}\|, \tag{2.8}$$

$\forall \hat{x} \in \partial^L F(x)$.

(iii) If F has the Lojasiewicz property and G is differentiable, then there exist $C, \epsilon > 0$ and $\theta \in [0, 1)$ such that $\forall x \in \{x \in \mathbb{R}^n : dist(x, \Omega) \leq \epsilon\}$, one has

$$|F(x) - F^*|^\theta \leq C\|\hat{x}\|, \tag{2.9}$$

$\forall \hat{x} \in \partial^L(-F)(x)$.

Proof 2.3 *(i) Since $\{x^k\}$ is bounded and Ω is the set of its limit points, Ω is a compact set. It follows from $\alpha = \inf_{x\in\mathbb{R}^n} F(x) > -\infty$ and (i) of Lemma 2.1 that the sequence $\{F(x^k)\}$ is non-increasing and bounded below. Thus, there exists $F^* = \lim_{k\to\infty} F(x^k)$. Let $x^* \in \Omega$. There exists a subsequence $\{x^{k_j}\}$ that converges to x^*. Without loss of generality, we can assume that $\{y^{k_j-1}\}$ converges to y^*. We note that*

$$x^{k_j} \in \arg\min\{G(x) - \langle y^{k_j-1}, x - v^{k_j-1}\rangle\}.$$

This implies that for all x,

$$G(x^{k_j}) - \langle y^{k_j-1}, x^{k_j} - v^{k_j-1}\rangle \ \leq G(x^*) - \langle y^{k_j-1}, x^* - v^{k_j-1}\rangle.$$

Taking $j \to +\infty$ gives us that

$$\limsup_{j\to+\infty} G(x^{k_j}) \leq G(x^*). \tag{2.10}$$

Therefore, we have

$$\begin{aligned}\limsup_{j\to\infty} F(x^{k_j}) &= \limsup_{j\to\infty}[G(x^{k_j}) - H(x^{k_j})]\\ &\le \limsup_{j\to\infty} G(x^{k_j}) - \liminf_{j\to\infty} H(x^{k_j})\\ &\le G(x^*) - \liminf_{j\to\infty} H(x^{k_j})\\ &\le G(x^*) - H(x^*) = F(x^*),\end{aligned}$$

where the second inequality follows from (2.10) *and the last inequality holds by the lower semicontinuity of* H*. On the other hand, from the lower semicontinuity of* F*, we obtain* $\liminf_{j\to\infty} F(x^{k_j}) \ge F(x^*)$*. Hence, by the uniqueness of limit, we have* $F^* = F(x^*)$*.*

Let us justify (ii) by noting that $\partial^L F(x^*) = \partial G(x^*) - \nabla H(x^*)$ *for all* $x^* \in \Omega$*. Hence,* Ω *is a subset of the limiting-critical points of* F*. According to Lemma 1 in [12], applied to the function* F*, there exist* $C, \epsilon > 0$ *and* $\theta \in [0, 1)$ *such that* $\forall x \in \mathbb{R}^n$*,* $dist(x, \Omega) \le \epsilon$*,* $\forall \hat{x} \in \partial^L F(x)$*, one has*

$$|F(x) - F^*|^\theta \le C\|\hat{x}\|.$$

By using a similar argument, we have the result (iii). ■

We now provide the asymptotic convergence rate of ADCA under the Kurdyka-Lojasiewicz (KL) assumption.

Theorem 2.2 *Consider the settings of Theorem 2.1. Suppose further that either* G *or* H *is differentiable with locally Lipschitz derivative. Let* $\{x^k\}$ *be sequence generated by ADCA with* $q = 0$*. Assume that* F *is lower semicontinuous and has the Lojasiewicz property, and* $\{x^k\}, \{y^k\}$ *are bounded. Denote by* θ *the parameter, which is defined as in Lemma 2.2. The following estimations hold*

(i) If $\theta = 0$*, then the sequence* $\{F(x^k)\}$ *converges to* F^* *in a finite number of steps.*

(ii) If $\theta \in (0, 1/2]$*, then the sequence* $\{F(x^k)\}$ *converges linearly to* F^**.*

(iii) If $\theta \in (1/2, 1)$*, then there exist positive constants* η *and* N_0 *such that* $F(x^k) - F^* \le \eta k^{-\frac{1}{2\theta-1}}$*, for all* $k \ge N_0$*.*

Proof 2.4 *Let us consider the following cases.*

Case 1. G *is differentiable and its derivative is locally Lipschitz. For each* $x \in \Omega$*, there exist* $L_x, \epsilon_x > 0$ *such that*

$$\|\nabla G(u) - \nabla G(v)\| \le L_x\|u - v\| \ \forall u, v \in B(x, \epsilon_x). \tag{2.11}$$

From the compactness of Ω*, there exist* $w^1, ..., w^m \in \Omega$ *such that* $\Omega \subset \cup_{i=1}^m B(w^i, \epsilon_{w^i}/4)$*, where* $B(w, \epsilon)$ *is the open ball with the center* w *and radius* ϵ*. Set* $L = \max\{L_{w^i} : i = 1, ..., m\}$ *and* $\epsilon = \min\{\epsilon_{w^i}/2 : i = 1, ..., m\}$*. It follows from the (ii) of Lemma 2.1 that* $\{v^k\}$ *and* $\{x^k\}$ *share the same the set of limit points* Ω*. Hence, there exists* $N_1 > 0$ *such that* $v^k \in \cup_{i=1}^m B(w^i, \epsilon_{w^i}/2)$ *and* $\|x^{k+1} - v^k\| \le \epsilon$ *whenever* $k \ge N_1$*. Thus, for any* $k \ge N_1$*, there is* w^i *such that* $x^{k+1}, v^k \in B(w^i, \epsilon_{w^i})$*. This implies that*

$$\|\nabla G(x^{k+1}) - \nabla G(v^k)\| \ \le L_{w^i}\|x^{k+1} - v^k\| \le L\|x^{k+1} - v^k\|. \tag{2.12}$$

On the other hand, since G *is differentiable and by the definition of* x^{k+1}*, we have*

$$\begin{aligned}\nabla G(x^{k+1}) - \nabla G(v^k) &= y^k - \nabla G(v^k)\\ &\in \partial H(v^k) - \nabla G(v^k) = \partial^L(-F)(v^k).\end{aligned}$$

Therefore, from the (iii) of Lemma 2.2 and dist$(v^k, \Omega) \to 0$, by increasing N_1 if necessary, we have for all $k \geq N_1$

$$|F(v^k) - F^*|^\theta \leq C\|G(x^{k+1}) - \nabla G(v^k)\|.$$

Combining this and $F(v^k) - F^* \geq F(x^{k+1}) - F^* \geq 0$, we get

$$\begin{aligned} |F(x^{k+1}) - F^*|^{2\theta} &\leq C^2\|G(x^{k+1}) - \nabla G(v^k)\|^2 \\ &\leq (CL)^2\|x^{k+1} - v^k\|^2 \\ &\leq \tfrac{2(CL)^2}{\mu(G)+\mu(H)}[F(x^k) - F(x^{k+1}], \end{aligned}$$

where the second inequality follows from (2.12) *and the last inequality follows from (i) of Lemma 2.1. Hence, by setting $r_k = F(x^k) - F^*$, we obtain*

$$r_{k+1}^{2\theta} \leq \frac{2(CL)^2}{\mu(G) + \mu(H)}[r_k - r_{k+1}]. \tag{2.13}$$

Case 2. H is differentiable and its derivative is locally Lipschitz. Similar to Case 1, we can find $L, N_2 > 0$ such that for any $k \geq N_2$,

$$\|\nabla H(v^k) - \nabla H(x^{k+1})\| \leq L\|v^k - x^{k+1}\|. \tag{2.14}$$

Since H is differentiable and by the definition of x^{k+1}, we have

$$\begin{aligned} \nabla H(v^k) - \nabla H(x^{k+1}) &= y^k - \nabla H(x^{k+1}) \\ &\in \partial G(x^{k+1}) - \nabla H(x^{k+1}) \\ &= \partial^L F(x^{k+1}). \end{aligned} \tag{2.15}$$

Therefore, from the (ii) of Lemma 2.2 and dist$(x^{k+1}, \Omega) \to 0$, by increasing N_2 if necessary, we have for all $k \geq N_2$,

$$\begin{aligned} |F(x^{k+1}) - F^*|^{2\theta} &\leq C^2\|\nabla H(v^k) - \nabla H(x^{k+1})\|^2 \\ &\leq (CL)^2\|v^k - x^{k+1}\|^2 \\ &\leq \tfrac{2(CL)^2}{\mu(G)+\mu(H)}[F(x^k) - F(x^{k+1})], \end{aligned}$$

where the second inequality follows from (2.14) *and the last inequality follows from the (i) of Lemma 2.1. Hence, we obtain*

$$r_{k+1}^{2\theta} \leq \frac{2(CL)^2}{\mu(G) + \mu(H)}[r_k - r_{k+1}]. \tag{2.16}$$

Thus, from (2.13) *and* (2.16)*, we have shown that, in both cases, there exists $\tau > 0$ such that*

$$r_{k+1}^{2\theta} \leq \tau[r_k - r_{k+1}]. \tag{2.17}$$

By using similar arguments in [74], it is easy to show that sequence $\{r_k\}$ satisfying the above inductive property converges to zero at different rates according to θ as stated in the theorem. ■

We consider now a special case of the standard DC program whose the objective function is the sum of a differentiable function with L-Lipschitz continuous gradient (possibly nonconvex) and a DC function. We exploit the special structure of the problem to propose an efficient DC decomposition for which the corresponding ADCA scheme is inexpensive.

2.3 Accelerated DCA for the sum of two nonconvex functions minimization problem

We consider now the sum of two nonconvex functions minimization problem (2.1). Recall that (2.1) is of the form

$$\min_{x\in\mathbb{R}^n} F(x) = f(x) + r(x),$$

where $f(x)$ is a differentiable non-convex function with L-Lipschitz continuous gradient and $r(x)$ is a DC function.

Since f is a function with L-Lipschitz continuous gradient, it can be expressed as a DC function : $f(x) = \frac{\rho}{2}\|x\|^2 - \left[\frac{\rho}{2}\|x\|^2 - f(x)\right]$, where $\rho \geq L$. Let $r(x) := g_r(x) - h_r(x)$, then a DC decomposition of $F(x) = f(x) + r(x)$ is given by

$$F(x) = G(x) - H(x), \tag{2.18}$$

where $G(x) = \frac{\rho}{2}\|x\|^2 + g_r(x)$ and $H(x) = \frac{\rho}{2}\|x\|^2 - f(x) + h_r(x)$. Thus, DCA and ADCA can be applied to solve (2.1). According to Algorithm 2.1, ADCA for solving the problem (2.1) is described in Algorithm 2.2.

Algorithm 2.2 ADCA for solving (2.1)

1: **Initialization :** Choose an initial point x^0, $z^0 = x^0$, $q \in \mathbb{N}$, $t_0 = (1+\sqrt{5})/2$, $\rho > L$, and $k \leftarrow 0$.
2: **repeat**
3: If $F(z^k) \leq \max_{t=\max(0,k-q),\ldots,k} F(x^t)$ then set $v^k = z^k$, otherwise set $v^k = x^k$.
4: Compute $y^k = \rho v^k - \nabla f(v^k) + \xi^k$, where $\xi^k \in \partial h_r(v^k)$.
5: Compute x^{k+1} by solving strongly convex problem

$$\min_{x\in\mathbb{R}^n} \left\{\frac{\rho}{2}\|x\|^2 + g_r(x) - \langle y^k, x\rangle\right\}. \tag{2.19}$$

6: Compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$ and $z^{k+1} = x^{k+1} + \frac{t_k - 1}{t_{k+1}}\left(x^{k+1} - x^k\right)$ if $k \geq 1$
7: $k \leftarrow k+1$.
8: **until** Stopping criterion.

Note that $\mu(g) > \rho > 0$ and the convergence results of Algorithm 2.2 are guaranteed by Theorem 2.1 and Theorem 2.2. Furthermore, with most of existing nonconvex regularizers r in the literature, the resolution of the sub-problem (2.19) is inexpensive or can be explicitly computed.

2.4 Application of ADCA on the sparse binary logistic regression problem

We consider now the sparse binary logistic regression problem. The problem can be described as follows. Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training set with observation vectors $x_i \in \mathbb{R}^d$ and labels $y_i \in \{-1, 1\}$. We aim to find a hyperplane $\langle w, x\rangle + b$ that separates the two classes. To find w and b, we maximize the log-likelihood function $-\frac{1}{n}\sum_{i=1}^n \log(1 + \exp(-y_i(x_i^T w + b)))$. On the other hand, to deal with irrelevant and redundant features in high-dimensional data, we use

features selection method which consists in minimizing the zero-norm of w. Hence the sparse binary logistic regression is formulated by

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i(x_i^T w+b))) + \lambda\|w\|_0 \right\}, \tag{2.20}$$

where $\lambda > 0$ is the trade-off parameter between the two terms. It is well-known that the minimization of zero-norm is NP-hard. The readers are referred to Chapter 5 for an review and our works on the minimization of zero-norm. In this work, we replace $\|x\|_0$ by a nonconvex approximation, namely the concave exponential function defined by $r_{exp}(w) = \sum_{i=1}^{d}(1-\exp(-\alpha|w_i|))$. Thus, the problem (2.20) becomes

$$\min_{(w,b)} \left\{ F(w,b) = \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i(x_i^T w+b))) + \lambda r_{exp}(w) \right\}. \tag{2.21}$$

Let $f(w,b) = \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i(x_i^T w+b)))$ and $r(w) = \lambda r_{exp}(w)$. Note that r is a DC function with the following DC decomposition $r = g - h$, where

$$g(w) = \lambda\alpha\|w\|_1 \text{ and } h(w) = \lambda\sum_{i=1}^{d}(\alpha|w_i| - 1 + \exp(-\alpha|w_i|)).$$

Thus, $F(w,b)$ is a DC function with

$$\begin{array}{ll} F(w,b) & = G(w,b) - H(w,b), \\ G(w,b) & = \frac{\rho}{2}\|(w,b)\|^2 + g(w), \\ H(w,b) & = \frac{\rho}{2}\|(w,b)\|^2 - f(w,b) + h(w) \end{array} \tag{2.22}$$

where $\rho \geq L$. On the other hand, f is a differentiable function whose gradient $\nabla f(w,b) = (\nabla_w f(w,b), \nabla_b f(w,b))$ is defined by

$$\begin{aligned} \nabla_w f(w,b) &= \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i}{1+\exp(y_i(x_i^T w+b))} x_i, \\ \nabla_b f(w,b) &= \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i}{1+\exp(y_i(x_i^T w+b))}. \end{aligned} \tag{2.23}$$

The hessian $\nabla^2 f$ of f is given by

$$\begin{aligned} \nabla_w^2 f(w,b) &= \frac{1}{n}\sum_{i=1}^{n} \frac{\exp(y_i(x_i^T w+b))}{\left(1+\exp(y_i(x_i^T w+b))\right)^2} x_i x_i^T, \\ \nabla_b^2 f(w,b) &= \frac{1}{n}\sum_{i=1}^{n} \frac{\exp(y_i(x_i^T w+b))}{\left(1+\exp(y_i(x_i^T w+b))\right)^2}. \end{aligned}$$

Hence, we can compute a bound of the spectral radius of $\nabla^2 f$ as follows.

$$\begin{aligned} \|\nabla^2 f(w,b)\| &\leq \frac{1}{n}\sum_{i=1}^{n} \frac{\exp(y_i(x_i^T w+b))}{\left(1+\exp(y_i(x_i^T w+b))\right)^2} \|(x_i,1)\|_2^2 \\ &\leq \frac{1}{4n}\sum_{i=1}^{n} (\|x_i\|^2+1). \end{aligned}$$

It follows that $f(w,b)$ has Lipschitz continuous gradient with a Lipschitz constant $L = \frac{1}{4n}\sum_{i=1}^{n}(\|x_i\|^2+1)$. Thus, the problem (2.21) takes the form of (2.1) and then can be solved by DCA and ADCA. In DCA and ADCA, at each iteration k, we need to compute $(u^k, v^k) \in \partial H(w^k, b^k)$ and solve a convex sub problem of the form

$$\min\left\{\frac{\rho}{2}\|(w,b)\|^2 + \lambda\alpha\|w\|_1 - \langle(u^k,v^k),(w,b)\rangle\right\}. \tag{2.24}$$

Recall that the gradient of f is given by (2.23) and $\xi^k \in \partial h(w^k)$ can be computed as

$$\xi^k = \lambda\alpha\sum_{i=1}^{d}(1-\exp(-\alpha|w_i^k|))\text{sign}(w_i^k). \tag{2.25}$$

Moreover, it is easy to see that the convex problem (2.24) can be explicitly solved by

$$\begin{array}{ll} w^{k+1} & = \mathcal{S}\left(u^k/\rho_k, \lambda\alpha/\rho_k\right), \\ b^{k+1} & = v^k/\rho_k. \end{array} \tag{2.26}$$

where $\mathcal{S}(x,t): \mathbb{R}^d \times \mathbb{R}_+ \to \mathbb{R}^d$ is the elementwise soft-thresholding operator defined by $\mathcal{S}(x,t)_i = \text{sign}(x_i)(|x_i|-t)_+$.

Finally, DCA and ADCA for solving (2.21) is given in Algorithm 2.3 and Algorithm 2.4.

Algorithm 2.3 DCA for solving (2.21)

1: **Initialization :** Choose an initial point (w^0, b^0), $\rho > L$, and $k \leftarrow 0$.
2: **repeat**
3: Compute $u^k = \rho w^k - \nabla_w f(w^k, b^k) + \xi$ and $v^k = \rho b^k - \nabla_b f(w^k, b^k)$ with ξ^k and ∇f defined in (2.23) and (2.25).
4: Compute (w^{k+1}, b^{k+1}) by (2.26).
5: **until** Stopping criterion.
6: $k \leftarrow k+1$.

Algorithm 2.4 ADCA for solving (2.21)

1: **Initialization :** Choose an initial point (w^0, b^0), $(\omega^0, \beta^0) = (w^0, b^0)$, $q \in \mathbb{N}$, $t_0 = (1+\sqrt{5})/2$, $\rho > L$, and $k \leftarrow 0$.
2: **repeat**
3: If $F(w^k, b^k) \le \max_{t=\max(0,k-q),\ldots,k} F(w^t, b^t)$ then set $(\bar{w}^k, \bar{b}^k) = (\omega^k, \beta^k)$, otherwise set $(\bar{w}^k, \bar{b}^k) = (w^k, b^k)$.
4: Compute $u^k = \rho\bar{w}^k - \nabla_w f(\bar{w}^k, \bar{b}^k) + \xi$ and $v^k = \rho\bar{b}^k - \nabla_b f(\bar{w}^k, \bar{b}^k)$ with ξ^k and ∇f defined in (2.23) and (2.25).
5: Compute (w^{k+1}, b^{k+1}) by (2.26).
6: Compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$ and $(\omega^{k+1}, \beta^{k+1}) = (w^{k+1}, b^{k+1}) + \frac{t_k-1}{t_{k+1}}\left((w^{k+1}, b^{k+1}) - (w^k, b^k)\right)$ if $k \ge 1$
7: $k \leftarrow k+1$.
8: **until** Stopping criterion.

2.4.1 Experiment setting

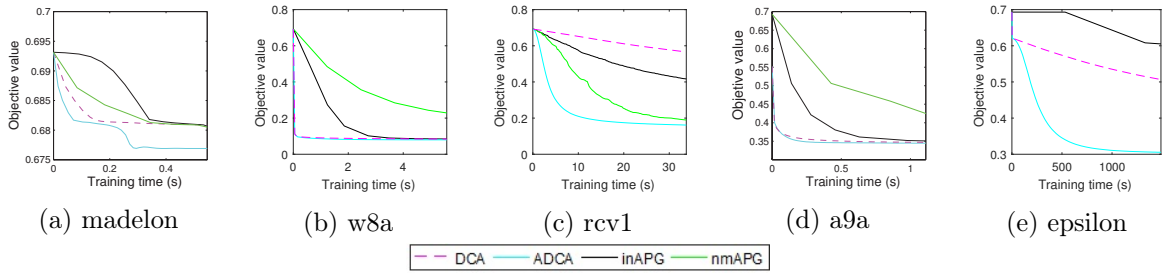
The numerical experiment was performed on data sets from the well-known repertory LibSVM. The detailed information of used datasets is summarized in the first column of Table 2.1. n_{train} (resp. n_{test}) represents the number of points in training set (resp. test set) while d is the number of features.

The two comparative algorithms are inexact APG (inAPG) [248] and monotone APG (nmAPG) [154] which are two variants of accelerated proximal gradient (APG) developed for (2.1). For DCA and ADCA, we estimated a Lipschitz constant L by computing a bound of Hessian matrix of logistic loss. Note that *inAPG* and nmAPG require to compute the proximal mapping of the DC function r_{exp}. However, this proximal mapping do not have a closed form. We therefore use DCA to compute the proximal mapping of r in inAPG and nmAPG. All the algorithms are terminated when the change of two consecutive objective function values is less than 10^{-5}. We also stop algorithms after 5h (18.000 seconds) of CPU time.

All experiments are performed on a PC Intel i7 CPU3770, 3.40 GHz of 8GB RAM and the codes were written in MATLAB. We fix $\alpha = 5$ as proposed in [35]. The parameter q is set to 5. The trade-off parameter λ is fixed to 10^{-4} on `rcv1, epsilon` and 10^{-3} on the other data sets. For the *epsilon* dataset, nmAPG algorithm did not furnish any result due to a out of memory problem.

2.4.2 Numerical results

The numerical results are summarized in Table 2.1.



(a) madelon (b) w8a (c) rcv1 (d) a9a (e) epsilon

FIGURE 2.1 – Objective value versus training time (in seconds)

We observe that ADCA gives the best classification accuracy for all 5 datasets (all four algorithms give the same result on *madelon*). As for the sparsity of solution, the four algorithms are comparable. All four algorithms give the same results on *madelon* and the best sparsity of solution on *a9a* is obtained by DCA, inAPG, nmAPG. nmAPG suppresses more features than the others on *rcv1* while ADCA gives the best sparsity of solution on *w8a, epsilon*. In terms of running time, ADCA is clearly the fastest one among the four compared algorithms. ADCA improves considerably the running time comparing to DCA : ADCA is up to 12.09 times faster than DCA (*epsilon*). ADCA is faster than inAPG and nmAPG which also use acceleration technique. We can observe from Figure 2.1 that the objective functions of ADCA decrease drastically in few first iterations comparing to the others three algorithms.

TABLE 2.1 – Comparative results. Bold values correspond to the best results for each dataset

Dataset	Method	Time (s)	Acc (%)	Sparsity (%)
`madelon`	DCA	1.14	**62.17**	**0.4**
n_{train}=2 000	ADCA	**0.54**	**62.17**	**0.4**
n_{test}=600	inAPG	0.86	**62.17**	**0.4**
d=500	nmAPG	1.23	**62.17**	**0.4**
`w8a`	DCA	32.8	98.43	16
n_{train}=49 749	ADCA	**5.58**	**98.51**	**15.67**
n_{test}=14 951	inAPG	36.42	98.45	17
d=300	nmAPG	54.81	98.4	19.33
`rcv1`	DCA	113.03	91.8	0.87
n_{train}=20 242	ADCA	**33.74**	**94.23**	0.79
n_{test}=677 399	inAPG	39.65	91.1	0.85
d=47 236	nmAPG	112.37	93.9	**0.72**
`a9a`	DCA	7.11	84.95	**32.52**
n_{train}=32 561	ADCA	**1.11**	**84.98**	33.33
n_{test}=16 281	inAPG	6.38	84.97	**32.52**
d=123	nmAPG	14.01	84.97	**32.52**
`real-sim`	DCA	33.25	94.49	2.71
n_{train}=57 847	ADCA	**7.74**	**94.5**	**2.59**
n_{test}=14 462	inAPG	33.49	94.42	2.62
d=20 958	nmAPG	53.11	94.39	2.82
`epsilon`	DCA	18000	87.53	7.77
n_{train}=400 000	ADCA	**1488**	**88.22**	**7.75**
n_{test}=100 000	inAPG	18000	73.14	12.6
d=2 000	nmAPG	NA	NA	NA

2.5 Conclusion

We have incorporated the Nesterov's acceleration technique into DCA which gives rise to ADCA. We have proved that ADCA converges to a critical point of the standard DC program. Furthermore, we have studied the convergence rate of ADCA under Kurdyka-Lojasiewicz (KL) assumption. ADCA is further developed for minimizing the sum of a (possibly nonconvex) differentiable function with L-Lipschitz continuous gradient and a DC function, which is a special case of the standard DC program. Exploiting the fact that $f(x)$ is differentiable with L-Lipschitz gradient, we propose, an efficient DC decomposition for which the corresponding ADCA scheme is inexpensive. To evaluate the performance of proposed algorithm, we consider the sparse binary logistic regression problem. ADCA for solving the latter is very inexpensive : it only required the soft thresholding operator which is explicitly computed. Numerical results showed that ADCA improves considerably the running time of DCA (up to 12.09 times faster than DCA) while giving similar or better classification accuracy and sparsity of solution. Furthermore, ADCA outperformed related accelerated proximal gradient methods such as non-monotone APG and inexact APG.

The present work is only one among several options of acceleration techniques which can be investigated into DCA. In future works we plan to develop

i) Nesterov acceleration technique for solving convex program in DCA (inner acceleration) ;

ii) Momentum based methods which accelerates the subgradient $y^k \in \partial h(x^k)$ in the first step of DCA (outer acceleration) ;

iii) Combined inner-outer accelerations.

Chapitre 3

DCA-Like and Accelerated DCA-Like for some classes of nonconvex optimization problems[1]

Abstract: In this chapter, we present two new variants of standard DCA, named DCA-Like and Accelerated DCA-Like to address two special classes of structured nonconvex minimization problems. The objective function in the first class is the sum of a differentiable nonconvex function with L-Lipschitz continuous gradient and a composite function while the one in the second class is the sum of a differentiable nonconvex function with L-Lipschitz continuous gradient and a DC function. Being common models of various applications, especially in machine learning, these classes of problems attract the attention of many researchers during the last years. DCA-Like is based on a new and efficient way to approximate the DC program without knowing a DC decomposition while Accelerated DCA-Like further improve DCA-Like by incorporating the Nesterov's acceleration technique into it. The convergence properties and convergence rate of the proposed algorithms are rigorously studied. We prove that, they subsequently converge from every initial point to a critical point of the considered problems. Moreover, we study their convergence rate with Kurdyka-Lojasiewicz assumption. Finally, we investigate the proposed algorithms for three important problems in machine learning : the sparse binary logistic regression, the group variables selection in multi-class logistic regression and the t-distributed stochastic neighbor embedding. Numerical experiments on several benchmark datasets illustrate the efficiency of our algorithms.

3.1 Introduction

In this chapter, we propose two new variants of standard DCA, named DCA-Like and Accelerated DCA-Like to address two special classes of nonconvex optimization problems.

The main idea of DCA relies on DC decompositions of the objective function. DCA consists in approximating a DC (nonconvex) program by a sequence of convex ones. Hence, to apply DCA, we have to highlight a DC decomposition of the objective function. However, it is not easy or always possible to find an explicit DC decomposition of a DC function. On another hand, the

1. The results presented in this chapter were published/submitted in
   - H.A. Le Thi, H.M. Le, D.N Phan, B. Tran, Novel DCA Based Algorithms for a special class of nonconvex problems with Application in Machine learning, Applied Mathematics and Computation, 409, 2021.
   - H.A. Le Thi, D.N Phan, H.M. Le, DCA-Like and its Boosted scheme for a class of Nonconvex Optimization Problems, submitted.

way to approximate the DC objective function by convex functions plays an important role in the design of DCA. Devising efficient ways to approximate the DC objective function without knowing a DC decomposition is a challenge in DC programming and DCA.

The first work addressing this challenge was [134], which considered the problems of minimizing the sum of a nonconvex, differentiable function with L-Lipschitz continuous gradient and composite functions. It takes the form

$$\min_{x\in X}\left\{F_1(x) := f(x) + \sum_{i=1}^{m} h_i(g_i(x_i))\right\}. \tag{3.1}$$

Here $f : \mathbb{R}^n \to \mathbb{R}$ is nonconvex differentiable with L-Lipschitz continuous gradient, $g_i : \mathbb{R}^{n_i} \to \mathbb{R}$ are continuous convex (possibly nonsmooth) where x_i (for $i = 1, \ldots, m$) are sub-vectors of x with $\sum_{i=1}^{m} n_i = n$, $h_i : \mathbb{R} \to \mathbb{R}$ are concave increasing and $X \subset \mathbb{R}^n$ is closed convex set. In [134], exploiting the special structure of the problem (3.1) we extended DCA to give rise to the so-named DCA-Like, which is based on a new and efficient way to approximate the DC objective function without knowing its DC decomposition. We further improve DCA-Like by incorporating the Nesterov's acceleration technique into it, and design the so named Accelerated DCA-Like (ADCA-Like in short).

DCA-Like is "like" DCA in the sense that they iteratively approximate the DC program (3.1) by a sequence of convex ones. However, DCA-Like is "unlike" DCA as it relaxes two key requirements in DCA :

i) in the manner to decompose $F_1 = G_\rho - H_\rho$, with a parameter ρ, in which H_ρ is not necessarily convex (i.e. we possibly do not have a DC decomposition of F_1), and ρ is updated so that the value of the surrogate convex function of F_1 at the current solution could be as close as possible to F_1 ;

ii) the affine minorant of H_ρ may not be a lower bound of H_ρ on the whole space but rather only at the current solution.

We prove that, fortunately, in spite of these modifications, the whole bounded sequence $\{x^k\}$ generated by DCA-Like still converges to a critical point of (3.1) under some conditions. Furthermore, we demonstrate that, under Kurdyka-Łojasiewicz assumption, the convergence rate of DCA-Like is at least sublinear $\mathcal{O}(1/k^\alpha)$ with $\alpha > 1$.

Late, in [150], following the same direction, we developed DCA-Like and ADCA-Like for the second class of problems which minimize the sum of a nonconvex differentiable function with L-Lipschitz continuous gradient and a DC function :

$$\min_{x\in\mathbb{R}^n}\{F_2(x) := f(x) + r(x)\}, \tag{3.2}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable nonconvex function with L-Lipschitz continuous gradient and $r(x)$ is a DC function of which a DC decomposition is available. The problem of minimizing F_2 under a convex set Δ is also of the type (3.2), as the convex constraint $x \in \Delta$ can be incorporated into the objective function F_2 via the indicator function χ_Δ.

The two above classes of problems are common mathematical models of various problems arising from different domains such as signal processing, finance, computational biology, machine learning,.... For instance, several problems in machine learning are formulated as minimizing the sum of a loss function f and a regularization term r (weighted by a trade-off parameter between these two terms) and in many cases, f is nonconvex and r is DC. In particular, the zero norm regularization is often considered in learning with sparsity, a very important area of

machine learning. Several approaches used a sparsity inducing function to approximate the zero norm and it has been showed in [147] that most existing sparsity inducing nonconvex functions are DC. On the other hand, they can also be expressed as a sum of composite functions $h_i(g_i(\cdot))$.

The two problems (3.1) and (3.2) are linked, but the special structure of composite functions requires special techniques to handle them. If the composite function $h_i(g_i(\cdot))$ in (3.1) is DC (it is not always the case), then (3.1) takes the form of (3.2). However, even if $h_i(g_i(x_i))$ is proved to be DC (and so is $\sum_{i=1}^{m} h_i(g_i(x_i))$), it is not easy to highlight a DC decomposition of it. We will see in the next section that the problem (3.1) can be reformulated in the form of (3.2) by using an additional variable $z \in \mathbb{R}^m$. Meanwhile, our algorithms in [134] exploit well the structure of composite functions, they work only on the variable x, then adding variable z doesn't make the algorithms more complicated.

The remainder of the chapter is organized as follows. Section 3.2 is devoted to DCA-Like and ADCA-Like for solving (3.1) as well as theirs convergence properties convergence and convergence rate. The resolution of (3.2) by DCALike/ADCA-Like is presented in Section 3.3. In Section 3.4 and 3.5, respectively, we show how to apply the proposed algorithms for Problem (3.1) on the t-distributed Stochastic Neighbor Embedding (t-SNE) and the sparse multi-class logistic regression. They are the two very important problems in machine learning. Section 3.6 deals with the proposed algorithms for Problem (3.2) applied on sparse binary logistic regression. Finally, in Section 3.7 we give some final remarks and conclude the chapter.

3.2 Minimizing the sum of a nonconvex, differentiable function with L-Lipschitz continuous gradient and composite functions

In this section, we address the sum of a nonconvex, differentiable function with L-Lipschitz continuous gradient and composite functions minimization problem (3.1).

In learning with sparsity problems (which involve the ℓ_0-norm), one often approximates the ℓ_0-norm by one of the sparse reducing regularizers in Table 3.1 and then solves the resulting problems which have the form of the problem (3.1). For instance, let us indicate some important applications.

Sparse signal recovery. Given a sensing matrix $A \in \mathbb{R}^{m\times n}$ $(m << n)$ and a measurement vector y of a signal $x \in \mathbb{R}^n$, say $y = Ax \in \mathbb{R}^m$. The recovery sparse signal consists in finding the sparsest signal x being consistent with its measurement. Its mathematical formulation is

$$\min\left\{\|Ax - y\|^2 + \lambda\|x\|_0 : x \in \mathbb{R}^n\right\}.$$

Sparse inverse covariance (resp. covariance) estimation. the sparse inverse covariance (resp. covariance) estimation problem consists in finding a sparse inverse covariance (resp. sparse covariance) matrix x from a given sample covariance matrix S, they can be formulated as

$$\min_{x\in X}\{-\log\det x + \operatorname{tr}(Sx) + \lambda\|(x)\|_0\} \text{ (resp. } \min_{x\in X}\{\log\det x + \operatorname{tr}(Sx^{-1}) + \lambda\|(x)\|_0\})$$

where $X = \mathcal{S}^n_{++}$ is the set of positive definite matrices or $X = \{x \in \mathcal{S}^n_{++} : \underline{\gamma}I \preceq x \preceq \bar{\gamma}I\}$ ($A \preceq B$ means that $B - A$ is positive semidefinite).

Sparse multi-class logistic regression (or group variable selection in multi-class logistic regression). Logistic regression, introduced by D. Cox in 1958 [58], is undoubtedly one of the most popular supervised learning methods. Logistic regression has been successfully applied in various real-life problems such as cancer detection [119], medical [33, 16, 218], social science [120], etc.

Especially, logistic regression combined with feature selection has been proved to be suitable for high dimensional problems, for instance, document classification [80] and microarray classification [155, 119]. We will sow in Section 3.5 that the problem of group variable selection in multi-class logistic regression using the mixed-norm regularizer $\ell_{q,0}$ can be formulated in the form of (3.1).

t-distributed Stochastic Neighbor Embedding (t-SNE) : the t-distributed Stochastic Neighbor Embedding (t-SNE) is a machine learning algorithm which models each high dimensional object by a lower dimensional point (specifically two or three dimensions in case of visualization) in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. t-SNE was first introduced in [161] as a visualization technique for high dimensional data, and later, it has been applied in many applications such as bioinformatic, cancer research, feature visualization in neural networks, etc. We will show in Section 3.4 that t-SNE can be formulated as an optimization problem of the form (3.1).

Our contributions. Firstly, we propose a standard DCA scheme for solving (3.1). With the assumptions on f, g_i, and h_i, one can in most cases show that F_1 is a DC function and then DCA can be applied. However, it is not evident to highlight a DC composition of the composite functions $h_i(g_i(\cdot))$ to get a DC decomposition of F_1. To tackle this difficulty, we equivalently reformulate (3.1) as an DC program whose the DC decomposition of the objective function is explicit, and then investigate DCA for solving the equivalent problem. It turns out that in this DCA scheme, one has to compute a parameter ρ greater or equal to the L-Lipschitz constant of f. In practice, it is difficult (or even impossible) to determine the exact value of L, and one usually estimates L by a quite large number. However, a large value of L could lead to a bad convex approximation of F, then DCA may converge rapidly to a *biased* critical point.

To overcome this drawback and improve the standard DCA, we propose a new variant of DCA, named DCA-Like. This constitutes our second contribution. We prove that, the whole bounded sequence $\{x^k\}$ generated by DCA-Like still converges to a critical point of (3.1) under some conditions. Furthermore, we demonstrate that, under Kurdyka-Łojasiewicz assumption, the convergence rate of DCA-Like is at least sublinear $\mathcal{O}(1/k^\alpha)$ with $\alpha > 1$.

Thirdly, we incorporate a Nesterov's acceleration technique into DCA-Like to give rise to Accelerated DCA-Like (ADCA-Like). We prove that the accelerated version enjoys similar convergence properties and convergence rate as of DCA-Like.

TABLE 3.1 – Regularization functions and expression of corresponding g_i and h_i ($x^{(1)}, ..., x^{(J)}$ are the sub-vectors of $x \in \mathbb{R}^n$).

	$r(x)$	Reformulation of $r(x)$	$g_i(x)$	$h_i(x)$
Lasso	$\lambda\|x\|_1$	$r(x) := \sum_{i=1}^{n} h_i(g_i(x_i))$	$g_i(x_i) = \|x_i\|$	$h_i(t) = \lambda t$
Fused Lasso	$\lambda\|x\|_1 + \gamma \sum_{i=2}^{n} \|x_i - x_{i-1}\|$	$r(x) := \sum_{i=1}^{2n-1} h_i(g_i(x_i))$	$g_i(x_i) = \|x_i\|$ for $i = 1, ..., n$;	$h_i(t) = \lambda t$ for i=1,...,n;
		$+ \sum_{j=1}^{n-1} h_{j+n}(g_{j+n}(x_j))$	$g_{j+n}(x_j) = \|x_{j+1} - x_j\|$	$h_i(t) = \gamma t$ otherwise.
			for $j = 1, ..., n-1$.	
Group Lasso	$\lambda \sum_{j=1}^{J} \|x^{(j)}\|_2$	$r(x) := \sum_{j=1}^{J} h_j(g_j(x^{(j)}))$	$g_j(x^{(j)}) = \|x^{(j)}\|_2$	$h_j(t) = \lambda t$
Sparse	$\lambda\|x\|_1 + \gamma \sum_{j=1}^{J} \|x^{(j)}\|_2$	$r(x) := \sum_{i=1}^{n} h_i(g_i(x_i))$	$g_i(x_i) = \|x_i\|$ for $i = 1, ..., n$;	$h_i(t) = \lambda t$ for $i = 1, ..., n$;
group Lasso		$+ \sum_{j=1}^{J} h_{j+n}(g_{j+n}(x^{(j)}))$	$g_{j+n}(x^{(j)}) = \|x^{(j)}\|_2$	$h_i(t) = \gamma t$ otherwise.
			for $j = 1, ..., J$.	
$\ell_{\infty,1}$	$\lambda \sum_{j=1}^{J} \|x^{(j)}\|_\infty$	$r(x) := \sum_{j=1}^{J} h_j(g_j(x^{(j)}))$	$g_j(x^{(j)}) = \|x^{(j)}\|_\infty$	$h_j(t) = \lambda t$
Log ($\epsilon > 0$)	$\lambda \sum_{i=1}^{n} \log(\|x_i\| + \epsilon)$	$r(x) := \sum_{i=1}^{n} h(g_i(x_i))$	$g_i(x_i) = \|x_i\|$	$h(t) = \lambda \log(t + \epsilon)$
ℓ_p ($0 \neq p < 1$)	$\lambda \sum_{i=1}^{n} \text{sign}(p)(\|x_i\| + \epsilon)^p$, $\epsilon > 0$	$r(x) := \sum_{i=1}^{n} h(g_i(x_i))$	$g_i(x_i) = \|x_i\|$	$h(t) = \lambda \text{sign}(p)(t + \epsilon)^p$
capped-ℓ_1 ($\theta > 0$)	$\lambda \sum_{i=1}^{n} \min(1, \theta\|x_i\|)$	$r(x) := \sum_{i=1}^{n} h(g_i(x_i))$	$g_i(x_i) = \|x_i\|$	$h(t) = \lambda \min(1, \theta t)$
Exp ($\theta > 0$)	$\lambda \sum_{i=1}^{n} 1 - \exp(-\theta\|x_i\|)$	$r(x) := \sum_{i=1}^{n} h(g_i(x_i))$	$g_i(x_i) = \|x_i\|$	$h(t) = \lambda(1 - \exp(-\theta t))$
SCAD ($\theta > 1$)	$\lambda \sum_{i=1}^{n} \int_0^{\|x_i\|} \min\left(1, \frac{[\theta\lambda - y]_+}{(\theta-1)\lambda}\right) dy$	$r(x) := \sum_{i=1}^{n} h(g_i(x_i))$	$g_i(x_i) = \|x_i\|$	$h(t) = \int_0^t \min\left(1, \frac{[\theta\lambda - y]_+}{(\theta-1)\lambda}\right) dy$
MCP ($\theta > 0$)	$\lambda \sum_{i=1}^{n} \int_0^{\|x_i\|} \left[1 - \frac{y}{\theta\lambda}\right]_+ dy$	$r(x) := \sum_{i=1}^{n} h(g_i(x_i))$	$g_i(x_i) = \|x_i\|$	$h(t) = \lambda \int_0^t \left[1 - \frac{y}{\theta\lambda}\right]_+ dy$

3.2.1 Standard DCA for solving (3.1)

Since f is a differentiable function with L-Lipschitz continuous gradient, it is a DC function. On the other hand, the second term of F_1, e.g. $\sum_{i=1}^{m} h_i(g_i(\cdot))$, can be generally rewritten as a DC function. For instance, it has been proved that if $g_i : \mathbb{A} \to \mathbb{B}$ and $h_i : \mathbb{B} \to \mathbb{R}$ are DC (with $\mathbb{A} \subset \mathbb{R}^{n_i}$ being a convex set, $\mathbb{B}$ being a convex and open set) then $h_i(g_i(\cdot))$ is DC on $\mathbb{A}$ [15, 95].

Nevertheless, it is not easy to highlight a DC decomposition of $h_i(g_i(x_i))$. Hence, we first reformulate the problem (3.1) as :

$$\min\Big\{\varphi(x,z) := \chi_\Omega(x,z) + f(x) + \sum_{i=1}^{m} h_i(z_i) : (x,z) \in \mathbb{R}^n \times \mathbb{R}^m\Big\}, \tag{3.3}$$

where $\Omega := \{(x,z) : x \in X, g_i(x_i) \leq z_i, i = 1,...,m\}$. Let $g(x)$ be the function defined by $g(x) = (g_1(x_1),...,g_m(x_m))$. The problems (3.1) and (3.3) are equivalent in the following sense.

Proposition 3.1 *A point $x^* \in X$ is a global (resp. local) solution to the problem (3.1) if and only if $(x^*, g(x^*))$ is a global (resp. local) solution to the problem (3.3).*

Proof 3.1 *Let x^* be a local solution of the problem (3.1). Hence there exists $\epsilon > 0$ such that $F_1(x) \geq F_1(x^*)$, for all $x \in \mathcal{B}(x^*, \epsilon)$ where $\mathcal{B}(x^*, \epsilon)$ is the ball of center x^* and radius ϵ in $\mathbb{R}^n$. We have $\|x - x^*\| \leq \|(x, g(x)) - (x^*, g(x^*))\|$. This implies that if $(x, g(x)) \in \mathcal{B}((x^*, g(x^*)), \epsilon)$ then $x \in \mathcal{B}(x^*, \epsilon)$. Hence, for all $(x,z) \in \mathcal{B}((x^*, g(x^*)), \epsilon)$, we have $\varphi(x,z) \geq F_1(x) \geq F_1(x^*) = \varphi(x^*, g(x^*))$. It follows that $(x^*, g(x^*))$ is a local solution to (3.3).*

Inversely, let $(x^*, g(x^*))$ be a local solution to (3.3). There exists $\epsilon > 0$ such that $\varphi(x,z) \geq \varphi(x^*, g(x^*))$, for all $(x,z) \in \mathcal{B}((x^*, g(x^*)), \epsilon)$. Since g is convex, it is locally Lipschitz continuous around x^*, and by shrinking ϵ if necessary, we can find $L_* > 0$ such that $\|g(x) - g(x^*)\| \leq L_*\|x - x^*\|$ for all $x \in \mathcal{B}(x^*, \epsilon)$. Therefore $\|(x, g(x)) - (x^*, g(x^*))\| \leq \sqrt{L_*^2 + 1}\|x - x^*\|$. It follows that, for all $x \in \mathcal{B}(x^*, \epsilon/\sqrt{L_*^2 + 1})$, $F_1(x) = \varphi(x, g(x)) \geq \varphi(x^*, g(x^*)) = F_1(x^*)$. This implies that x^* is a local solution of (3.1).

Suppose that $x^* \in X$ is a global solution of (3.1). For any (x', z'), let $\epsilon > \|(x', z') - (x^*, g(x^*))\|$. Clearly, $F_1(x) \geq F_1(x^*)$, for all $x \in \mathcal{B}(x^*, \epsilon)$. As above, we have $\varphi(x,z) \geq \varphi(x^*, g(x^*))$ for all $(x,z) \in \mathcal{B}((x^*, g(x^*)), \epsilon)$. Thus, $\varphi(x', z') \geq \varphi(x^*, g(x^*))$. This is true for any (x', z'), hence $(x^*, g(x^*))$ is a global solution to (3.3).

Inversely, suppose that $(x^*, g(x^*))$ is a global solution to (3.3). For any x', let $\epsilon > \sqrt{L_*^2 + 1}\,\|x' - x^*\|$. Obviously, $\varphi(x,z) \geq \varphi(x^*, g(x^*))$ for all $(x,z) \in \mathcal{B}((x^*, g(x^*)), \epsilon)$. As above, for all $x \in \mathcal{B}(x^*, \epsilon/\sqrt{L_*^2 + 1})$, $F_1(x) \geq F_1(x^*)$. Thus, $F_1(x') \geq F_1(x^*)$. This is true for any x', so x^* is a global solution to (3.1). ■

Since the problems (3.1) and (3.3) are equivalent, in the remaining of the chapter, we consider the problem (3.3) instead of (3.1). The main advantage of (3.3) is that the objective function φ does not contain composite functions, and it is easy to highlight its DC decompositions.

Standard DCA for solving (3.3).

As h_i is concave, the function $\sum_{i=1}^{m} h_i$ is concave too. Then (3.3) can be reformulated as a DC program, for any $\rho \geq L$,

$$\min\{\varphi(x,z) = G_\rho(x,z) - H_\rho(x,z) : (x,z) \in \mathbb{R}^n \times \mathbb{R}^m\}, \tag{3.4}$$

with $G_\rho(x,z) := \frac{\rho}{2}\|x\|^2 + \chi_\Omega(x,z)$, $H_\rho(x,z) := \frac{\rho}{2}\|x\|^2 - f(x) - \sum_{i=1}^m h_i(z_i)$. Standard DCA applied to to (3.4) amounts, at each iteration k, to solving the convex sub-problem

$$\min\left\{\frac{\rho}{2}\|x\|^2 - \langle y^k, x\rangle + \sum_{i=1}^m(-\xi_i^k)z_i : (x,z) \in \Omega\right\}, \tag{3.5}$$

where $\xi_i^k \in \partial(-h_i)(z_i^k)$ and $y^k = \rho x^k - \nabla f(x^k)$. Furthermore, we prove that an optimal solution of (3.5) can be obtained by solving a strongly convex problem of only variable x.

Proposition 3.2 *If x^{k+1} is an optimal solution to the following strongly convex problem*

$$\min\{\frac{\rho}{2}\|x\|^2 - \langle y^k, x\rangle + \sum_{i=1}^m(-\xi_i^k)g_i(x_i) : x \in X\}, \tag{3.6}$$

then $(x^{k+1}, g(x^{k+1}))$ is an optimal solution to (3.5).

Proof 3.2 *Assume that x^{k+1} is the optimal solution to* (3.6). *We have*

$$\begin{aligned}&\tfrac{\rho}{2}\|x\|^2 - \langle y^k, x\rangle + \sum_{i=1}^m(-\xi_i^k)z_i \geq \tfrac{\rho}{2}\|x\|^2 - \langle y^k, x\rangle + \sum_{i=1}^m(-\xi_i^k)g_i(x_i)\\ &\geq \tfrac{\rho}{2}\|x^{k+1}\|^2 - \langle y^k, x^{k+1}\rangle + \sum_{i=1}^m(-\xi_i^k)g_i(x_i^{k+1}),\ \forall (x,z) \in \Omega,\end{aligned}$$

where the first inequality follows from $-\xi_i^k \geq 0$ and $z_i \geq g(x_i)$, and the second inequality comes from the fact that x^{k+1} is the optimal solution to (3.6). *This implies that $(x^{k+1}, g(x^{k+1}))$ is an optimal solution to* (3.5). ■

Finally, the standard DCA for solving (3.3) is described in Algorithm 3.1.

Algorithm 3.1 Standard DCA for solving (3.3)

1: **Initialization :** Choose an initial point x_0, $\rho \geq L$ and set $k \leftarrow 0$.
2: **repeat**
3: Compute $\xi_i^k \in \partial(-h_i)(g_i(x_i^k))$, $y^k = \rho x^k - \nabla f(x^k)$
4: Compute x^{k+1} by solving the strongly convex problem (3.6)
5: $k \leftarrow k+1$.
6: **until** Stopping criterion.

Remark 3.1 *Thanks to the use of the variable z in* (3.3) *and the DC decomposition* (3.4), *the above DCA scheme is not affected by the fact that the function g_i in the composite function $h_i(g_i(\cdot))$ is smooth or not.*

The convergence properties of Algorithm 3.1 are provided in Theorem 3.1. We recall that, by the definition of the critical point mentioned above, a point $(x^*, z^*) \in \mathbb{R}^n \times \mathbb{R}^m$ is called a critical point of the problem (3.3) if and only if

$$[(\nabla f(x^*), 0_m) + \mathcal{N}_\Omega(x^*, z^*)] \bigcap [0_n \times \partial(-h_1)(z_1^*) \times ... \times \partial(-h_m)(z_m^*)] \neq \emptyset,$$

where 0_d denotes the zero vector in $\mathbb{R}^d$, and $\mathcal{N}_\Omega(u^*)$ is the normal cone of Ω at u^* which is defined by $\mathcal{N}_\Omega(u^*) = \{v : \langle v, u - u^*\rangle \leq 0, \forall u \in \Omega\}$.

Theorem 3.1 *Let $\{x^k\}$ be the sequence generated by Algorithm 3.1. The following statements hold.*

(i) The sequence $\{\varphi(x^k, g(x^k))\}$ is decreasing.

(ii) If $\alpha = \inf \varphi(x, z) > -\infty$ then $\sum_{k=0}^{+\infty} \|x^{k+1} - x^k\|^2 < +\infty$, and therefore $\lim_{k\to+\infty} \|x^{k+1} - x^k\| = 0$.

(iii) If $\alpha = \inf \varphi(x, z) > -\infty$, then any limit point of $\{(x^k, g(x^k))\}$ is a critical point of (3.3).

These results are direct consequences of Theorem 3 about convergence properties of standard DCA in [189].

3.2.2 DCA-Like

In Algorithm 3.1, if ρ is very large, then the convex approximation of the DC objective function F_1 could be bad. Hence, we introduce DCA-Like, aiming to get better convex approximations than the one in DCA. The main idea of DCA-Like is to keep the parameter ρ as small as possible while finding a convex approximation of F_1. DCA-Like relaxes the two key requirements of standard DCA which are i) the second component H_ρ must be convex, and ii) the affine minorant of H_ρ^k must be a lower bound of H_ρ on the whole space (note that in DCA the condition i) and the way to define H_ρ^k imply the condition ii)). In fact, at each iteration k, we only need to find ρ_k such that

$$H_{\rho_k}(x^{k+1}, z^{k+1}) \geq H_{\rho_k}^k(x^{k+1}, z^{k+1}), \text{ with } (x^{k+1}, z^{k+1}) \in \arg\min \varphi_{\rho_k}^k(x, z). \tag{3.7}$$

The DCA-Like algorithm for solving (3.3) is described in Algorithm 3.2.

Algorithm 3.2 DCA-Like for solving (3.3)

Initialization : Choose an initial point x^0, a small enough positive parameter ρ_0, $\eta > 1$ and $0 < \delta < 1$. Set $k \leftarrow 0$.
repeat
  Compute $\xi_i^k \in \partial(-h_i)(g_i(x_i^k))$ and $\nabla f(x^k)$.
  Set $\rho_k = \max\{\rho_0, \delta\rho_{k-1}\}$ if $k > 0$.
  Compute x^{k+1} by solving (3.6) with $\rho = \rho_k$ and $y^k = \rho_k x^k - \nabla f(x^k)$.
  while $H_{\rho_k}(x^{k+1}, g(x^{k+1})) < H_{\rho_k}^k(x^{k+1}, g(x^{k+1}))$ **do**
    $\rho_k \leftarrow \eta\rho_k$.
    Update x^{k+1} by solving (3.6) with $\rho = \rho_k$ and $y^k = \rho_k x^k - \nabla f(x^k)$.
  end while
  $k \leftarrow k + 1$.
until Stopping criterion.

Remark 3.2 *a) It is easy to show that the while loop stops after finite steps. Indeed, it follows from the convexity of $-h_i$ that for $i = 1, ..., m$*

$$-h_i(g_i(x^{k+1})) \geq -h_i(g_i(x^k)) + \langle \xi_i^k, g_i(x^{k+1}) - g_i(x^k) \rangle. \tag{3.8}$$

Since f has L-Lipschitz gradient, for $\rho_k \geq L$ we have

$$f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{\rho_k}{2}\|x^{k+1} - x^k\|^2 \geq f(x^{k+1}). \tag{3.9}$$

Summing inequalities (3.8) *and* (3.9) *we see that the condition* (3.7) *holds if* $\rho_k \geq L$. *Therefore, there also exists* $\beta > 0$ *such that* $\rho_k \leq \beta L$ *for all* k.

b) The condition (3.7) *does not imply that* ρ_k *is large enough to ensure the convexity of* H_{ρ_k}. *However, we can prove that the convergence properties of DCA-Like are still guaranteed.*

c) In fact, the condition (3.7) *can be expressed as*

$$F_1(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{\rho_k}{2}\|x^{k+1} - x^k\|^2 - \langle \xi^k, g(x^{k+1}) - g(x^k) \rangle \geq F_1(x^{k+1}).$$

d) The hyper parameters δ *and* η *are used to suitably update* ρ_k. *From computational point of view, their values may affect the behavior of DCA-Like. A neutral and reasonable choice could be* $\eta = 2$, $\delta = 1/2$. *The choice of* ρ_0 *is also important. As we want to keep* ρ_k *as small as possible, we have an interest in choosing a small enough* ρ_0 *(for instance* 10^{-6} *in our experiments).*

We now study the convergence properties of DCA-Like in the following theorem.

Theorem 3.2 *Let* $\{x^k\}$ *be the sequence generated by DCA-Like (Algorithm 3.2). The following statements hold.*

(i) The sequence $\{\varphi(x^k, g(x^k))\}$ *is decreasing. Moreover, we have*

$$\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_k}{2}\|x^{k+1} - x^k\|^2.$$

(ii) If $\alpha = \inf \varphi(x, z) > -\infty$ *then* $\sum_{k=0}^{+\infty} \|x^{k+1} - x^k\|^2 < +\infty$, *and therefore* $\lim_{k\to+\infty} \|x^{k+1} - x^k\| = 0$.

(iii) If $\alpha = \inf \varphi(x, z) > -\infty$, *then any limit point of* $\{(x^k, g(x^k))\}$ *is a critical point of* (3.3).

Proof 3.3 *(i) From the* ρ_k*-convexity of* G_{ρ_k} *on* x *and* $(y^k, \xi^k) \in \partial G_{\rho_k}(x^{k+1}, g(x^{k+1}))$, *we have*

$$\begin{aligned} G_{\rho_k}(x^k, g(x^k)) \geq G_{\rho_k}(x^{k+1}, g(x^{k+1})) &+ \langle y^k, x^k - x^{k+1} \rangle \\ &+ \langle \xi^k, g(x^k) - g(x^{k+1}) \rangle + \frac{\rho_k}{2}\|x^k - x^{k+1}\|^2. \end{aligned}$$

This inequality and $H_{\rho_k}(x^{k+1}, g(x^{k+1})) \geq H^k_{\rho_k}(x^{k+1}, g(x^{k+1}))$ *imply that*

$$\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_k}{2}\|x^k - x^{k+1}\|^2. \tag{3.10}$$

(ii) For $\rho_0 \leq \rho_k$, *we obtain from* (3.10) *that*

$$\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_0}{2}\|x^k - x^{k+1}\|^2.$$

Summing the above inequality over $k = 0, ..., N$ *we have*

$$\varphi(x^0, g(x^0)) - \varphi(x^{N+1}, g(x^{N+1})) \geq \frac{\rho_0}{2}\sum_{k=0}^{N} \|x^k - x^{k+1}\|^2. \tag{3.11}$$

It follows from (3.11), $\rho_0 > 0$ *and* $\varphi(x^{N+1}, g(x^{N+1})) \geq \alpha$ *that*

$$\frac{2}{\rho_0}[\varphi(x^0, g(x^0)) - \alpha] \geq \sum_{k=0}^{N} \|x^k - x^{k+1}\|^2.$$

Finally, passing to the limit over the sequence $\{N\}_{N\in\mathbb{N}}$ *we get* $\sum_{k=0}^{+\infty}\|x^k - x^{k+1}\|^2 < +\infty$, *and therefore* $\lim_{k\to+\infty}\|x^{k+1} - x^k\| = 0$.
(iii) Let (x^*, z^*) *be a limit point of the sequence* $\{(x^k, g(x^k))\}$. *By the continuity of* g, *we can take a subsequence* $\{x^{k_j}\}$ *of* $\{x^k\}$ *that converges to* x^* *and* $z^* = g(x^*) = \lim_{j\to+\infty} g(x^{k_j})$. *It follows from (ii) that* $\lim_{j\to+\infty} x^{k_j+1} = x^*$. *In addition, without loss of generality, we suppose that the subsequence* $\{\xi^{k_j}\}$ *converges to* ξ^*. *By the continuity of* g_i *and closed property of the subdifferential mapping* $\partial(-h_i)$, *we have* $\xi_i^* \in \partial(-h_i)(g_i(x_i^*))$.
On another hand, since $(x^{k_j+1}, g(x^{k_j+1}))$ *is an optimal solution of the convex problem*

$$\min\left\{G_{\rho_{k_j}}(x,z) - \langle y^{k_j}, x\rangle - \langle \xi^{k_j}, z\rangle\right\}, \tag{3.12}$$

we have

$$\begin{array}{l}\frac{\rho_{k_j}}{2}\|x^{k_j+1}\|^2 + \chi_\Omega(x^{k_j+1}, g(x^{k_j+1}) - \langle \rho_{k_j}x^{k_j} - \nabla f(x^{k_j}), x^{k_j+1}\rangle - \langle \xi^{k_j}, g(x^{k_j+1})\rangle \\ \le \frac{\rho_{k_j}}{2}\|x^*\|^2 + \chi_\Omega(x^*, g(x^*)) - \langle \rho_{k_j}x^{k_j} - \nabla f(x^{k_j}), x^*\rangle - \langle \xi^{k_j}, g(x^*)\rangle.\end{array}$$

Therefore

$$\begin{array}{ll}\frac{\rho_{k_j}}{2}\|x^{k_j+1} - x^{k_j}\|^2 & +\chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) \le \frac{\rho_{k_j}}{2}\|x^* - x^{k_j}\|^2 + \chi_\Omega(x^*, g(x^*)) \\ & +\langle \nabla f(x^{k_j}), x^* - x^{k_j+1}\rangle - \langle \xi^{k_j}, g(x^*) - g(x^{k_j+1})\rangle.\end{array}$$

From Remark 3.2 (a), the sequence $\{\rho_{k_j}\}$ *is bounded, i.e.,* $\rho_0 \le \rho_k \le \beta L$. *Hence, taking* $j \to +\infty$ *we get*

$$\limsup_{j\to+\infty} \chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) \le \chi_\Omega(x^*, g(x^*)). \tag{3.13}$$

Combining (3.13) *with the lower semi-continuity of the function* χ *we get*

$$\limsup_{j\to+\infty} \chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) = \chi_\Omega(x^*, g(x^*)). \tag{3.14}$$

Similarly, it follows from (3.12) *and* $\rho_0 \le \rho_k \le \beta L$ *that*

$$\begin{array}{ll}\frac{\rho_0}{2}\|x^{k_j+1} - x^{k_j}\|^2 & +\chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) \le \frac{\beta L}{2}\|x - x^{k_j}\|^2 + \chi_\Omega(x, z) \\ & +\langle \nabla f(x^{k_j}), x - x^{k_j+1}\rangle - \langle \xi^{k_j}, z - g(x^{k_j+1})\rangle.\end{array} \tag{3.15}$$

Passing to the limit and combining with (3.14) *we get*

$$\chi_\Omega(x^*, g(x^*)) \le \frac{\beta L}{2}\|x - x^*\|^2 + \chi_\Omega(x, z) + \langle \nabla f(x^*), x - x^*\rangle - \langle \xi^*, z - g(x^*)\rangle.$$

This implies

$$0 \in (\nabla f(x^*), -\xi^*) + \mathcal{N}_\Omega(x^*, g(x^*)). \tag{3.16}$$

It follows from (3.16) *and* $\xi_i^* \in \partial(-h_i)(g_i(x_i^*))$ *that*

$$\begin{array}{rl}(0, \xi^*) \in & [(\nabla f(x^*), 0_m) + \mathcal{N}_\Omega(x^*, g(x^*))] \cap \\ & [0_n \times \partial(-h_1)(g_1(x_1^*)) \times ... \times \partial(-h_m)(g_m(x_m^*))].\end{array}$$

Therefore, $(x^*, g(x^*))$ *is a critical point of* (3.3). ■

Remark 3.3 *Since the basic convergence properties of DCA were proved by using the conjugate of convex functions, we cannot use these techniques in the case that the second component may be nonconvex. Unlike DCA, we justified the convergence properties of DCA-Like based on three key facts :*

i) There exists $\beta > 0$ *such that* $\rho_0 \leq \rho_k \leq \beta L$ *for all* k.

ii) $\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_k}{2}\|x^{k+1} - x^k\|^2$ *for all* k.

iii) If h_i *is differentiable with locally Lipschitz derivative, there exists a non-negative integer* N *and* $\pi > 0$ *such that*

$$dist(0, \partial^L \varphi(x^{k+1}, g(x^{k+1}))) \leq (\rho_k + L + \pi)\|x^{k+1} - x^k\| \ \forall k > N \tag{3.17}$$

Clearly, these key properties were obtained without the convexity of the second component.

In Theorem 3.3, we provide the sufficient conditions that guarantee the convergence of the whole sequence $\{x^k\}$ generated by DCA-Like. These conditions include the KL property of φ and the differentiability with locally Lipschitz derivative of h_i. Moreover, if the function ψ appearing in the KL inequality has the form $\psi(s) = cs^{1-\theta}$ with $\theta \in [0, 1)$ and $c > 0$, then we obtain the rates of convergence for both sequences $\{x^k\}$ and $\{\varphi(x^k, g(x^k))\}$.

Theorem 3.3 *Suppose that* $\inf \varphi(x, z) > -\infty$, *and* h_i *is differentiable with locally Lipschitz derivative. Assume further that* φ *has the KL property at any point* $(x, z) \in dom\ \partial^L \varphi$. *If* $\{x^k\}$ *generated by DCA-Like is bounded, then the whole sequence* $\{x^k\}$ *converges to* x^* *and* $(x^*, g(x^*))$ *is a critical point of* (3.3). *Moreover, if the function* ψ *appearing in the KL inequality has the form* $\psi(s) = cs^{1-\theta}$ *with* $\theta \in [0, 1)$ *and* $c > 0$, *then the following statements hold :*

i) If $\theta = 0$, *then the sequences* $\{x^k\}$ *and* $\{\varphi(x^k, g(x^k))\}$ *converge in a finite number of steps to* x^* *and* φ^*, *respectively.*

ii) If $\theta \in (0, 1/2]$, *then the sequences* $\{x^k\}$ *and* $\{\varphi(x^k, g(x^k))\}$ *converge linearly to* x^* *and* φ^*, *respectively.*

iii) If $\theta \in (1/2, 1)$, *then there exist positive constants* δ_1, δ_2, *and* N_0 *such that* $\|x^k - x^*\| \leq \delta_1 k^{-\frac{1-\theta}{2\theta-1}}$ *and* $\varphi(x^k, g(x^k)) - \varphi^* \leq \delta_2 k^{-\frac{1}{2\theta-1}}$ *for all* $k \geq N_0$.

To prove Theorem 3.3, we use the following Lemma 3.2 and Lemma 3.3. We also need the Lemma below given in [30].

Lemma 3.1 ([30]) *Let* Λ *be a compact set and let* $\sigma : \mathbb{R}^d \to (-\infty, +\infty]$ *be a proper and lower semicontinuous function. Assume that* σ *satisfies the KL property at each point of* Λ *on which* σ *is constant. Then, there exist* $\eta > 0, \epsilon > 0$ *and* $\psi \in \mathcal{M}_\eta$ *such that for all* $u \in \{u : dist(u, \Lambda) < \epsilon\} \cap \{u : \sigma(u^*) < \sigma(u) < \sigma(u^*) + \eta\}$, *one has*

$$\psi'(\sigma(u) - \sigma(u^*))dist(0, \partial^L \sigma(u)) \geq 1.$$

Lemma 3.2 *Let* $\{x^k\}$ *be the sequence generated by DCA-Like (Algorithm 3.2). If* $\inf \varphi(x, z) > -\infty$, *and the set of limit points* $\mathcal{C}$ *of* $\{x^k\}$ *is not empty, then* $\lim_{k\to+\infty} \varphi(x^k, g(x^k)) = \varphi(x^*, g(x^*))$ *for some* $x^* \in \mathcal{C}$. *Thus,* φ *has the same value on* $\Lambda = \{(x^*, g(x^*)) : x^* \in \mathcal{C}\}$.

Proof 3.4 (Proof of Lemma 3.2) *Since* $\inf \varphi(x, z) > -\infty$, *it follows from (i) of Theorem 3.2 that* $\{\varphi(x^k, g(x^k))\}$ *is non-increasing and bounded below. Thus, there exists* $\varphi^* = \lim_{k\to+\infty} \varphi(x^k, g(x^k))$. *Let* x^* *be a limit point of* $\{x^k\}$. *We therefore can find a subsequence*

$\{x^{k_j}\}$ *that converges to* x^*. *We have*

$$\begin{aligned}\limsup_{j\to+\infty}\varphi(x^{k_j},g(x^{k_j})) &= \limsup_{j\to+\infty}[\chi_\Omega(x^{k_j},g(x^{k_j}))+f(x^{k_j})-\sum_{i=1}^{m}(-h_i)(g_i(x_i^{k_j}))]\\ &\le \limsup_{j\to+\infty}\chi_\Omega(x^{k_j},g(x^{k_j}))+\limsup_{j\to+\infty}f(x^{k_j})-\liminf_{j\to+\infty}\sum_{i=1}^{m}(-h_i)(g_i(x_i^{k_j}))\\ &\le \chi_\Omega(x^*,g(x^*))+f(x^*)-\sum_{i=1}^{m}(-h_i)(g_i(x_i^*))=\varphi(x^*,g(x^*)),\end{aligned}$$

where the last inequality holds by (3.14) *and the lower semicontinuity of* $-h_i$. *From the above inequality and the lower semicontinuity of* φ, *we obtain* $\lim_{j\to+\infty}\varphi(x^{k_j},g(x^{k_j}))=\varphi(x^*,g(x^*))$. *Hence, by the uniqueness of the limit, we have* $\varphi^*=\varphi(x^*,g(x^*))$. ■

Lemma 3.3 *Let* $\{s_k\}$ *be a sequence in* $\mathbb{R}_+$ *and let* α,γ *be some non-negative constants. Assume that* $s_k\to 0$ *as* $k\to+\infty$ *and there exists* $N>0$ *such that*

$$s_{k+1}^\alpha \le \gamma(s_k-s_{k+1}), \forall k\ge N. \tag{3.18}$$

Then, the following statements hold.

i) If $\alpha=0$, *then* $\{s_k\}$ *converges in a finite number of steps to* 0.

ii) If $\alpha\in(0,1]$, *then* $\{s_k\}$ *converges linearly to* 0 *with rate* $\frac{\gamma}{1+\gamma}$.

iii) If $\alpha>1$, *then there exist positive constants* δ *and* N_0 *such that*

$$s_k\le \delta k^{-\frac{1}{\alpha-1}}, \forall k\ge N_0.$$

Proof 3.5 (Proof of Lemma 3.3) *The proof of Lemma 3.3 is similar to the one of Theorem 2 in [12].*

(i) If $\alpha=0$, *then we have* $s_k-s_{k+1}\ge\frac{1}{\gamma}$. *This implies (i).*

(ii) Assume that $\alpha\in(0,1]$. *Since* $\{s_k\}$ *converges to* 0, *there exists* N_1 *such that* $s_k<1$ *whenever* $k>N_1$. *Hence, we have* $s_{k+1}\le s_{k+1}^\alpha\le\gamma(s_k-s_{k+1})$. *Therefore* $s_{k+1}\le\frac{\gamma}{1+\gamma}s_k, \forall k>N_1$, *which implies (ii).*

(iii) The condition (3.18) *implies that the sequence* $\{s_k\}_{k\ge N}$ *is decreasing and non-negative. Thus, if there is* $k_0\ge N$ *such that* $s_{k_0}=0$ *then* $s_k=0$ *for any* $k\ge k_0$ *and iii) trivially holds. Now we assume that* $s_k>0$ *for all* $k\ge N$. *Due to convexity of the function* $s\in(0,\infty)\mapsto s^{1-\alpha}$ *and the condition* (3.18), *we have*

$$s_{k+1}^{1-\alpha}-s_k^{1-\alpha}\ge(1-\alpha)s_k^{-\alpha}(s_{k+1}-s_k)\ge\frac{s_{k+1}^\alpha}{s_k^\alpha}\frac{\alpha-1}{\gamma},\quad\forall k\ge N.$$

Therefore, if $s_k^\alpha\le 2s_{k+1}^\alpha$, *we have*

$$s_{k+1}^{1-\alpha}-s_k^{1-\alpha}\ge\frac{\alpha-1}{2\gamma},\quad\forall k\ge N.$$

Otherwise, if $s_k^\alpha>2s_{k+1}^\alpha$, *by taking both sides to the power of* $\frac{1-\alpha}{\alpha}<0$ *we obtain*

$$s_k^{1-\alpha}<2^{\frac{1-\alpha}{\alpha}}s_{k+1}^{1-\alpha}\quad\Longrightarrow\quad s_{k+1}^{1-\alpha}-s_k^{1-\alpha}>(1-2^{\frac{1-\alpha}{\alpha}})s_{k+1}^{1-\alpha}.$$

Since $1-2^{\frac{1-\alpha}{\alpha}}>0$ *and* $0<s_k\to 0$, *there exists* $\bar{\rho}>0$ *such that*

$$(1-2^{\frac{1-\alpha}{\alpha}})s_{k+1}^{1-\alpha}\geq\bar{\rho},\quad\forall k\geq N.$$

Letting $\rho=\min\{\frac{\alpha-1}{2\gamma},\bar{\rho}\}>0$, *we have*

$$s_{k+1}^{1-\alpha}-s_k^{1-\alpha}\geq\rho,\quad\forall k\geq N.$$

By summing these inequalities, we obtain that

$$s_k^{1-\alpha}=\sum_{i=N}^{k-1}(s_{i+1}^{1-\alpha}-s_i^{1-\alpha})+s_N^{1-\alpha}\geq(k-N)\rho,\quad\forall k\geq N.$$

Thus, with $N_0=2N$ *and* $\delta=(\frac{\rho}{2})^{\frac{1}{1-\alpha}}>0$, *we have*

$$s_k^{1-\alpha}\geq\frac{\rho}{2}k\quad or\quad s_k\leq\delta k^{\frac{1}{1-\alpha}},\quad\forall k\geq N_0.$$

■

Proof 3.6 (Proof of Theorem 3.3) *Denote by* $\mathcal{C}$ *the set of limit points of* $\{x^k\}$. *It follows from the continuity of* g *that the set of limit points* Λ *of* $\{(x^k,g(x^k))\}$ *takes the form* $\Lambda=\{(x,g(x)):x\in\mathcal{C}\}$. *By the Lemma 3.2,* φ *has the same value on* Λ, *which is denoted by* φ^*. *Thus,* $\lim_{k\to+\infty}\varphi(x^k,g(x^k))=\varphi^*$. *If there exists* $k\geq 1$ *such that* $\varphi(x^k,g(x^k))=\varphi^*$, *then* $\varphi(x^k,g(x^k))=\varphi(x^{k+p},g(x^{k+p}))$ *for any* $p\geq 0$ *due to the decreasing property of* $\{\varphi(x^k,g(x^k))\}$. *Hence,* $x^k=x^{k+p}$ *for all* $p\geq 0$ *and DCA-Like terminates after a finite number of steps. Hence, without loss of generality, we can assume that* $\varphi(x^k,g(x^k))>\varphi^*$ *for all* k.

Since $\{x^k\}$ *is bounded,* $\mathcal{C}$ *is a compact set. Define* $H(z)=\sum_{i=1}^m(-h_i)(z_i)$. *By the locally Lipschitz property of* ∇H *and* g, *for each* $x\in\mathcal{C}$, *there exist* L_x *and* ϵ_x *such that* $\|\nabla H(g(u))-\nabla H(g(v))\|\leq L_x\|u-v\|$ *for all* $u,v\in\mathcal{B}(x,\epsilon_x)$. *By the compactness of* $\mathcal{C}$, *there exist* $w_1,...,w_p\in\mathcal{C}$ *such that* $\mathcal{C}\subset\cup_{i=1}^p\mathcal{B}(w_i,\epsilon_{w_i}/4)$. *Set* $\pi=\max\{L_{w_i}:i=1,...,p\}$ *and* $\epsilon=\min\{\epsilon_{w_i}/2:i=1,...,p\}$. *Since* $\mathcal{C}$ *is the set of limit points of* $\{x^k\}$, *we have* $\lim_{k\to+\infty}dist(x^k,\mathcal{C})=0$. *From this and (ii) of Theorem 3.2, there exists* $N_1>0$ *such that* $x^{k-1}\in\cup_{i=1}^p\mathcal{B}(w_i,\epsilon_{w_i}/2)$ *and* $\|x^k-x^{k-1}\|<\epsilon$ *whenever* $k\geq N_1$. *Hence, there are some* w_i *such that* $x^k,x^{k-1}\in\mathcal{B}(w_i,\epsilon_{w_i})$. *Thus,*

$$\|\nabla H(g(x^k))-\nabla H(g(x^{k-1}))\|\leq L_{w_i}\|x^k-x^{k-1}\|\leq\pi\|x^k-x^{k-1}\|,\forall k\geq N_1.\tag{3.19}$$

Thank to the compactness of $\mathcal{C}$ *and the continuity of* g, Λ *is also a compact set. Applying Lemma 3.1 to the function* φ *and by shrinking* ϵ *if necessary, we can find* $\eta>0$ *and* $\psi\in\mathcal{M}_\eta$ *such that*

$$\psi'(\varphi(x,z)-\varphi^*)dist(0,\partial^L\varphi(x,z))\geq 1\tag{3.20}$$

$\forall(x,z)\in U:=\{(x,z):dist((x,z),\Lambda)<\epsilon\}\cap\{(x,z):\varphi^*<\varphi(x,z)<\varphi^*+\eta\}$. *From the definition of* $\Lambda:\lim_{k\to+\infty}dist((x^k,g(x^k)),\Lambda)=0$. *Thus, there exists* $N_2>0$ *such that* $dist((x^k,g(x^k)),\Lambda)<\epsilon$ *whenever* $k\geq N_2$. *By (i) of Theorem 3.2 and Lemma 3.2, there exists* $N_3>0$ *such that* $\varphi^*<\varphi(x^k,g(x^k))<\varphi^*+\eta$ *for all* $k\geq N_3$. *Taking* $N=\max\{N_1,N_2,N_3\}$ *we get* $(x^k,g(x^k))\in U$ *for all* $k\geq N$. *Hence, it follows from* (3.20) *that for any* $k\geq N$,

$$\psi'(\varphi(x^k,g(x^k))-\varphi^*)dist(0,\partial^L\varphi(x^k,g(x^k)))\geq 1.\tag{3.21}$$

By the differentiability of h_i and the definition of $\{x^k\}$, we have

$$(y^{k-1}, \xi^{k-1}) \in \partial G_{\rho_k}(x^k, g(x^k)) = \rho_k x^k + \partial \chi_\Omega(x^k, g(x^k)).$$

This implies that

$$(\rho_k(x^{k-1} - x^k) - \nabla f(x^{k-1}) + \nabla f(x^k), \xi^{k-1}) \in \partial \chi_\Omega(x^k, g(x^k)) + (\nabla f(x^k), 0).$$

Therefore

$$\begin{aligned}&(\rho_k(x^{k-1} - x^k) - \nabla f(x^{k-1}) + \nabla f(x^k), \xi^{k-1} - \xi^k)\\ &\in \partial \chi_\Omega(x^k, g(x^k)) + (\nabla f(x^k), 0) - (0, \nabla H(g(x^k)))\partial^L \varphi(x^k, g(x^k)).\end{aligned} \tag{3.22}$$

By the Lipschitz continuity of ∇f and (3.19)*, for all $k \geq N$, we have*

$$\begin{aligned}\|(\rho_k(x^{k-1} - x^k) &- \nabla f(x^{k-1}) + \nabla f(x^k), \xi^{k-1} - \xi^k)\|\\ &\leq (\rho_k + L + \pi)\|x^{k-1} - x^k\| \leq ((\beta + 1)L + \pi)\|x^{k-1} - x^k\|,\end{aligned}$$

where the last inequality follows from $\rho_k \leq \beta L$. Combining this with (3.22) *we obtain, for any $k \geq N$,*

$$dist(0, \partial^L \varphi(x^k, g(x^k))) \leq M\|x^{k-1} - x^k\|, \text{ where } M = (\beta + 1)L + \pi. \tag{3.23}$$

It follows from this, (3.21)*, and the concavity of ψ that*

$$\begin{aligned}\forall k \geq N:\ M\|x^{k-1} - x^k\|[\psi(\varphi(x^k, g(x^k)) - \varphi^*) - \psi(\varphi(x^{k+1}, g(x^{k+1})) - \varphi^*)]\\ \geq dist(0, \partial^L \varphi(x^k, g(x^k)))\psi'(\varphi(x^k, g(x^k)) - \varphi^*)(\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})))\\ \geq \varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_0}{2}\|x^k - x^{k+1}\|^2,\end{aligned}$$

where the last inequality comes from (i) of Theorem 3.2. This implies that

$$\begin{aligned}\forall k \geq N: \psi(\varphi(x^k, g(x^k)) - \varphi^*) &- \psi(\varphi(x^{k+1}, g(x^{k+1})) - \varphi^*)\\ &\geq \frac{\rho_0}{2M}\frac{\|x^k - x^{k+1}\|^2}{\|x^{k-1} - x^k\|} \geq \frac{\rho_0}{2M}[\|x^k - x^{k+1}\| - \frac{1}{4}\|x^{k-1} - x^k\|],\end{aligned}$$

where the last inequality follows from $a^2/b \geq a - b/4$ for all $a, b > 0$. Thus,

$$\begin{aligned}\forall k \geq N:\ \|x^k - x^{k+1}\| \leq &\frac{1}{4}\|x^{k-1} - x^k\| + \frac{2M}{\rho_0}[\psi(\varphi(x^k, g(x^k)) - \varphi^*)\\ &- \psi(\varphi(x^{k+1}, g(x^{k+1})) - \varphi^*)].\end{aligned}$$

Summing the above inequality from N to $N + k - 1$ we get

$$\begin{aligned}\sum_{j=1}^{k} \|x^{N+j} - x^{N+j-1}\| \leq\ &\tfrac{8M}{3\rho_0}[\psi(\varphi(x^N, g(x^N)) - \varphi^*) + \tfrac{1}{3}\|x^{N-1} - x^N\|\\ &-\psi(\varphi(x^{N+k}, g(x^{N+k})) - \varphi^*)].\end{aligned}$$

By the non-negativity of φ, we have

$$\sum_{j=1}^{k} \|x^{N+j} - x^{N+j-1}\| \leq \frac{1}{3}\|x^{N-1} - x^N\| + \frac{8M}{3\rho_0}\psi(\varphi(x^N, g(x^N)) - \varphi^*).$$

Taking the limit we obtain

$$\sum_{k=N}^{+\infty} \|x^{k+1} - x^k\| \leq \frac{1}{3}\|x^{N-1} - x^N\| + \frac{8M}{3\rho_0}\psi(\varphi(x^N, g(x^N)) - \varphi^*) < +\infty. \tag{3.24}$$

This implies that $\{x^k\}$ *is a Cauchy sequence, and hence it converges to a point* x^**. By Theorem 3.2,* $(x^*, g(x^*))$ *is a critical point of* (3.3)*. From* (3.21) *and* (3.23)*, for all* $k \geq N$*, we have*

$$\psi'(\varphi(x^k, g(x^k)) - \varphi^*)M\|x^{k-1} - x^k\| \geq 1. \tag{3.25}$$

By the assumption $\psi(t) = ct^{1-\theta}$*, (ii) of Theorem 3.2, and the above inequality, we obtain, for all* $k \geq N$*,*

$$\begin{aligned} \|\varphi(x^k, g(x^k)) - \varphi^*\|^{2\theta} &\leq [c(1-\theta)M]^2\|x^{k-1} - x^k\|^2 \\ &\leq \frac{2[c(1-\theta)M]^2}{\rho_0}(\varphi(x^{k-1}) - \varphi(x^k)). \end{aligned} \tag{3.26}$$

Set $r_k = \varphi(x^k, g(x^k)) - \varphi^*$*. It follows from* (3.26) *that*

$$r_k^{2\theta} \leq \frac{2[c(1-\theta)M]^2}{\rho_0}(r_{k-1} - r_k), \forall k \geq N. \tag{3.27}$$

Therefore, taking $\alpha = 2\theta$*,* $\gamma = \frac{2[c(1-\theta)M]^2}{\rho_0}$ *and applying Lemma 3.3 on the sequence* $\{r_k\}$*, we get the convergence rate of* $\{\varphi(x^k, g(x^k))\} \to \varphi^*$ *related to different values of* θ *as stated in this theorem.*

Consider now the convergence rate of $\{x^k\}$*. By setting* $s_i = \sum_{k=i}^{+\infty} \|x^{k+1} - x^k\|$*, it follows from* (3.24) *that* s_i *is finite. Since* $\|x^i - x^*\| \leq s_i$*, the rate of convergence of* $\{x^i\}$ *to* x^* *can be deduced from the rate of convergence of* $\{s_i\}$ *to 0. By* (3.24) *and* $\psi(t) = ct^{1-\theta}$ *we get*

$$(\varphi(x^i, g(x^i)) - \varphi^*)^{1-\theta} \geq \frac{\rho_0}{8Mc}[3s_i - \|x^{i-1} - x^i\|], \ \forall i \geq N. \tag{3.28}$$

Combining $\psi(t) = ct^{1-\theta}$ *and* (3.25)*, we have*

$$c(1-\theta)M\|x^{i-1} - x^i\| \geq (\varphi(x^i, g(x^i)) - \varphi^*)^\theta, \forall i \geq N. \tag{3.29}$$

This implies that if $\theta = 0$*, then* $\|x^{i-1} - x^i\| \geq \frac{1}{cM}$ *for all* $i \geq N$*. Hence, by (i) of Theorem 3.2, we have*

$$\varphi(x^{i-1}, g(x^{i-1})) - \varphi(x^i, g(x^i)) \geq \frac{\rho_k}{2}\|x^{i-1} - x^i\|^2 \geq \frac{\rho_0}{2[cM]^2}.$$

Hence, DCA-Like must terminate after a finite number of iterations. If $\theta \in (0, 1)$*, according to* (3.29) *and* (3.28) *we have, for all* $i \geq N$*,*

$$(c(1-\theta)M\|x^{i-1} - x^i\|)^{\frac{1-\theta}{\theta}} \geq \frac{\rho_0}{8Mc}[3s_i - \|x^{i-1} - x^i\|]. \tag{3.30}$$

Since $\|x^{i-1} - x^i\| \to 0$ *as* $i \to +\infty$*, by increasing* N *if necessary, we have* $\|x^{i-1} - x^i\| < 1$ *for all* $i \geq N$*. Let* $\phi(\theta) = \min(1, \frac{1-\theta}{\theta})$*. Hence, it follows from* (3.30) *that*

$$\begin{aligned} \frac{3\rho_0}{8Mc}s_i &\leq [(c(1-\theta)M)^{\frac{1-\theta}{\theta}} + \frac{\rho_0}{8Mc}]\|x^{i-1} - x^i\|^{\phi(\theta)} \\ &= [(c(1-\theta)M)^{\frac{1-\theta}{\theta}} + \frac{\rho_0}{8Mc}](s_{i-1} - s_i)^{\phi(\theta)}. \end{aligned} \tag{3.31}$$

This implies that $s_i^{\frac{1}{\phi(\theta)}} \leq (\frac{8Mc}{3\rho_0}(c(1-\theta)M)^{\frac{1-\theta}{\theta}} + \frac{1}{3})^{\frac{1}{\phi(\theta)}}(s_{i-1} - s_i), \forall i \geq N$*. Taking* $\alpha = 1/\phi(\theta)$*,* $\gamma = (\frac{8Mc}{3\rho_0}(c(1-\theta)M)^{\frac{1-\theta}{\theta}} + \frac{1}{3})^{1/\phi(\theta)}$ *and applying (ii) and (iii) of Lemma 3.3 on the sequence* $\{s_i\}$*, we get the convergence rate of* $\{s_i\} \to 0$*, which is the convergence rate of* $\{x^k\} \to x^*$*, related to different values of* θ *as stated in (ii) and (iii) of this theorem.* ■

3.2.3 Accelerated DCA-Like

Following the same idea of Accelerated DCA, we introduce an accelerated version of DCA-Like (ADCA-Like), which is described in Algorithm 3.3.

Algorithm 3.3 ADCA-Like for solving (3.3)

1: **Initialization :** Choose an initial point x_0, $w^0 = x^0$, $t_0 = (1+\sqrt{5})/2$, a small enough positive parameter ρ_0, $\eta > 1$ and $0 < \delta < 1$ and $k \leftarrow 0$.
2: **repeat**
3: **If** $\varphi(w^k, g(w^k)) \leq \varphi(x^k, g(x^k))$ **then** set $v^k = w^k$;
else set $v^k = x^k$.
4: Compute $\xi_i^k \in \partial(-h_i)(z_i^k)$ with $z_i^k = g_i(v_i^k)$ and $\nabla f(v^k)$.
5: Set $\rho_k = \max\{\rho_0, \delta\rho_{k-1}\}$ if $k > 0$
6: Compute x^{k+1} by solving (3.6) with $\rho = \rho_k$ and $y^k = \rho_k v^k - \nabla f(v^k)$.
7: **while** $H_{\rho_k}(x^{k+1}, g(x^{k+1})) < H_{\rho_k}^{(v^k, g(v^k))}(x^{k+1}, g(x^{k+1}))$ **do**
8: $\rho_k \leftarrow \eta\rho_k$.
9: Update x^{k+1} by solving (3.6) with $\rho = \rho_k$ and $y^k = \rho_k v^k - \nabla f(v^k)$.
10: **end while**
11: Compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$ and $w^{k+1} = x^{k+1} + \frac{t_k - 1}{t_{k+1}}\left(x^{k+1} - x^k\right)$.
12: $k \leftarrow k + 1$.
13: **until** Stopping criterion.

The convergence properties of ADCA-Like are provided in the following theorems. In Theorem 3.4, we provide the convergence properties of ADCA-Like.

Theorem 3.4 *Let $\{x^k\}$ be the sequence generated by Algorithm 3.3. The following statements hold.*

(i) The sequence $\{\varphi(x^k, g(x^k))\}$ is decreasing. More precisely, we have

$$\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_k}{2}\|x^{k+1} - v^k\|^2.$$

(ii) If $\alpha = \inf \varphi(x, z) > -\infty$ then $\sum_{k=0}^{+\infty} \|x^{k+1} - v^k\|^2 < +\infty$ and therefore $\lim_{k\to+\infty} \|x^{k+1} - v^k\| = 0$.

(iii) If $\alpha = \inf \varphi(x, z) > -\infty$, then any limit point of $\{(x^k, g(x^k))\}$ is a critical point of (3.3).

Proof 3.7 *Similar to DCA-Like, the proof of convergence properties of ADCA-Like are mainly based on the key results analogue to (ii) and (iii) of Remark 3.3, with the use of the intermediate variable v^k.*

(i) By the definition of x^{k+1}, we have $(y^k, \xi^k) \in \partial G_{\rho_k}(x^{k+1}, g(x^{k+1}))$. Hence, it follows from the ρ_k-convexity of G_{ρ_k} on x that

$$\begin{aligned} G_{\rho_k}(v^k, g(v^k)) \geq G_{\rho_k}(x^{k+1}, g(x^{k+1})) &+ \langle y^k, v^k - x^{k+1}\rangle \\ &+ \langle \xi^k, g(v^k) - g(x^{k+1})\rangle + \frac{\rho_k}{2}\|v^k - x^{k+1}\|^2. \end{aligned}$$

Combining the above inequality with $H_{\rho_k}(x^{k+1}, g(x^{k+1})) \geq H_{\rho_k}(v^k, g(v^k)) + \langle y^k, x^{k+1} - v^k\rangle + \langle \xi^k, g(x^{k+1}) - g(v^k)\rangle$ we get

$$\varphi(v^k, g(v^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_k}{2}\|v^k - x^{k+1}\|^2.$$

From this and $\varphi(v^k, g(v^k)) \leq \varphi(x^k, g(x^k))$*, we obtain*

$$\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_k}{2}\|v^k - x^{k+1}\|^2. \tag{3.32}$$

(ii) From (3.32) *and* $\rho_k \geq \rho_0$*, we have*

$$\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})) \geq \frac{\rho_0}{2}\|v^k - x^{k+1}\|^2.$$

Summing the above inequality over $k = 0, ..., N$ *we get*

$$\varphi(x^0, g(x^0)) - \varphi(x^{N+1}, g(x^{N+1})) \geq \frac{\rho_0}{2}\sum_{k=0}^{N}\|v^k - x^{k+1}\|^2. \tag{3.33}$$

It follows from (3.33)*,* $\rho_0 > 0$ *and* $\varphi(x^{N+1}, g(x^{N+1})) \geq \alpha$ *that*

$$\frac{2}{\rho_0}[\varphi(x^0, g(x^0)) - \alpha] \geq \sum_{k=0}^{N}\|v^k - x^{k+1}\|^2.$$

Passing to the limit over the sequence $\{N\}_{N\in\mathbb{N}}$*, we obtain* $\sum_{k=0}^{+\infty}\|v^k - x^{k+1}\|^2 < +\infty$*, and therefore* $\lim_{k\to+\infty}\|x^{k+1} - v^k\| = 0$*.*
(iii) Let (x^*, z^*) *be a limit point of the sequence* $\{(x^k, g(x^k))\}$*. By the continuity of* g*, we can take a subsequence* $\{x^{k_j+1}\}$ *of* $\{x^k\}$ *that converges to* x^* *and* $z^* = g(x^*) = \lim_{j\to+\infty} g(x^{k_j+1})$*. It follows from (ii) that* $\lim_{j\to+\infty} v^{k_j} = x^*$*. In addition, without loss of generality, we can suppose that the subsequence* $\{\xi^{k_j}\}$ *converges to* ξ^**. By the continuity of* g_i *and the property of the subdifferential mapping* $\partial(-h)$*, we have* $\xi_i^* \in \partial(-h)(g_i(x_i^*))$*. We note that* $(x^{k_j+1}, g(x^{k_j+1}))$ *is a solution of the following convex problem*

$$\min\left\{G_{\rho_{k_j}}(x, z) - \langle y^{k_j}, x\rangle - \langle \xi^{k_j}, z\rangle\right\}. \tag{3.34}$$

This implies that

$$\begin{array}{l}\frac{\rho_{k_j}}{2}\|x^{k_j+1}\|^2 + \chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) - \langle \rho_{k_j}v^{k_j} - \nabla f(v^{k_j}), x^{k_j+1}\rangle - \langle \xi^{k_j}, g(x^{k_j+1})\rangle \\ \leq \frac{\rho_{k_j}}{2}\|x^*\|^2 + \chi_\Omega(x^*, g(x^*)) - \langle \rho_{k_j}v^{k_j} - \nabla f(v^{k_j}), x^*\rangle - \langle \xi^{k_j}, g(x^*)\rangle.\end{array}$$

Therefore

$$\begin{array}{ll}\frac{\rho_{k_j}}{2}\|x^{k_j+1} - v^{k_j}\|^2 & +\chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) \leq \frac{\rho_{k_j}}{2}\|x^* - v^{k_j}\|^2 + \chi_\Omega(x^*, g(x^*)) \\ & +\langle \nabla f(v^{k_j}), x^* - x^{k_j+1}\rangle - \langle \xi^{k_j}, g(x^*) - g(x^{k_j+1})\rangle.\end{array}$$

Since there exists $\beta > 0$ *such that* $\rho_0 \leq \rho_k \leq \beta L$*, we have*

$$\begin{array}{ll}\frac{\rho_0}{2}\|x^{k_j+1} - v^{k_j}\|^2 & +\chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) \leq \frac{\beta L}{2}\|x^* - v^{k_j}\|^2 + \chi_\Omega(x^*, g(x^*)) \\ & +\langle \nabla f(v^{k_j}), x^* - x^{k_j+1}\rangle - \langle \xi^{k_j}, g(x^*) - g(x^{k_j+1})\rangle.\end{array}$$

As $\lim_{j\to+\infty} x^{k_j+1} = \lim_{j\to+\infty} v^{k_j} = x^*$*, taking* $j \to +\infty$ *we obtain*

$$\limsup_{j\to+\infty} \chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) \leq \chi_\Omega(x^*, g(x^*)). \tag{3.35}$$

Combining (3.35) *with the lower semi-continuity of the function* χ*, we get*

$$\limsup_{j\to+\infty} \chi_\Omega(x^{k_j+1}, g(x^{k_j+1})) = \chi_\Omega(x^*, g(x^*)). \tag{3.36}$$

Similarly, it follows from (3.34) *that*

$$\begin{array}{ll} \frac{\rho_0}{2}\|x^{k_j+1}-v^{k_j}\|^2 & +\chi_\Omega(x^{k_j+1},g(x^{k_j+1}))\leq \frac{\beta L}{2}\|x-v^{k_j}\|^2+\chi_\Omega(x,z)\\ & +\langle\nabla f(v^{k_j}),x-x^{k_j+1}\rangle-\langle\xi^{k_j},z-g(x^{k_j+1})\rangle. \end{array} \tag{3.37}$$

Passing to the limit and combining with (3.36) *we obtain*

$$\chi_\Omega(x^*,g(x^*))\leq\frac{\beta L}{2}\|x-x^*\|^2+\chi_\Omega(x,z)+\langle\nabla f(x^*),x-x^*\rangle-\langle\xi^*,z-g(x^*)\rangle.$$

This implies that $0\in(\nabla f(x^*),-\xi^*)+\mathcal{N}_\Omega(x^*,g(x^*))$. *Consequently*

$$\begin{array}{ll}(0,\xi^*) & \in[(\nabla f(x^*),0)+\mathcal{N}_\Omega(x^*,g(x^*))]\\ & \cap[0\times\partial(-h_i)(g_1(x_1^*))\times\ldots\times\partial(-h_i)(g_m(x_m^*))].\end{array}$$

Therefore, $(x^*,g(x^*))$ *is a critical point of* (3.3). ■

The sufficient descent property (i) of Theorem 3.4 is different from Theorem 3.2 due to the intermediate variable v^k. Hence, neither the convergence of the whole sequence $\{x^k\}$ nor convergence rate for $\{\|x^k-x^*\|\}$ can be achieved. However, we can still obtain some interesting results for the sequence $\{\varphi(x^k,g(x^k))\}$ under the KL assumption. These properties are presented in Theorem 3.5.

Theorem 3.5 *Suppose that* $\inf\varphi(x,z)>-\infty$, *and* h_i *is differentiable with locally Lipschitz derivative. Assume further that* φ *has the KL property at any point* $(x,z)\in\text{dom}\,\partial^L\varphi$ *with* $\psi(s)=cs^{1-\theta}$ *for some* $\theta\in[0,1)$ *and* $c>0$. *If* $\{x^k\}$ *generated by accelerated DCA-Like is bounded, then the following statements hold.*

(i) If $\theta=0$, *then* $\{\varphi(x^k,g(x^k))\}$ *converges in a finite number of steps to* φ^*.

(ii) If $\theta\in(0,1/2]$, *then* $\{\varphi(x^k,g(x^k))\}$ *converges linearly to* φ^*.

(iii) If $\theta\in(1/2,1)$, *then there exist positive constants* δ *and* N_0 *such that* $\varphi(x^k,g(x^k))-\varphi^*\leq\delta k^{-\frac{1}{2\theta-1}}$ *for all* $k\geq N_0$.

To prove Theorem 3.5, we use the following Lemma 3.4 whose the proof is similar to the proof of Lemma 3.2.

Lemma 3.4 *Let* $\{x^k\}$ *be the sequence generated by ADCA-Like (Algorithm 3.3). If* $\inf\varphi(x,z)>-\infty$, *and the set of limit points* $\mathcal{C}$ *of* $\{x^k\}$ *is not empty, then* $\lim_{k\to+\infty}\varphi(x^k,g(x^k))=\varphi(x^*,g(x^*))$ *for some* $x^*\in\mathcal{C}$. *Thus,* φ *has the same value on* $\Lambda=\{(x^*,g(x^*)):x^*\in\mathcal{C}\}$.

Proof 3.8 (Proof of Theorem 3.5) *(i) By Theorem 3.4 (ii), the sequences* $\{x^k\}$ *and* $\{v^k\}$ *share the same set of limit points* $\mathcal{C}$. *Since* $\{x^k\}$ *is bounded,* $\mathcal{C}$ *is compact. As* $H(z)=\sum_{i=1}^m(-h_i)(z_i)$ *is differentiable with locally Lipschitz derivative, by the continuity of* g *and by (ii) of Theorem 3.4 we see that there exist positive constants* N_1 *and* π *such that*

$$\|\nabla H(g(x^{k+1}))-\nabla H(g(v^k))\|\leq\pi\|x^{k+1}-v^k\|,\forall k\geq N_1. \tag{3.38}$$

Since g *is continuous, the set of limit points of* $\{(x^k,g(x^k))\}$ *is given by* $\Lambda=\{(x,g(x)):x\in\mathcal{C}\}$. *Thus,* Λ *is also a compact set. By Lemma 3.4,* φ *has the same value* φ^* *on* Λ. *According to Lemma 3.1, there exist* $\epsilon>0$, $\eta>0$ *and* $\psi\in\mathcal{M}_\eta$ *such that for all* $\forall(x,z)\in U:=\{(x,z):dist((x,z),\Lambda)<\epsilon\}\cap\{(x,z):\varphi^*<\varphi(x,z)<\varphi^*+\eta\}$, *we have*

$$\psi'(\varphi(x,z)-\varphi^*)dist(0,\partial^L\varphi(x,z))\geq 1. \tag{3.39}$$

Similar to the proof of Theorem 3.3, without loss of generality, we can assume that $\varphi(x^k, g(x^k)) > \varphi^*$ *for all* k. *By Lemma 3.4, there exists* N_2 *such that* $\varphi^* < \varphi(x^k, g(x^k)) < \varphi^* + \eta$ *whenever* $k \geq N_2$. *On the other hand, it follows from* $\lim_{k\to+\infty} dist((x^k, g(x^k)), \Lambda) = 0$ *that there is* N_3 *such that* $dist((x^k, g(x^k)), \Lambda) < \epsilon$ *for all* $k \geq N_3$. *Let* $N = \max\{N_1, N_2, N_3\}$, *we have* $(x^k, g(x^k)) \in U$. *It follows this and* (3.39) *that for all* $k \geq N$,

$$\psi'(x^k, g(x^k)) - \varphi^*) dist(0, \partial^L \varphi(x^k, g(x^k))) \geq 1. \tag{3.40}$$

Since H *is differentiable, by the definition of* x^{k+1} *we have*

$$(y^k, \nabla H(g(v^k)) \in \partial G_{\rho_k}(x^{k+1}, g(x^{k+1})) = \rho_k x^{k+1} + \partial \chi_\Omega(x^{k+1}, g(x^{k+1})).$$

This implies that

$$\begin{aligned}(\rho_k(v^k - x^{k+1}) - \nabla f(v^k) + \nabla f(x^{k+1}), \nabla H(g(v^k)))\\ \in \partial \chi_\Omega(x^{k+1}, g(x^{k+1})) + (\nabla f(x^{k+1}), 0).\end{aligned}$$

Therefore

$$\begin{aligned}&(\rho_k(v^k - x^{k+1}) - \nabla f(v^k) + \nabla f(x^{k+1}), \nabla H(g(v^k)) - \nabla H(g(x^{k+1})))\\ \in&\partial \chi_\Omega(x^{k+1}, g(x^{k+1})) + (\nabla f(x^{k+1}), 0) - (0, \nabla H(g(x^{k+1})))\\ =&\partial^L \varphi(x^{k+1}, g(x^{k+1})).\end{aligned} \tag{3.41}$$

By the Lipschitz continuity of ∇f, $\rho_k \leq \beta L$ *and* (3.38) *for any* $k \geq N$, *we have* $\|(\rho_k(v^k - x^{k+1}) - \nabla f(v^k) + \nabla f(x^{k+1}), \nabla H(g(v^k)) - \nabla H(g(x^{k+1})))\| \leq M\|x^{k+1} - v^k\|$, *where* $M = \beta L + L + \pi$. *Combining this and* (3.41) *we get* $dist(0, \partial^L \varphi(x^{k+1}, g(x^{k+1}))) \leq M\|x^{k+1} - v^k\|$. *Denote* $r_{k+1} = \varphi(x^{k+1}, g(x^{k+1})) - \varphi^* > 0$. *It follows from the above inequality and* (3.40) *that, for all* $k \geq N$,

$$\begin{aligned}1 &\leq [\psi'(\varphi(x^{k+1}, g(x^{k+1})) - \varphi^*) M\|x^{k+1} - v^k\|]^2\\ &\leq [\psi'(r_{k+1})]^2 M^2 \frac{2(\varphi(x^k, g(x^k)) - \varphi(x^{k+1}, g(x^{k+1})))}{u_k}\\ &\leq [\psi'(r_{k+1})]^2 M^2 \frac{2(r_k - r_{k+1})}{u_0},\end{aligned} \tag{3.42}$$

where the second inequality follows from (i) of Theorem 3.4, and the last inequality holds by $\rho_k \geq \rho_0$. *Since* ψ *takes the form* $\psi(s) = cs^{1-\theta}$, *we get* $\psi'(s) = c(1-\theta)s^{-\theta}$. *Therefore, the inequality* (3.42) *becomes*

$$r_{k+1}^{2\theta} \leq \frac{2c(1-\theta)M^2}{\rho_0}(r_k - r_{k+1}),\ \forall k \geq N. \tag{3.43}$$

Finally, applying Lemma 3.3 with $s_k = r_k$, $\alpha = 2\theta$ *and* $\gamma = \frac{2c(1-\theta)M^2}{\rho_0}$ *gives us that the sequence* $\{r_k\}$ *converges to* 0 *with the rates corresponding to different values of* θ *as started in this theorem.* ■

3.3 Minimizing the sum of a nonconvex differentiable function with L-Lipschitz continuous gradient and a DC function

We now consider le sum of a nonconvex differentiable function with L-Lipschitz continuous gradient and a DC function minimization problem (3.2). Recall that, in Chapter 2, we have

developed standard DCA and Accelerated DCA to solve this problem. The readers are referred to Section 2.1 for more details on the description and existing works of (3.2). The standard DCA and Accelerated DCA for solving (3.2) are presented in Chapter 2 Section 2.3.

Similarly to the problem (3.1), the DC decomposition of (3.2) requires a parameter ρ which is greater ou equal to the L-Lipschitz constant of f. Hence, in order to avoid a bad convex approximation of the objective function by using a large value of ρ, we will develop DCA-Like for (3.2). DCA-Like and ADCA-Like for (3.2) as well as their convergence properties and convergence rate are presented in the following sub-sections.

3.3.1 DCA-Like

In order to keep the parameter ρ as small as possible, at each iteration k of DCA-Like, we only need to find ρ_k such that $H_{\rho_k}(x, x^k) := H_\rho(x^k) + \langle \rho x^k - \nabla f(x^k) + \xi^k, x - x^k \rangle$ is a lower bound of $H_{\rho_k}(x)$ at x^{k+1} but not on the whole space, i.e.,

$$H_{\rho_k}(x^{k+1}) \geq H_{\rho_k}(x^{k+1}, x^k). \tag{3.44}$$

The DCA-Like algorithm for solving (3.2) is described in Algorithm 3.4.

Algorithm 3.4 DCA-Like for solving (3.2)

1: **Initialization :** Choose an initial point x^0, a small enough positive parameter ρ_0 $\eta > 1, 0 < \delta < 1$, and $k \leftarrow 0$.
2: **repeat**
3: Compute $\xi^k \in \partial h(x^k)$ and $\nabla f(x^k)$.
4: Set $\rho_k = \max\{\rho_0, \delta\rho_{k-1}\}$ if $k > 0$.
5: Compute x^{k+1} by solving the convex program

$$\min_{x \in \mathbb{R}^n} \left\{ \frac{\rho_k}{2}\|x\|^2 + g(x) - \langle \rho_k x^k - \nabla f(x^k) + \xi^k, x \rangle \right\}. \tag{3.45}$$

6: **while** $H_{\rho_k}(x^{k+1}) < H_{\rho_k}(x^{k+1}, x^k)$ **do**
7: $\rho_k \leftarrow \eta\rho_k$.
8: Compute x^{k+1} by solving (3.45).
9: **end while**
10: $k \leftarrow k + 1$.
11: **until** Stopping criterion.

Remark 3.4 *a) It is easy to show that the while loop stops after finite steps. Indeed, it follows from the convexity of h that*

$$h(x^{k+1}) \geq h(x^k) + \langle \xi^k, x^{k+1} - x^k \rangle. \tag{3.46}$$

Since f is L - Lipschitz gradient, when $\rho_k \geq L$, we have

$$f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{\rho_k}{2}\|x^{k+1} - x^k\|^2 \geq f(x^{k+1}). \tag{3.47}$$

Summing the inequalities (3.46) *and* (3.47) *we obtain that the condition* (3.44) *holds when $\rho_k \geq L$. Consequently, there exists $\beta > 0$ such that $\rho_0 \leq \rho_k \leq \beta L$ for all k.*

b) The condition (3.44) *does not imply that* ρ_k *is large enough to ensure the convexity of* H_{ρ_k}. *However, we can prove that the convergence properties of DCA-Like are still guaranteed.*

c) The hyper parameters δ *and* η *are used to suitably update* ρ_k. *From computational point of view, their values may affect the behavior of DCA-Like. A neutral and reasonable choice could be* $\eta = 2$, $\delta = 1/2$. *The choice of* ρ_0 *is also important. As we want to keep* ρ_k *as small as possible, we have an interest in choosing a small enough* ρ_0 *(for instance* 10^{-6} *in our experiments).*

We now study the convergence properties and convergence rate of DCA-Like. In Theorem 3.6, we give the properties of the limit points of the sequence $\{x^k\}$ generated by DCA-Like. We recall that a point x^* is a critical point of (3.2) if

$$[\nabla f(x^*) + \partial g(x^*)] \cap \partial h(x^*) \neq \emptyset.$$

The modulus of strong convexity of σ on Ω, denoted by $\mu(\sigma, \Omega)$ or $\mu(\sigma)$ if $\Omega = \mathbb{R}^n$, is given by

$$\mu(\sigma, \Omega) = \sup\{\rho \geq 0 : \sigma - (\rho/2)\|.\|^2 \text{ is convex on } \Omega\}.$$

One says that σ is *strongly convex* on Ω if $\mu(\sigma, \Omega) > 0$.

Theorem 3.6 *Let* $\{x^k\}$ *be the sequence generated by Algorithm 3.4. The following statements hold.*

i) The sequence $\{F_2(x^k)\}$ *is decreasing. More precisely, we have*

$$F_2(x^k) - F_2(x^{k+1}) \geq \frac{\rho_k + \mu(g)}{2}\|x^{k+1} - x^k\|^2.$$

ii) If $\alpha = \inf_{x\in\mathbb{R}^n} F_2(x) > -\infty$, *then* $\sum_{k=1}^{+\infty} \|x^{k+1} - x^k\|^2 < +\infty$ *and therefore* $\lim_{k\to+\infty} \|x^{k+1} - x^k\| = 0$.

iii) If $\alpha = \inf_{x\in\mathbb{R}^n} F_2(x) > -\infty$, *then for any subsequence* $\{x^{k_j}\}$ *of* $\{x^k\}$, *converging to* x^*, *the limit point* x^* *is a critical point of* (3.2).

Proof 3.9 *i) It follows from the computation of* x^{k+1} *that* $y^k - \rho_k x^{k+1} \in \partial g(x^{k+1})$, *where* $y^k = \rho_k x^k - \nabla f(x^k) + \xi^k$). *Hence, we obtain*

$$g(x^k) \geq g(x^{k+1}) + \langle y^k - \rho_k x^{k+1}, x^k - x^{k+1}\rangle + \frac{\mu(g)}{2}\|x^k - x^{k+1}\|^2.$$

By the condition (3.44), *we have*

$$\frac{\rho_k}{2}\|x^{k+1}\|^2 - f(x^{k+1}) + h(x^{k+1}) \geq \frac{\rho_k}{2}\|x^k\|^2 - f(x^k) + h(x^k) + \langle y^k, x^{k+1} - x^k\rangle.$$

Summing the two above inequalities, we get

$$F_2(x^k) \geq F_2(x^{k+1}) + \frac{\rho_k + \mu(g)}{2}\|x^k - x^{k+1}\|^2. \tag{3.48}$$

ii) Substituting ρ_k *by* ρ_0 ($\rho_k \geq \rho_0$) *in the inequality* (3.48) *that gives us*

$$F_2(x^k) - F_2(x^{k+1}) \geq \frac{\rho_0 + \mu(g)}{2}\|x^k - x^{k+1}\|^2. \tag{3.49}$$

By summing the above inequality over $k = 0, ..., N$ *we obtain*

$$F_2(x^0) - F_2(x^{N+1}) \geq \frac{\rho_0 + \mu(g)}{2} \sum_{k=0}^{N} \|x^k - x^{k+1}\|^2. \tag{3.50}$$

Since $\rho_0 > 0$ *and* $F_2(x^{N+1}) \geq \alpha$ *we have*

$$\frac{2}{\rho_0 + \mu(g)} \left(F_2(x^0) - \alpha\right) \geq \sum_{k=0}^{N} \|x^k - x^{k+1}\|^2.$$

By passing to the limit, we get

$$\sum_{k=0}^{+\infty} \|x^k - x^{k+1}\|^2 \leq +\infty,$$

which implies that $\lim_{k\to+\infty} \|x^k - x^{k+1}\| = 0$.
iii) Let $\{x^{k_j}\}$ *be a subsequence of* $\{x^k\}$ *which converges to* x^*. *It follows from (ii) that* $\lim_{j\to+\infty} x^{k_j+1} = x^*$. *Since* $\{x^{k_j}\}$ *is bounded and finite convexity of* h, $\{\xi^{k_j}\}$ *is bounded. Without loss of generality, we can assume that* $\{\xi^{k_j}\}$ *converges to* ξ^*. *By the closed property of the subdifferential mapping* ∂h, *we have* $\xi^* \in \partial h(x^*)$. *Furthermore, we have*

$$\rho_{k_j}(x^{k_j} - x^{k_j+1}) - \nabla f(x^{k_j}) + \xi^{k_j} \in \partial g(x^{k_j+1}).$$

Since $\{\rho_{k_j}\}$ *is bounded and by the closedness property of the subdifferential, passing to the limit we get*

$$-\nabla f(x^*) + \xi^* \in \partial g(x^*),$$

where we use the fact that $\lim_{j\to+\infty} x^{k_j} = \lim_{j\to+\infty} x^{k_j+1} = x^*$. *Therefore,*

$$\xi^* \in [\nabla f(x^*) + \partial g(x^*)] \cap \partial h(x^*).$$

■

In the Theorem 3.7, we will study the sufficient conditions which guarantee the convergence of the whole sequence $\{x^k\}$ generated by DCA-Like. Moreover, we provide the rates of convergence for the both sequences $\{x^k\}$ and $\{F_2(x^k)\}$.

Theorem 3.7 *Assume that* $\inf_{x\in\mathbb{R}^n} F_2(x) > -\infty$, *and* F_2 *is lower semicontinuous. Suppose further that* $\{x^k\}$ *generated by DCA-Like is bounded, and one of the following assumptions is satisfied :*

A) g *is differentiable with locally Lipschitz gradient and* F_2 *has the strong KL property at any* $x \in dom\, F_2$.

B) h *is differentiable with locally Lipschitz gradient and* F_2 *has KL property at any* $x \in dom\, F_2$.

Then the whole sequence $\{x^k\}$ *converges to a critical point* x^* *of* (3.2). *Moreover, if the function* ψ *in the KL inequality has the form* $\psi(s) = cs^{1-\theta}$ *with* $\theta \in [0, 1)$ *and* $c > 0$, *then we have*

i) If $\theta = 0$, *then the sequences* $\{x^k\}$ *and* $\{F_2(x^k)\}$ *converge to* x^* *and* F_2^* *in a finite number of steps, respectively.*

ii) If $\theta \in (0, 1/2]$, *then the sequences* $\{x^k\}$ *and* $\{F_2(x^k)\}$ *converge linearly to* x^* *and* F_2^*, *respectively.*

iii) If $\theta \in (1/2, 1)$, then there exist positive constants δ_1, δ_2, and N_0 such that for all $k \geq N_0$,

$$\|x^k - x^*\| \leq \delta_1 k^{-\frac{1-\theta}{2\theta-1}}$$
$$F_2(x^k) - F_2^* \leq \delta_2 k^{-\frac{1}{2\theta-1}}.$$

To prove Theorem 3.7, we use the following lemma. The lemma is a result related to the strong KL property, which is similar to the one connected with the KL property ([30], Lemma 6). It can be proved in a similar way of [30] by using the strong KL property instead of the KL property.

Lemma 3.5 *Let Ω be a compact set and $F : \mathbb{R}^n \to (-\infty, +\infty]$ be a proper lower semicontinuous function. If F has the strong KL property at each point of Ω on which F is constant, then there exist $\epsilon > 0$, $\eta > 0$, and $\psi \in \mathcal{M}_\eta$ such that for all $x \in \{x : dist(x, \Omega) < \epsilon\} \cap \{x : F^* < F_1(x) < F_1^* + \eta\}$, one has*

$$\psi'(F_1(x) - F^*) dist(0, \partial^C F_1(x)) \geq 1.$$

Proof 3.10 (Proof of Theorem 3.7) *We first prove that $\{x^k\}$ satisfies three conditions H1, H2, H3 in [14]. Obviously, the suffucient decrease condition (H1) is satisfied by the (i) of Theorem 3.6. We next verify the relative error condition (H2), i.e., there exist $N, \pi > 0$ such that for all $k \geq N$*

$$dist(0, \partial^C F_2(x^k)) \leq \pi \|x^{k+1} - x^k\|, \tag{3.51}$$

if the assumption (A) is satisfied, and

$$dist(0, \partial^L F_2(x^{k+1})) \leq \pi \|x^k - x^{k+1}\|, \tag{3.52}$$

if the assumption (B) is satisfied. Let us consider the following cases.

Case 1, (A) is satisfied. Since g is differentiable and its gradient is locally Lipschitz, it is easy to prove that there exists $N, L_1 > 0$ such that for any $k \geq N$

$$\|\nabla g(x^{k+1}) - \nabla g(x^k)\| \leq L_1 \|x^{k+1} - x^k\|. \tag{3.53}$$

On the other hand, by the definition of x^{k+1}, we have

$$\rho_k(x^k - x^{k+1}) - \nabla f(x^k) + \xi^k = \nabla g(x^{k+1}).$$

Combining the last equality with $\xi^k \in \partial h(x^k)$, we get

$$\begin{aligned} \rho_k(x^k - x^{k+1}) + \nabla g(x^k) - \nabla g(x^{k+1}) &= \nabla f(x^k) + \nabla g(x^k) - \xi^k \\ &\in \nabla f(x^k) + \nabla g(x^k) - \partial h(x^k) \\ &= \partial^C F_2(x^k). \end{aligned}$$

Hence the last equality holds since f and g are differentiable, and h is continuous. Therefore, combining the above equation with (3.53) we obtain that for any $k \geq N_1$,

$$\begin{aligned} dist(0, \partial^C F_2(x^k)) &\leq \|\rho_k(x^k - x^{k+1}) + \nabla g(x^k) - \nabla g(x^{k+1})\| \\ &\leq \|\rho_k(x^k - x^{k+1})\| + \|\nabla g(x^k) - \nabla g(x^{k+1})\| \\ &\leq (\beta L + L_1)\|x^{k+1} - x^k\|, \end{aligned} \tag{3.54}$$

knowing that $\rho_k \leq \beta L$.

Case 2, (B) is satisfied. Since h is differentiable with locally Lipschitz derivative, similar to Case 1, we can find $L_2 > 0$ such that for any $k \geq N$ (increasing N if necessary),

$$\|\nabla h(x^k) - \nabla h(x^{k+1})\| \leq L_2 \|x^k - x^{k+1}\|. \tag{3.55}$$

By the definition of x^{k+1}, we have

$$\begin{aligned} &\Big(\rho_k(x^k - x^{k+1}) - \nabla f(x^k) + \nabla h(x^k)\Big) + \nabla f(x^{k+1}) - \nabla h(x^{k+1}) \\ \in &\partial g(x^{k+1}) + \nabla f(x^{k+1}) - \nabla h(x^{k+1}) = \partial^L F_2(x^{k+1}). \end{aligned} \tag{3.56}$$

On the other hand, by the Lipschitz continuity of ∇f and (3.55), we obtain

$$\begin{aligned} &\| \Big(\rho_k(x^k - x^{k+1}) - \nabla f(x^k) + \nabla h(x^k)\Big) + \nabla f(x^{k+1}) - \nabla h(x^{k+1})\| \\ \leq &(\beta L + L + L_2) \|x^k - x^{k+1}\|. \end{aligned}$$

It follows from the above inequality and (3.56) that for any $k \geq N$,

$$dist(0, \partial^L F_2(x^{k+1})) \leq (\beta L + L + L_2) \|x^k - x^{k+1}\|. \tag{3.57}$$

We now verify the continuity condition (H3). Since the boundedness of $\{x^k\}$, there exists a sub-sequence $\{x^{k_j+1}\}$ that converges to a point x^*. We need to prove that $\lim_{j\to\infty} F_2(x^{k_j+1}) = F_2(x^*)$. Indeed, we have

$$x^{k_j+1} \in \arg\min \left\{ \frac{\rho_{k_j}}{2} \|x - x^{k_j}\|^2 + g(x) - \langle \xi^{k_j} - \nabla f(x^{k_j}), x - x^{k_j} \rangle \right\},$$

which implies that

$$\begin{aligned} &\frac{\rho_{k_j}}{2} \|x^{k_j+1} - x^{k_j}\|^2 + g(x^{k_j+1}) - \langle \xi^{k_j} - \nabla f(x^{k_j}), x^{k_j+1} - x^{k_j} \rangle \\ \leq &\frac{\rho_{k_j}}{2} \|x^* - x^{k_j}\|^2 + g(x^*) - \langle \xi^{k_j} - \nabla f(x^{k_j}), x^* - x^{k_j} \rangle. \end{aligned}$$

Using the limsup, we get

$$\limsup_{j\to+\infty} g(x^{k_j+1}) \leq g(x^*), \tag{3.58}$$

where we have used the bounded properties of $\{\rho_k\}$ and $\{\xi^k\}$, the continuity of ∇f, and (ii) of Theorem 3.6. Therefore, we have

$$\begin{aligned} \limsup_{j\to\infty} F_2(x^{k_j+1}) &= \limsup_{j\to\infty} [f(x^{k_j+1}) + g(x^{k_j+1}) - h(x^{k_j+1})] \\ &\leq \limsup_{j\to\infty} f(x^{k_j+1}) + \limsup_{j\to\infty} g(x^{k_j+1}) - \liminf_{j\to\infty} h(x^{k_j+1}) \\ &\leq f(x^*) + g(x^*) - \liminf_{j\to\infty} h(x^{k_j+1}) \\ &\leq f(x^*) + g(x^*) - h(x^*) = F_2(x^*), \end{aligned}$$

where the second inequality follows from the continuity of f and (3.58), and the last inequality holds by the lower semicontinuity of h. On the other hand, from the lower semicontinuity of F_2, we obtain $\liminf_{j\to\infty} F_2(x^{k_j+1}) \geq F_2(x^*)$. Hence, we get $\lim_{j\to\infty} F_2(x^{k_j+1}) = F_2(x^*)$.

Since the convergence of whole sequence $\{x^k\}$ to a critical point now can follow from Lemma 3.5 and similar arguments of the proof for [14, Theorem 2.9] while the second part of Theorem can be justified by using similar proof techniques of [29, Theorem 2] and [74, Theorem 3.4], we omit the detail of the rest of the proof. ■

3.3.2 Accelerated DCA-Like

We now present the accelerated version of DCA-Like to solve (3.2). Note that the acceleration step is more general than the one in ADCA-Like for (3.1) (Algorithm 3.3). Here, we compare the extrapolated point z^k with last q iterates $\{x^{k-q}, \ldots, x^{k-1}, x^k\}$ while we only do the comparison with the latest iterate in Algorithm 3.3. Theoretically, a large-value of q increases the chance of using the extrapolated points z^k in ADCA and consequently increases its chance to accelerate

Algorithm 3.5 ADCA-Like for solving (3.2)

1: **Initialization :** Choose an initial point x^0, $z^0 = x^0$, $q \in \mathbb{N}$, $t_0 = (1+\sqrt{5})/2$, a small enough positive parameter ρ_0, $\eta > 1, 0 < \delta < 1$, and $k \leftarrow 0$.
2: **repeat**
3: **If** $F_2(z^k) \leq \max\limits_{t=[k-q]_+,\ldots,k} F_2(x^t)$, **then** set $v^k = z^k$, **otherwise** set $v^k = x^k$.
4: Compute $\xi^k \in \partial h(v^k)$ and $\nabla f(v^k)$.
5: Set $\rho_k = \max\{\rho_0, \delta\rho_{k-1}\}$ if $k > 0$.
6: Compute x^{k+1} by solving (3.45).

$$\min_{x\in\mathbb{R}^n} \left\{ \frac{\rho_k}{2}\|x\|^2 + g(x) - \langle \rho_k v^k - \nabla f(v^k) + \xi^k, x\rangle \right\}. \tag{3.59}$$

7: **while** $H_{\rho_k}(x^{k+1}) < H_{\rho_k}(x^{k+1}, v^k)$ **do**
8: $\rho_k \leftarrow \eta\rho_k$.
9: Compute x^{k+1} by solving (3.45).
10: **end while**
11: Compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$ and $z^{k+1} = x^{k+1} + \frac{t_k - 1}{t_{k+1}}\left(x^{k+1} - x^k\right)$
12: $k \leftarrow k + 1$.
13: **until** Stopping criterion.

The convergence properties and convergence rates of ADCA-Like are provided in the following theorems. Denote by $\{\Gamma^k\}$ and $\{\phi(k)\}$ the sequences respectively defined as $\Gamma^k = \max\limits_{t=[k-q]_+,\ldots,k} F_2(x^t)$ and $\phi(k) = \operatorname{arg\,min}\limits_{t=k+1,\ldots,k+1+q} \frac{\rho_{t-1}+\mu(g)}{2}\|x^t - v^{t-1}\|^2$.

Theorem 3.8 *Let $\{x^k\}$ and $\{v^k\}$ be the sequences generated by Algorithm 3.5. The following statements hold.*

i) For any $k = 0, 1, ...,$

$$\Gamma^k - \Gamma^{k+1+q} \geq \frac{\rho_{\phi(k)-1} + \mu(g)}{2}\|x^{\phi(k)} - v^{\phi(k)-1}\|^2. \tag{3.60}$$

As a result, by choosing $q = 0$, we get the monotone property of $\{F_2(x^k)\}$, i.e., $F_2(x^k) - F_2(x^{k+1}) \geq \frac{\rho_k+\mu(g)}{2}\|x^{k+1} - v^k\|^2$.

ii) If $\alpha = \inf_{x\in\mathbb{R}^n} F_2(x) > -\infty$, then

$$\sum_{k=0}^{+\infty} \|x^{\phi(k)} - v^{\phi(k)-1}\|^2 < +\infty,$$

and therefore $\lim_{k\to+\infty} \|x^{\phi(k)} - v^{\phi(k)-1}\| = 0$.

iii) If $\alpha = \inf_{x\in\mathbb{R}^n} F_2(x) > -\infty$, *then for any subsequence* $\{x^{\phi(k_j)}\}$ *of* $\{x^{\phi(k)}\}$, *converging to* x^*, *the limit point* x^* *is a critical point of* (3.2).

Proof 3.11 *i) From* $y^k - \rho_k x^{k+1} \in \partial g(x^{k+1})$, *where* $y^k = \rho_k v^k - \nabla f(v^k) + \xi^k$, *we have*

$$g(v^k) \geq g(x^{k+1}) + \langle y^k - \rho_k x^{k+1}, v^k - x^{k+1}\rangle + \frac{\mu(g)}{2}\|v^k - x^{k+1}\|^2.$$

Combining the above inequality and $H_{\rho_k}(x^{k+1}) \geq H_{\rho_k}(x^{k+1}, v^k)$, *we get the following inequality*

$$F_2(v^k) - F_2(x^{k+1}) \geq \frac{\rho_k + \mu(g)}{2}\|x^{k+1} - v^k\|^2. \tag{3.61}$$

Observe that $F_2(v^k) \leq \max_{t=[k-d]_+,\ldots,k} F_2(x^t) = \Gamma^k$. *It follows that*

$$F_2(x^{k+1}) \leq \Gamma^k - \frac{\rho_k + \mu(g)}{2}\|x^{k+1} - v^k\|^2. \tag{3.62}$$

This implies that $F_2(x^{k+1}) \leq \Gamma^k$. *We prove by induction that for all* $t = 0, ..., q$

$$F_2(x^{k+1+t}) \leq \Gamma^k - \frac{\rho_{k+t} + \mu(g)}{2}\|x^{k+1+t} - v^{k+t}\|^2. \tag{3.63}$$

Indeed, it follows from (3.62) *that the claim holds for* $t = 0$. *We suppose that it also holds for* $t = 0, ..., p-1$ *with* $1 \leq p \leq q$. *Thus, we have*

$$\begin{aligned} F_2(x^{k+1+p}) &\leq \Gamma^{k+p} - \frac{\rho_{k+p} + \mu(g)}{2}\|x^{k+1+p} - v^{k+p}\|^2 \\ &\leq \max(\Gamma^k, F_2(x^{k+1}), ..., F_2(x^{k+p})) - \frac{\rho_{k+t} + \mu(g)}{2}\|x^{k+1+p} - v^{k+p}\|^2 \\ &\leq \Gamma^k - \frac{\rho_{k+p} + \mu(g)}{2}\|x^{k+1+p} - v^{k+p}\|^2, \end{aligned}$$

where last inequality follows from $F_2(x^{k+1+t}) \leq \Gamma^k$ *for* $t = 0, ..., p-1$. *Therefore, we obtain*

$$\begin{aligned} \Gamma^{k+q+1} &= \max_{t=k+1,\ldots,k+q+1} F_2(x^t) \\ &\leq \Gamma^k - \min_{t=k+1,\ldots,k+1+q} \frac{\rho_{t-1} + \mu(g)}{2}\|x^t - v^{t-1}\|^2 \\ &= \Gamma^k - \frac{\rho_{\phi(k)-1} + \mu(g)}{2}\|x^{\phi(k)} - v^{\phi(k)-1}\|^2. \end{aligned}$$

ii) Note that $\Gamma^k \geq \alpha$ *for all* k. *By substituting* $\rho_k \geq \rho_0$ *in* (3.60), *we get*

$$\frac{\rho_0 + \mu(g)}{2}\|x^{\phi(k)} - v^{\phi(k)-1}\|^2 \leq \Gamma^k - \Gamma^{k+q+1}. \tag{3.64}$$

Summing the inequality over $k = 0, ..., N$, *we obtain*

$$\begin{aligned} \frac{\rho_0 + \mu(g)}{2}\sum_{k=0}^{N}\|x^{\phi(k)} - v^{\phi(k)-1}\|^2 &\leq \sum_{t=0}^{q}(\Gamma^t - \Gamma^{N+t+1}) \\ &\leq (q+1)(\max_{t=0,\ldots,q} F_2(x^t) - \alpha). \end{aligned}$$

Therefore

$$\sum_{k=0}^{N} \|x^{\phi(k)} - v^{\phi(k)-1}\|^2 \leq \frac{2(q+1)(\max\limits_{t=0,\ldots,q} F_2(x^t) - \alpha)}{\rho_0 + \mu(g)}.$$

Passing to the limit over the sequence $\{N\}_{N\in\mathbb{N}}$ *we get*

$$\sum_{k=0}^{+\infty} \|x^{\phi(k)} - v^{\phi(k)-1}\|^2 < +\infty, \tag{3.65}$$

and therefore $\lim_{k\to+\infty} \|x^{\phi(k)} - v^{\phi(k)-1}\| = 0$.
iii) Let $\{x^{\phi(k_j)}\}$ *be a subsequence of* $\{x^{\phi(k)}\}$ *that converges to* x^*. *It follows from (ii) that* $\lim_{j\to+\infty} v^{\phi(k_j)-1} = x^*$. *Since the sequence* $\{v^{\phi(k_j)-1}\}$ *is bounded,* $\{\xi^{\phi(k_j)-1}\}$ *is bounded too. Without loss of generality, we can suppose that the sequence* $\{\xi^{\phi(k_j)-1}\}$ *converges to* ξ^*. *By the closed property of the subdifferential mapping* ∂h, *we have* $\xi^* \in \partial h(x^*)$. *We note that*

$$\rho_{\phi(k_j)-1}(v^{\phi(k_j)-1} - x^{\phi(k_j)}) - \nabla f(v^{\phi(k_j)-1}) + \xi^{\phi(k_j)-1} \in \partial g(x^{\phi(k_j)}). \tag{3.66}$$

Passing the limit of (3.66) *and taking into account the closedness of* ∂g, *we obtain* $-\nabla f(x^*)+\xi^* \in \partial g(x^*)$. *Therefore,* $\xi^* \in [\nabla f(x^*) + \partial g(x^*)] \cap \partial h(x^*)$ *which implies that* x^* *is a critical point of* (3.2). ■

The flowing theorem shows rates of convergence of ADCA-Like in terms of objective value. Denote $\ell(k) = \arg\max_{t=[k-q]_+,\ldots,k} F_2(x^t)$ and $i(k) = \ell((q+1)k)$. From the proof of Theorem 3.7, we can show that $\{F_2(x^{\ell(k)})\}$ is non-increasing. Hence, there exists L^* such that $L^* = \lim_{k\to+\infty} F_2(x^{\ell(k)})$.

Theorem 3.9 *Assume that* $\inf_{x\in\mathbb{R}^n} F_2(x) > -\infty$, *and* F_2 *is is lower semicontinuous. Suppose further that* $\{x^k\}$ *generated by ADCA-Like is bounded, and one of the following assumptions is satisfied :*

A) g *is differentiable with locally Lipschitz gradient and* F_2 *has the strong KL property at any* $x \in dom\, F_2$.

B) h *is differentiable with locally Lipschitz gradient and* F_2 *has KL property at any* $x \in dom\, F_2$.

Moreover, if the function ψ *in the KL inequality has the form* $\psi(s) = cs^{1-\theta}$ *with* $\theta \in [0,1)$ *and* $c > 0$, *then we have the following conclusions*

i) If $\theta = 0$, *there exists* N *such that* $F_2(x^k) - F_2^* \leq 0$ *for all* $k \geq N$.

ii) If $\theta \in (0, 1/2]$, *there exists* $\delta \in (0,1), c > 0$, *and* N *such that* $F_2(x^k) - F_2^* \leq c\delta^k$ *for all* $k \geq N$.

iii) If $\theta \in (1/2, 1)$, *there exist positive constants* η *and* N *such that*

$$F_2(x^k) - F_2^* \leq \eta k^{-\frac{1}{2\theta-1}},$$

for all $k \geq N$.

Proof 3.12 *Like the proof of Theorem 3.7, we verify three similar conditions H1, H2, and H3, but they are different from those in the proof of Theorem 3.7 in the way that there is the sequence* $\{v^{i(k)}\}$. *We first verify the condition H1. Substituting* $i(k) - 1$ *for* k *in the inequality* (3.11), *we obtain*

$$\frac{\rho_{i(k)-1} + \mu(g)}{2} \|x^{i(k)} - v^{i(k)-1}\|^2 \leq F_2(x^{\ell(i(k)-1)}) - F_2(x^{i(k)}). \tag{3.67}$$

In addition, we have

$$i(k) - 1 \geq (q+1)k - q - 1 = (q+1)(k-1). \tag{3.68}$$

It follows from this and (3.11) *that* $F_2(x^{\ell(i(k)-1)}) \leq F_2(x^{i(k-1)})$. *Therefore, we get*

$$\frac{\rho_0 + \mu(g)}{2}\|x^{i(k)} - v^{i(k)-1}\|^2 \leq F_2(x^{i(k-1)}) - F_2(x^{i(k)}). \tag{3.69}$$

To verify the condition H2, we consider the following cases.

Case 1, (A) is satisfied. Since g is differentiable and its gradient is locally Lipschitz, there exists $N, L_1 > 0$ *such that for any* $k \geq N$

$$\|\nabla g(x^{i(k)}) - \nabla g(v^{i(k)-1})\| \leq L_1\|x^{i(k)} - v^{i(k)-1}\|. \tag{3.70}$$

On the other hand, by the definition of $x^{i(k)}$, *we have*

$$\rho_{i(k)-1}(v^{i(k)-1} - x^{i(k)}) - \nabla f(v^{i(k)-1}) + \xi^{i(k)-1} = \nabla g(x^{i(k)}).$$

Combining this with $\xi^{i(k)-1} \in \partial h(v^{i(k)-1})$, *we get*

$$\rho_{i(k)-1}(v^{i(k)-1} - x^{i(k)}) + \nabla g(v^{i(k)-1}) - \nabla g(x^{i(k)}) \in \partial^C F_2(v^{i(k)-1}).$$

Therefore, combining this with (3.70), *we obtain that for any* $k \geq N_1$,

$$dist(0, \partial^C F_2(v^{i(k)-1})) \leq \pi\|x^{i(k)} - v^{i(k)-1}\|, \tag{3.71}$$

where $\pi = \beta L + L_1$.

Case 2, (B) is satisfied. Since h is differentiable with locally Lipschitz gradient, we can find $L_2 > 0$ *such that for any* $k \geq N$ *(increasing N if necessary),*

$$\|\nabla h(v^{i(k)-1}) - \nabla h(x^{i(k)})\| \leq L_2\|v^{i(k)-1} - x^{i(k)}\|. \tag{3.72}$$

By the definition of $x^{i(k)}$, *we have*

$$\begin{aligned}&\Big(\rho_{i(k)-1}(v^{i(k)-1} - x^{i(k)}) - \nabla f(v^{i(k)-1}) + \nabla h(v^{i(k)-1})\Big) + \nabla f(x^{i(k)})\\ &\quad -\nabla h(x^{i(k)}) \in \partial g(x^{i(k)}) + \nabla f(x^{i(k)}) - \nabla h(x^{i(k)}) = \partial^L F_2(x^{i(k)}).\end{aligned} \tag{3.73}$$

Therefore, for any $k \geq N$, *we obtain*

$$dist(0, \partial^L F_2(x^{i(k)})) \leq \pi\|v^{i(k)-1} - x^{i(k)}\|, \tag{3.74}$$

where $\pi = \beta L + L + L_2$.

We now verify the condition H3, i.e., we have to show that there exists a sub-sequence $\{x^{i(k_j)}\}$ *converging to a point* x^* *such that* $\lim_{j\to\infty} F_2(x^{i(k_j)}) = F_2(x^*)$. *Indeed, since the boundedness of* $\{x^{i(k)}\}$, *there exists* $\{x^{i(k_j)}\}$ *that converges to a point* x^*. *We have*

$$x^{i(k_j)} \in \arg\min\left\{\frac{\rho_{i(k_j)-1}}{2}\|x - v^{i(k_j)-1}\|^2 + g(x) - \langle \xi^{i(k_j)-1} - \nabla f(v^{i(k_j)-1}), x - v^{i(k_j)-1}\rangle\right\},$$

which implies that

$$\begin{aligned}&\frac{\rho_{i(k_j)-1}}{2}\|x^{i(k_j)} - v^{i(k_j)-1}\|^2 + g(x^{i(k_j)}) - \langle \xi^{i(k_j)-1} - \nabla f(v^{i(k_j)-1}), x^{i(k_j)} - v^{i(k_j)-1}\rangle\\ \leq&\frac{\rho_{i(k_j)-1}}{2}\|x^* - v^{i(k_j)-1}\|^2 + g(x^*) - \langle \xi^{i(k_j)-1} - \nabla f(v^{i(k_j)-1}), x^* - v^{i(k_j)-1}\rangle.\end{aligned}$$

Using the limsup, we get

$$\limsup_{j\to+\infty} g(x^{i(k_j)}) \leq g(x^*), \tag{3.75}$$

where we have used the bounded properties of $\{\rho_{i(k)-1}\}$ *and* $\{\xi^{i(k)-1}\}$*, the continuity of* ∇f*, and the fact that* $\|x^{i(k_j)} - v^{i(k_j)-1}\| \to 0$ *as* $j \to +\infty$*. Therefore, we have*

$$\begin{aligned}\limsup_{j\to\infty} F_2(x^{i(k_j)}) &= \limsup_{j\to\infty}[f(x^{i(k_j)}) + g(x^{i(k_j)}) - h(x^{i(k_j)})]\\ &\leq \limsup_{j\to\infty} f(x^{i(k_j)}) + \limsup_{j\to\infty} g(x^{i(k_j)}) - \liminf_{j\to\infty} h(x^{i(k_j)})\\ &\leq f(x^*) + g(x^*) - \liminf_{j\to\infty} h(x^{i(k_j)})\\ &\leq f(x^*) + g(x^*) - h(x^*) = F_2(x^*),\end{aligned}$$

where the second inequality follows from the continuity of f *and* (3.58)*, and the last inequality holds by the lower semicontinuity of* h*. On the other hand, from the lower semicontinuity of* F_2*, we obtain* $\liminf_{j\to\infty} F_2(x^{i(k_j)}) \geq F_2(x^*)$*. Hence, we get* $\lim_{j\to\infty} F_2(x^{i(k_j)}) = F_2(x^*)$*.*

The results now can follow from Lemma 3.5, similar arguments of the proof for [14, Theorem 2.9] and [29, Theorem 2], and the facts that $F_2(x^{i(k)}) \leq F_2(v^{i(k)-1}) \leq F_2(x^{i(k-1)})$ *and for all* $k \in [(q+1)t - q, (q+1)t]$ *with any* $t \geq 1$*,* $F_2(x^k) \leq F_2(x^{i(t)})$*.* ■

3.4 Application to the t-distributed Stochastic Neighbor Embedding problem

The t-SNE can be described as follows. Given a data set of N (high-dimensional) input objects $\mathcal{D} = \{a_1, ..., a_N\}$ with $a_i \in \mathbb{R}^d$, the t-SNE aims to find, for each input object a_i, a low-dimensional embedding point $x_i \in \mathbb{R}^s$ in a way that respects similarities between points. To this end, t-SNE first defines joint probabilities p_{ij} (using Gaussian distribution) and q_{ij} (using a Student t-distribution with one degree of freedom [161]) that measure, respectively, the pairwise similarity between objects a_i and a_j in the original space, and the one between two points x_i and x_j in the embedding space $\mathcal{E}$, which are given by

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N};\ p_{j|i} = \frac{\exp(-\|a_i - a_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|a_i - a_k\|^2/2\sigma_i^2)} \text{ if } i \neq j, \text{ 0 otherwise,}$$

where σ_i is the variance of the Gaussian that is centered at a_i, and

$$q_{ij} = \frac{(1 + \|x_i - x_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|x_k - x_l\|^2)^{-1}} \text{ if } i \neq j, \text{ and 0 otherwise.}$$

Then t-SNE learns a s-dimension map $x_1, \ldots, x_N \in \mathbb{R}^s$ which minimizes the Kullback-Leibler divergence between the two joint distributions $P = (p_{ij})$ and $Q = (q_{ij})$:

$$\min_x \left\{ F(x) = KL(P||Q) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \right\}. \tag{3.76}$$

The nonconvex optimization problem (3.76) has been studied in several works ([161, 246, 229]), and the most noticeable was presented in [247] where the authors presented and compared

Majorization Minimization algorithm (MM) with five state-of-the-arts methods, such as gradient descent, gradient descent with momentum [161], spectral direction [229], FPHSSNE [246] and Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS [178]. The numerical results showed that MM [247] outperforms all five state-of-the-art optimization methods.

We now show that (3.76) takes the form of (3.1). Indeed, the function F in (3.76) can be rewritten as

$$F(x) = f(x) + \sum_{i,j} h_{ij}(g_{ij}(x_i, x_j)), \tag{3.77}$$

where

$$f(x) := \sum_{i \neq j} p_{ij} \log p_{ij} + \log(\sum_{i \neq j} (1 + \|x_i - x_j\|^2)^{-1}), \tag{3.78}$$

$$h_{ij}(t) = p_{ij} \log(1 + t),\ g_{ij}(x_i, x_j) = \|x_i - x_j\|^2. \tag{3.79}$$

It is obvious that g_{ij} are convex functions, and h_{ij} are concave increasing functions. Moreover, the function f is differentiable with L-Lipschitz continuous gradient by the following proposition.

Proposition 3.3 *The function $f(x)$ in (3.78) is smooth with Lipschitz gradient, where we can choose a Lipschitz constant $L = 4$.*

Proof 3.13 *We recall the definition of function f*

$$f(x) := \sum_{i \neq j} p_{ij} \log p_{ij} + \log \left(\sum_{i \neq j} (1 + \|x_i - x_j\|^2)^{-1} \right).$$

Note that p_{ij} is known in advance hence the first term of f is a constant. Let's consider the second term of f

$$\varphi(x) = \log \left(\sum_{i \neq j} (1 + \|x_i - x_j\|^2)^{-1} \right).$$

For $i \neq j$, let's define the function

$$z_{ij}(x) = \|x_i - x_j\|^2.$$

Then, we have

$$\varphi(x) = \log \left(\sum_{i \neq j} \frac{1}{1 + z_{ij}(x)} \right),$$

or equivalently

$$e^{\varphi(x)} = \sum_{i \neq j} \frac{1}{1 + z_{ij}(x)}.$$

Taking derivatives in x both sides of the above equation, we get

$$\nabla \varphi(x)\ e^{\varphi(x)} = \sum_{i \neq j} \frac{-1}{(1 + z_{ij}(x))^2} \nabla z_{ij}(x).$$

Taking derivatives in x both sides of the above equation, we get

$$\nabla^2\varphi(x)\ e^{\varphi(x)} + \nabla\varphi(x)(\nabla\varphi(x))^T\ e^{\varphi(x)} = \sum_{i\neq j}\left(\frac{2}{(1+z_{ij}(x))^3}\nabla z_{ij}(x)(\nabla z_{ij}(x))^T + \frac{-1}{(1+z_{ij}(x))^2}\nabla^2 z_{ij}(x)\right)$$

Note that z_{ij} are convex then $\nabla^2 z_{ij}(x) \succeq 0$. The above equation implies that

$$\begin{aligned}\nabla^2\varphi(x)\ e^{\varphi(x)} &\preceq \sum_{i\neq j}\frac{2}{(1+z_{ij}(x))^3}\nabla z_{ij}(x)(\nabla z_{ij}(x))^T\\ &\preceq \sum_{i\neq j}\frac{2\|\nabla z_{ij}(x)\|^2}{(1+z_{ij}(x))^3}\ \mathbf{I}.\end{aligned}$$

Moreover,

$$\|\nabla z_{ij}(x)\|^2 = 8\|x_i - x_j\|^2 = 8z_{ij}(x) \leq 2(1+z_{ij}(x))^2.$$

Thus,

$$\nabla^2\varphi(x)\ e^{\varphi(x)} \preceq \sum_{i\neq j}\frac{4}{1+z_{ij}(x)}\ \mathbf{I} = 4\ e^{\varphi(x)}\ \mathbf{I},$$

or equivalently,

$$\nabla^2\varphi(x) \preceq 4\mathbf{I}.$$

Therefore, $\varphi(x)$ is Lipschitz gradient with a Lipschitz constant $L = 4$. ■

As (3.76) takes the form of (3.1), DCA-Like (Algorithm 3.2) and ADCA-Like (Algorithm 3.3) can be developed to solve it. For applying DCA based algorithms on (3.76), at each iteration, we have to compute

$$\xi_{ij}^k = \nabla(-h_{ij})(g_{ij}(x_i^k, x_j^k)) = -\frac{p_{ij}}{1+\|x_i^k - x_j^k\|^2}$$

and $\nabla f(x^k)$ by

$$\nabla_{x_i} f(x^k) = \sum_{j=1}^n \frac{-4(x_i^k - x_j^k)(1+\|x_i^k - x_j^k\|^2)^{-2}}{\sum_{l\neq m}(1+\|x_l^k - x_m^k\|^2)^{-1}}, \tag{3.80}$$

and solve the following convex problem to compute x^{k+1}

$$\min_x \left\{\frac{\rho_k}{2}\|x - x^k\|^2 + \langle\nabla f(x^k), x\rangle + \sum_{i,j} -\xi_{ij}^k\|x_i - x_j\|^2\right\}.$$

The solution x^{k+1} of the above problem is given by

$$x^{k+1} = (2\mathcal{L}_{-\xi^k - (\xi^k)^T} + \rho_k I)^{-1}(-\nabla f(x^k) + \rho_k x^k), \tag{3.81}$$

where the matrix ξ^k is defined by the elements ξ_{ij}^k and $\mathcal{L}_A$ denotes the matrix with $(\mathcal{L}_A)_{ij} = -A_{ij}$ if $i \neq j$ and $-A_{ii} + \sum_{l=1}^n A_{il}$ otherwise. Note that all the above DCA based algorithms are in explicit form and very inexpensive. We also observe that, from the update rule (3.81) for x^{k+1} and this stopping criterion for searching ρ_k, the algorithm MM in [247] for (3.76) is a special version of DCA-Like.

Algorithm 3.6 DCA-Like for solving t-SNE problem (3.76)

Initialization Choose x^0, $\eta > 1, 0 < \delta < 1$, $\rho_0 > 0$ and $k \leftarrow 0$.
repeat
  Compute $\xi_{ij}^k = -\frac{p_{ij}}{1+\|x_i^k - x_j^k\|^2}$ and $\nabla f(x^k)$ by (3.80).
  Set $\rho_k = \max\{\rho_0, \delta\rho_{k-1}\}$ if $k > 0$.
  Compute x^{k+1} by (3.81).
  while $U_{\rho_k}^{tsne}(x^{k+1}, x^k) < F(x^{k+1})$ **do**
    $\rho_k \leftarrow \eta\rho_k$.
    Compute x^{k+1} by (3.81).
  end while
  $k \leftarrow k + 1$.
until Stopping criterion

According to the Algorithm 3.2, at each iteration k, we only need to find ρ_k such that

$$U_{\rho_k}^{tsne}(x^{k+1}, x^k) \geq F(x^{k+1}), \tag{3.82}$$

where $U_{\rho_k}^{tsne}(x^{k+1}, x^k) = F(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{\mu_k}{2}\|x^{k+1} - x^k\|^2 - \langle \xi^k, g(x^{k+1}) - g(x^k) \rangle$.

Finally, DCA-Like for solving (3.76) is given in Algorithm 3.6.

ADCA-Like for (3.76) is obtained by adding the acceleration steps (Step 3 and 11 of Algorithm 3.3) to Algorithm 3.6.

We recall that all semi-algebraic functions and subanalysis functions satisfy the KL property [13], for examples, real polynomial functions, logarithm function, ℓ_p-norm with $p \geq 0$. In addition, finite sums, products, generalized inverse, compositions of semi-algebraic functions are also semi-algebraic. This implies that the objective function of (3.76) satisfies the KL property. Hence, DCA-Like and ADCA-Like for solving (3.76) enjoy all convergence properties provided in Theorems 3.6–3.9.

3.4.1 Experiment setting

The numerical experiments were realized on several data sets taken from the UCI data repository. We compare our algorithms with MM [247] and DC Proximal Newton (DC-PN) [198]. Recall that Yang et al. (2015) [247] have developed a MM method for t-SNE and showed that their method outperformed many other methods for t-SNE such as gradient descent, gradient descent with momentum, spectral direction, FPHSSNE, etc. And as MM for t-SNE [247] is a special case of DCA-Like, it is not necessary to compare our methods with other non DCA-based methods. On the other hand, DC-PN is chosen since it is one of the most recent paper and has been successfully applied to several non-convex optimization problems, especially in Machine Learning.

As mentioned before, in DCA and ADCA schemes for solving (3.76) we have to estimate the L-Lipschitz constant of f. According to Proposition 3.3, we can choose $L = 4$. However, in practice, this value is still large and consequently DCA could converge rapidly to a bad solution. Hence, we incorporate a ρ updating procedure into DCA and ADCA. We start with a small value of ρ and increase ρ if the objective value increases in DCA/ADCA scheme. For all algorithms, the initial value of ρ is set to be $\rho_0 = 10^{-6}$. We set $\eta = \frac{1}{\delta} = 2$ as proposed in [247].

We adopt the same test procedure as described in [247]. The initial point x^0 is drawn from normal distribution $\mathcal{N}(0, 10^{-8})$ for all methods. We use the early exaggeration technique as

proposed in [161] which consists in running the algorithm in 2 stages. In the first stage, we run the algorithm from the initial point x^0 with modified value of p_{ij} (for instance, p_{ij} is multiplied by 4). Then the solution of the first stage is used as a starting point for the second stage in which we use the true value of p_{ij}. By using large value of p_{ij} in the first stage, the algorithm is encouraged to focus on modelling p_{ij} by fairly large q_{ij} in low dimensional space and consequently it tends to form tight widely separated clusters in the map. This creates a lot of empty space in the map, which makes it much easier for the clusters to move around in order to find a good global organization in the second stage [161].

Recent research on t-SNE suggests using Barnes-Hut tree approximation to reduce the running time by computing approximately the objective function and gradient in t-SNE [160]. This technique is well-known in Neighbor Embedding problems, which allows to reduce greatly the running time while having a small loss in gradients and the objective function. The parameter $\theta_{\text{Barnes-Hut}}$ is set to be 0.5. It has also been shown that Barnes-Hut tree approximation works very well with sparse matrix P. In our experiment, k-Nearest Neighbor (with $k = 10$) is employed to construct a sparse matrix P as follows : $p_{ij} = \frac{\bar{p}_{ij}}{\sum_{k,l} \bar{p}_{kl}}$ where $\bar{p}_{ij} = 1$ if data point j (reps. i) is one of k nearest neighbors of data point j (reps. i) and $\bar{p}_{ij} = 0$ otherwise. We use the same way to compute variances σ_i of Gaussian centered at a_i as in [161]. More precisely, one performs a binary search for the value of σ_i that produces a probability distribution P_i with a fixed perplexity given by the user.

The stopping conditions of all algorithms are the same, by either (a) number of iterations exceeds 10000 or (b) $\|x^k - x^{k-1}\|/\|x^{k-1}\| \leq 10^{-8}$. Throughout our experiment, the number of embedding dimension is set to $s = 2$.

The performance of each algorithm is evaluated on three criteria : the objective value $F(x)$, the number of iterations and the running time (measured in seconds). We run each algorithm 10 times and then report the mean and standard deviation for each criterion in Table 3.2.

3.4.2 Numerical results

TABLE 3.2 – Comparative results on t-SNE problem. Bold values indicate best results.

Dataset	Algorithm	Objective		Iteration		Time (sec.)	
		Mean	STD	Mean	STD	Mean	STD
gisette	DCA	3.52	0.04	**27**	0	**15.3**	0.2
$N = 7000$	ADCA	3.33	0.00	118	31	39.5	7.9
$d = 5000$	DCA-Like	3.34	0.01	283	81	209.0	57.9
	ADCA-Like	**3.32**	0.02	133	24	62.9	10.0
	DC-PN	3.53	0.02	2187	186	1218.3	194.3
usps	DCA	2.41	0.01	**29**	1	**23.1**	1.8
$N = 9298$	ADCA	3.92	1.36	70	107	27.9	39.0
$d = 256$	DCA-Like	**2.34**	0.00	106	18	62.5	11.0
	ADCA-Like	**2.34**	0.01	107	15	64.9	6.0
	DC-PN	2.57	0.07	2083	538	1315.2	171.7
magic	DCA	2.40	0.00	850	373	413.4	170.4
$N = 19020$	ADCA	2.46	0.01	215	9	150.5	9.8
$d = 10$	DCA-Like	**2.36**	0.00	168	39	135.2	31.1
	ADCA-Like	**2.36**	0.00	**106**	7	**101.7**	4.8
	DC-PN	2.80	0.07	3498	237	4321.0	557.8

							s
letters	DCA	1.58	0.02	1186	159	652.8	98.2
$N = 20000$	ADCA	1.49	0.02	369	104	268.4	86.8
$d = 16$	DCA-Like	**1.48**	0.01	164	24	149.0	19.3
	ADCA-Like	**1.48**	0.01	**90**	7	**96.8**	7.4
	DC-PN	2.14	0.07	1326	253	1997.1	539.0
shuttle	DCA	1.60	0.02	3520	459	6056.5	782.9
$N = 58000$	ADCA	1.48	0.04	760	96	1781.7	215.1
$d = 9$	DCA-Like	1.45	0.02	333	9	981.7	33.1
	ADCA-Like	**1.42**	0.00	**152**	23	**553.7**	123.6
	DC-PN	2.54	0.03	4693	68	26784.4	3633.2
sensorless	DCA	3.18	0.02	1788	454	3232.1	837.3
$N = 58509$	ADCA	**3.13**	0.01	418	24	912.8	50.0
$d = 48$	DCA-Like	3.21	0.02	313	41	953.7	114.5
	ADCA-Like	3.18	0.02	**153**	28	**499.3**	91.2
	DC-PN	3.81	0.04	4255	144	20586.9	4488.6
mnist	DCA	3.44	0.00	3894	111	13752.7	644.9
$N = 70000$	ADCA	3.45	0.01	960	60	2659.1	63.3
$d = 784$	DCA-Like	3.46	0.01	371	67	1576.8	259.7
	ADCA-Like	**3.43**	0.01	**196**	27	**834.1**	127.5
	DC-PN	4.12	0.01	3703	308	21486.7	3643.1
miniboone	DCA	3.55	0.05	3401	1151	20473.3	6471.2
$N = 130064$	ADCA	**3.47**	0.06	454	338	2917.1	2050.2
$d = 50$	DCA-Like	3.53	0.05	469	61	4841.7	691.5
	ADCA-Like	3.53	0.02	**170**	12	**1560.6**	209.1
	DC-PN	5.36	0.15	471	29	7071.7	844.0
covertype	DCA	2.14	0.00	3998	35	71591.8	1407.2
$N = 581012$	ADCA	1.88	0.00	1670	0	29954.8	0.0
$d = 54$	DCA-Like	2.10	0.04	1217	66	37123.1	3546.5
	ADCA-Like	**1.68**	0.01	**330**	10	**8338.4**	191.8
	DC-PN	4.24	0.00	5741	45	140122.8	5126.4

• Comparison between DCA-Like and DCA.

For two smallest datasets (*gisette* and *ups*), DCA is faster than DCA-Like while the later is better than the former in terms of objective value. One possible reason is that the value of ρ is quite large in DCA and then it converges quickly to a "bad" critical point.

In datasets of over $10,000$ samples, DCA-Like is superior to DCA in all three criteria. In terms of running time, the number of iterations of DCA-Like is from 3.2 to 10 times less than DCA. Consequently, the computing time of DCA-Like is improved from 1.9 to 8.7 times comparing to DCA. Furthermore, DCA-Like gives lower objective value than DCA in six out of nine datasets (*gisette*, *usps*, *magic*, *letters*, *shuttle*, *miniboone*, and *covertype*), whereas the difference of two algorithms is neglectable for other three datasets.

In summary, DCA-Like improves the standard DCA on both quality of solution and rapidity.

• We study now the benefit of acceleration technique via the comparison between two pairs : ADCA versus DCA and ADCA-Like versus DCA-Like.

Not surprisingly, the number of iterations of ADCA-Like is always smaller (up to 3.6 times smaller) than that of DCA-Like. The gains in running time are also considerable, ADCA-Like is

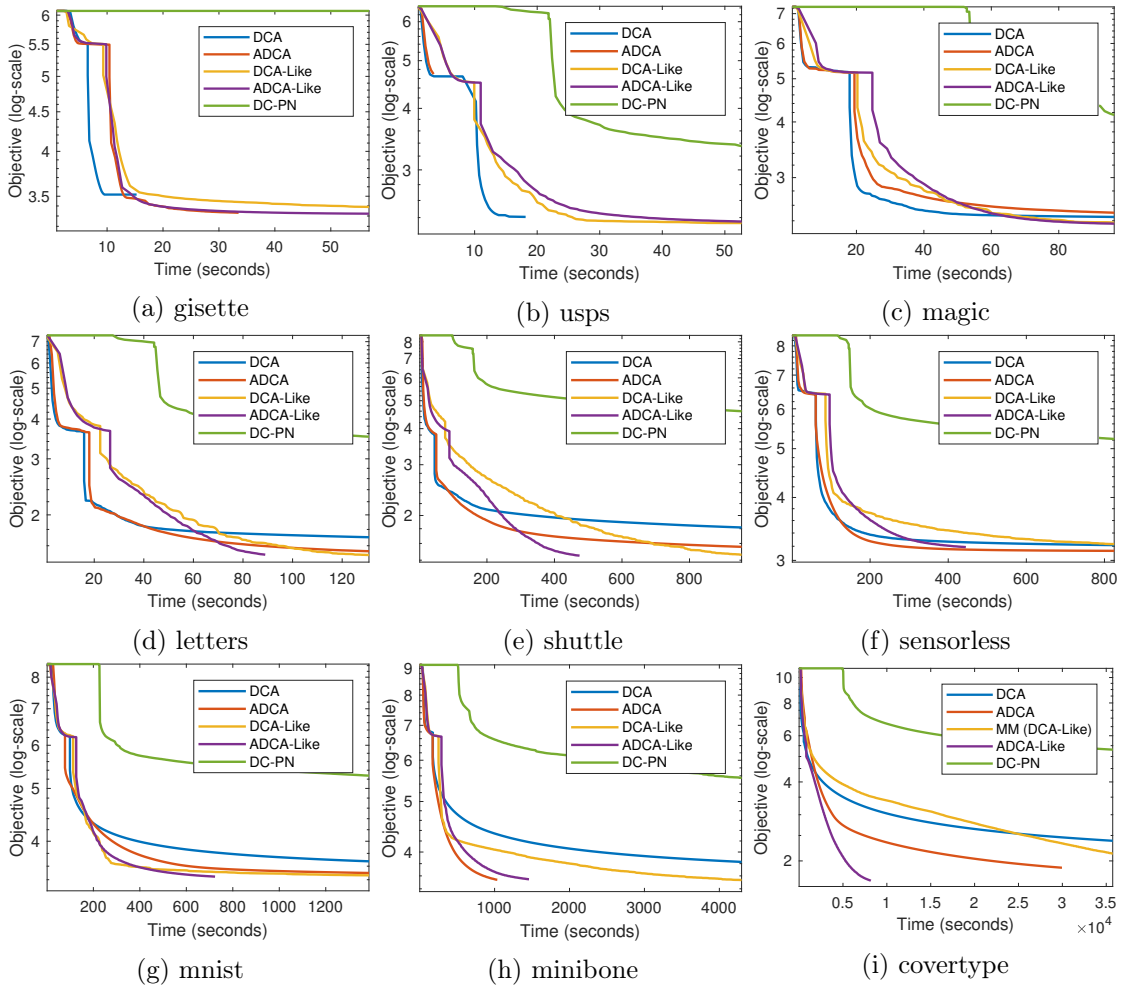


FIGURE 3.1 – Objective value versus running time (average of ten runs)

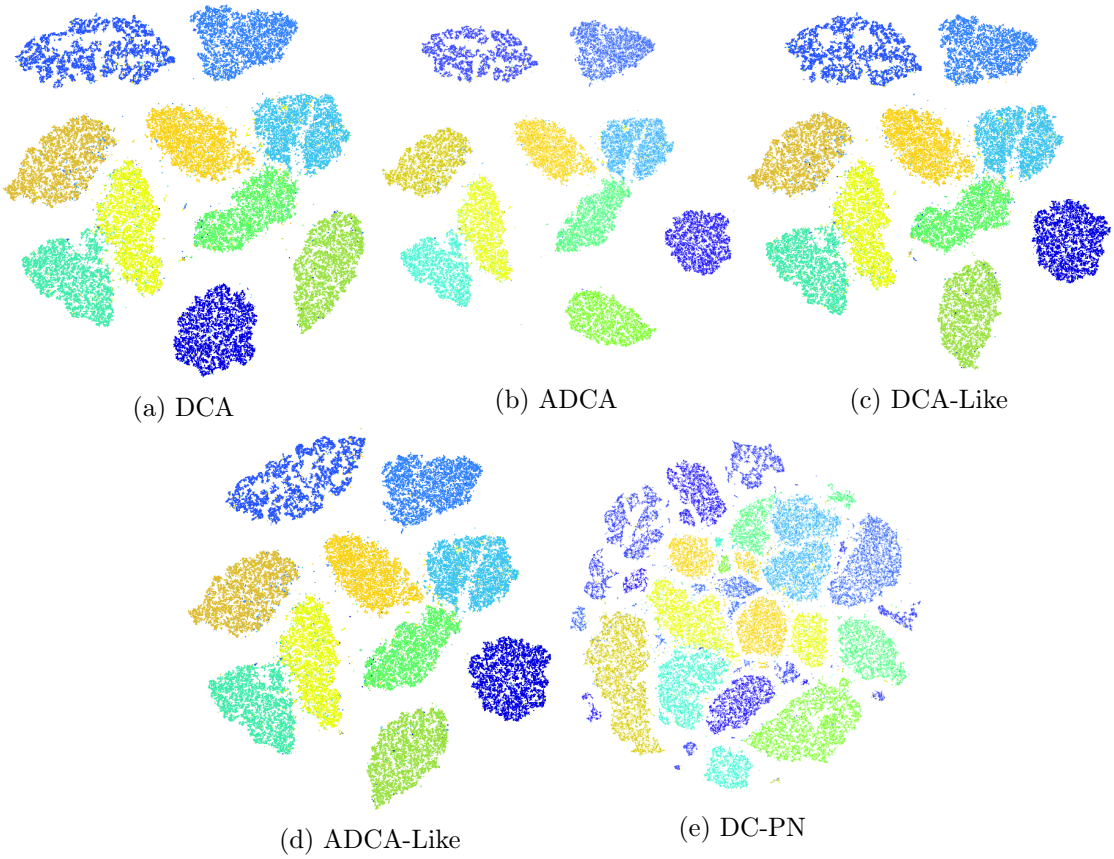
faster than DCA-Like from 1.5 to 5 times. On another hand, the two algorithms furnish the same objective value on three datasets (*ups*, *magic* and *letters*) while ADCA-Like gives better results than DCA-Like in the other six datasets. Thus, we can say that ADCA-Like further improves the performance of DCA-Like.

As for the comparison of DCA and ADCA, in most of the cases, ADCA is better than DCA. In five out of nine datasets (*letters*, *shuttle*, *sensorless*, *miniboone* and *covertype*), ADCA is superior to DCA in all three aspects. The ratio of gains in terms of running time is from 2.4 to 7.0 times. In the two datasets (*magic* and *mnist*), ADCA is faster, but is slightly worse in terms of objective value.

• Among the five comparative algorithms, ADCA-Like gives the best results. In terms of running time, ADCA-Like, followed by DCA-Like, is the fastest in seven over nine datasets. As for the objective value, ADCA-Like gives the best result in seven datasets, followed by DCA-Like in three datasets. Note that, the gains of ADCA-Like versus the second-best algorithm, in each comparative criterion, are noticeable. This illustrates the superior of ADCA-Like versus the other algorithms.

In Figure 3.1, we plot the objective value against the running time.

We observe that DCA performs thoroughly at the beginning but then it is left behind, while



(a) DCA (b) ADCA (c) DCA-Like

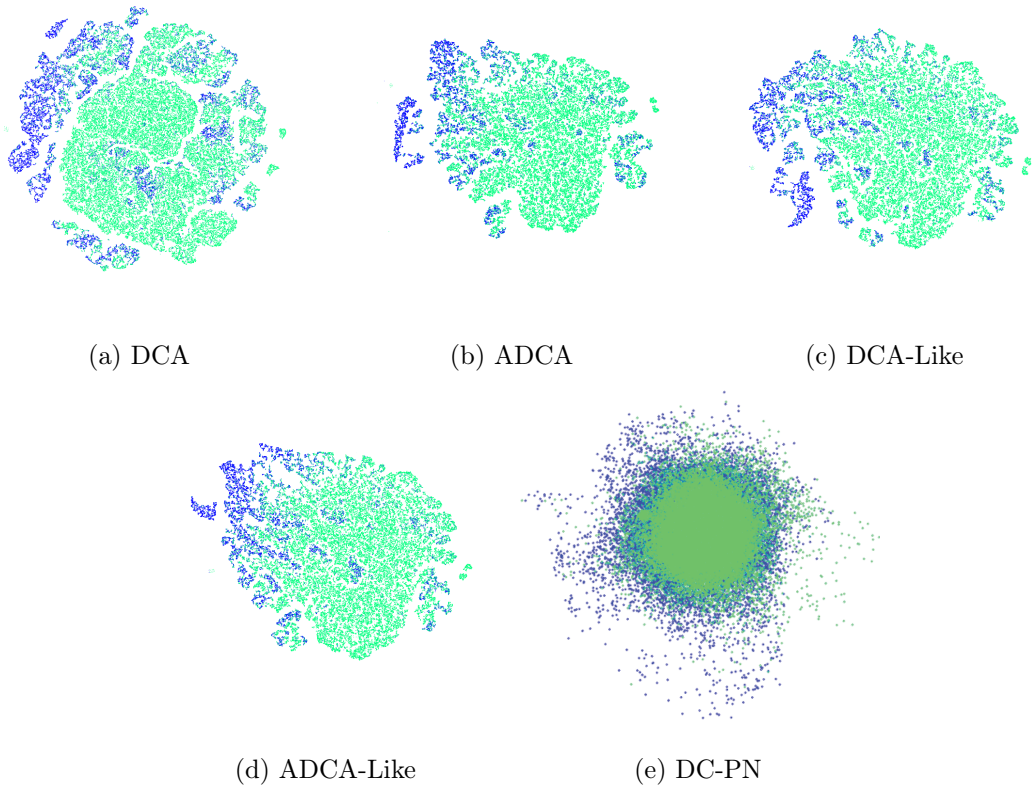
(d) ADCA-Like (e) DC-PN

FIGURE 3.2 – Visualization of embedding space on *mnist* dataset. Colors represent classes of data (0-9).

DCA-Like, ADCA and ADCA-Like improve swiftly over time. It is noticeable that, in the plot of *magic*, *letters*, *shuttle*, although ADCA-Like struggle at first iterations, it soon catches up and surpasses others algorithms to produce the best result while being the fastest. Also, for the biggest dataset (*covertype*), the effectiveness of accelerated algorithms is clearly demonstrated : ADCA-Like (reps. ADCA) surpass DCA-Like (reps. DCA) after a short amount of time.

Figure 3.2 visualizes the results on the well-known dataset *mnist*. This dataset consists of 70000 gray-scale 28×28 images over 10 classes of handwritten digits. *mnist* can be considered as the benchmark dataset for SNE-based algorithms, since they are able to capture both local and global structure of this dataset, especially in 2D embedding space. As we can see, in the embedding space, all four DCA-based algorithms managed to keep the structure of dataset of the original space, whereas DC-PN failed to maintain the structure. Four images of DCA-based algorithms in Figure 3.2 are quite similar since their objective values are fairly similar.

• We now study the effect of ρ_0 on the performance of DCA-Like. We run DCA-Like with different values of ρ_0 taken from the interval $\left[10^{-7}, \ldots, 1, 4\right]$. This experiment is performed on *letters* - an arbitrary chosen dataset. In Figure 3.4 (a) we report the objective value given by DCA-Like as ρ_0 varies. We observe that the best values of objective function are achieved when ρ_0 is in the vicinity of 10^{-7} , and the objective value increases quickly as $\rho_0 \geq 10^{-6}$. Figure 3.4 (b) shows the percentage of number of iterations in inner while loop in DCA-Like over the total number of iterations (blue curve) and the percentage of time spent on the inner while loop over the total



(a) DCA (b) ADCA (c) DCA-Like

(d) ADCA-Like (e) DC-PN

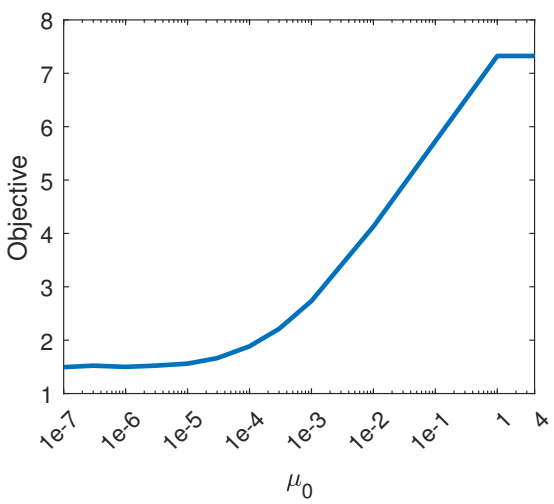
FIGURE 3.3 – Visualization of embedding space on *miniboone* dataset (2 classes). Colors represent classes of data.

running time (red curve). We observe that with the smallest value $\rho_0 = 10^{-7}$, DCA-Like needs a large number of iterations in the inner loop for finding a suitable value of ρ_k. In contrast, the number of inner iterations decreases drastically as ρ_0 increases and for $\rho_0 \geq 10^{-5}$ it is equal to 0. It means that for a too large value of ρ_0, the condition (3.44) is always verified and DCA-Like does not enjoy its advantages. It seems that $\rho_0 = 10^{-6}$ could be a good initial value of ρ_0 to realize the trade-off between the objective value and the running time.

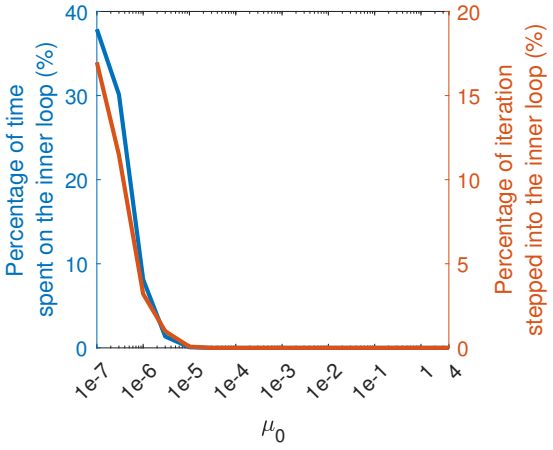
Overall, ADCA-Like gives the best results in all three comparison criteria while the results furnished by DC-PN are worst off. We can safely conclude that DCA-Like considerably improves DCA and ADCA-Like further enhances DCA-Like.

3.5 Application to group variables selection in multi-class logistic regression

Logistic regression, introduced by D. Cox in 1958 [58], is undoubtedly one of the most popular supervised learning methods. Logistic regression has been successfully applied in various real-life problems such as cancer detection [119], medical [33, 16, 218], social science [120], etc. Especially, logistic regression combined with feature selection has been proved to be suitable for high dimensional problems, for instance, document classification [80] and microarray classification [155, 119].



(a) Objective value given by DCA-Like as ρ_0 varies



(b) Percentage of number of iterations in DCA-Like where the inner while loop is activated and the time spent on inner while loop. Values of both lines are 0 for $\rho_0 \geq 10^{-5}$.

FIGURE 3.4 – Result of DCA-Like with different value of ρ_0 on dataset letters.

The multi-class logistic regression problem can be described as follows. Let $\{(x_i, y_i) : i = 1, ..., n\}$ be a training set with observation vectors $x_i \in \mathbb{R}^d$ and labels $y_i \in \{1, ..., Q\}$ where Q is the number of classes. Let W be the $d \times Q$ matrix whose columns are $W_{:,1}, ..., W_{:,Q}$ and $b = (b_1, ..., b_Q) \in \mathbb{R}^Q$. The couple $(W_{:,i}, b_i)$ forms the hyperplane $f_i := W_{:,i}^T x + b_i$ that separates the class i from the other classes.

In the multi-class logistic regression problem, the conditional probability $p(Y = y|X = x)$ that an instance x belongs to a class y is defined as

$$p(Y = y|X = x) = \frac{\exp(b_y + W_{:,y}^T x)}{\sum_{k=1}^{Q} \exp(b_k + W_{:,k}^T x)}. \tag{3.83}$$

We aim to find (W, b) for which the total probability of the training observations x_i belonging to its correct classes y_i is maximized. A natural way to estimate (W, b) is to minimize the negative log-likelihood function which is defined by

$$\mathcal{L}(W, b) := \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, W, b) \tag{3.84}$$

where $\ell(x_i, y_i, W, b) = -\log p(Y = y_i|X = x_i)$. Moreover, in high-dimensional settings, there are many irrelevant and/or redundant features. Hence, we need to select important features in order to reduce overfitting of the training data. A feature j is to be removed if and only if all components in the row j of W are zero. Therefore, it is reasonable to consider rows of W as groups. Denote by $W_{j,:}$ the j-th row of the matrix W. The $\ell_{2,0}$-norm of W, i.e., the number of non-zero rows of W, is defined by

$$\|W\|_{2,0} = |\{j \in \{1, ..., d\} : \|W_{j,:}\|_2 \neq 0\}|.$$

Hence, the $\ell_{2,0}$ regularized multi-class logistic regression problem is formulated as follows

$$\min_{W,b}\left\{\frac{1}{n}\sum_{i=1}^{n}\ell(x_i,y_i,W,b)+\lambda\|W\|_{2,0}\right\}. \tag{3.85}$$

In this work, we use a non-convex approximation of the $\ell_{2,0}$-norm based on the function Capped-ℓ_1 regularization which is defined by $\eta_\alpha(s) := \min\{1,\alpha s\}$. The Capped-$\ell_1$ is chosen since it has some interesting theoretical properties (c.f. Chapter 5 for more details on sparse inducing regularizations) and has shown its effectiveness in several problems, for instance sparse optimal scoring problem [149], sparse covariance matrix estimation problem [194]. Note that, our algorithms presented below are still available for other nonconvex regularizations such as concave exponentiel, SCAD, etc.

The corresponding approximate problem of (3.85) takes the form :

$$\min_{W,b}\left\{\frac{1}{n}\sum_{i=1}^{n}\ell(x_i,y_i,W,b)+\lambda\sum_{j=1}^{d}\eta_\alpha(\|W_{j,:}\|_2)\right\}. \tag{3.86}$$

It is easy to see that the objective function $F(w,b)$ of (3.86) can take the form of (3.1), where $f(W,b)=\frac{1}{n}\sum_{i=1}^{n}\ell(x_i,y_i,W,b)$, $g_j(t)=\lambda\min(\alpha\,|t|\,,1)$, $h_j(W)=\|W_{j,:}\|_q$. In deed, the function f is differential with L-Lipschitz continuous gradient, g_j are non-smooth convex functions, and h_j are concave increasing functions. As for the problem (3.1), we equivalently reformulate the problem (3.86) to the following problem in order to remove the composite function from the objective funtion

$$\min_{(W,b,t)}\left\{F(w,b,z)=\frac{1}{n}\sum_{i=1}^{n}\left[\ell(x_i,y_i,W,b)+\chi_\Omega(W,b,z)+\lambda\sum_{j=1}^{d}\eta_\alpha(z_j)\right]\right\}, \tag{3.87}$$

where $\Omega=\{(W,b,z)\in\mathbb{R}^{d\times Q}\times\mathbb{R}^Q\times\mathbb{R}^d : \|W_{j,:}\|_2\le z_j, j=1,...,d\}$.

The problem (3.87) is in the form of (3.3), thus DCA and DCA-Like can be developed to solve it. DCA applied to (3.87) consists in computing, at each iteration l, $(U^k,v^k,t^k)\in\partial H(W^k,b^k,z^k)$, and solving the convex sub-problem

$$\min_{(W,b,t)}\left\{\frac{\rho}{2}\|(W,b)\|^2+\chi_\Omega(W,b,z)-\langle U^k,W\rangle-\langle v^k,b\rangle-\langle t^k,z\rangle\right\}. \tag{3.88}$$

The computation of (U^k,v^k,t^k) is explicitly defined as follows.

$$(U^k,v^k,t^k)=\frac{1}{n}\sum_{i=1}^{n}(U_i^k,v_i^k,t_i^k),(U_i^k,v_i^k,t_i^k)\in\partial h_i(W^k,b^k,z^k),$$

$$\begin{aligned}
(U_i^k)_{:,l}&=\rho W_{:,l}^k-\left(p_k^k(x_i)-\delta_{ly_i}\right)x_i, l=1,...Q,\\
(v_i^k)_l&=\rho b_l^k-\left(p_l^k(x_i)-\delta_{ly_i}\right), l=1,...Q,\\
(t_i^k)_j&=-\lambda\alpha \text{ if } \alpha z_j^k\le 1, \text{ and 0 otherwise,} \qquad j=1,\dots,d,
\end{aligned} \tag{3.89}$$

with $p_l^k(x_i)=\exp(b_l^k+(W_{:,l}^k)^Tx_i)/(\sum_{h=1}^{Q}b_h^k+(W_{:,h}^k)^Tx_i))$, $\delta_{ly_i}=1$ if $l=y_i$ and 0 otherwise.

The convex sub-problem (3.88) can be solved as follows (note that $t_j^k \leq 0$ for $j = 1, \ldots, d$)

$$W^{k+1} = \arg\min_W \left\{ \frac{\rho}{2}\|W\|^2 + \sum_{j=1}^{d}(-z_j^k)\|W_{j,:}\|_2 - \langle U^k, W\rangle \right\}, \tag{3.90}$$

$$b^{k+1} = \arg\min_b \left\{ \frac{\rho}{2}\|b\|^2 - \langle v^k, b\rangle \right\} = \frac{1}{\rho}v^k, \tag{3.91}$$

$$z_j^{k+1} = \|W_{j,:}^{k+1}\|_2, j = 1, ..., d. \tag{3.92}$$

Since the problem (3.90) is separable in rows of W, solving it amounts to solving d independent sub-problems

$$W_{j,:}^{k+1} = \arg\min_{W_{j,:}} \left\{ \frac{\rho}{2}\|W_{j,:}\|^2 + (-t_j^k)\|W_{j,:}\|_2 - \langle U_{j,:}^k, W_{j,:}\rangle \right\} \tag{3.93}$$

$W_{j,:}^{k+1}$ can be explicitly computed as follows. From (3.93) we can write

$$W_{j,:}^{k+1} = \arg\min_{W_{j,:}} \left\{ \frac{1}{2}\|W_{j,:} - U_{j,:}^k/\rho\|^2 + (-t_j^k)\|W_{j,:}\|_2 \right\}.$$

$$W_{j,:}^{k+1} = \mathbf{prox}_{(-t_j^k)/\rho\|\cdot\|_2}\left(U_{j,:}^k/\rho\right),$$

where the proximal operator $\mathbf{prox}_f(\nu)$ is defined by

$$\mathbf{prox}_f(\nu) = \arg\min_t \left\{ \frac{1}{2}\|t - \nu\|^2 + f(t) \right\}.$$

The proximal operator of $(-t_j^k)/\rho\|\cdot\|_q$ can be efficiently computed [184]. The computation of $\mathbf{prox}_{(-t_j^k)/\rho\|\cdot\|_q}(\nu/\rho)$ is given by

$$\mathbf{prox}_{(-t_j^k)/\rho\|\cdot\|_q}(\nu/\rho) = \begin{cases} \left(1 - \frac{-t_j^k}{\|U_{j,:}^k\|_2}\right) U_{j,:}^k/\rho & \text{if } \|U_{j,:}^k\|_2 > -t_j^k \\ 0 & \text{if } \|U_{j,:}^k\|_2 \leq -t_j^k. \end{cases} \tag{3.94}$$

Hence, DCA for solving (3.87) is described in Algorithm 3.7.

Algorithm 3.7 DCA-$\ell_{2,0}$: DCA for solving (3.87)

1: **Initialization :** Choose $(W^0, b^0) \in \mathbb{R}^{d\times Q} \times \mathbb{R}^Q$, $\rho > L$ and $k \leftarrow 0$.
2: **repeat**
3: Compute (U^k, v^k, t^k) by (3.89).
4: Compute $(W^{k+1}, b^{k+1}, z^{k+1})$ according to (3.91), (3.92) and (3.94).
5: $k \leftarrow k + 1$.
6: **until** Stopping criterion.

In the DCA-Like scheme for solving (3.87), we observe that the while loop stops if the following inequality holds

$$U_{\rho_k}^{LR}((W, b, z)^{k+1}, (W, b, z)^k) \geq F((W, b, z)^{k+1}), \tag{3.95}$$

where $U_{\rho_l}^{LR}((W, b, z)^{k+1}, (W, b, z)^k) = F((W, b, z)^k) + \langle \nabla f((W, b, z)^k), (W, b, z)^{k+1} - (W, b, z)^k\rangle + \frac{\rho_k}{2}\|(W, b, z)^{k+1} - (W, b, z)^k\|^2 - \langle z^k, g(W^{k+1}) - g(W^k)\rangle$. Thus, DCA-Like for solving problem (3.87) is presented in Algorithm 3.8.

ADCA-Like for solving (3.87) is obtained by adding the steps (Step 3 and 11 of Algorithm 3.3) to Algorithm 3.8.

Algorithm 3.8 DCA-Like for solving (3.87)

Initialization : Choose $(W^0, b^0) \in \mathbb{R}^{d\times Q} \times \mathbb{R}^Q$, $\rho_0 > 0$ and $k \leftarrow 0$.
repeat
  Compute $(U, v, t)^k$ by (3.89).
  Set $\rho_l = \max\{\rho_0, \delta\rho_{l-1}\}$ if $l > 0$.
  Compute $(W, b, z)^{k+1}$ according to (3.91), (3.92) and (3.94).
  while $U^{LR}_{\rho_l}((W, b, z)^{k+1}, (W, b, z)^k) < F((W, b, z)^{k+1})$ **do**
    $\rho_l \leftarrow \eta\rho_k$.
    Compute $(W, b, z)^{k+1}$ according to (3.91), (3.92) and (3.94).
  end while
  $k \leftarrow k + 1$.
until Stopping criterion

3.5.1 Experiment setting

The comparisons are performed between 6 algorithms : DCA, DCA-Like, ADCA, ADCA-Like, nm-APG (non-monotone APG) [154] and DC-PN (DC Proximal Newton) [198]. Recall that nm-APG is an accelerated proximal gradient based method for minimizing $f(x) + r(x)$ where $f(x)$ is a differentiable function with L-Lipschitz gradient and $r(x)$ is a nonconvex function. nm-APG requires to compute the proximal mapping of the DC function η_α. However, this proximal mapping does not have a closed form. We therefore use DCA to compute the proximal mapping of η_α in nm-APG.

For DCA and ADCA, the Lipschitz constant L is estimated by an upper bound of Hessian matrix of logistic loss function which clearly too large. Hence, similarly as in the first application, we incorporate a ρ-updating procedure into DCA and ADCA. We set the initial value of ρ to $\rho_0 = 10^{-1}$.

We performed numerical experiments on several benchmark datasets taken from UCI and LIBSVM data repositories. The detailed information of used datasets is summarized in the first column of Table 3.3. n represents the number of points in dataset, d is the number of features, and Q is the number of classes. We randomly take 80% of the whole dataset as a training set and the rest is used as test set (20%).

In order to evaluate the performance of algorithms, we consider the following criteria : the classification accuracy (percentage of well-classified point on test set), the sparsity of obtained solution (percentage of selected features), the running time (measured in seconds) and the number of iterations.

3.5.2 Numerical results

The numerical results are reported in Table 3.3.

• *Comparison between DCA and DCA-Like.* Concerning the running time, DCA-Like is clearly faster than DCA, with the gains from 1.6 to 8.8 times. In term of classification accuracy and sparsity, DCA-Like is slightly better than DCA. In 4 cases (*satimage*, *mushroom*, *shuttle* and *sensorless*), DCA-Like achieves better classification accuracy (up to 1%) whereas choosing the same number of features. In the remaining 2 cases, the difference between them are neglectable.

• *Comparison between ADCA and ADCA-Like.* In terms of classification accuracy, ADCA-Like is slightly better than DCA-Like. In five over six datasets (except *mushroom* dataset), ADCA-Like shows the superior over DCA-Like in both classification accuracy and sparsity : ADCA-Like

TABLE 3.3 – Comparative results on the group variable selection in multiclass logistic regression problem. Bold values correspond to best results for each dataset. n, d and Q are the number of instances, dimensions and classes respectively.

Dataset	Algorithm	Accuracy (%)		Time (sec.)		Sparsity (%)	
		Mean	STD	Mean	STD	Mean	STD
dna	DCA	93.41	0.54	3.95	0.19	8.89	0.56
$n = 3186$	ADCA	93.30	0.18	**0.60**	0.02	8.89	0.56
$d = 180$	DCA-Like	93.20	0.74	0.98	0.13	**7.78**	0.96
$Q = 3$	ADCA-Like	**93.88**	0.83	0.92	0.05	**7.78**	2.00
	nm-APG	93.30	0.65	3.30	0.05	8.52	0.32
	DC-PN	93.56	0.16	1.20	0.13	29.07	2.51
satimage	DCA	84.25	0.20	4.02	1.15	**44.44**	2.78
$n = 6435$	ADCA	83.61	0.95	2.90	2.58	62.04	25.66
$d = 36$	DCA-Like	84.51	0.25	0.46	0.05	**44.44**	2.78
$Q = 6$	ADCA-Like	**84.67**	0.43	**0.28**	0.03	49.07	1.60
	nm-APG	81.84	0.70	1.98	0.05	100.00	0.00
	DC-PN	78.63	0.88	4.24	0.26	47.22	16.90
mushroom	DCA	98.44	0.30	5.71	0.08	4.50	0.00
$n = 8124$	ADCA	98.24	0.30	0.48	0.03	6.31	3.12
$d = 112$	DCA-Like	**99.41**	0.22	0.25	0.02	4.50	0.00
$Q = 2$	ADCA-Like	**99.41**	0.22	0.20	0.09	**3.60**	0.00
	nm-APG	98.44	0.30	12.07	0.47	4.50	0.00
	DC-PN	97.81	0.66	**0.17**	0.04	27.93	1.80
phishing	DCA	92.52	0.18	2.60	0.08	33.82	2.55
$n = 11055$	ADCA	92.64	0.07	0.38	0.01	28.92	0.85
$d = 68$	DCA-Like	92.36	0.32	0.19	0.01	27.94	1.47
$Q = 2$	ADCA-Like	**92.66**	0.15	**0.14**	0.01	**27.45**	0.85
	nm-APG	92.40	0.32	6.95	0.18	28.43	1.70
	DC-PN	92.66	0.11	0.38	0.03	61.76	5.30
shuttle	DCA	95.97	0.11	16.50	5.13	**59.26**	6.42
$n = 58000$	ADCA	96.11	0.26	15.93	4.43	**59.26**	6.42
$d = 9$	DCA-Like	96.06	0.05	1.81	0.20	**59.26**	6.42
$Q = 7$	ADCA-Like	**96.13**	0.08	**1.43**	0.21	**59.26**	6.42
	nm-APG	96.06	0.05	21.27	0.30	66.67	0.00
	DC-PN	92.56	0.61	3.04	0.73	92.59	6.42
sensorless	DCA	78.55	0.45	58.76	0.15	**12.50**	0.00
$n = 58509$	ADCA	79.03	0.44	12.47	0.27	**12.50**	0.00
$d = 54$	DCA-Like	79.51	0.41	55.10	1.32	**12.50**	0.00
$Q = 11$	ADCA-Like	**79.56**	0.36	36.02	0.79	**12.50**	0.00
	nm-APG	78.72	0.41	46.53	0.86	**12.50**	0.00
	DC-PN	77.64	1.20	**8.43**	1.11	78.47	12.73

yields higher classification accuracy while choosing a smaller subset of features.
• *Comparison between our algorithms and existing methods (nm-APG and DC-PN).* In term of classification accuracy, ADCA-Like produces the best results in all six datasets. As for the sparsity of solution, ADCA-Like also furnishes good results : the best result on four datasets and the second-best result on the other two datasets. It is worth to mention that in two datasets (*sensorless* and *shuttle*), while most algorithms choose the same percentage of features, ADCA-Like has the highest classification accuracy. We also observe that DC-PN often chooses the biggest subset of features while giving lower classification accuracy. In term of computing time, DCA-Like and ADCA-Like are arguably faster than the other algorithms, whereas nm-APG is the slowest. Overall, both ADCA-Like and ADCA achieve better results than nm-APG and DC-PN.

3.6 Application to the sparse binary logistic regression problem

Consider now the sparse binary logistic regression problem whose the description is given in Section 2.4 of Chapter 2. Recall that we have developed ADCA and ADCA to solve this problem in Chapter 2. Let us recall that the considered problem is written as

$$\min_{(w,b)} \left\{ F(w,b) = \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i(x_i^T w + b))) + \lambda r_{exp}(w) \right\}. \tag{3.96}$$

with $r_{exp}(w) = \sum_{i=1}^{d}(1-\exp(-\alpha|w_i|))$. In Section 2.4, we have shown that $f(w,b) = \frac{1}{n}\sum_{i=1}^{n}\log(1+\exp(-y_i(x_i^T w + b)))$ is a Lipschitz continuous gradient with a Lipschitz constant $L = \frac{1}{4n}\sum_{i=1}^{n}(\|x_i\|^2+1)$ and $r_{exp}(w)$ is a DC function. Thus, the problem (3.96) takes the form of (3.2).

In DCA-Like for solving (3.96), we observe that the while loop stops if the following inequality holds

$$H_{\rho_k}(w^{k+1},b^{k+1}) \geq H_{\rho_k}(w^{k+1},b^{k+1}),(w^k,b^k)), \tag{3.97}$$

where $H_{\rho_k}((w,b),(w^k,b^k)) := H_\rho(w^k,b^k) + \langle (u^k,v^k),(w,b)-(w^k,b^k)\rangle$.

Thus, DCA-Like for solving (3.96) is presented in Algorithm 3.9.

Algorithm 3.9 DCA-Like for solving (3.96)

1: **Initialization :** Choose an initial point (w^0,b^0), $0<\delta<1$, $\rho_0>0$ and $k \leftarrow 0$.
2: **repeat**
3: Compute $u^k = \rho w^k - \nabla_w f(w^k,b^k) + \xi$ and $v^k = \rho b^k - \nabla_b f(w^k,b^k)$ with ξ^k and ∇f defined in (2.23) and (2.25).
4: Set $\rho_k = \max\{\rho_0, \delta\rho_{k-1}\}$ if $k>0$.
5: Compute (w^{k+1},b^{k+1}) by (2.26).
6: **while** $H_{\rho_k}((w^{k+1},b^{k+1}) < H_{\rho_k}((w^{k+1},b^{k+1},(w^k,b^k))$ **do**
7: $\rho_k \leftarrow \eta\rho_k$.
8: Compute (w^{k+1},b^{k+1}) by (2.26).
9: **end while**
10: **until** Stopping criterion.
11: $k \leftarrow k+1$.

ADCA-Like for solving (3.96) is obtained by adding the steps (Step 3 and 11 of Algorithm 3.5) to Algorithm 3.9.

3.6.1 Experiment setting

Comparative algorithms. The comparison is performed between 6 algorithms :

— DCA,

— ADCA,

— DCA-Like,

— ADCA-Like,

— nmAPG [154] : nonmonotone accelerated proximal gradient method,

— inAPG [248] : inexact accelerated proximal gradient algorithm.

DCA and ADCA require to compute the Lipschitz constant L of function f. In our numerical experiments, L is estimated by a large value $L = \frac{1}{4n}\sum_{i=1}^{n}(\|x_i\|^2 + 1)$ as showed above.

For inAPG and nmAPG, one has to compute the proximal mapping of the DC function r_{exp}. Unfortunately, this proximal mapping do not have a closed form. We therefore use DCA for computing the proximal mapping of r_{exp} in inAPG and nmAPG. Furthermore, the step size in inAPG and nmAPG is computed by a backtracking line search initialized with Barzilai-Borwein rule in which the parameters are used as proposed by [154].

Setting. The numerical experiments were performed on several datasets taken from the well-known UCI and LibSVM data repertories. The detailed information of used datasets is given in the first column of Table. 3.4 where n_{train} (resp. n_{test}) represents the number of points in training set (resp. test set) while d stands for the number of features.

All experiments are performed on a PC Intel i7 CPU3770, 3.40 GHz of 8GB RAM and the codes have been written in MATLAB. We set $\rho_0 = 1/n$, $\eta = 2$ and $\delta = 0.5$ for ADCA and ADCA-Like. The parameter α for controlling the tightness of zero-norm approximation is set to 5. We set the trade-off parameter $\lambda = 10^{-4}$ on `rcv1` and $\lambda = 10^{-3}$ on the other datasets.

All the algorithms are terminated when the difference between objective function values of two consecutive iterations is smaller than $\epsilon = 10^{-5}$. We set the initial points to zero.

In order to evaluate the performance of algorithms, we consider the following three criteria : the classification accuracy (percentage of well classified point on test set), the sparsity of obtained solution and the running time (measured in seconds). The sparsity is computed as the percentage of selected variables.

3.6.2 Numerical results

3.6.2.1 Experiment 1

We report in Table 3.4 the running time in seconds, the classification accuracy on test set and the sparsity (percentage of selected features) of solutions. We also plot the curves of objective function values versus training time in Figure 3.5. Note that the parameter q of acceleration step in ADCA and ADCA-Like is set to 5 (c.f the experiment 2 in Section 3.6.2.2).

Table 3.4 – Comparative results. Bold values correspond to best results for each dataset

Dataset	Method	Time(s)	Accuracy(%)	Sparsity (%)
`madelon`	DCA	1.14	62.17	**0.4**
n_{train}=2000	ADCA	0.54	62.17	**0.4**
n_{test}=600	DCA-Like	0.42	62.17	**0.4**

d=500	ADCA-Like	**0.13**	**62.83**	**0.4**
	inAPG	0.86	62.17	**0.4**
	nmAPG	1.23	62.17	**0.4**
`gisette`	DCA	687.27	**96.9**	2.54
n_{train}=6000	ADCA	51.47	96	2
n_{test}=1000	DCA-Like	51.9	96	1.35
d=5000	ADCA-Like	**17.02**	96	1.37
	inAPG	172.74	95.9	1.43
	nmAPG	559.81	95.9	**1.33**
`w8a`	DCA	32.8	98.43	16
n_{train}=49 749	ADCA	**5.58**	98.51	15.67
n_{test}=14 951	DCA-Like	25.92	98.51	**15.33**
d=300	ADCA-Like	6.37	**98.52**	**15.33**
	inAPG	36.42	98.45	17
	nmAPG	54.81	98.4	19.33
`a9a`	DCA	108.3	84.88	47.97
n_{train}=32 561	ADCA	9.3	85.01	44.72
n_{test}=16 281	DCA-Like	6.06	84.98	32.52
d=123	ADCA-Like	**0.95**	**85.04**	**30.08**
	inAPG	6.38	84.98	32.52
	nmAPG	14.01	84.97	32.52
`rcv1`	DCA	113.03	91.8	0.87
n_{train}=20 242	ADCA	33.74	94.23	0.79
n_{test}=677 399	DCA-Like	62.03	94.29	0.71
d = 47236	ADCA-Like	**9.31**	**94.4**	**0.68**
	inAPG	39.65	91.1	0.85
	nmAPG	112.37	93.9	0.72
`real-sim`	DCA	33.25	94.49	2.71
n_{train}=57 847	ADCA	7.74	**94.5**	2.59
n_{test}=14 462	DCA-Like	21.71	**94.5**	2.63
d=20 958	ADCA-Like	**6.01**	**94.5**	**2.54**
	inAPG	33.49	94.42	2.62
	nmAPG	53.11	94.39	2.82
`epsilon`	DCA	18000	87.53	7.77
n_{train}=400 000	ADCA	1488	88.22	7.75
n_{test}=100 000	DCA-Like	15608.01	88.1	**6.9**
d=2000	ADCA-Like	**975.73**	**88.3**	7.15
	inAPG	18000	73.14	12.6
	nmAPG	NA	NA	NA
`url`	DCA	2857.08	97.27	0.004
n_{train}=1 916 904	ADCA	396.77	**97.47**	**0.0038**
n_{test}=479 226	DCA-Like	1891.05	97.3	**0.0038**
d=3 231 961	ADCA-Like	**234.75**	**97.47**	**0.0038**
	inAPG	7786.16	97.45	0.0045
	nmAPG	5213.68	97.45	0.0051

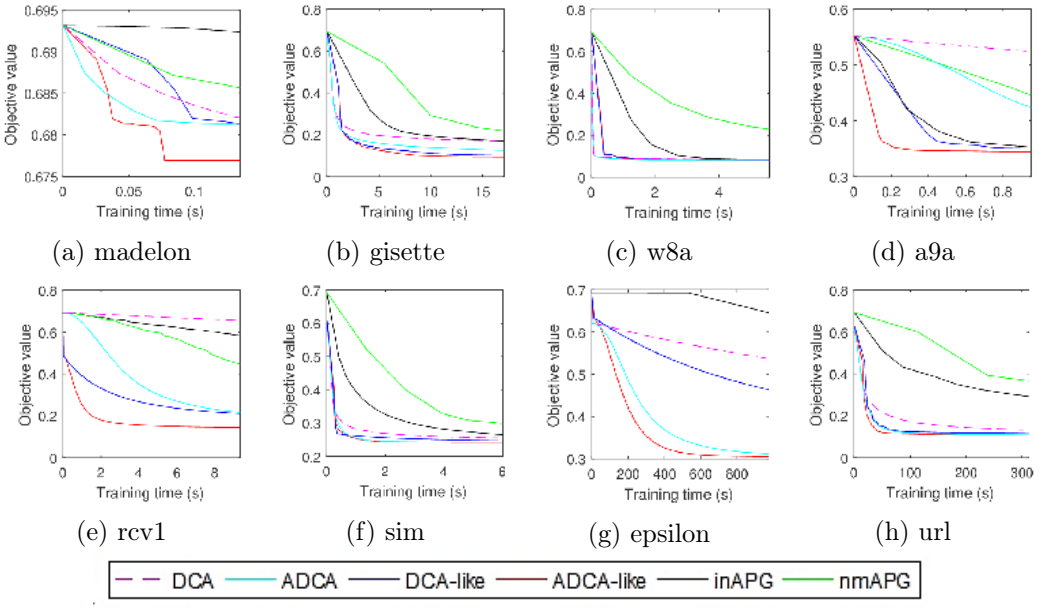


FIGURE 3.5 – Objective value versus running time (average of ten runs)

Concerning the classification accuracy, all 6 algorithms are comparable. Although ADCA-Like gives better accuracy than other algorithms on 6 out of 8 datasets, the gain is quite small. Overall, DCA-Like and ADCA-Like furnish better classification accuracy than DCA, except for the dataset gisette.

As for the sparsity of solution, we observe again that DCA-Like and ADCA-Like improve the result of DCA on 7 out of 8 datasets (all 6 algorithms give the same sparsity of solution on madelon). ADCA-Like suppresses more features than the other algorithms on 3 datasets (with an important gain of 17.89% on dataset a9a). On dataset w8a, DCA-like and ADCA-Like give the best result while all three variants of DCA give the same result on url. Overall, ADCA-Like seems to give better result than other five algorithms.

Concerning the rapidity, DCA-Like and ADCA-Like improve considerably the running time of DCA. ADCA is always faster than DCA with an important gain of 11.6 times on dataset a9a and 13.3 times on dataset gisette. DCA-Like also is faster than DCA with a gain up to 13.2 times (on gisette). Not surprisingly, ADCA-Like, which incorporates an acceleration technique into DCA-Like is the fastest algorithm. ADCA-Like is up to 40.4 times faster than DCA. We also observe from Figure 3.5 that the objective function of DCA-Like and ADCA-Like decreases drastically in few first iterations.

Overall, among the six compared algorithms, ADCA-Like is the fastest one while giving better or similar classification accuracy and sparsity of solution. Furthermore, DCA-Like and ADCA-Like improve DCA on all three criteria.

3.6.2.2 Experiment 2

In this experiment, we study the influence of parameter q in ADCA and ADCA-like. Recall that, in ADCA and ADCA-Like, the parameter q allows to control how often the algorithms use the extrapolated points z^k instead of the last iterate x^k in acceleration step.

We randomly choose a data set (madelon) and vary the value of q from 0 to 9. We report the results in the Table 3.5. The last column (#iter) presents the number of times (in percentage of the total number of iterations of ADCA-Like/ADCA) where the extrapolated points z^k is better than one of last q iterates.

TABLE 3.5 – Numerical results on madelon data set with different value of q

q	ADCA				ADCA-Like			
	Time (s)	Accuracy (%)	Sparsity (%)	#iter (%)	Time (s)	Accuracy (%)	Sparsity (%)	#iter (%)
0	0.78	62	0.4	90	0.23	62.17	0.4	90.9
1	0.65	62.17	0.4	91.3	0.19	63.34	0.4	92.3
2	0.63	62.17	0.4	96.6	0.16	62.35	0.4	97.2
3	0.62	62.17	0.4	99	0.16	62.83	0.4	97.3
4	0.62	67.17	0.4	99	0.15	62.83	0.4	98
5	0.54	67.17	0.4	100	0.13	62.83	0.4	100
6	0.54	67.17	0.4	100	0.13	62.83	0.4	100
7	0.54	67.17	0.4	100	0.13	62.83	0.4	100
8	0.54	67.17	0.4	100	0.13	62.83	0.4	100
9	0.54	67.17	0.4	100	0.13	62.83	0.4	100

We observe from the last column that the value of #iter logically increases as the value of q increases. With $q \geq 5$, #iter achieves 100%, i.e., the extrapolated point z^k is always used instead of the last iterate x^k and consequently decreases the running time of ADCA-Like/ADCA as we can observe from the second column (Times). As we can see, the the accuracy and sparity are the same for q from 5 to 10. Thus, we will use $q = 5$ for ADAC-Like and DCA-Like for our experiments.

3.7 Final remarks and Conclusion

We saw that the two classes of problems considered in this chapter are formulated or reformulated as DC programs (recall that (3.1) is equivalent to (3.3) which is a DC program on variable (x, z)). Therefore standard DCA and its acceleration version can be applied to solve these problems. The DC compositions of their objective function are essentially based on the one of f. As f is differentiable with L-Lipschitz continuous gradient, we use the natural DC decomposition $f = (1/2)\rho\|.\|^2 - ((1/2)\rho\|.\|^2 - f)$. with $\rho \geq L$. In practice, it is difficult (or even impossible) to determine the exact value of L, and one usually estimates L by a quite large number. However, a large value of L could lead to a bad convex approximation of F_1 and F_2, then DCA may converge rapidly to a *biased* critical point. In DCA-Like we do not need to evaluate L, as the algorithm starts with a small value of ρ and then updates ρ accordingly to get a convex majorization which may not be a upper bound of the objective function on the whole space (as in standard DCA) but rather only at the current solution.

The idea of DCA-Like is original and its advantage is double. Firstly, while standard DCA works with a convex majorization of the objective function on the whole space via an available DC decomposition, DCA-Like seeks a better convex approximation at the current solution

through a decomposition which is not necessarily DC. Secondly, and more importantly in several practical problems, DCA-Like works even when one can not highlight a DC decomposition. It turns out that, fortunately, despite these modifications we can prove that DCA-Like still enjoys the convergence properties of DCA.

While the two DCA-Like investigated to (3.3) and (3.2) are similar and enjoy the same convergence properties, the acceleration steps in the two algorithms 3.3 and 3.5 are different, by the way, these two ADCA-Like schemes do not have the same convergence properties. In fact, they are only the same if we take $q = 0$ in all iterations of Algorithm 3.5. Taking $q > 0$ allows the objective function to increase and consequently to escape from a potential bad local minimum.

The efficiency of DCA-Like and ADCA-Like was proved on three applications in machine learning : the sparse binary logistic regression, the group variables selection multi-class logistic regression and the t-distributed Stochastic Embedding (t-SNE) problem. The numerical showed that DCA-Like and ADCA-Like improve considerably the running time of standard DCA while giving similar or better solution. For instance, DCA-Like is faster than DCA with a gain up to 13.2 times while ADCA-Like is up to 40.4 times faster than DCA in the sparse binary logistic regression problem.

We are convinced that DCA-Like and the accelerated versions can be efficiently exploited for solving broader classes of nonconvex problems, and it is the purpose of our future works.

Chapitre 4

Stochastic DCA for minimizing a large sum of DC functions[1]

Abstract: We consider the problem of minimizing a large sum of DC functions which appears in several different areas, especially in stochastic optimization and machine learning. Two DCA based algorithms are proposed : Stochastic DCA and Inexact Stochastic DCA. We prove that the convergence to a critical point of both algorithms is guaranteed with probability one. Furthermore, we develop our Stochastic DCA for solving an important problem in multi-task learning, namely group variables selection in multi class logistic regression. The corresponding stochastic DCA is very inexpensive, all computations are explicit. Numerical experiments on several benchmark datasets and synthetic datasets illustrate the efficiency of our algorithms and their superiority over existing methods, with respect to classification accuracy, sparsity of solution as well as running time.

4.1 Introduction

We address the so called large sum of DC functions minimization problem which takes the form

$$\min_{x\in\mathbb{R}^d}\left\{F(x) := \frac{1}{n}\sum_{i=1}^{n} F_i(x)\right\}, \tag{4.1}$$

where F_i are DC functions, i.e., $F_i(x) = g_i(x) - h_i(x)$ with g_i and h_i being lower semi-continuous proper convex functions, and n is a very large integer number. Note that, the problem of minimizing F under a convex set Ω is also of the type (4.1), as the convex constraint $x \in \Omega$ can be incorporated into the objective function F via the indicator function χ_Ω on Ω. The problem (4.1) appears in several different contexts, especially in stochastic optimization and machine learning. Indeed, the large-sum structure arises naturally in empirical risk minimization in stochastic

1. The results presented in this chapter were published in :
   - H.A. Le Thi, H.M. Le, D.N Phan, B. Tran, Stochastic DCA for minimizing a large sum of DC functions with application to multi-class logistic regression, Neural Networks, 132 :220-231, 2020.
   - Stochastic DCA for the Large-sum of Non-convex Functions Problem and its Application to Group Variable Selection in Classification, Internationale Conference on Machine learning ICML, 3394-3403, 2017.
   - Stochastic DCA for Sparse Multiclass Logistic Regression, Advances in Intelligent Systems and Computing, Vol. 629, 1-12, 2017.

programming. For example, let us consider the expected loss minimization problem

$$\min_{x\in\Omega} \mathbb{E}[f(x,\xi)], \tag{4.2}$$

where f is a loss function of a real variable x and a random variable ξ. A standard approach for solving (4.3) is the sample average method [98] which approximates the problem (4.2) by

$$\min_{x\in\Omega} \frac{1}{n}\sum_{i=1}^{n} f(x,\xi_i), \tag{4.3}$$

where $\xi_1, ..., \xi_n$ are independent variables, identically distributed realizations of ξ. When the loss function f is DC, the problem (4.3) takes the form of (4.1) with $F_i(x) = f(x,\xi_i) + \chi_\Omega(x)$. Obviously, the larger n is, the better approximation will be. Hence, a good approximate model of the form (4.3) in average sample methods requires an extremely large number n.

Another similar example is the mean-variance minimization problem of the form

$$\min_{x\in\Omega}\{\mathbb{E}[f(x,\xi)] + \lambda VaR[f(x,\xi)], \tag{4.4}$$

where $f(x,\xi)$ is a loss function, ξ is a random vector and λ is a regularization parameter. It is proved in [238] that VaR is a DC function in the case finite scenarios and $f(x,\xi) = x^T\xi$. Hence, by using the sample average approximation method, the resulting problem of (4.4) takes the form of (4.1).

Furthermore, in the era of big data, the large-sum structure is one of the most popular forms encountered in practice to model big data-driven problems. The objective function is usually composed of a loss function (the data-fitting term) and a regularization term (to encourage some desired properties on the found solutions or to model constraints on x). For instance, let us mention an important problem in machine learning, the multi-task learning. Let T be the number of tasks. For the j-th task, the training set $\mathcal{D}_j$ consists of n_j labeled data points in the form of ordered pairs $(x_i^j, y_i^j), i = 1, ..., n_j$, with $x_i^j \in \mathbb{R}^d$ and its corresponding output $y_i^j \in \mathbb{R}$. Multi-task learning aims to estimate T predictive functions $f_\theta^j(x) : \mathbb{R}^d \to \mathbb{R}^m, j = 1, ..., T$, which fit well the data. The multi-task learning can be formulated as

$$\min_{\theta}\left\{\sum_{j=1}^{T}\sum_{i=1}^{n_j}\mathcal{L}(y_i^j, f_\theta^j(x_i^j)) + \lambda p(\theta)\right\}, \tag{4.5}$$

where $\mathcal{L}$ denotes the loss function, p is a regularization term and $\lambda > 0$ is a trade-off parameter. For a good learning process, T and n_j are, in general, very large numbers. Clearly, this problem takes the form of (4.1) when $\mathcal{L}$ and p are DC functions. We observe that numerous loss functions in machine learning (e.g. least square loss, squared hing loss, ramp loss, logistic loss, etc) are DC. On another hand, most of existing regularizations can be expressed as DC functions. For instance, in learning with sparsity problems involving the zero norm (which include, among of others, variable / group variable selection in classification, sparse regression, compressed sensing) all standard nonconvex regularizations studied in the literature are DC functions [147]. Moreover, in many applications dealing with big data, the number of both variables and samples are very large.

The problem (4.1) has a double difficulties due to the nonconvexity of F_i and the large value of n. Meanwhile, the sum structure of F enjoys an advantage : one can work on F_i instead of the whole function F. Since all F_i are DC functions, F is DC too, and therefore (4.1) is a standard

DC program on which DCA can be applied. Nevertheless, although DCA is a scalable approach in nonconvex optimization, the large-sum structure is difficult to be handled by deterministic algorithms. Therefore, we are suggested to investigate stochastic DCA for solving (4.1).

Related works on stochastic algorithms.

To the best of our knowledge, although several methods have been developed for solving different special cases of (4.1), our approach [133], named stochastic DCA, is the first in the literature solving the general nonsmooth nonconvex problem (4.1).

In the convex setting, the literature on stochastic optimization is vast. A well-known approach is the stochastic gradient (SG) which was first introduced in [205] for convex functions (in fact, it is initially introduced in form of a Markov chain). The SG method chooses $i_l \in \{1, ..., n\}$ randomly and takes the update

$$x^{l+1} = x^l - \alpha_l \nabla f(x^l, \xi_{i_l}), \tag{4.6}$$

where α_l is the step size and $\nabla f(x^l, \xi_{i_l})$ is a stochastic gradient. This simple scheme has inspired a large number of works, here we name a few. In [25, 24], the authors proposed the proximal stochastic subgradient methods (also referred as incremental proximal methods) for solving (4.3) in convex case, i.e., Ω is a closed convex set and $f(\cdot, \xi_i)$ are convex functions. The computational cost per iteration of these basic SG methods is very cheap, however, due to the variance introduced by random sampling, their convergence rates are slower than the "full" gradient methods. Hence, some SG methods for solving (4.3) in unconstrained differentiable convex case use either the average of the stored past gradients or a multi-stage scheme to progressively reduce the variance of the stochastic gradient (see e.g [210, 213, 60, 61, 117, 173]). In [8, 214], the convexity of each function $f(\cdot, \xi_i)$ is relaxed (say, it is not necessarily convex), but the overall objective function is still required to be convex.

In the nonconvex setting, each function $f(\cdot, \xi_i)$ is usually required to be L-smooth (or slightly milder : the objective is L-average-smooth)[6, 7, 71, 174, 175, 259, 153]. In [31], the author analyzed the SG scheme for differentiable objective function under the condition that is closely related to convexity, then extended the analysis to the nonconvex case where the higher order of smoothness is imposed. In the practical perspective, the work [152] presents various tricks to use SG in training neural networks. To meet the practical demands in modelling, some works add a (nonsmooth) convex function (to capture the regularizer that usually appears in machine learning) into the L-smooth large sum (conventionally called the "composite" setting) [112, 233, 188]. Note that, even with the composite setting, the literature remains really limited [188].

The problem considered (4.1) is more general : the objective is a large sum of DC functions whose both DC components can be nonsmooth. For nonsmooth DC large-sum problems, the current body of research is even more limited. There were some recent works that coupled stochastic algorithms and DCA. The first stochastic DCA was proposed in [131] for a large-sum of (nonconvex) L-smooth functions with the DC regularizer $\|x\|_{2,0}$. The work in this chapter extends [131] to the general problem (4.1) where the L-smooth assumption is relaxed. Another approach was studied in [244] where the data-fitting term is large-sum DC and the regularizer is nonconvex, nonsmooth whose proximal operator can be efficiently computed. By using the Moreau envelope (which is a DC function) of the regularizer, the authors approximated the original problem by a DC program and proposed DCA schemes for solving it. Their algorithms can be regarded as the standard DCA in which the (large-sum) convex subproblems are solved by stochastic algorithms (e.g., Adagrad [69], SVRG [242]) up to a certain level of accuracy.

Our contributions.

To tackle the difficulty due to the large value of n, we first propose Stochastic DCA, by taking the advantage of the sum structure of F, for solving (4.1). The basic idea of stochastic DCA is to update, at each iteration, the minorant of only some randomly chosen h_i while keeping the minorant of the other h_i. Hence the main advantage of the stochastic DCA versus standard DCA is the computational reduction in the step of computing a subgradient of H. Meanwhile, the convex subproblem is the same in both standard DCA and stochastic DCA. The first work in this direction was published in the conference paper [131] where we only considered a machine learning problem which is a special case of (4.1), namely

$$\min_x \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \lambda\|x\|_{2,0},$$

where f_i are L-Lipschitz functions. We rigorously studied the convergence properties of this stochastic DCA and proved that its convergence is guaranteed with probability one. In this chapter, we present our approach for the general model (4.1), and the same convergence properties of stochastic DCA is proved.

Secondly, to deal with the computational aspect in large-scale setting, we further propose an inexact stochastic DCA version in which both subgradient of H and optimal solution of the resulting convex program are approximately computed. We show that the convergence properties of stochastic DCA are still valid for the inexact stochastic DCA.

Thirdly, we show how to develop the proposed stochastic DCA for the group variables selection in multi-class logistic regression, a very important problem in machine learning which takes the form (1). We consider three nonconvex regularizations $\ell_{q,0}$ ($q \in \{1, 2, \infty\}$) with two approximation functions of $\ell_{q,0}$, and develop 12 algorithms (6 versions of standard DCA and 6 versions of stochastic DCA) for this problem. For the first time, a careful study is proposed in the literature. Thanks to a suitable DC decomposition, all versions of our proposed algorithms are explicit, i.e. all calculations are explicitly defined and no supplement solver is needed. Numerical experiments on very large synthetic and real-world datasets show that our approach is much more efficient, in both quality and rapidity, than four related methods.

The remainder of the chapter is organized as follows. Solution methods based on stochastic DCA for solving (4.1) are developed in Section 4.2 while the stochastic DCA for the group variables selection in multi-class logistic regression is presented in Section 4.3.

4.2 Stochastic DCA for minimizing a large sum of DC functions

Now, let us introduce a stochastic version of DCA, named SDCA, for solving (4.1). A natural DC formulation of the problem (4.1) is

$$\min\left\{F(x) = G(x) - H(x) : x \in \mathbb{R}^d\right\}, \tag{4.7}$$

where

$$G(x) = \frac{1}{n}\sum_{i=1}^{n} g_i(x) \text{ and } H(x) = \frac{1}{n}\sum_{i=1}^{n} h_i(x).$$

According to the generic DCA scheme, DCA for solving the problem (4.7) consists of computing, at each iteration l, a subgradient $v^l \in \partial H(x^l)$ and solving the convex subproblem of the form

$$\min\left\{G(x) - \langle v^l, x\rangle : x \in \mathbb{R}^d\right\}. \tag{4.8}$$

As $H = \frac{1}{n}\sum_{i=1}^{n} h_i$, the computation of subgradients of H requires the one of all functions h_i. This may be expensive when n is very large. The main idea of SDCA is to update, at each iteration, the computation of subgradient of only some randomly chosen h_i while keeping the one of the other h_i.

SDCA for solving the problem (4.7) is described in Algorithm 4.1 below.

Algorithm 4.1 SDCA for solving the problem (4.1)

1: **Initialization :** Choose $x^0 \in \text{dom } \partial H$, $s_0 = \{1, ..., n\}$, and $l \leftarrow 0$.
2: **repeat**
3: Compute $v_i^l \in \partial h_i(x^l)$ if $i \in s_l$ and keep $v_i^l = v_i^{l-1}$ if $i \notin s_l$, $l > 0$. Set $v^l = \frac{1}{n}\sum_{i=1}^{n} v_i^l$.
4: Compute x^{l+1} by solving the convex problem (4.8).
5: Set $l \leftarrow l + 1$ and randomly choose a small subset $s_l \subset \{1, ..., n\}$.
6: **until** Stopping criterion

Remark 4.1 *SDCA differs from DCA only at the step* 3. *Instead of computing* ∂H *(by computing* ∂h_i *for all* $i = 1, \ldots, n$*) as in DCA, SDCA updates only a small number of* ∂h_i*. Clearly, this technique reduces considerably the running time of DCA, and, intuitively, SDCA should be much faster than DCA, in particular when the computation of* ∂h_i *is expensive. Naturally, such a technique may reduce the quality of solution, and the crucial matter is the convergence properties of SDCA : do we get a critical point as in DCA ?*

The following theorem shows that the convergence properties of SDCA are guaranteed with probability one.

Theorem 4.1 *Assume that* $\alpha^* = \inf F(x) > -\infty$*, and* $|s_l| = b$ *for all* $l > 0$*. Let* $\{x^l\}$ *be a sequence generated by SDCA , the following statements hold.*

a) $\{F(x^l)\}$ *is the almost sure convergent sequence.*

b) If $\min_i \rho(h_i) > 0$*, then* $\sum_{l=1}^{\infty} \|x^l - x^{l-1}\|^2 < +\infty$ *and* $\lim_{l\to\infty} \|x^l - x^{l-1}\| = 0$*, almost surely.*

c) If $\min_i \rho(h_i) > 0$*, then every limit point of* $\{x^l\}$ *is a critical point of* F *with probability one.*

Proof 4.1 *a) Let* x_i^0 *be the copies of* x^0*. We set* $x_i^{l+1} = x^{l+1}$ *for all* $i \in s_{l+1}$ *and* $x_j^{l+1} = x_j^l$ *for* $j \notin s_{l+1}$*. We then have* $v_i^l \in \partial h_i(x_i^l)$ *for* $i = 1, ..., n$*. Let* T_i^l *be the function given by*

$$T_i^l(x) = g_i(x) - h_i(x_i^l) - \left\langle x - x_i^l, v_i^l \right\rangle.$$

It follows from $v_i^l \in \partial h_i(x_i^l)$ *that*

$$h_i(x) \geq h_i(x_i^l) + \left\langle x - x_i^l, v_i^l \right\rangle.$$

That implies $T_i^l(x) \geq F_i(x) \geq F_i(x)$ *for all* $l \geq 0$*,* $i = 1, ..., n$*. We also observe that* x^{l+1} *is a solution to the following convex problem*

$$\min_x T^l(x) := \frac{1}{n}\sum_{i=1}^{n} T_i^l(x), \tag{4.9}$$

Therefore

$$\begin{aligned} T^l(x^{l+1}) \leq T^l(x^l) &= T^{l-1}(x^l) + \frac{1}{n}\sum_{i\in s_l}[T_i^l(x^l) - T_i^{l-1}(x^l)] \\ &= T^{l-1}(x^l) + \frac{1}{n}\sum_{i\in s_l}[F_i(x^l) + 2\epsilon^l - T_i^{l-1}(x^l)], \end{aligned} \tag{4.10}$$

where the second equality follows from $T_i^l(x^l) = F_i(x^l)$ *for all* $i \in s_l$. *Let* $\mathcal{F}_l$ *denote the* σ*-algebra generated by the entire history of SDCA up to the iteration* l, *i.e.,* $\mathcal{F}_0 = \sigma(x^0)$ *and* $\mathcal{F}_l = \sigma(x^0, ..., x^l, s_0, ..., s_{l-1})$ *for all* $l \geq 1$. *By taking the expectation of the inequality* (4.10) *conditioned on* $\mathcal{F}_l$, *we have*

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b}{n}\left[T^{l-1}(x^l) - F(x^l)\right].$$

By applying the supermartingale convergence theorem [172, 23] to the nonnegative sequences $\{T^{l-1}(x^l) - \alpha^*\}, \{\frac{b}{n}[T^{l-1}(x^l) - F(x^l)]\}$ *and* $\{0\}$, *we conclude that the sequence* $\{T^{l-1}(x^l, y^l) - \alpha^*\}$ *converges to* $T^* - \alpha^*$ *and*

$$\sum_{l=1}^{\infty}\left[T^{l-1}(x^l) - F(x^l)\right] < \infty, \tag{4.11}$$

with probability 1. *Therefore* $\{F(x^l)\}$ *converges almost surely to* T^*.

b) By $v_i^{l-1} \in \partial h_i(x_i^{l-1})$, *we have*

$$h_i(x) \geq h_i(x_i^{l-1}) + \langle x - x_i^{l-1}, v_i^{l-1}\rangle + \frac{\rho(h_i)}{2}\|x - x_i^{l-1}\|^2, \ \forall x \in \mathbb{R}^d.$$

This implies

$$F_i(x) \leq T_i^{l-1}(x) - \frac{\rho(h_i)}{2}\|x - x_i^{l-1}\|^2. \tag{4.12}$$

From (4.10) *and* (4.12) *with* $x = x^l$, *we have*

$$T^l(x^{l+1}) \leq T^{l-1}(x^l) - \frac{1}{n}\sum_{i\in s_l}\frac{\rho(h_i)}{2}\|x - x_i^{l-1}\|^2. \tag{4.13}$$

Taking the expectation of the inequality (4.13) *conditioned on* $\mathcal{F}_l$, *we obtain*

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b}{4n^2}\sum_{i=1}^{n}\rho(h_i)\|x^l - x_i^{l-1}\|^2 + \left(\frac{2b}{n} + 1\right)\epsilon^l.$$

Combining this and $\rho = \min_{i=1,...,n}\rho(h_i) > 0$ *gives us*

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b\rho}{2n^2}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2.$$

Applying the supermartingale convergence theorem to the nonnegative sequences $\{T^{l-1}(x^l) - \alpha^*\}, \{\frac{b\rho}{2n^2}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2\}$ *and* $\{0\}$, *we get*

$$\sum_{l=1}^{\infty}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2 < \infty,$$

with probability 1. *In particular, for* $i = 1, ..., n$, *we have*

$$\sum_{l=1}^{\infty} \|x^l - x_i^{l-1}\|^2 < \infty, \tag{4.14}$$

and hence $\lim_{l\to\infty} \|x^l - x_i^{l-1}\| = 0$ *almost surely.*

c) Assume that there exists a sub-sequence $\{x^{l_k}\}$ *of* $\{x^l\}$ *such that* $x^{l_k} \to x^*$ *almost surely. From* (4.14), *we have* $\|x^{l_k+1} - x_i^{l_k}\| \to 0$ *almost surely. Therefore, by the finite convexity of* h_i, *without loss of generality, we can suppose that the sub-sequence* $v_i^{l_k}$ *tends to* v_i^* *almost surely. Since* $v_i^{l_k} \in \partial h_i(x_i^{l_k})$ *and by the closed property of the subdifferential mapping* ∂h_i, *we have* $v_i^* \in \partial h_i(x^*)$. *As* x^{l_k+1} *is a solution of the problem* $\min_x T^{l_k}(x)$, *we obtain*

$$0 \in \partial T^{l_k}(x^{l_k+1}). \tag{4.15}$$

This is equivalent to

$$0 \in \partial \frac{1}{n}\sum_{i=1}^{n} g_i(x^{l_k+1}) - \frac{1}{n}\sum_{i=1}^{n} v_i^{l_k} = \partial G(x^{l_k+1}) - \frac{1}{n}\sum_{i=1}^{n} v_i^{l_k}. \tag{4.16}$$

Hence, $\frac{1}{n}\sum_{i=1}^{n} v_i^{l_k} \in \partial G(x^{l_k+1})$. *By the closedness property of the subdifferential mapping* ∂G, *we obtain* $v^* = \frac{1}{n}\sum_{i=1}^{n} v_i^* \in \partial G(x^*)$ *with probability one. Therefore,*

$$v^* \in \partial G(x^*) \cap \partial H(x^*), \tag{4.17}$$

with probability 1. *This implies that* x^* *is a critical point of* F *with probability* 1 *and the proof is then complete.* ■

4.2.1 Inexact Stochastic DCA

The SDCA scheme requires the exact computations of v_i^l and x^{l+1}. Observing that, for standard DCA these computations are not necessarily exact ([144]), we are suggested to introduce an inexact version of SDCA. This could be useful when the exact computations of v_i^l and x^{l+1} are expensive. The inexact version of SDCA computes ϵ-subgradients $v_i^l \in \partial_{\epsilon^l} h_i(x^l)$ and an ϵ^l-solution x^{l+1} of the convex problem (4.8) instead of exactly computing. The inexact version of SDCA, named ISDCA, is described as follows.

Algorithm 4.2 ISDCA for solving the problem (4.1)

1: **Initialization :** Choose $x^0 \in \text{dom } \partial H$, $s_0 = \{1, ..., n\}$, $\epsilon^0 \geq 0$ and $l \leftarrow 0$.
2: **repeat**
3: Compute $v_i^l \in \partial_{\epsilon^l} h_i(x^l)$ if $i \in s_l$ and keep $v_i^l = v_i^{l-1}$ if $i \notin s_l$, $l > 0$. Set $v^l = \frac{1}{n}\sum_{i=1}^{n} v_i^l$.
4: Compute an ϵ^l-solution x^{l+1} of the convex problem (4.8).
5: Set $l \leftarrow l + 1$, randomly choose a small subset $s_l \subset \{1, ..., n\}$, and update $\epsilon^l \geq 0$.
6: **until** Stopping criterion.

Remark 4.2 *ISDCA should be less expensive than SDCA when the computation of* ∂h_i *and/or the solution of convex subproblems* (4.8) *is very hard and expensive. In such cases ISDCA could be an effective algorithm. If the computation of all* ∂h_i *as well as the solution of convex subproblems* (4.8) *are defined in explicit forms, then this version ISDCA is useless.*

Under an assumption that $\sum_{l=0}^{\infty} \epsilon^l < +\infty$, the ISDCA has the same convergence properties as SDCA, which are stated in the following theorem.

Theorem 4.2 *Assume that $\alpha^* = \inf F(x) > -\infty$, and $|s_l| = b$ for all $l > 0$. Let $\{x^l\}$ be a sequence generated by ISDCA with respect to a nonnegative sequence $\{\epsilon^l\}$ such that $\sum_{l=0}^{\infty} \epsilon^l < +\infty$ almost surely. The following statements hold.*

a) $\{F(x^l)\}$ is the almost sure convergent sequence.

b) If $\min_i \rho(h_i) > 0$, then $\sum_{l=1}^{\infty} \|x^l - x^{l-1}\|^2 < +\infty$ and $\lim_{l\to\infty} \|x^l - x^{l-1}\| = 0$, almost surely.

c) If $\min_i \rho(h_i) > 0$, then every limit point of $\{x^l\}$ is a critical point of F with probability one.

To prove Theorem 4.2, we will use the following lemma.

Lemma 4.1 *Let $f : \mathbb{R}^d \to \mathbb{R} \cap \{+\infty\}$ be a ρ-convex function. For any $\epsilon \geq 0$ and any $v \in \partial_\epsilon f(x)$ with $x \in dom\ f$, we have*

$$2\epsilon + f(y) \geq f(x) + \langle v, y - x\rangle + \frac{\rho}{4}\|y - x\|^2,\ \forall y \in \mathbb{R}^d.$$

Proof 4.2 *Since $v \in \partial_\epsilon f(x)$, we have*

$$\epsilon + f(z) \geq f(x) + \langle v, z - x\rangle,\ \forall z \in \mathbb{R}^d.$$

Replacing z with $x + t(y - x)$ in this inequality gives that

$$\epsilon + f(x + t(y - x)) \geq f(x) + t\langle v, y - x\rangle,\ \forall y \in \mathbb{R}^d.$$

It follows from the ρ-convexity of f that for $y \in \mathbb{R}^d$ and $t \in (0, 1)$,

$$tf(y) + (1 - t)f(x) \geq f(x + t(y - x)) + \frac{\rho t(1 - t)}{2}\|y - x\|^2.$$

Summing the two above inequalities gives us

$$\epsilon + tf(y) \geq tf(x) + t\langle v, y - x\rangle + \frac{\rho t(1 - t)}{2}\|y - x\|^2.$$

Thus, the conclusion follows from this inequality with $t = 1/2$.

Proof 4.3 (Proof of Theorem 4.2) *a) Let x_i^0 be the copies of x^0. We set $x_i^{l+1} = x^{l+1}$ for all $i \in s_{l+1}$ and $x_j^{l+1} = x_j^l$ for $j \notin s_{l+1}$. Set $\epsilon_i^0 = \epsilon^0$ and $\epsilon_i^{l+1} = \epsilon^{l+1}$ if $i \in s_{l+1}$, ϵ_i^l otherwise. We then have $v_i^l \in \partial_{\epsilon_i^l} h_i(x_i^l)$ for $i = 1, ..., n$. Let T_i^l be the function given by*

$$T_i^l(x) = g_i(x) - h_i(x_i^l) - \left\langle x - x_i^l, v_i^l \right\rangle + 2\epsilon_i^l.$$

It follows from $v_i^l \in \partial_{\epsilon_i^l} h_i(x_i^l)$ that

$$\epsilon_i^l + h_i(x) \geq h_i(x_i^l) + \left\langle x - x_i^l, v_i^l \right\rangle.$$

This implies $T_i^l(x) \geq F_i(x) + \epsilon_i^l \geq F_i(x)$ for all $l \geq 0$, $i = 1, ..., n$. We also observe that x^{l+1} is an ϵ^l-solution of the following convex problem

$$\min_x T^l(x) := \frac{1}{n}\sum_{i=1}^{n} T_i^l(x) \tag{4.18}$$

Therefore

$$\begin{aligned} T^l(x^{l+1}) \leq T^l(x^l) + \epsilon^l &= T^{l-1}(x^l) + \frac{1}{n}\sum_{i\in s_l}[T_i^l(x^l) - T_i^{l-1}(x^l)] + \epsilon^l \\ &= T^{l-1}(x^l) + \frac{1}{n}\sum_{i\in s_l}[F_i(x^l) + 2\epsilon^l - T_i^{l-1}(x^l)] + \epsilon^l, \end{aligned} \tag{4.19}$$

where the second equality follows from $T_i^l(x^l) = F_i(x^l) + 2\epsilon^l$ *for all* $i \in s_l$. *Let* $\mathcal{F}_l$ *denote the* σ*-algebra generated by the entire history of ISDCA up to the iteration* l, *i.e.,* $\mathcal{F}_0 = \sigma(x^0, \epsilon^0)$ *and* $\mathcal{F}_l = \sigma(x^0, ..., x^l, \epsilon^0, ..., \epsilon^l, s_0, ..., s_{l-1})$ *for all* $l \geq 1$. *By taking the expectation of the inequality* (4.19) *conditioned on* $\mathcal{F}_l$, *we have*

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b}{n}\left[T^{l-1}(x^l) - F(x^l)\right] + \left(\frac{2b}{n} + 1\right)\epsilon^l.$$

Since $\sum_{l=0}^{\infty}\epsilon_i^l < +\infty$ *with probability* 1, *by applying the supermartingale convergence theorem [172, 23] to the nonnegative sequences* $\{T^{l-1}(x^l) - \alpha^*\}, \{\frac{b}{n}[T^{l-1}(x^l) - F(x^l)]\}$ *and* $\{(\frac{2b}{n}+1)\epsilon^l\}$, *we conclude that the sequence* $\{T^{l-1}(x^l, y^l) - \alpha^*\}$ *converges to* $T^* - \alpha^*$ *and*

$$\sum_{l=1}^{\infty}\left[T^{l-1}(x^l) - F(x^l)\right] < \infty, \tag{4.20}$$

with probability 1. *Therefore* $\{F(x^l)\}$ *converges almost surely to* T^*.

b) By $v_i^{l-1} \in \partial_{\epsilon_i^{l-1}} h_i(x_i^{l-1})$ *and Lemma 4.1, we have*

$$2\epsilon_i^{l-1} + h_i(x) \geq h_i(x_i^{l-1}) + \langle x - x_i^{l-1}, v_i^{l-1}\rangle + \frac{\rho(h_i)}{4}\|x - x_i^{l-1}\|^2, \ \forall x \in \mathbb{R}^d.$$

This implies

$$F_i(x) \leq T_i^{l-1}(x) - \frac{\rho(h_i)}{4}\|x - x_i^{l-1}\|^2. \tag{4.21}$$

From (4.19) *and* (4.21) *with* $x = x^l$, *we have*

$$T^l(x^{l+1}) \leq T^{l-1}(x^l) - \frac{1}{n}\sum_{i\in s_l}\frac{\rho(h_i)}{4}\|x - x_i^{l-1}\|^2 + \left(\frac{2b}{n} + 1\right)\epsilon^l. \tag{4.22}$$

Taking the expectation of the inequality (4.22) *conditioned on* $\mathcal{F}_l$, *we obtain*

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b}{4n^2}\sum_{i=1}^{n}\rho(h_i)\|x^l - x_i^{l-1}\|^2 + \left(\frac{2b}{n} + 1\right)\epsilon^l.$$

Combining this and $\rho = \min_{i=1,...,n}\rho(h_i) > 0$ *gives us*

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b\rho}{4n^2}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2 + \left(\frac{2b}{n} + 1\right)\epsilon^l.$$

Applying the supermartingale convergence theorem to the nonnegative sequences $\{T^{l-1}(x^l) - \alpha^*\}, \{\frac{b}{4\rho n^2}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2\}$ *and* $\{(\frac{2b}{n}+1)\epsilon^l\}$, *we get*

$$\sum_{l=1}^{\infty}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2 < \infty,$$

with probability 1. *In particular, for* $i = 1, ..., n$, *we have*

$$\sum_{l=1}^{\infty} \|x^l - x_i^{l-1}\|^2 < \infty, \tag{4.23}$$

and hence $\lim_{l\to\infty} \|x^l - x_i^{l-1}\| = 0$ *almost surely.*

c) Assume that there exists a sub-sequence $\{x^{l_k}\}$ *of* $\{x^l\}$ *such that* $x^{l_k} \to x^*$ *almost surely. From* (4.23), *we have* $\|x^{l_k+1} - x_i^{l_k}\| \to 0$ *almost surely. Without loss of generality, we can suppose that the sub-sequence* $v_i^{l_k} \to v_i^*$ *almost surely. From the proof of (a), we have*

$$\frac{1}{n}\sum_{i=1}^{n} \epsilon_i^l \leq T^l(x^{l+1}) - F(x^{l+1}).$$

From this and (4.20) *it follows that* ϵ_i^l *converges to* 0 *as* $l \to +\infty$ *with probability* 1. *Since* $v_i^{l_k} \in \partial_{\epsilon_i^{l_k}} h_i(x_i^{l_k})$, $\epsilon_i^{l_k} \to 0$ *with probability* 1, *and by the closedness property of the* ϵ*-subdifferential mapping* $\partial_{\epsilon_i^{l_k}} h_i$, *we have* $v_i^* \in \partial h_i(x^*)$. *Since* x^{l_k+1} *is a* ϵ^{l_k}*-solution of the problem* $\min_x T^{l_k}(x)$, *we obtain*

$$T^{l_k}(x^{l_k+1}) \leq T^{l_k}(x) + \epsilon^{l_k}. \tag{4.24}$$

Taking $k \to \infty$ *gives us*

$$\limsup_{l_k\to+\infty} G(x^{l_k+1}) \leq G(x) - \langle x - x^*, v^*\rangle, \ \forall x \in \mathbb{R}^d,$$

with probability 1, *where* $v^* = \frac{1}{n}\sum_{i=1}^n v_i^* \in \partial H(x^*)$ *almost surely. It follows from this with* $x = x^*$ *that*

$$\limsup_{l_k\to+\infty} G(x^{l_k+1}) \leq G(x^*),$$

almost surely. Combining this with the lower semi-continuity of G *gives us that*

$$\lim_{l_k\to+\infty} G(x^{l_k+1}) = G(x^*),$$

almost surely. Thus, we have

$$G(x^*) \leq G(x) - \langle x - x^*, v^*\rangle, \ \forall x \in \mathbb{R}^d,$$

almost surely. This implies

$$v^* \in \partial G(x^*), \tag{4.25}$$

with probability one. Therefore,

$$v^* \in \partial G(x^*) \cap \partial H(x^*), \tag{4.26}$$

with probability 1. *This implies that* x^* *is a critical point of* F *with probability* 1 *and the proof is then complete.* ■

4.3 Application to group variables selection in multi-class logistic regression

Consider now the group variables selection in multi-class logistic regression whose the description is given in Section 3.5 of Chapter 3. Recall that we have developed DCA-Like and ADCA-Like to solve this problem in Chapter 3 with $\ell_{2,0}$ regularization. In this section, we consider a broader case with $\ell_{2,0}$ regularization defined as follows

$$\|W\|_{q,0} = |\{j \in \{1, ..., d\} : \|W_{j,:}\|_q \neq 0\}|,$$

with $q \in \{1, 2, \infty\}$. Hence, the $\ell_{q,0}$ regularized multi-class logistic regression problem is formulated as follows

$$\min_{W,b} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, W, b) + \lambda \|W\|_{q,0} \right\}. \tag{4.27}$$

We use a non-convex approximation of the $\ell_{q,0}$-norm based on the following two penalty functions $\eta_\alpha(s)$:

$$\begin{aligned} \text{Concave exponential :} \quad \eta_\alpha^{\exp}(s) &= 1 - \exp(-\alpha s), \\ \text{Capped-}\ell_1 : \quad \eta_\alpha^{\text{cap-}\ell_1}(s) &= \min\{1, \alpha s\}. \end{aligned}$$

These penalty functions have shown their effectiveness in several problems, for instance, individual variables selection in SVM [35, 128], sparse optimal scoring problem [149], sparse covariance matrix estimation problem [194], and bi-level/group variables selection [151, 193]. For a more complete references on nonconvex approximation approaches for the ℓ_0 norm, refer to Chapter 5.

The corresponding approximate problem of (4.27) takes the form :

$$\min_{W,b} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, W, b) + \lambda \sum_{j=1}^{d} \eta_\alpha(\|W_{j,:}\|_q) \right\}. \tag{4.28}$$

Since η_α is increasing on $[0, +\infty)$, the problem (4.28) can be equivalently reformulated as follows

$$\min_{(W,b,t)} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\ell(x_i, y_i, W, b) + \chi_\Omega(W, b, t) + \lambda \sum_{j=1}^{d} \eta_\alpha(t_j) \right] \right\}, \tag{4.29}$$

where $\Omega = \{(W, b, t) \in \mathbb{R}^{d \times Q} \times \mathbb{R}^Q \times \mathbb{R}^d : \|W_{j,:}\|_q \leq t_j, j = 1, ..., d\}$. Moreover, as $\ell(x_i, y_i, W, b)$ is differentiable with L-Lipschitz continuous gradient and η_α is concave, the problem (4.29) takes the form of (4.1) where the function $F_i(W, b, t)$ is given by

$$F_i(W, b, t) = \ell(x_i, y_i, W, b) + \chi_\Omega(W, b, t) + \lambda \sum_{j=1}^{d} \eta_\alpha(t_j) := g_i(W, b, t) - h_i(W, b, t),$$

where the DC components g_i and h_i are defined by (with $\rho > L$) :

$$\begin{aligned} g_i(W, b, t) &= \frac{\rho}{2} \|(W, b)\|^2 + \chi_\Omega(W, b, t), \\ h_i(W, b, t) &= \frac{\rho}{2} \|(W, b)\|^2 - \ell(x_i, y_i, W, b) - \lambda \sum_{j=1}^{d} \eta_\alpha(t_j). \end{aligned}$$

Before presenting SDCA for solving (4.29), let us show how to apply standard DCA to this problem.

4.3.1 Standard DCA for solving the problem (4.29)

We consider three norms corresponding to $q \in \{1, 2, \infty\}$. DCA applied to (4.29) consists of computing, at each iteration l, $(U^l, v^l, z^l) \in \partial H(W^l, b^l, t^l)$, and solving the convex sub-problem

$$\min_{(W,b,t)} \left\{ \frac{\rho}{2}\|(W,b)\|^2 + \chi_\Omega(W,b,t) - \langle U^l, W\rangle - \langle v^l, b\rangle - \langle z^l, t\rangle \right\}. \tag{4.30}$$

The computation of (U^l, v^l, z^l) is explicitly defined as follows.

$$(U^l, v^l, z^l) = \frac{1}{n}\sum_{i=1}^{n}(U_i^l, v_i^l, z_i^l), (U_i^l, v_i^l, z_i^l) \in \partial h_i(W^l, b^l, t^l),$$

$$\begin{aligned}
(U_i^l)_{:,k} &= \rho W_{:,k}^l - \left(p_k^l(x_i) - \delta_{ky_i}\right) x_i, k = 1, ...Q,\\
(v_i^l)_k &= \rho b_k^l - \left(p_k^l(x_i) - \delta_{ky_i}\right), k = 1, ...Q,\\
(z_i^l)_j &= \begin{cases} -\lambda\alpha \exp(-\alpha t_j^l), & j = 1, \ldots, d \quad \text{if } \eta_\alpha = \eta_\alpha^{\exp},\\ -\lambda\alpha \text{ if } \alpha t_j^l \le 1, \text{ and 0 otherwise}, & j = 1, \ldots, d, \quad \text{if } \eta_\alpha = \eta_\alpha^{\text{cap}-\ell_1}, \end{cases}
\end{aligned} \tag{4.31}$$

with $p_k^l(x_i) = \exp(b_k^l + (W_{:,k}^l)^T x_i)/(\sum_{h=1}^{Q} b_h^l + (W_{:,h}^l)^T x_i))$, $\delta_{ky_i} = 1$ if $k = y_i$ and 0 otherwise.

The convex sub-problem (4.30) can be solved as follows (note that $z_j^l \le 0$ for $j = 1, \ldots, d$)

$$W^{l+1} = \arg\min_W \left\{ \frac{\rho}{2}\|W\|^2 + \sum_{j=1}^{d}(-z_j^l)\|W_{j,:}\|_q - \langle U^l, W\rangle \right\}, \tag{4.32}$$

$$b^{l+1} = \arg\min_b \left\{ \frac{\rho}{2}\|b\|^2 - \langle v^l, b\rangle \right\} = \frac{1}{\rho}v^l, \tag{4.33}$$

$$t_j^{l+1} = \|W_{j,:}^{l+1}\|_q, j = 1, ..., d. \tag{4.34}$$

Since the problem (4.32) is separable in rows of W, solving it amounts to solving d independent sub-problems

$$W_{j,:}^{l+1} = \arg\min_{W_{j,:}} \left\{ \frac{\rho}{2}\|W_{j,:}\|^2 + (-z_j^l)\|W_{j,:}\|_q - \langle U_{j,:}^l, W_{j,:}\rangle \right\} \tag{4.35}$$

$W_{j,:}^{l+1}$ can be explicitly computed as follows. From (4.35) we can write

$$W_{j,:}^{l+1} = \arg\min_{W_{j,:}} \left\{ \frac{1}{2}\|W_{j,:} - U_{j,:}^l/\rho\|^2 + (-z_j^l)\|W_{j,:}\|_q \right\}.$$

$$W_{j,:}^{l+1} = \mathbf{prox}_{(-z_j^l)/\rho\|\cdot\|_q}\left(U_{j,:}^l/\rho\right),$$

where the proximal operator $\mathbf{prox}_f(\nu)$ is defined by

$$\mathbf{prox}_f(\nu) = \arg\min_t \left\{ \frac{1}{2}\|t - \nu\|^2 + f(t) \right\}.$$

The proximal operator of $(-z_j^l)/\rho\|\cdot\|_q$ can be efficiently computed [184]. The computation of $\mathbf{prox}_{(-z_j^l)/\rho\|.\|_q}(\nu/\rho)$ can be summarized in Table 4.1.

Hence, DCA based algorithms for solving (4.29) with $q \in \{1, 2, \infty\}$ are described as follows.

TABLE 4.1 – Computation of $W_{j,:}^{l+1} = \mathbf{prox}_{(-z_j^l)/\rho\|.\|_q}\left(U_{j,:}^l/\rho\right)$ corresponding to $q \in \{1, 2, \infty\}$.

q	$\mathbf{prox}_{(-z_j^l)/\rho\|.\|_q}\left(U_{j,:}^l/\rho\right)$
1	$\left(\lvert U_{j,:}^l\rvert/\rho - (-z_j^l)/\rho\right)_+ \circ \operatorname{sign}(U_{j,:}^l)$
2	$\begin{cases} \left(1 - \frac{-z_j^l}{\|U_{j,:}^l\|_2}\right) U_{j,:}^l/\rho & \text{if } \|U_{j,:}^l\|_2 > -z_j^l \\ 0 & \text{if } \|U_{j,:}^l\|_2 \leq -z_j^l. \end{cases}$
∞	$\begin{cases} U_{j,:}^l/\rho - \left(\frac{1}{-z_j^l}\lvert U_{j,:}^l\rvert - \delta\right)_+ \circ \operatorname{sign}(U_{j,:}^l) & \text{if } \|U_{j,:}^l\|_1 > -z_j^l \\ 0 & \text{if } \|U_{j,:}^l\|_1 \leq -z_j^l, \end{cases}$ where δ satisfies $\sum_{k=1}^Q \left(\frac{1}{-z_j^l}\lvert U_{j,k}^l\rvert - \delta\right)_+ = 1$.

Algorithm 4.3 DCA-$\ell_{q,0}$: DCA for solving (4.29) with $q \in \{1, 2, \infty\}$

1: **Initialization :** Choose $(W^0, b^0) \in \mathbb{R}^{d\times Q} \times \mathbb{R}^Q$, $\rho > L$ and $l \leftarrow 0$.
2: **repeat**
3: Compute $(U^l, v^l, z^l) = \frac{1}{n}\sum_{i=1}^n (U_i^l, v_i^l, z_i^l)$, where (U_i^l, v_i^l, z_i^l), $i = 1, ..., n$ are defined in (4.31).
4: Compute $(W^{l+1}, b^{l+1}, t^{l+1})$ according to Table 4.1, (4.33) and (4.34).
5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

4.3.2 Stochastic DCA for solving the problem (4.29)

In SDCA, at each iteration l, we have to compute $(U_i^l, v_i^l, z_i^l) \in \partial h_i(W^l, b^l, t^l)$ for $i \in s_l$ and keep $(U_i^l, v_i^l, z_i^l) = (U_i^{l-1}, v_i^{l-1}, z_i^{l-1})$ for $i \notin s_l$, where s_l is a randomly chosen subset of the indexes, and solve the convex sub-problem taking the form of (4.30). Hence, SDCA for solving (4.29) is described below.

Algorithm 4.4 SDCA-$\ell_{q,0}$: DCA for solving (4.29) with $q \in \{1, 2, \infty\}$

1: **Initialization :** Choose $(W^0, b^0) \in \mathbb{R}^{d\times Q} \times \mathbb{R}^Q$, $\rho > L$ and $l \leftarrow 0$.
2: **repeat**
3: Compute (U_i^l, v_i^l, z_i^l) by (4.31) if $i \in s_l$ and keep $(U_i^l, v_i^l, z_i^l) = (U_i^{l-1}, v_i^{l-1}, z_i^{l-1})$ if $i \notin s_l$. Set $(U^l, v^l, z^l) = \frac{1}{n}\sum_{i=1}^n (U_i^l, v_i^l, z_i^l)$.
4: Compute $(W^{l+1}, b^{l+1}, t^{l+1})$ according to Table 4.1, (4.33) and (4.34).
5: $l \leftarrow l + 1$.
6: **until** Stopping criterion

We observe that all calculations of (U_i^l, v_i^l, z_i^l) and $(W^{l+1}, b^{l+1}, t^{l+1})$ in DCA and SDCA for the problem (4.29) are exactly defined in explicit form. Therefore the inexact version ISDCA is useless (Remark 4.2). Hence in our experiments we consider only SDCA.

4.3.3 Experiment setting

Datasets. To evaluate the performances of algorithms, we performed numerical experiments on two types of data : real datasets (*covertype*, *madelon*, *miniboone*, *protein*, *sensit* and *sensorless*) and simulated datasets (*sim_ 1*, *sim_ 2* and *sim_ 3*). All real-world datasets are taken from the well-known UCI and LibSVM data repositories. We generate three synthetic datasets (*sim_ 1*, *sim_ 2* and *sim_ 3*) by the same process proposed in [236]. The information of datasets are given in following.

— *covertype* belongs to the Forest Cover Type Prediction from strictly cartographic variables challenge (`https://archive.ics.uci.edu/ml/datasets/Covertype`). It is a very large dataset containing $581,012$ points described by 54 variables.

— *madelon* is one of five datasets used in the NIPS 2003 feature selection challenge (`https://archive.ics.uci.edu/ml/datasets/Madelon`). The dataset contains 2600 points, each point is represented by 500 variables. Among 500 variables, there are only 5 informative variables and 15 redundant variables (which are created by linear combinations of 5 informative variables). The 480 others variables were added and have no predictive power. Notice that *madelon* is a highly non-linear dataset.

— *miniboone* is taken form the MiniBooNE experiment to observe neutrino oscillations (`https://archive.ics.uci.edu/ml/datasets/MiniBooNE+particle+identification`), containing $130,065$ data points.

— *protein* (`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`) is a dataset for classifying protein second structure state (α, β, and coil) of each residue in amino acid sequences, including $24,387$ data points.

— *sensit* [4.3.3] dataset obtained from distributed sensor network for vehicle classification. It consists of $98,528$ data points categorized into 3 classes : Assault Amphibian Vehicle (AAV), Dragon Wagon (DW) and noise.

— *sensorless* measures electric current drive signals from different operating conditions, which is classified into 11 different classes (`https://archive.ics.uci.edu/ml/datasets/Dataset+for+Sensorless+Drive+Diagnosis`). It is a huge dataset, which contains $58,509$ data points, described by 48 variables.

— For *sim_ 1* : we generate a four-classes classification problem. Each class is assumed to have a multivariate normal distribution $\mathcal{N}(\mu_k, I)$, $k = 1, 2, 3, 4$ with dimension of $d = 50$. The first 10 components of μ_1 are 0.5, $\mu_{2j} = 0.5$ if $11 \leq j \leq 20$, $\mu_{3j} = 0.5$ if $21 \leq j \leq 30$, $\mu_{4j} = 0.5$ if $31 \leq j \leq 40$ and 0 otherwise. We generate $250,000$ instances with equal probabilities.

— For *sim_ 2* : this synthetic dataset contains three classes of multivariate normal distributions $\mathcal{N}(\mu_k, \Sigma)$, $k = 1, 2, 3$, each of dimension $d = 50$. The components of $\mu_1 = 0$, $\mu_{2j} = 0.4$ and $\mu_{3j} = 0.8$ if $j \leq 40$ and 0 otherwise. The covariance matrix Σ is the block diagonal matrix with five blocks of dimension 10×10 whose element (j, j') is $0.6^{|j-j'|}$. We generate $150,000$ instances.

— For *sim_ 3* : this synthetic dataset consists of four classes. For class $k = 1, 2, 3, 4$, $i \in C_k$ then $X_{ij} \sim \mathcal{N}(0, 1)$ for $j > 100$, and $X_{ij} \sim \mathcal{N}(\frac{k-1}{3}, 1)$ otherwise, where $\mathcal{N}(\mu, \sigma^2)$ denotes the Gaussian distribution with mean μ and variance σ^2. We generate $62,500$ data points for each class.

Comparative algorithms. We will compare the performance of the proposed SDCA with standard DCA as well as four other algorithms : the first two algorithms, named `SPGD-`$\ell_{2,1}$ and

`msgl`, use the convex regularization $\ell_{2,1}$ instead of $\ell_{2,0}$ and deal with the following problem

$$\min_{W,b} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, W, b) + \lambda \|W\|_{2,1} \right\}. \tag{4.36}$$

`SPGD-`$\ell_{2,1}$ is a version of stochastic gradient descent algorithm [31, 152], the most popular stochastic algorithm in machine learning, while `msgl` ([227]) is a coordinate gradient descent method. The last two algorithms, named `nm-APG` ([154]) and `DC-PN` ([198]), are the most recent which consider nonconvex regularization $\ell_{2,0}$. `nm-APG` [154] is an accelerated proximal gradient based method for minimizing $f(x) + r(x)$ where $f(x)$ is a differentiable function with L-Lipschitz gradient and $r(x)$ is a nonconvex function. `nm-APG` requires to compute the proximal mapping of the DC function η_α. However, this proximal mapping does not have a closed form. We therefore use DCA to compute the proximal mapping of η_α in `nm-APG`. `DC-PN` was proposed in [198] for minimizing $f(x) + h(x)$, where $f(x) = f_1(x) - f_2(x)$ is a DC function, twice differentiable, with $f(x)$ verifying the L-Lipschitz gradient property ; $h(x) = h_1(x) - h_2(x)$ where both h_1 and h_2 are convex functions and (possibly) non-differential. Note that, in `DC-PN`, L-BFGS is employed for approximating the Hessian matrix H_k ; and the sub-problem is solved by `minFunc` solver (`minFunc` : unconstrained differentiable multivariate optimization in Matlab. `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`).

Setting. We randomly split each dataset into a training set and a test set. The training set contains 80% of the total number of points and the remaining 20% are used as a test set.

We use the early-stopping condition for `SDCA` and `SPGD-`$\ell_{2,1}$. Early-stopping is a well-know technique in machine learning, especially in stochastic learning which permits to avoid the overfitting in learning. More precisely, after each epoch, we compute the classification accuracy on a validation set which contains 20% randomly chosen data points of training set. We stop `SDCA` and `SPGD-`$\ell_{2,1}$ if the classification accuracy is not improved after $n_{patience} = 5$ epochs. The batch size of stochastic algorithms (`SDCA` and `SPGD-`$\ell_{2,1}$) is set to 10%. `DCA` is stopped if the difference between two consecutive objective functions is smaller than a threshold $\epsilon_{stop} = 10^{-6}$. For *msgl*, we use its default stopping parameters as in [227]. We also stop algorithms if they exceed 2 hours of running time in the training process.

The parameter α for controlling the tightness of zero-norm approximation is chosen in the set $\{0.5, 1, 2, 5\}$. We use the solution-path procedure for the trade-off parameter λ. Let $\lambda_1 > \lambda_2 > ... > \lambda_l$ be a decreasing sequence of λ. At step k, we solve the problem (4.27) with $\lambda = \lambda_k$ from the initial point chosen as the solution of the previous step $k-1$. Starting with a large value of λ, we privilege the sparsity of solution (i.e. selecting very few variables) over the classification ability. Then by decreasing the value λ, we select more variables in order to increase the classification accuracy. In our experiments, the sequence of λ is set to $\{10^4, 3 \times 10^3, 10^3, \ldots, 3 \times 10^{-3}, 10^{-3}\}$.

All experiments are performed on a PC Intel (R) Xeon (R) E5-2630 v2 @2.60 GHz with 32GB RAM. In order to evaluate the performance of algorithms, we consider the following three criteria : the classification accuracy (percentage of well-classified points on the test set), the sparsity of obtained solutions and the running time (measured in seconds). The sparsity is computed as the percentage of selected variables. Note that a variable $j \in \{1, \ldots, d\}$ is considered to be removed if all components of the row j of W are smaller than a threshold, i.e., $|W_{j,i}| \leq 10^{-8}, \forall i \in 1, \ldots, Q$. We perform each algorithm 10 times and report the average result over 10 runs.

4.3.4 Numerical results

Experiment 1 : Comparison between SDCA with other algorithms. For this purpose, among three regularizations $\ell_{q,0}$ and two approximations of the ℓ_0 norm we fix the $\ell_{2,0}$ regulari-

zation and the capped-ℓ_1 approximation. As the capped-ℓ_1 function is nonsmooth, the resulting approximate problem is nonsmooth (and nonconvex) (whereas the exponential approximation is a smooth function). We perform comparative experiments on SDCA-$\ell_{2,0}$- capℓ_1, DCA-$\ell_{2,0}$-capℓ_1, as well as the four comparative algorithms mentioned above. The comparative results (the average on 10 runs) are reported in Table 4.2. To simplify the presentation, in this experiment we use SDCA, DCA and SPGD instead of SDCA-$\ell_{2,0}$-capℓ_1, DCA-$\ell_{2,0}$-capℓ_1 and SPGD-$\ell_{2,1}$ respectively.

TABLE 4.2 – Experiment 1 : The average results on 10 runs of six algorithms on all datasets. n, d and Q is the number of instances, the number of variables and the number of classes respectively. NA means that the algorithm fails to furnish a result after 12 hours.

Dataset	Performance	SDCA	DCA	SPGD	mgsl	mn-APG	DC-PN
covertype	Accuracy%	70.40	70.40	66.97	71.22	NA	71.43
n=581,012	Sparsity%	57.41	57.41	100	68.52	NA	93.21
d=54, Q=7	Time(s)	7.47	61.15	60.59	525.49	NA	64.95
madelon	Accuracy%	62.18	61.28	61.79	60.48	56.99	59.49
n=2,600	Sparsity%	0.40	0.93	1.00	0.67	0.40	56.87
d=500,Q=2	Time(s)	0.15	0.21	1.07	23.92	2.43	0.55
miniboone	Accuracy%	83.31	83.74	83.86	81.99	83.95	83.57
n=130,065	Sparsity%	6.00	6.00	11.00	10.00	6.00	14.00
d=50, Q=2	Time(s)	1.18	7.04	8.77	121.17	56.51	7.96
protein	Accuracy%	66.41	67.04	66.59	67.34	67.04	67.39
n=24,387	Sparsity%	22.64	50.47	92.70	47.15	44.29	63.30
d=357, Q=3	Time(s)	1.13	3.35	11.73	5.59	31.04	4.13
sensit	Accuracy%	78.59	78.92	79.52	79.02	79.16	79.20
n=98,528	Sparsity%	33.80	56.67	27.00	23.00	14.33	50.33
d=100,Q=3	Time(s)	2.94	26.36	22.44	11.16	332.58	10.42
sensorless	Accuracy%	84.77	89.60	86.07	85.06	89.09	87.66
n=58,509	Sparsity%	68.06	53.47	88.89	50.00	36.81	97.22
d=48, Q=11	Time(s)	2.45	24.75	8.16	199.00	360.02	30.11
sim_1	Accuracy%	72.24	72.24	71.48	72.33	71.02	72.18
n=100,000	Sparsity%	80.00	80.00	83.50	82.00	80.00	96.00
d=50, Q=4	Time(s)	0.50	0.30	7.16	214.83	344.65	2.77
sim_2	Accuracy%	68.50	67.70	67.62	68.42	66.96	67.50
n=150,000	Sparsity%	80.00	80.00	82.00	82.00	80.00	88.00
d=50, Q=3	Time(s)	1.02	4.29	7.77	367.29	319.28	3.93
sim_3	Accuracy%	99.69	99.88	99.70	99.93	99.86	99.85
n=250,000	Sparsity%	80.00	80.00	80.00	80.20	80.00	80.13
d=500, Q=4	Time(s)	21.95	249.74	212.71	1581.44	571.19	144.72

• *Stochastic SDCA versus standard DCA.* As expected, in terms of classification accuracy, SDCA produces fairly similar results comparing with DCA, with a gap less than 1% on 8 out of 9 datasets. More precisely, the gap is zero in two datasets, in favor of SDCA in 2 datasets, and in favor of DCA in 5 datasets. For the remaining dataset *sensorless* the gap is 4.83% in favor of DCA.

As for the sparsity of solution, the two algorithms give the same result on 5 out of 9 datasets, SDCA is better than DCA on 2 datasets (the gap is 27.83% and 22.87%), and DCA is better

than SDCA on 2 datasets (the gap is 0.53% and 14.59%).

Concerning the running time, not surprisingly, SDCA is much faster than DCA, the ratio of gain varies from 1.41 (madelon) to 11.38 (*sim_ 3*) times on 8 out of 9 datasets. Except for *sim_ 1* where DCA is slightly faster than SDCA (0.5 versus 0.3 s).

Overall, SDCA achieves similar solutions' quality in a much shorter time than DCA, that is the aim of our work.

• *SDCA and DCA versus other algorithms.* In terms of classification accuracy, SDCA and DCA produce fairly similar results compared with the best results given by the four algorithms SPGD, mgsl, mn-APG, DC-PN with a gap less than 1% on 8 out of 9 datasets (4 in favor of SDCA/DCA, 4 in favor of the best among the four other algorithms) and equal to 1.03% in the remaining dataset (*covertype*). More concretely, comparing with the two $\ell_{2,1}$ (convex) regularization algorithms SPGD and mgsl, the gap versus SPGD (resp. mgsl) is in favor of SDCA/DCA on 6 (resp. 5) datasets. As for the two $\ell_{2,0}$ (nonconvex) regularization algorithms mn-APG, DC-PN, the gap versus mn-APG as well as DC-PN is in favor of SDCA/DCA on 6 datasets, and the gap versus mn-APG is zero on 1 dataset.

Regarding the sparsity, SDCA gives the sparsest solutions on 7 datasets, while DCA as well as mn-APG get the best result on 5 datasets (these three algorithms furnish the same sparsity of solutions on 4 datasets). The gain of SDCA versus the three remaining algorithms is considerable. It varies from 0.68% (resp. 0.27%) to 70.06% (resp. 24.51%) comparing with SPGD (resp. mgsl). Relating to DC-PN, the gap varies from 16% to 56.47% on 6 datasets, and from 0.13% to 8% on 3 datasets. Altogether, SDCA is the best while DC-PN is the worst.

Overall, combining the two criteria - accuracy and sparsity, SDCA is the best, followed by DCA, and then mn-APG (not counting *covertype* for which mn-APG is failed after 12 hours), while DC-PN is the worse.

In the matter of rapidity (running time), SDCA (resp. DCA) is the fastest on 8 (resp. 1) dataset(s). The gain of SDCA versus the four other algorithms is huge. The ratio of gain varies from 3.33 (resp. 4.95) to 14.32 (resp. 360.09) times comparing with SPGD (resp. mgsl), and from 16.20 (resp. 3.65) to 689.30 (resp. 12.29) times compared with mn-APG (resp. DC-PN). Altogether, SDCA is the fastest algorithm, followed by DCA, and then DC-PN, while mn-APG is the slowest, in particular it fails to get a solution after 12 hours on *covertype*.

In summary, as expected, SDCA reduces considerably the running time of DCA while achieving equivalent classification accuracy and better sparsity. Moreover, SDCA outperforms the four related algorithms SPGD, msgl, mn-APG and DC-PN. Hence, SDCA is the best algorithm on both quality (accuracy and sparsity) and rapidity.

Experiment 2 : Comparison on feature selection. For the purpose of feature selection, the effect of regularization terms as well as of approximation functions is an important matter. In this experiment we study the effectiveness of SDCA in terms of different non-convex regularizations $\ell_{q,0}$ ($q \in \{1, 2, \infty\}$) by comparing three algorithms SDCA-$\ell_{1,0}$-exp, SDCA-$\ell_{2,0}$-exp and SDCA-$\ell_{\infty,0}$-exp (the exponential approximation is fixed for this experiment). The results are summarized in Table 4.3 (the columns 3-5). We also study the effect of the approximation functions (capped-ℓ_1 and exponential approximation) by comparing two algorithms : SDCA-$\ell_{2,0}$-exp and SDCA-$\ell_{q,0}$-capℓ_1. The results are given in the same Table 4.3 (4th and 6th column).

•*SDCA with three regularizations $\ell_{1,0}$, $\ell_{2,0}$, $\ell_{\infty,0}$.* In terms of classification accuracy, the three algorithms SDCA-$\ell_{1,0}$-exp, SDCA-$\ell_{2,0}$-exp and SDCA-$\ell_{\infty,0}$-exp get similar results with a gap less than 1% on all datasets. In particular, the gap between SDCA-$\ell_{1,0}$-exp and SDCA-$\ell_{2,0}$-exp is lower than 0.3% on 6 datasets. The gain is in favor of SDCA-$\ell_{1,0}$-exp on 4 datasets, of SDCA-$\ell_{\infty,0}$-exp on 4 datasets (they have the same best result on *sim_ 1*) and of SDCA-$\ell_{2,0}$-exp on 2

TABLE 4.3 – The average results on 10 runs of Experiment 2 (SDCA with different regularization $\ell_{q,0}, q \in 1, 2, \infty$)

Dataset	Performance	$\ell_{1,0}$-exp	$\ell_{2,0}$-exp	$\ell_{\infty,0}$-exp	$\ell_{2,0}$-capℓ_1
covertype	Accuracy%	71.34	71.62	69.92	70.40
	Sparsity%	69.91	61.11	60.49	57.41
	Time(s)	10.27	4.74	11.93	7.47
madelon	Accuracy%	61.92	62.12	61.68	62.18
	Sparsity%	0.65	0.40	0.70	0.40
	Time(s)	0.14	0.16	0.16	0.15
miniboone	Accuracy%	83.90	83.84	83.10	83.31
	Sparsity%	8.00	6.00	8.00	6.00
	Time(s)	1.57	3.60	1.62	1.18
protein	Accuracy%	67.23	67.84	68.13	66.41
	Sparsity%	63.67	64.89	92.79	22.64
	Time(s)	1.47	1.28	1.36	1.13
sensit	Accuracy%	79.64	78.67	79.73	78.59
	Sparsity%	34.00	28.33	53.67	33.80
	Time(s)	3.11	3.48	1.61	2.94
sensorless	Accuracy%	87.33	86.52	86.69	84.77
	Sparsity%	54.69	37.50	97.92	68.06
	Time(s)	1.40	1.47	1.41	2.25
sim_1	Accuracy%	72.24	72.22	72.24	72.24
	Sparsity%	80.00	80.00	80.00	80.00
	Time(s)	0.46	0.46	0.56	0.50
sim_2	Accuracy%	68.48	68.53	68.71	68.50
	Sparsity%	80.00	80.00	80.00	80.00
	Time(s)	0.73	0.79	0.97	1.02
sim_3	Accuracy%	99.93	99.69	99.56	99.69
	Sparsity%	80.00	80.00	80.73	80.00
	Time(s)	10.74	36.61	22.11	21.45

datasets.

As for the sparsity of solution, SDCA-$\ell_{2,0}$-exp is the best on 7 datasets (except for *covertype* and protein), while SDCA-$\ell_{\infty,0}$-exp is the worse on 8 datasets (except for *covertype* it gives the best result). The gain of SDCA-$\ell_{2,0}$-exp versus SDCA-$\ell_{1,0}$-exp (resp. SDCA-$\ell_{\infty,0}$-exp) varies from 0.25% to 17.19% (resp. from 0.30% to 60.42%), and the gain of SDCA-$\ell_{1,0}$-exp versus SDCA-$\ell_{\infty,0}$-exp varies from 0.05% to 43.23%. Overall, combining the two criteria - accuracy and sparsity, SDCA-$\ell_{1,0}$-exp and SDCA-$\ell_{2,0}$-exp realize a better trade-off between classification and sparsity of solution than SDCA-$\ell_{\infty,0}$-exp, and SDCA-$\ell_{2,0}$-exp is the best.

In terms of rapidity, the three algorithms are comparable on 6 datasets where the running time is less than 1.5 s. As for the 3 remaining datasets, each algorithm is winner on 1 dataset. SDCA-$\ell_{2,0}$-exp is the winner on *covertype* with the ratio of gain being 2.17 and 2.52 versus SDCA-$\ell_{1,0}$-exp and SDCA-$\ell_{\infty,0}$-exp respectively. SDCA-$\ell_{1,0}$-exp is 3.4 (resp. 2.06) times faster than SDCA-$\ell_{2,0}$-exp (resp. SDCA-$\ell_{\infty,0}$-exp) on *sim_ 3* while SDCA-$\ell_{\infty,0}$-exp is 1.93 (resp. 2.16) times faster than SDCA-$\ell_{1,0}$-exp (resp. SDCA-$\ell_{2,0}$-exp) on *sensit*.

In summary, the two algorithms SDCA-$\ell_{1,0}$-exp and SDCA-$\ell_{2,0}$-exp are comparable and they are better than SDCA-$\ell_{\infty,0}$-exp on both quality and rapidity, in particular in terms of sparsity. Hence, for feature selection purpose, it is suggested to use the first two algorithms.

•*SDCA with the two approximations - exponential and capℓ_1 functions.*

In terms of classification accuracy SDCA-$\ell_{2,0}$-exp is slightly better than SDCA-$\ell_{2,0}$-capℓ_1 on 8 datasets with a gap less than 1% on 5 datasets and less than 1.8% on 3 datasets. For the remain dataset (*sim_ 1*) the gain is only 0.02% in favor of SDCA-$\ell_{2,0}$-capℓ_1. Regarding sparsity, the two algorithms get the same result on 5 datasets, SDCA-$\ell_{2,0}$-capℓ_1 is winner on *covertype* and *protein* with the gain 3.7% and 42.25% respectively, while SDCA-$\ell_{2,0}$-exp is winner on *sensit* and *sensorless* with the gain 5.47% and 43.56% respectively.

As for running time, SDCA-$\ell_{2,0}$-exp is faster than SDCA-$\ell_{2,0}$-capℓ_1 from 1.08 to 1.57 times on 4 datasets, while the later is faster than the former from 1.13 to 3.05 times on 5 datasets.

Overall, SDCA-$\ell_{2,0}$-exp seems to be better for the purpose of classification, and it realizes a better trade-off between quality and rapidity in most cases.

Experiment 3 : Comparison between SDCA, DCA-Like and ADCA-Like.

In this experiment, we aim to evaluate the performance of our three proposed algorithms, namely SDCA, DCA-Like and ADCA-Like for high-dimensional datasets. For this purpose, we choose $q = 2$ and concave exponential approximation. The results are reported in the Table 4.4.

We observe that

— The classification accuracy of DCA-Like and ADCA-Like are similar ($< 0.1\%$ in 8 over 9 cases) ; and both are slightly higher than SDCA by up to 0.5% (in protein and sim_1). Overall, the differences are neglectable.

— Similarly, in terms of sparsity, DCA-Like and ADCA-Like often choose the least number of variables, and is much smaller than SDCA. The gap between DCA-Like (reps. ADCA-Like) and SDCA is up to 9%, which is equivalent to 2 to 6 times bigger subset of features in some datasets (*CLL_ SUB_ 111*, *ORL* and *miniboone*).

— In terms of running time, it is clear that SDCA is faster than DCA-Like and ADCA-Like, especially for large dataset. Except the three smallest datasets (*CLL_ SUB_ 111*, *Carcinom* and *lung*) where DCA-Like (reps. ADCA-Like) is faster than SDCA ; SDCA is faster. The reduction in running time is from 1.3 to 8 times ; especially in the two biggest datasets where SDCA is faster than the second fastest ones by 4 and 8 times respectively.

In conclusion, ADCA-Like is faster than DCA-Like, while having similar classification accuracy and sparsity. SDCA is much faster than ADCA-Like (reps. DCA-Like) while having similar

TABLE 4.4 – Comparative results on group variable selection for multi-class logistic regression. Bold values correspond to best results for each dataset. *NA* means that the algorithm fails to furnish a result. n, d and Q are the number of instances, dimensions and classes respectively. Unit of time is second.

Dataset	Algorithm	Accuracy (%)		Time (sec.)		Sparsity (%)	
		Mean	STD	Mean	STD	Mean	STD
CLL_SUB_111	DCA-Like	**78.79**	**6.94**	0.75	0.03	0.32	0.18
$n \times d = 111 \times 11340$	ADCA-Like	**78.79**	**6.94**	**0.63**	**0.06**	**0.18**	**0.04**
$Q = 3$	SDCA	**78.79**	**5.25**	5.25	1.64	1.95	1.09
Carcinom	DCA-Like	89.52	8.25	3.06	0.05	**0.41**	**0.01**
$n \times d = 174 \times 9182$	ADCA-Like	**90.48**	**6.60**	**1.81**	**0.11**	**0.41**	**0.01**
$Q = 11$	SDCA	**90.48**	**6.60**	4.16	2.74	0.66	0.25
lung	DCA-Like	**91.06**	**3.73**	0.71	0.01	**0.47**	**0.02**
$n \times d = 203 \times 3312$	ADCA-Like	**91.06**	**3.73**	**0.42**	**0.08**	0.50	0.02
$Q = 5$	SDCA	**91.06**	**3.73**	0.86	0.10	0.57	0.11
ORL	DCA-Like	**94.58**	**5.05**	10.76	0.56	**6.02**	**0.31**
$n \times d = 400 \times 1024$	ADCA-Like	**94.58**	**2.89**	7.43	0.76	7.71	0.34
$Q = 40$	SDCA	**94.58**	**5.05**	**3.04**	**1.38**	16.76	6.69
BASEHOCK	DCA-Like	94.65	1.26	5.72	2.53	4.24	0.87
$n \times d = 1993 \times 4862$	ADCA-Like	94.65	1.38	5.47	1.51	**2.28**	**0.69**
$Q = 2$	SDCA	**94.74**	**0.25**	**3.54**	**0.28**	2.73	0.16
protein	DCA-Like	68.26	0.50	2.92	0.43	**77.90**	**2.03**
$n \times d = 24387 \times 357$	ADCA-Like	**68.29**	**0.51**	4.16	0.90	78.00	2.13
$Q = 3$	SDCA	67.94	0.62	**1.17**	**0.35**	80.43	0.58
sim_1	DCA-Like	**72.22**	**0.50**	0.66	0.00	**80.00**	**0.00**
$n \times d = 100000 \times 50$	ADCA-Like	**72.22**	**0.50**	0.63	0.02	**80.00**	**0.00**
$Q = 4$	SDCA	72.18	0.57	**0.49**	**0.06**	**80.00**	**0.00**
miniboone	DCA-Like	83.78	0.07	4.82	0.42	**6.00**	**0.00**
$n \times d = 130064 \times 50$	ADCA-Like	83.78	0.07	3.68	0.60	**6.00**	**0.00**
$Q = 2$	SDCA	**84.25**	**0.34**	**0.92**	**0.23**	11.33	1.15
sim_3	DCA-Like	**99.93**	**0.01**	358.40	7.38	**80.00**	**0.00**
$n \times d = 250000 \times 500$	ADCA-Like	**99.93**	**0.01**	159.95	16.26	**80.00**	**0.00**
$Q = 4$	SDCA	**99.93**	**0.02**	**17.86**	**3.28**	**80.00**	**0.00**

accuracy ; however ADCA-Like selects more compact subset of variables than SDCA.

4.4 Conclusion

We have proposed two novel DCA-based algorithms, stochastic DCA and inexact stochastic DCA with the aim of reducing the computation cost of DCA in large-scale setting. The sum structure of the objective function F permits us to work separately on the component functions F_i. The stochastic DCA is then proposed to tackle problems with huge numbers of F_i while the inexact stochastic DCA aims to address large-scale setting and big data. In the algorithmic point of view, our SDCA scheme enjoys some variance reduction design : the SDCA can be regarded as a combination of the standard DCA and the SAG estimator (which is a variance-reduced gradient) introduced in [210]. We proved that both algorithms converge to a critical point with probability 1. We applied DCA and SDCA to group variables selection in multi-class logistic regression, an important problem in machine learning. By using a suitable DC decomposition of the objective function we have designed a DCA scheme in which all computations are explicit and inexpensive. Consequently SDCA is very inexpensive. Numerical results showed that, as expected, SDCA reduces considerably the running time of standard DCA while achieving equivalent classification accuracy and better sparsity.

Our works, more particularly the paper [131] was the first contribution which initialized a very attracting research direction coupled DCA and stochastic framework. Since then, several algorithms in this direction were developed in the literature (e.g. [127, 158, 136, 177, 244]. As for us, we are continuing to develop stochastic DCA in the following works (which are not presented in this chapter) :

i) Online stochastic DCA [148] which can be used in the context of online learning and big data ;

ii) Stochastic DCA with Variance Reduction for solving a class of structured DC problems [135].

In future works we plan to

1. develop new stochastic DCA schemes with competitive complexity compared with existing algorithms ;
2. develop stochastic based general DCA for stochastic DC programs with stochastic DC constraints (Stochastic General DCA) ;
3. deploy novel Stochastic General DCA schemes on several important areas and their applications such as Chance constrained optimization, Bayesian optimization, Interference networks in Telecommunications, Distributionally Robust Chance Constrained Programming based on Generative adversarial networks.

Deuxième partie

Sparse Optimization and its applications

This part is devoted to the sparse optimization problem which refers to an optimization problem involving the ℓ_0-norm in objective or constraints. Formally, a sparse optimization problem takes the form

$$\inf \{F(x,y) = f(x,y) + \lambda \|x\|_0 : (x,y) \in K \subset \mathbb{R}^n \times \mathbb{R}^m \},$$

where the function f corresponds to a given criterion and $\lambda > 0$ is the trade-off between the criterion f and the sparsity of x.

We develop an unifying framework, with solid theoretical tools as well as efficient algorithms based on DC programming and DCA for sparse optimization. Our unified DC programming framework shed a new light on sparse nonconvex programming. It permits to establish the crucial relations among existing sparsity-inducing methods and therefore to exploit, in an elegant way, the nice effect of DC decompositions of objective functions.

In Chapter 5, we present two approaches for sparse optimization, namely nonconvex approximation approach and nonconvex exact reformulation approach. In nonconvex approximation approach (Section 5.2), the zero-norm is approximated by a nonconvex continuous function. Considering a class of DC approximation functions of the zero-norm including all usual sparse inducing approximation functions in literature, we provide several important results from both a theoretical and an algorithmic point of view. We prove the consistency between global minimums (resp. local minimums) of approximate and original problems. We show that, in several cases, some global minimizers (resp. local minimizers) of the approximate problem are also those of the original problem. Furthermore, we prove that for some particular approximations, the approximate problem with suitable parameters is equivalent to the original problem. The efficiency of several sparse inducing penalty functions is carefully analyzed. Four DCA schemes are developed that cover all standard algorithms in nonconvex sparse approximation approaches as special versions. In nonconvex exact reformulation approach (Section 5.3), the sparse optimization problem is equivalently reformulated as DC program using the exact penalty technique. Then, the latter can be solved by DCA based algorithm.

In Chapter 6, we apply the two proposed approaches in three applications : feature selection in Support Vector Machine (Section 6.1), feature selection in Semi-Supervised Support Vector Machine (Section 6.2) and sparse recovery signal (Section 6.3).

Chapitre 5

DCA based algorithms for sparse optimization[1]

Abstract: In this chapter, we developed an unifying framework, with solid theoretical tools as well as efficient algorithms based on DC programming and DCA, to tackle the sparse optimization problem. Two approaches are developed, namely nonconvex approximation and nonconvex exact reformulation. In nonconvex approximation approach, using a common DC approximation of the zero-norm including all standard sparse inducing penalty functions, we studied the consistency between global minimums (resp. local minimums) of approximate and original problems. We showed that, in several cases, some global minimizers (resp. local minimizers) of the approximate problem are also those of the original problem. Furthermore, we proved that for some particular approximations, the approximate problem with suitable parameters is equivalent to the original problem. The efficiency of several sparse inducing penalty functions have been fully analyzed. Four DCA schemes were developed that cover all standard algorithms in nonconvex sparse approximation approaches as special versions. They can be viewed as, an ℓ_1-perturbed algorithm / reweighted-ℓ_1 algorithm / reweighted-ℓ_1 algorithm. In nonconvex exact reformulation approach, we equivalently reformulated the original problem as a DC program thanks to the exact penalty technique and then developed DCA to solve it. Moreover, we established the link between the exact reformulation with the approximate problem using the convex approximation ℓ_1 and the nonconvex approximation Capped-ℓ_1. By these results, we unified all nonconvex approaches for treating the zero-norm into the DC programming and DCA.

5.1 Introduction

The zero-norm on $\mathbb{R}^n$, denoted ℓ_0-norm or $\|.\|_0$, is defined by

$$\|x\|_0 := |\{i = 1, ..., n : x_i \neq 0\}|,$$

where $|S|$ is the cardinality of the set S. The useful notation $|.|_0$ denoting the ℓ_0-norm on $\mathbb{R}$ allows to express the separability of $\|.\|_0$ on $\mathbb{R}^n$

$$\|x\|_0 = \sum_{i=1}^{n} |x_i|_0 .$$

1. The results presented in this chapter were published in :
   - H.A. Le Thi, T. Pham Dinh, H.M. Le, X.T. Vo, DC approximation approaches for sparse optimization, European Journal of Operational Research,244(1) :26-46, 2015.
   - H.A. Le Thi, H.M. Le and T. Pham Dinh, Feature Selection in machine learning : an exact penalty approach using a Different of Convex function Algorithm, Machine Learning, 101(1) :163-186, 2015.

The ℓ_0-norm is an important concept for modelling the sparsity of data and plays a crucial role in optimization problems where one has to select representative variables. Sparse optimization, which refers to an optimization problem involving the ℓ_0-norm in objective or constraints, has many applications in various domains (in particular in machine learning, image processing and finance), and draws increased attention from many researchers in recent years. The function ℓ_0, apparently very simple, is lower-semicontinuous on $\mathbb{R}^n$, but its discontinuity at the origin makes nonconvex programs involving $\|.\|_0$ challenging. Note that although one uses the term "norm" to design $\|.\|_0$, $\|.\|_0$ is not a norm in the mathematical sense. Indeed, for all $x \in \mathbb{R}^n$ and $\lambda \neq 0$, one has $\|\lambda x\|_0 = \|x\|_0$, which is not true for a norm.

Formally, a sparse optimization problem takes the form

$$\inf \{F(x,y) = f(x,y) + \lambda \|x\|_0 : (x,y) \in K \subset \mathbb{R}^n \times \mathbb{R}^m \}, \tag{5.1}$$

where the function f corresponds to a given criterion and λ is a positive number, called the regularization parameter, that makes the trade-off between the criterion f and the sparsity of x [235, 254]. In some applications, one wants to control the sparsity of solutions, the ℓ_0-term is thus put in constraints, and the corresponding optimization problem is

$$\inf\{f(x,y) : (x,y) \in K, \|x\|_0 \leq k\}. \tag{5.2}$$

Let us mention some important applications of sparse optimization corresponding to these models.

Feature selection in classification learning. Feature selection is one of fundamental problems in machine learning. In many applications such as text classification, web mining, gene expression, micro-array analysis, combinatorial chemistry, image analysis, etc, data sets contain a large number of features, many of which are irrelevant or redundant. Feature selection is often applied to high-dimensional data prior to classification learning. The main goal is to select a subset of features of a given data set while preserving or improving the discriminative ability of the classifier. The embedded feature selection in classification can be formulated as $\min f(x) + \lambda \|x\|_0$ where $f(x)$ is the loss function corresponding to the classification method and $\lambda > 0$ is the trade-off between two terms. Numerous sparse optimization methods have been developed for feature selection in classification, for instance in SVM [35, 121], S3VM [121, 245], Gaussian Mixtures Model [176], Logistique regression [192, 132], etc

Sparse Regression. Given a training data set $\{b_i, a_i\}_{i=1}^q$ of q independent and identically distributed samples composed of explanatory variables $a_i \in \mathbb{R}^n$ (inputs) and response variables $b_i \in \mathbb{R}$ (ouputs). Let $b := (b_i)_{i=1,\ldots,q}$ denote the vector of outputs and $A := (a_{i,j})_{i=1,\ldots,q}^{j=1,\ldots,n}$ denote the matrix of inputs. The problem of the regression consists in looking for a relation which can possibly exist between A and b, in other words, relating b to a function of A and a model parameter x. Such a model parameter x can be obtained by solving the optimization problem

$$\min \left\{ f(x) := \sum_{i=1}^{q} L(b_i, a_i^T x) : x \in \mathbb{R}^n \right\}, \tag{5.3}$$

where $L : \mathbb{R}^n \to \mathbb{R}$ is called loss function. The *sparse regression* problem aims to find a sparse solution of the above regression model, it takes the form of (5.1) :

$$\min_{x \in \mathbb{R}^n} \left\{ \sum_{i=1}^{q} L(b_i, a_i^T x) + \rho \|x\|_0 \right\}. \tag{5.4}$$

Regression problems have many important applications, among them sparse signal/image recovery and feature selection in classification.

Sparse Fisher Linear Discriminant Analysis. Discriminant analysis captures the relationship between multiple independent variables and a categorical dependent variable in the usual multivariate way, by forming a composite of the independent variables. Given a set of q independent and identically distributed samples composed of explanatory variables $a_i \in \mathbb{R}^n$ and binary response variables $b_i \in \{-1, 1\}$. The idea of Fisher linear discriminant analysis is to determine a projection of variables onto a straight line that best separables the two classes. The line is so determined to maximize the ratio of the variances of between and within classes in this projection, i.e. maximize the function $f(\alpha) = \frac{\langle \alpha, S_B \alpha \rangle}{\langle \alpha, S_W \alpha \rangle}$, where S_B and S_W are, respectively, the between and within classes scatter matrix (they are symmetric positive semidefinite) given by

$$S_B := (q_+ - q_-)(q_+ - q_-)^T,\ S_W = S_+ + S_-,$$

$$S_+ = \sum_{i=1, b_i=+1}^{q} (x_i - q_+)(x_i - q_+)^T,\ S_- = \sum_{i=1, b_i=-1}^{q} (x_i - q_-)(x_i - q_-)^T.$$

Here, for $j \in \{\pm\}$, q_j is the mean vector of class j, l_j is the number of labeled samples in class j. If α is an optimal solution of the problem, then the classifier is given by $F(a) = \alpha^T a + c$, $c = 0.5\alpha^T(q_+ - q_-)$.

The sparse Fisher Discriminant model is defined by ($\rho > 0$)

$$\min\{\alpha^T S_W \alpha + \rho \|\alpha\|_0 : \alpha^T(q_+ - q_-) = b\}.$$

Sparse Covariance Matrix Estimation. The estimation of a covariance matrix is a fundamental problems in statisticsand emerges from many applications such as portfolio management and risk assessment, high-dimensional classification, finding quantitative trait loci based on longitudinal data, etc [194]. Let $Y = (Y_1, \ldots, Y_p)^T$ be p-dimensional random vector with the covariance matrix $\Sigma = [\Sigma]_{ij}]_{1 \le i,j \le p}$ where $\Sigma]_{ij}$ is the covariance between Y_i and Y_j. Suppose that we observe a sample including n observational data points $X_1, \ldots, X_n$ from a multivariate normal distribution $\mathcal{N}(0, \Sigma)$. The sparse covariance matrix extimation is formulated as

$$\min \left\{ \log\det \Sigma + tr(\Sigma^{-1} S) + \lambda \|\Sigma\|_0 \right\}$$

where $S = \frac{1}{n} \sum_{i=1}^{n} X_i X_i^T$ is the sample covariance matrix.

Compressed sensing. Compressed sensing refers to techniques for efficiently acquiring and reconstructing signals via the resolution of underdetermined linear systems. Compressed sensing concerns sparse signal representation, sparse signal recovery and sparse dictionary learning which can be formulated as sparse optimization problems of the form (5.1).

Portfolio selection problem with cardinality constraint. In portfolio selection problem, given a set of available securities or assets, we want to find the optimum way of investing a particular amount of money in these assets. Each of the different ways to diversify this money among the several assets is called a portfolio. In portfolio management one wants to limit the number of assets to be investigated in the portfolio, that leads to a problem of the form (5.2).

Other applications : Other applications of sparse optimization include Sensor networks ([18]), Error correction ([43, 42]), Digital photography ([219]), etc.

State of the art of sparse optimization

During the last decades, research is very active in models and methods optimization involving the zero-norm. Works can be divided into three categories according to the way to treat the zero-norm : convex approximation, nonconvex approximation, and nonconvex exact reformulation.

Convex approximation. In the machine learning community, one of the best known approaches, belonging to the group "convex approximation", is the ℓ_1 regularization approach proposed in [223] in the context of linear regression, called LASSO (Least Absolute Shrinkage and Selection Operator), which consists in replacing the ℓ_0 term $\|x\|_0$ by $\|x\|_1$, the ℓ_1 -norm of the vector x. In [88], the authors have proved that, under suitable assumptions, a solution of the ℓ_0- regularizer problem over a polyhedral set can be obtained by solving the ℓ_1- regularizer problem. However, these assumptions are quite restrictive. Since its introduction, several works have been developed to study the ℓ_1-regularization technique, from the theoretical point of view to efficient computational methods (see [96], Chapter 18 for more discussions on ℓ_1-regularized methods). The LASSO penalty has been shown to be, in certain cases, inconsistent for variable selection and biased [263]. Hence, the Adaptive LASSO is introduced in [263] in which adaptive weights are used for penalizing different coefficients in the ℓ_1-penalty.

Nonconvex approximations. At the same time, nonconvex continuous approaches, belonging to the second group "nonconvex approximation" (the ℓ_0 term $\|x\|_0$ is approximated by a nonconvex continuous function) were extensively developed. A variety of sparsity-inducing penalty functions have been proposed to approximate the ℓ_0 term : exponential concave function [35], ℓ_p-norm with $0 < p < 1$ [76] and $p < 0$ [203], Smoothly Clipped Absolute Deviation (SCAD) [70], Logarithmic function [235], Capped-ℓ_1 [186]. Using these approximations, several algorithms have been developed for resulting optimization problems, most of them are in the context of feature selection in classification, sparse regressions or more especially for sparse signal recovery : Successive Linear Approximation (SLA) algorithm [35], DCA based algorithms [53, 56, 79, 91, 121, 128, 140, 139, 137, 171, 180, 121, 253, 252, 261], Local Linear Approximation (LLA) [264], Two-stage ℓ_1 [256], Adaptive Lasso [263], reweighted-ℓ_1 algorithms [44], reweighted-ℓ_2 algorithms such as Focal Underdetermined System Solver (FOCUSS) ([85, 203, 202]), Iteratively reweighted least squares (IRLS) and Local Quadratic Approximation (LQA) algorithm [70, 264].

Nonconvex exact reformulation. In the third category named nonconvex exact reformulation approaches, the ℓ_0-regularized problem is reformulated as a continuous nonconvex program. There are a few works in this category. In [163], the author reformulated the problem (5.1) in the context of feature selection in SVM as a linear program with equilibrium constraints (LPEC). However, this reformulation is generally intractable for large-scale datasets. In [221, 190] an exact penalty technique in DC programming is used to reformulate (5.1) and (5.2) as DC programs. In [222] this technique is used for Sparse Eigenvalue problem with ℓ_0-norm in constraint functions

$$\max\{x^T Ax : x^T x = 1, \|x\|_0 \leq k\}, \tag{5.5}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric and k an integer, and a DCA based algorithm was investigated for the resulting problem.

Beside the three above categories, heuristic methods are developed to tackle directly the original problem (5.1) by greedy based algorithms, e.g. matching pursuit, orthogonal matching pursuit, [162, 185], etc.

Challenges in sparse optimization. Convex regularization approaches involve convex optimization problems which are so far "easy" to solve, but they don't attain the solution of the ℓ_0-regularizer problem. Nonconvex approximations are, in general, deeper than convex relaxations, and then can produce good sparsity, but the resulting optimization problems are still difficult since they are nonconvex and there are many local minima which are not global. The exact reformulation approach has the same difficulty since the equivalent problems are nonconvex and require efficient methods to solve. Many issues have not yet been studied or proved in the existing approximation approaches. First, the consistency between the approximate problems and the original problem is a very important question but still is open. Only a weak result has been proved for two special cases in [36] (resp. [204]) when f is concave, bounded below on a polyhedral convex set K and the approximation term is an exponential concave function (resp. a logarithm function and/or ℓ_p-norm ($p < 1$)). It has been shown in these works that the intersection of the solution sets of the approximate problem and the original problem is nonempty. Moreover no result on the consistency between local minimum of approximate and original problems has been available, while most of the proposed algorithms furnish local minima. Second, several existing algorithms lack a rigorous mathematical proof of convergence. Hence the choice of a "good" approximation remains relevant. Two crucial questions should be studied for solving large scale problems, that are, *how to suitably approximate the zero-norm* and *which computational method to use* for solving the resulting optimization problem. The development of new models and algorithms for sparse optimization problems is always a challenge for researchers in optimization and machine learning.

Our contributions

We consider in this chapter the sparse optimization problem (5.1) where K is a polyhedral convex set in $\mathbb{R}^n \times \mathbb{R}^m$ and f is a finite DC function with the following decomposition

$$f(x, y) = g(x, y) - h(x, y) \quad \forall (x, y) \in \mathbb{R}^n \times \mathbb{R}^m, \tag{5.6}$$

where g, h are finite convex functions on $\mathbb{R}^n \times \mathbb{R}^m$.

We address all challenge cited above and develop an unifying approach based on DC programming and DCA, from both a theoretical and a computational point of view.

Firstly, considering a common DC approximate function, we prove the consistency between the approximate problem and the original problem by showing the link between their global minimizers as well as their local minimizers. We demonstrate that any optimal solution of the approximate problem is in a $\epsilon-$neighbourhood of an optimal solution to the original problem (5.1). More strongly, if f is concave and the objective function of the approximate problem is bounded below on K, then some optimal solutions of the approximate problem are exactly solutions of the original problem. These new results are important and very useful for justifying the performance of approximation approaches.

Secondly, thanks to the exact penalty technique in DC programming, we equivalently reformulate the optimization problem (5.1) and (5.2) as continuous optimization problems which are DC program. DCA is then developed for solving the continuous exact reformulation of (5.1). Moreover, we show that the ℓ_1 approximation is nothing but the linear relaxation of our exact formulation. Note that in our paper [130], we have developed the continuous exact reformulation of (5.1) and (5.2) with f being convex. Here, we consider a broader case where f is a DC function.

Thirdly, we provide an in-depth analysis of usual sparsity-inducing functions and compare them according to suitable parameter values. This study suggests the choice of good approximations of the zero-norm as well as that of good parameters for each approximation. A reasonable comparison via suitable parameters identifies Capped -ℓ_1 and SCAD as the best approximations.

Fourthly, we prove, via an exact reformulation approach by exact penalty techniques that, with suitable parameters ($\theta > \theta_0$), nonconvex approximate problems resulting from Capped -ℓ_1 or SCAD functions are equivalent to the original problem. Moreover, when the set K is a box, we can show directly (without using exact penalty techniques) the equivalence between the original problem and the approximate Capped -ℓ_1 problem and give the value of θ_0 such that this equivalence holds for all $\theta > \theta_0$. These interesting and significant results justify our analysis on usual sparsity-inducing functions and the pertinence of these approximation approaches. It opens the door to study other approximation approaches which are consistent with the original problem.

Fifthly, we develop solution methods for all DC approximation approaches. Our algorithms are based on DC programming and DCA, because our main motivation is to exploit the efficiency of DCA to solve this hard problem. We propose three DCA schemes for three different formulations of a common model to all concave approximation functions. We show that these DCA schemes include all standard algorithms as special versions. The fourth DCA scheme is concerned with the resulting DC program given by the DC approximation (nonconcave piecewise linear) function in [122]. Using DC programming framework, we unify all solution methods into DC, and then convergence properties of our algorithms are guaranteed, thanks to general convergence results of the generic DCA scheme. It permits to exploit, in an elegant way, the nice effect of DC decompositions of the objective functions to design various versions of DCA. It is worth mentioning here the flexibility/versatility of DC programming and DCA : the four algorithms can be viewed as an ℓ_1-perturbed algorithm / a reweighted-ℓ_1 algorithm (intimately related to the ℓ_1-penalized LASSO approach) / a reweighted-ℓ_2 algorithm in case of convex objective functions.

The remainder of the chapter is organized as follows. In Section 5.2, we present the theoretical tools and different DCA algorithms for the nonconvex approximation approach. Section 5.3 is devoted to the exact reformulation approach and its resolution by DCA.

5.2 DC approximation approach for sparse optimization

In this session, we focus on the sparse optimization formulation (5.1) which involves the ℓ_0 norm in objective function, i.e.,

$$\inf \{F(x,y) = f(x,y) + \lambda \|x\|_0 : (x,y) \in K \subset \mathbb{R}^n \times \mathbb{R}^m \}.$$

Let us define the step function $s : \mathbb{R} \to \mathbb{R}$ by $s(t) = 1$ for $t \neq 0$ and $s(t) = 0$ otherwise. Then $\|x\|_0 = \sum_{i=1}^n s(x_i)$. The idea of approximation methods is to replace the discontinuous step function by a continuous approximation r_θ, where $\theta > 0$ is a parameter controling the tightness of approximation. This leads to the approximate problem of the form

$$\min \left\{ F_{r_\theta}(x,y) = f(x,y) + \lambda \sum_{i=1}^n r_\theta(x_i) : (x,y) \in K \right\}. \tag{5.7}$$

Assumption 5.1 *$\{r_\theta\}_{\theta>0}$ is a family of functions $\mathbb{R} \to \mathbb{R}$ satisfying the following properties :*

i) $\lim_{\theta \to +\infty} r_\theta(t) = s(t)$, $\forall t \in \mathbb{R}$.

ii) For any $\theta > 0$, r_θ is even, i.e. $r_\theta(t) = r_\theta(|t|)$ $\forall t \in \mathbb{R}$) and r_θ is increasing on $[0, +\infty)$.

iii) For any $\theta > 0$, r_θ is a DC function which can be represented as

$$r_\theta(t) = \varphi_\theta(t) - \psi_\theta(t) \quad t \in \mathbb{R},$$

where $\varphi_\theta, \psi_\theta$ are finite convex functions on $\mathbb{R}$.

iv) $t\mu \geq 0\ \forall t \in \mathbb{R}, \mu \in \partial r_\theta(t)$.*where* $\partial r_\theta(t) = \{u - v : u \in \partial\varphi_\theta(t), v \in \partial\psi_\theta(t)\}$.

v) For any $a \leq b$ *and* $0 \notin [a,b]$ *:* $\lim\limits_{\theta\to+\infty} \sup\{|z| : z \in \partial r_\theta(t), t \in [a,b]\} = 0$.

We now study the consistency between the approximate problems and the original one.

5.2.1 DC approximation approaches : consistency results

First of all, we observe that by assumption ii) above, we get another equivalent form of (5.7)

$$\min_{(x,y,z)\in\Omega_1} \overline{F}_{r_\theta}(x,y,z) := f(x,y) + \lambda \sum_{i=1}^{n} r_\theta(z_i), \tag{5.8}$$

where

$$\Omega_1 = \{(x,y,z) : (x,y) \in K, |x_i| \leq z_i \ \ \forall i = 1,\ldots,n\}.$$

Indeed, (5.7) and (5.8) are equivalent in the following sense.

Proposition 5.1 ([147]) *A point* $(x^*,y^*) \in K$ *is a global (resp. local) solution of the problem* (5.7) *if and only if* $(x^*,y^*,|x^*|)$ *is a global (resp. local) solution of the problem* (5.8)*. Moreover, if* (x^*,y^*,z^*) *is a global solution of* (5.8) *then* (x^*,y^*) *is a global solution of* (5.7)*.*

Proof 5.1 *Since* r_θ *is an increasing function on* $[0,+\infty)$*, we have*

$$\overline{F}_{r_\theta}(x,y,z) \geq \overline{F}_{r_\theta}(x,y,|x|) = F_{r_\theta}(x,y) \quad \forall (x,y,z) \in \Omega_1.$$

Then the conclusion concerning global solutions is trivial. The result on local solutions also follows by remarking that if $(x,y,z) \in B((x^*,y^*,z^*),\delta)$ *(*$B(u^*,\delta)$ *stands for the set of vectors* $u \in \mathbb{R}^d$ *such that* $\|u-u^*\| < \delta$*) then* $(x,y) \in B((x^*,y^*),\delta)$*, and if* $(x,y) \in B((x^*,y^*),\frac{\delta}{2})$ *then* $(x,y,|x|) \in B((x^*,y^*,|x^*|),\delta)$*.* ■

In standard nonconvex approximation approaches to ℓ_0-problem, all the proposed approximation functions r_θ are even and concave increasing on $[0,+\infty)$ (see Table 5.1 below) and the approximate problems were often considered in the form (5.8). Here we study the general case where r_θ is a DC function and consider both problems (5.7) and (5.8) in order to exploit the nice effect of DC decompositions of a DC program.

Now we show the link between the original problem (5.1) and the approximate problem (5.7). This result gives *a mathematical foundation* of approximation methods.

Theorem 5.1 *Let* $\mathcal{P}, \mathcal{P}_\theta$ *be the solution sets of the problem* (5.1) *and* (5.7) *respectively.*

i) Let $\{\theta_k\}$ *be a sequence of nonnegative numbers such that* $\theta_k \to +\infty$ *and* $\{(x^k,y^k)\}$ *be a sequence such that* $(x^k,y^k) \in \mathcal{P}_{\theta_k}$ *for any* k*. If* $(x^k,y^k) \to (x^*,y^*)$*, then* $(x^*,y^*) \in \mathcal{P}$*.*

ii) If K *is compact, then for any* $\epsilon > 0$ *there is* $\theta(\epsilon) > 0$ *such that*

$$\mathcal{P}_\theta \subset \mathcal{P} + B(0,\epsilon) \quad \forall \theta \geq \theta(\epsilon).$$

iii) If there is a finite set $\mathcal{S}$ *such that* $\mathcal{P}_\theta \cap \mathcal{S} \neq \emptyset\ \forall \theta > 0$*, then there exists* $\theta_0 \geq 0$ *such that*

$$\mathcal{P}_\theta \cap \mathcal{S} \subset \mathcal{P} \quad \forall \theta \geq \theta_0.$$

Proof 5.2 *i) Let (x, y) be arbitrary in K. For any k, since $(x^k, y^k) \in \mathcal{P}_{\theta_k}$, we have*

$$f(x,y) + \lambda \sum_{i=1}^{n} r_{\theta_k}(x_i) \geq f(x^k, y^k) + \lambda \sum_{i=1}^{n} r_{\theta_k}(x_i^k). \tag{5.9}$$

By Assumption 5.1 ii), if $x_i^* = 0$, we have

$$\liminf_{k\to+\infty} r_{\theta_k}(x_i^k) \geq \liminf_{k\to+\infty} r_{\theta_k}(0) = 0.$$

If $x_i^* \neq 0$, there exist $a_i \leq b_i$ and $k_i \in \mathbb{N}$ such that $0 \neq [a_i, b_i]$ and $x_i^k \in [a_i, b_i]$ for all $k \geq k_i$. Then we have

$$|r_{\theta_k}(x_i^k) - s(x_i^*)| \leq \max\{|r_{\theta_k}(a_i) - s(a_i)|, |r_{\theta_k}(b_i) - s(b_i)|\} \quad \forall k \geq k_i.$$

Since $\lim_{k\to+\infty} r_{\theta_k}(a_i) = s(a_i)$ and $\lim_{k\to+\infty} r_{\theta_k}(b_i) = s(b_i)$, we have $\lim_{k\to+\infty} r_{\theta_k}(x_i^k) = s(x_i^*)$. Note that f is continuous, taking lim inf *of both sides of* (5.9)*, we get*

$$f(x,y) + \lambda \sum_{i=1}^{n} s(x_i) \geq f(x^*, y^*) + \lambda \sum_{i=1}^{n} \liminf_{k\to\infty} r_{\theta_k}(x_i^k) \geq f(x^*, y^*) + \lambda \sum_{i=1}^{n} s(x_i^*).$$

Thus, $F(x, y) \geq F(x^*, y^*)$ for any $(x, y) \in K$, or $(x^*, y^*) \in \mathcal{P}$.

ii) We assume by contradiction that there exists $\epsilon > 0$ and a sequence $\{\theta_k\}$ such that $\theta_k \to +\infty$, and for any k there is $(x^k, y^k) \in \mathcal{P}_{\theta_k} \backslash (\mathcal{P} + B(0, \epsilon))$. Since $\{(x^k, y^k)\} \subset K$ and K is compact, there exists a subsequence $\{(x^{k_l}, y^{k_l})\}$ of $\{(x^k, y^k)\}$ converges to a point $(x^*, y^*) \in K$. By i), we have $(x^*, y^*) \in \mathcal{P}$. However, $\{(x^{k_l}, y^{k_l})\} \subset K \backslash (\mathcal{P} + B(0, \epsilon))$ that is a closed set, so $(x^*, y^*) \in K \backslash (\mathcal{P} + B(0, \epsilon))$. This contradicts the fact that $(x^*, y^*) \in \mathcal{P}$.

iii) Assume by contradiction that there is a sequence $\{\theta_k\}$ such that $\theta_k \to +\infty$, and for any k there is $(x^k, y^k) \in (\mathcal{P}_{\theta_k} \cap \mathcal{S}) \backslash \mathcal{P}$. Since $\mathcal{S}$ is finite, we can extract a subsequence such that $(x^{k_l}, y^{k_l}) = (\overline{x}, \overline{y})$ $\forall l$. Then we have $(\overline{x}, \overline{y}) \notin \mathcal{P}$. This contradicts the fact that $(\overline{x}, \overline{y}) \in \mathcal{P}$ following i). ■

Remark 5.1 *The assumption that r_θ is an even function is not needed for proving this theorem. More precisely, the theorem still holds when the assumption ii) is replaced by "for any $\theta > 0$, r_θ is decreasing on $(-\infty, 0]$ and is increasing on $[0, +\infty)$". For the zero-norm, since the step function is even, it is natural to consider its approximation r_θ as an even function.*

Theorem 5.1 shows that any optimal solution of the approximate problem (5.7) is in a $\epsilon-$neighboohord of an optimal solution to the original problem (5.1), and the tighter approximation of ℓ_0-norm is, the better approximate solutions are. Moreover, if there is a finite set $\mathcal{S}$ such that $\mathcal{P}_\theta \cap \mathcal{S} \neq \emptyset$ $\forall \theta > 0$, then any optimal solution of the approximate problem (5.7) contained in $\mathcal{S}$ solves also the problem (5.1). By considering the equivalent problem (5.8), we show in the following Corollary that such a set $\mathcal{S}$ exists in several contexts of applications (for instance, in feature selection in SVM).

Corollary 5.1 *Suppose that r is concave on $[0, +\infty)$, K is a polyhedral convex set having at least a vertex and f is concave, bounded below on K. Then Ω_1 defined in* (5.8) *is also a polyhedral convex set having at least a vertex. Let $\mathcal{V}$ be the vertex set of Ω_1 and*

$$\overline{\mathcal{P}}_\theta = \{(x, y) : \exists z \in \mathbb{R}^n \text{ s.t. } (x, y, z) \in \mathcal{V} \text{ is a global solution of } (5.8)\}.$$

Then $\overline{\mathcal{P}}_\theta \neq \emptyset$ $\forall \theta > 0$ and there exists $\theta_0 > 0$ such that $\overline{\mathcal{P}}_\theta \subset \mathcal{P}$, $\forall \theta \geq \theta_0$.

Proof 5.3 *By the assumptions, we have $\overline{F}_{r_\theta}$ is concave, bounded below on Ω_1, so $\overline{\mathcal{P}}_\theta \neq \emptyset$ $\forall \theta > 0$. Let $\mathcal{S} = \{(x, y) : (x, y, z) \in \mathcal{V}$ for some $z \in \mathbb{R}^n\}$. By Proposition 5.1, we have $\overline{\mathcal{P}}_\theta \subset \mathcal{P}_\theta \cap \mathcal{S}$ $\forall \theta > 0$. Since $\mathcal{V}$ is finite, so is $\mathcal{S}$. The property iii) of Theorem 5.1 implies the existence of $\theta_0 > 0$ such that*

$$\overline{\mathcal{P}}_\theta \subset \mathcal{P}_\theta \cap \mathcal{S} \subset \mathcal{P} \ \forall \theta \geq \theta_0.$$

■

Note that the consistency between the solution of the approximate problem and the original problem have been carried out in [36] (resp. [204]) for the case where f is concave, bounded below on the polyhedral convex set K and r is the exponential approximation defined in Table 5.1 below (resp. r is the logarithm function and/or ℓ_p-norm ($p < 1$)). Here, besides general results carried out in Theorem 5.1, our Corollary 5.1 gives a much stronger result than those in [36, 204] where they only ensure that $\overline{\mathcal{P}}_\theta \cap \mathcal{P} \neq \emptyset$ $\forall \theta \geq \theta_0$.

Observing that the approximate problem is still nonconvex for which, in general, only local algorithms are available, we are motivated by the study of the consistency between local minimizers of the original and approximate problems. For this purpose, first, we need to describe characteristics of local solutions of these problems.

Proposition 5.2 *i) A point $(x^*, y^*) \in K$ is a local optimum of the problem* (5.1) *if and only if (x^*, y^*) is a local optimum of the problem*

$$\min\{f(x, y) : (x, y) \in K(x^*)\}, \tag{5.10}$$

where $K(x^*) = \{(x, y) \in K : x_i = 0 \ \forall i \notin supp(x^*)\}$.

ii) If $(x^*, y^*) \in K$ is a local optimum of the problem (5.1) *then*

$$\langle \overline{x}^*, x - x^* \rangle + \langle \overline{y}^*, y - y^* \rangle \geq 0 \quad \forall (x, y) \in K(x^*), \tag{5.11}$$

for some $(\overline{x}^*, \overline{y}^*) \in \partial f(x^*, y^*)$.

Proof 5.4 *i) The forward implication is obvious, we only need to prove the backward one. Assume that (x^*, y^*) is a local solution of the problem* (5.10)*. There exists a neighbourhood $\mathcal{V}$ of (x^*, y^*) such that*

$$supp(x^*) \subset supp(x) \quad and \quad |f(x, y) - f(x^*, y^*)| < \lambda \quad \forall (x, y) \in \mathcal{V},$$

and

$$f(x^*, y^*) \leq f(x, y) \quad \forall (x, y) \in \mathcal{V} \cap K(x^*).$$

For any $(x, y) \in \mathcal{V} \cap K$, two cases occur :

- If $(x, y) \in K(x^*)$, then $\|x\|_0 = \|x^*\|_0$ and $f(x^*, y^*) \leq f(x, y)$.

- If $(x, y) \notin K(x^*)$, then $\|x^*\|_0 \leq \|x\|_0 - 1$ and $f(x^*, y^*) < f(x, y) + \lambda$.

In both cases, we have $f(x^*, y^*) + \lambda\|x^*\|_0 \leq f(x, y) + \lambda\|x\|_0$. Thus, (x^*, y^*) is a local solution of the problem (5.2)*.*

ii) Since $f = g - h$ is a DC function, (5.10) *is a DC program. Therefore, the necessary local condition of the problem* (5.10) *can be stated by*

$$0 \in \partial(g + \chi_{K(x^*)})(x^*, y^*) - \partial h(x^*, y^*),$$

or equivalently, there exists $(\overline{x}^*, \overline{y}^*) \in \partial f(x^*, y^*)$ such that

$$-(\overline{x}^*, \overline{y}^*) \in \partial \chi_{K(x^*)}(x^*, y^*) \Leftrightarrow \langle \overline{x}^*, x - x^* \rangle + \langle \overline{y}^*, y - y^* \rangle \geq 0 \quad \forall (x, y) \in K(x^*).$$

■

As for the characteristics of local solutions of the problem (5.7), we follow the necessary local optimality conditions for a DC program

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset \tag{5.12}$$

Writing the problem (5.7) in form of a DC program

$$\min_{x,y}\{F_{r_\theta}(x,y) := G(x,y) - H(x,y)\}, \tag{5.13}$$

with

$$G(x,y) = \chi_K(x,y) + g(x,y) + \lambda\sum_{i=1}^{n}\varphi_\theta(x_i),\ H(x,y) = h(x,y) + \lambda\sum_{i=1}^{n}\psi_\theta(x_i). \tag{5.14}$$

Then, for a point $(x^*, y^*) \in K$, the necessary local optimality condition (5.12) can be expressed as

$$0 \in \partial G(x^*, y^*) - \partial H(x^*, y^*),$$

which is equivalent to

$$\langle \overline{x}^*, x - x^*\rangle + \langle \overline{y}^*, y - y^*\rangle + \langle \overline{z}^*, x - x^*\rangle \geq 0 \quad \forall (x,y) \in K, \tag{5.15}$$

for some $(\overline{x}^*, \overline{y}^*) \in \partial f(x^*, y^*)$ and $\overline{z}_i^* \in \lambda\partial r_\theta(x_i^*)\ \ \forall i = 1, \ldots, n$.

Now we are able to state consistency results of local optimality.

Theorem 5.2 *Let $\mathcal{L}$ and $\mathcal{L}_\theta$ be the sets of $(x,y) \in K$ satisfying the conditions* (5.11) *and* (5.15) *respectively.*

i) Let $\{\theta_k\}$ be a sequence of nonnegative numbers such that $\theta_k \to +\infty$ and $\{(x^k, y^k)\}$ be a sequence such that $(x^k, y^k) \in \mathcal{L}_{\theta_k}, \forall k$. If $(x^k, y^k) \to (x^*, y^*)$, we have $(x^*, y^*) \in \mathcal{L}$.

ii) If K is compact then, for any $\epsilon > 0$, there is $\theta(\epsilon) > 0$ such that

$$\mathcal{L}_\theta \subset \mathcal{L} + B(0,\epsilon) \quad \forall \theta \geq \theta(\epsilon).$$

iii) If there is a finite set $\mathcal{S}$ such that $\mathcal{L}_\theta \cap \mathcal{L} \neq \emptyset, \forall \theta > 0$, then there exists $\theta_0 \geq 0$ such that

$$\mathcal{L}_\theta \cap \mathcal{S} \subset \mathcal{L} \quad \forall \theta \geq \theta_0.$$

Proof 5.5 *i) By definition, there is a sequence $\{(\overline{x}^k, \overline{y}^k, \overline{z}^k)\}$ such that for all $k = 1, 2, \ldots$*

$$(\overline{x}^k, \overline{y}^k) \in \partial f(x^k, y^k),\ \text{and}\ \overline{z}_i^k \in \lambda\partial r_{\theta_k}(x_i^k)\ \ i = 1, \ldots, n,$$

$$\langle \overline{x}^k, x - x^k\rangle + \langle \overline{y}^k, y - y^k\rangle + \langle \overline{z}^k, x - x^k\rangle \geq 0 \quad \forall (x,y) \in K. \tag{5.16}$$

For $k = 1, 2, \ldots,$ we have

$$(\overline{x}^k, \overline{y}^k) = (x_g^k, y_g^k) - (x_h^k, y_h^k),$$

where $(x_g^k, y_g^k) \in \partial g(x^k, y^k)$, and $(x_h^k, y_h^k) \in \partial h(x^k, y^k)$.

Since $\{(x^k, y^k)\}$ converges to (x^*, y^*), there is $k_0 \in \mathbb{N}$ and a compact set $\mathcal{S} \subset \mathbb{R}^n \times \mathbb{R}^m$ such that $(x^k, y^k) \in \mathcal{S},\ \forall k \geq k_0$. It follows by Theorem 24.7 ([206]) that $\partial g(\mathcal{S}) := \cup_{x\in\mathcal{S}}\partial g(x)$ and $\partial h(\mathcal{S}) := \cup_{x\in\mathcal{S}}\partial h(x)$ are compact sets. Thus, there is an infinite set $\mathcal{K} \subset \mathbb{N}$ such that the sequence $\{(x_g^k, y_g^k)\}_{k\in\mathcal{K}}$ converges to a point $(x_g^*, y_g^*) \in \partial g(\mathcal{S})$ and the sequence $\{(x_h^k, y_h^k)\}_{k\in\mathcal{K}}$

converges to a point $(x_h^*, y_h^*) \in \partial h(\mathcal{S})$. *By Theorem 24.4 ([206]), we have* $(x_g^*, y_g^*) \in \partial g(x^*, y^*)$ *and* $(x_h^*, y_h^*) \in \partial h(x^*, y^*)$. *Therefore, the sequence* $\{(\overline{x}^k, \overline{y}^k)\}_{k\in\mathcal{K}}$ *converges to* $(\overline{x}^*, \overline{y}^*) = (x_g^*, y_g^*) - (x_h^*, y_h^*) \in \partial f(x^*, y^*)$.

By Assumption 5.1 iv), we have $\overline{z}_i^k x_i^k \geq 0\ \forall i, k$. *Moreover, for any* $i \in supp(x^*)$, *there exist* $a_i \leq b_i$ *and* $k_i \in \mathbb{N}$ *such that* $0 \notin [a_i, b_i]$ *and* $x_i^k \in [a_i, b_i]$ *for all* $k \geq k_i$. *By Assumption 5.1 v), we deduce that* $\overline{z}_i^k \to 0$ *as* $k \to +\infty$.

For arbitrary $(x, y) \in K(x^*)$, (5.16) *implies that*

$$\begin{aligned}\langle \overline{x}^k, x - x^k\rangle + \langle \overline{y}^k, y - y^k\rangle &\geq \sum_{i\notin supp(x^*)} \overline{z}_i^k x_i^k - \sum_{i\in supp(x^*)} \overline{z}_i^k (x_i - x_i^k)\\ &\geq -\sum_{i\in supp(x^*)} \overline{z}_i^k (x_i - x_i^k) \quad \forall k.\end{aligned}$$

Taking $k \in \mathcal{K}, k \to +\infty$, *we get*

$$\langle \overline{x}^*, x - x^*\rangle + \langle \overline{y}^*, y - y^*\rangle \geq 0 \quad \forall (x, y) \in K(x^*).$$

Thus, $(x^*, y^*) \in \mathcal{L}$.

ii) and iii) are proved similarly as in Theorem 5.1. ■

5.2.2 DC approximation functions

First, let us mention, in chronological order, the approximation functions proposed in the literature in different contexts, but we don't indicate the related works concerning algorithms using these approximations). The first was concave exponential approximation proposed in [35] in the context of feature selection in SVM, and ℓ_p-norm with $0 < p < 1$ for sparse regression ([76]). Later, the ℓ_p-norm with $p < 0$ was studied in [203] for sparse signal recovery, and then the Smoothly Clipped Absolute Deviation (SCAD) [70] in the context of regression, the logarithmic approximation [235] for feature selection in SVM, and the Capped-ℓ_1 ([186]) applied on sparse regression.

A common property of these approximations is they are all even, concave increasing functions on $[0, +\infty)$. It is easy to verify that these function satisfy the conditions in Assumption 1 and so they are particular cases of our DC approximation r. More general DC approximation functions are also investigated, e.g., PiL ([122]) that is a (nonconcave) piecewise linear function defined in Table 5.1.

Note that, some of these approximation functions, namely logarithm (log), SCAD and ℓ_p-norm defined by

$$Log : \log(|t| + \epsilon),\ \epsilon > 0, \quad \ell_p : \text{sign}(p)(|t| + \epsilon)^p,\ 0 \neq p \leq 1, \epsilon > 0; \tag{5.17}$$

$$SCAD : \begin{cases} \gamma|t| & \text{if } 0 \leq |t| \leq \gamma, \quad \frac{(a+1)\gamma^2}{2} \text{if } |t| \geq a\gamma \\ \frac{-t^2 + 2a\gamma|t| - \gamma^2}{2(a-1)} & \text{if } \gamma < |t| < a\gamma \end{cases} \quad , a > 1, \gamma > 0 \tag{5.18}$$

do not directly approximate ℓ_0-norm. But they become approximations of ℓ_0-norm if we multiply them by an appropriate factor (which can be incorporated into the parameter λ), and add an appropriate term (such a procedure doesn't affect the original problem). The resulting approximation forms of these functions are given in Table 5.1. We see that r_{scad} is obtained by

TABLE 5.1 – ℓ_0-approximation functions r and the first DC decomposition φ. The second DC decomposition is $\psi = \varphi - r$.

Approximation	Function r	Function φ
Exp ([35])	$r_{exp}(t) = 1 - e^{-\theta\|t\|}$	$\theta\|t\|$
$\ell_p(0 < p < 1)$([76])	$r_{\ell_p^+}(t) = (\|t\| + \epsilon)^{1/\theta}$	$\frac{\epsilon^{1/\theta-1}}{\theta}\|t\|$
$\ell_p(p < 0)$([203])	$r_{\ell_p^-}(t) = 1 - (1 + \theta\|t\|)^p$, $p < 0$	$-p\theta\|t\|$
Log ([235])	$r_{log}(t) = \frac{\log(1+\theta\|t\|)}{\log(1+\theta)}$	$\frac{\theta}{\log(1+\theta)}\|t\|$
SCAD ([70])	$r_{scad}(t) = \begin{cases} \frac{2\theta}{a+1}\|t\| & 0 \le \|t\| \le \frac{1}{\theta} \\ \frac{-\theta^2t^2+2a\theta\|t\|-1}{a^2-1} & \frac{1}{\theta} < \|t\| < \frac{a}{\theta} \\ 1 & \|t\| \ge \frac{a}{\theta} \end{cases}$	$\frac{2\theta}{a+1}\|t\|$
Capped-ℓ_1 ([186])	$r_{cap}(t) = \min\{1, \theta\|t\|\}$	$\theta\|t\|$
PiL [122]	$r_{PiL} = \min\left\{1, \max\left\{0, \frac{\theta\|t\|-1}{a-1}\right\}\right\}$	$\frac{\theta}{a-1}\max\left\{\frac{1}{\theta}, \|t\|\right\}$

multiplying the SCAD function by $\frac{2}{(a+1)\gamma^2}$ and setting $\theta = \frac{1}{\gamma}$. Similarly, by taking $\theta = \frac{1}{\epsilon}$, we have

$$r_{log}(t) = \frac{\log(|t| + \epsilon)}{\log(1 + 1/\epsilon)} - \frac{\log \epsilon}{\log(1 + 1/\epsilon)}, \text{ and } r_{\ell_p^-}(t) = -\frac{(|t| + \epsilon)^p}{\epsilon^p} + 1.$$

For using ℓ_p-norm approximation with $0 < p < 1$, we take $\theta = \frac{1}{p}$. Note that $\lim_{\theta\to\infty} |t|^{1/\theta} = s(t)$. To avoid singularity at 0, we add a small $\epsilon > 0$. In this case, we require $\epsilon = \epsilon(\theta)$ satisfying $\lim_{\theta\to\infty} \epsilon(\theta)^{1/\theta} = 0$ to ensure that $\lim_{\theta\to\infty} r_{\ell_p^+}(t) = s(t)$.

All these functions satisfy Assumption 5.1 (for proving the condition iii) of Assumption 5.1 we indicate in Table 5.1 a DC decomposition of the approximation functions), so the consistency results stated in Theorems 5.1 and 5.2 are applicable.

Discussion. Except r_{PiL} that is differentiable at 0 with $r'_{PiL}(0) = 0$, the other approximations have the right derivative at 0 depending on the approximation parameter θ. Clearly the tightness of each approximation depends on related parameters. Hence, a suitable way to compare them is using the parameter θ such that their right derivatives at 0 are equal, namely

$$\theta_{cap} = \frac{2}{a+1}\theta_{scad} = \theta_{exp} = -p\theta_{\ell_p^-}.$$

In this case, by simple calculation we have

$$0 \le r_{\ell_p^-} \le r_{exp} \le r_{scad} \le r_{cap} \le s. \tag{5.19}$$

Comparing r_{cap} and r_{scad} with different values θ, we get

$$\begin{cases} 0 \le r_{scad} \le r_{cap} \le s, & \text{if } \frac{2\theta_{scad}}{a+1} \le \theta_{cap} \\ 0 \le r_{cap} \le r_{scad} \le s, & \text{if } \theta_{cap} \le \frac{\theta_{scad}}{a}. \end{cases} \tag{5.20}$$

Inequalities in (5.19) show that, with the parameter θ such that their right derivatives at 0 are equal, r_{scad} and r_{cap} are closer to the step function s than $r_{\ell_p^-}$ and r_{exp}.

As for r_{log} and $r_{\ell_p^+}$, we see that they tend to $+\infty$ when $t \to +\infty$, so they have poor approximation for t large. Whereas, the other approximations are minorants of s and larger t is, closer to s they are. For easier seeing, we depict these approximations in Figure 5.1.

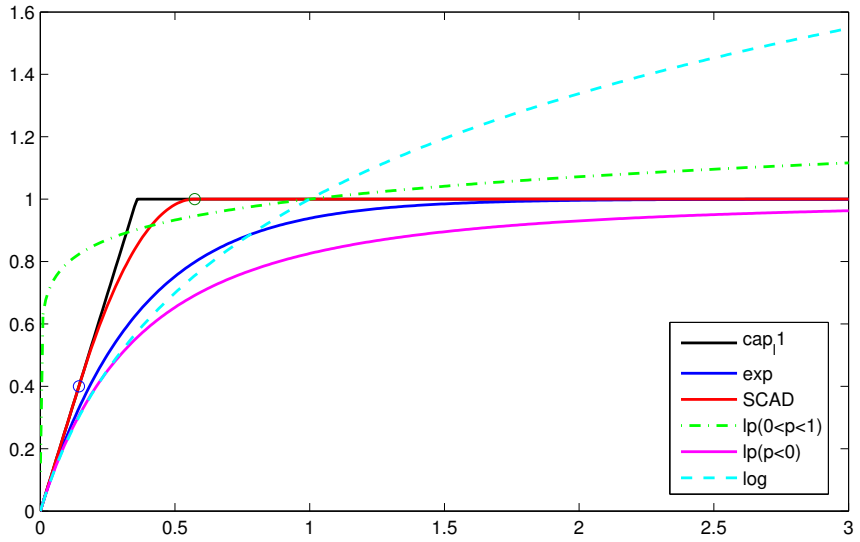


FIGURE 5.1 – Graphs of approximation functions. Except ℓ_p-norm$(0 < p < 1)$ and PiL, the others have the same derivative at 0. Here $\theta_{log} = 10$ for Log, $a = 4$ for SCAD, $p = -2$ for ℓ_p-norm$(p < 0)$. For ℓ_p-norm$(0 < p < 1)$, $\epsilon = 10^{-9}$ and $p = 0.2$. For PiL, $a = 5$ and $\theta_{PiL} = a\theta_{exp}$.

Now, we give a deeper study on Capped-ℓ_1 approximation. Using exact penalty techniques related to ℓ_0-norm developed in Section 5.3 we prove a much stronger result for this approximation, that is the approximation problem (5.7) is equivalent to the original problem with appropriate parameters θ when K is a compact polyhedral convex set (this case quite often occurs in applications, in particular in machine learning contexts). Furthermore, when K is a box, we show (directly, without using the exact penalty techniques) that the Capped-ℓ_1 approximation problem is equivalent to the original problem and we compute an exact value θ_0 such that the equivalence holds for all $\theta > \theta_0$.

5.2.3 A deeper study on Capped-ℓ_1 approximation problem

5.2.3.1 Link between the continuous exact formulation and Capped-ℓ_1 approximation problem

The Capped-ℓ_1 approximation is defined by :

$$\Psi_\theta(x) := \sum_{i=1}^{n} r_{cap}(x_i), \forall x = (x_i) \in \mathbb{R}^n, \text{ with } r_{cap}(t) := \min\{\theta\,|t|\,, 1\},\ t \in \mathbb{R}. \tag{5.21}$$

Hence, the approximate problem of (5.1) using Capped-ℓ_1 approximation is written as

$$\beta(\theta) := \inf\left\{ f(x,y) + \lambda \sum_{i=1}^{n} r_{cap}\,(x_i) : (x,y) \in K \right\}. \tag{5.22}$$

We will demonstrate that the problem (5.22) is equivalent to the continuous exact reformulation of (5.1) with suitable values of parameters λ, τ and θ. The complete detail of the exact reformulation approach is presented in the Section 5.3. For the reader convenience, let us briefly give

the continuous exact reformulation of (5.1). Define the binary variable $u_i \in \{0,1\}$ as

$$u_i = |x_i|_0 = \begin{cases} 1 \text{ if } x_i \neq 0 \\ 0 \text{ if } x_i = 0, \end{cases} \qquad \forall i = 1...n.$$

and $p(u) := \sum_{i=1}^{n} \min\{u_i, 1-u_i\}$, the continuous exact reformulation of (5.1) is given by (τ being the positive penalty parameter)

$$\alpha(\tau) := \inf\{f(x,y) + \lambda e^T u + \tau p(u) : \quad (x,y) \in K, u \in [0,1]^n. |x_i| \leq c_i u_i, \ i = 1, ..., n\}, \quad (5.23)$$

We now establish the link between the continuous exact reformulation (5.23) and the Capped-ℓ_1 approximation problem.

Let $M = \max\{c_i : i = 1, \ldots, n\}$, consider the problem (5.23) in the form

$$\alpha(\tau) := \inf\{f(x,y) + \lambda e^T u + \tau p(u) : (x,y) \in K, u \in [0,1]^n, |x_i| \leq M u_i, \ i = 1, \ldots, n\}. \quad (5.24)$$

Let $\varsigma : \mathbb{R} \to \mathbb{R}$ be the function defined by $\varsigma(t) = \min\{t, 1-t\}$. Then $p(u) = \sum_{i=1}^{n} \varsigma(u_i)$ and the problem (5.24) can be rewritten as

$$\alpha(\tau) := \inf\left\{ f(x,y) + \lambda \sum_{i=1}^{n} \left(u_i + \frac{\tau}{\lambda} \varsigma(u_i) \right) : (x,y) \in K, \frac{|x_i|}{M} \leq u_i \leq 1, \ i = 1, \ldots, n \right\}, \quad (5.25)$$

or again

$$\alpha(\tau) := \inf\left\{ f(x,y) + \lambda \sum_{i=1}^{n} \pi\left(u_i\right) : (x,y) \in K, \frac{|x_i|}{M} \leq u_i \leq 1, \ i = 1, \ldots, n \right\} \quad (5.26)$$

where $\pi : \mathbb{R} \to \mathbb{R}$ be the function defined by $\pi(t) := t + \frac{\tau}{\lambda}\varsigma(t)$.

Proposition 5.3 *Let $\theta := \frac{\tau+\lambda}{\lambda M}$. For all $\tau \geq \lambda$ problems (5.26) and (5.22) are equivalent in the following sense : (x^*, y^*) is an optimal solution of (5.22) iff (x^*, y^*, u^*) is an optimal solution of (5.26), where $u_i^* \in \left\{ \frac{|x_i^*|}{M}, 1 \right\}$ such that $\pi(u_i^*) = r_{cap}(x_i^*)$ for $i = 1, \ldots, n$. Moreover, $\alpha(\tau) = \beta(\theta)$.*

Proof 5.6 *If (x^*, y^*, u^*) is an optimal solution of (5.26), then u_i^* is an optimal solution of the following problem, for every $i = 1, \ldots, n$*

$$\min\left\{ \pi(u_i) : \frac{|x_i^*|}{M} \leq u_i \leq 1 \right\}. \quad (5.27)$$

Since ς is a concave function, so is π. Consequently

$$\min\left\{ \pi(u_i) : \frac{|x_i^*|}{M} \leq u_i \leq 1 \right\} = \min\left\{ \pi\left(\frac{|x_i^*|}{M} \right), \pi(1) \right\} = \min\left\{ \left(1 + \frac{\tau}{\lambda}\right) \frac{|x_i^*|}{M}, 1 \right\} = r_{cap}(x_i^*).$$

For an arbitrary $(x,y) \in K$, we will show that

$$f(x^*, y^*) + \lambda \sum_{i=1}^{n} r_{cap}(x_i^*) \leq f(x,y) + \lambda \sum_{i=1}^{n} r_{cap}(x_i). \quad (5.28)$$

By the assumption that (x^*, y^*, u^*) *is an optimal solution of* (5.26)*, we have*

$$f(x^*, y^*) + \lambda \sum_{i=1}^{n} \pi(u_i^*) \leq f(x, y) + \lambda \sum_{i=1}^{n} \pi(u_i) \tag{5.29}$$

for any feasible solution (x, y, u) *of* (5.26)*. Let*

$$u_i^x \in \arg\min \left\{ \pi(\xi) \ : \xi \in \left\{ \frac{|x_i|}{M}, 1 \right\} \right\} \subset \arg\min \left\{ \pi(\xi) : \frac{|x_i|}{M} \leq \xi \leq 1 \right\},$$

for all $i = 1, \ldots, n$*. Then* (x, y, u^x) *is a feasible solution of* (5.24) *and*

$$\pi(u_i^x) = \min \left\{ \pi(\xi) : \frac{|x_i|}{M} \leq \xi \leq 1 \right\} \ = r_{cap}(x_i), \quad \forall i = 1, \ldots, n.$$

Combining (5.29) in which u_i *is replaced by* u_i^x *and the last equation we get (5.28), which implies that* (x^*, y^*) *is an optimal solution of* (5.22)*.*

Conversely, if (x^*, y^*) *is a solution of* (5.22)*, and let* $u_i^* \in \left\{ \frac{|x_i^*|}{M}, 1 \right\}$ *such that* $\pi(u_i^*) = r_{cap}(x_i^*)$ *for* $i = 1, \ldots, n$*. Then* (x^*, y^*, u^*) *is a feasible solution of* (5.26) *and for an arbitrary feasible solution* (x, y, u) *of* (5.26)*, we have*

$$\begin{aligned} f(x, y) + \lambda \sum_{i=1}^{n} \pi(u_i) &\geq f(x, y) + \lambda \sum_{i=1}^{n} r_{cap}(x_i) \\ \geq f(x^*, y^*) + \lambda \sum_{i=1}^{n} r_{cap}(x_i^*) &= f(x^*, y^*) + \lambda \sum_{i=1}^{n} \pi(u_i^*). \end{aligned}$$

Thus, (x^*, y^*, u^*) *is an optimal solution of* (5.26)*. The equality* $\alpha(\tau) = \beta(\theta)$ *is immediately deduced from the equality* $\pi(u_i^*) = r_{cap}(x_i^*)$*.* ■

We conclude from the above results that for $\theta = \frac{\tau+\lambda}{\lambda M}$ with $\tau > \max\{\lambda, \tau_0\}$, or equivalently $\theta > \theta_0 := \max\{\frac{2}{M}, \frac{\tau_0+\lambda}{\lambda M}\}$, the approximate problem (5.22) is equivalent to the origial problem (5.1). The result justifies the goodness of the Capped-ℓ_1 approximation studied in Section 5.2.2 above.

5.2.3.2 Link between the original problem (5.1) and Capped-ℓ_1 approximation problem

In particular, for a special structure of K, we get the following result.

Proposition 5.4 *Suppose that* $K = \prod_{i=1}^{n}[-l_i, l_i] \times Y$ $(0 \leq l_i \leq +\infty \ \forall i, Y \subset \mathbb{R}^m)$ *and* $\kappa > 0$ *is a constant satisfying*

$$|f(x, y) - f(x', y)| \leq \kappa \|x - x'\|_2 \quad \forall (x, y), (x', y) \in K, \|x - x'\|_0 \leq 1. \tag{5.30}$$

Then for $\theta > \frac{\kappa}{\lambda}$*, the problems* (5.1) *and* (5.22) *are equivalent.*

Proof 5.7 *We observe that if* $(x, y) \in K$ *such that* $0 < |x_{i_0}| < \frac{1}{\theta}$ *for some* i_0*, let* $(x', y) \in K$ *determined by* $x_i' = x_i \ \forall i \neq i_0$ *and* $x_{i_0}' = 0$*, then*

$$f(x, y) + \lambda\Phi(x) > f(x', y) + \lambda\Phi(x'),$$

where $\Phi(x) = \sum_{i=1}^{n} r_{cap}(x_i)$. *Indeed, this inequality follows the facts that*

$$|f(x,y) - f(x',y)| \leq \kappa\|x - x'\| = \kappa|x_{i_0}|$$

and

$$\Phi(x) - \Phi(x') = r_{cap}(x_{i_0}) = \theta|x_{i_0}| > \frac{\kappa}{\lambda}|x_{i_0}|.$$

For $x \in \mathbb{R}^n$, *we define* $t^x \in \mathbb{R}^n$ *by* $t_i^x = 0$ *if* $|x_i| < \frac{1}{\theta}$ *and* $t_i^x = x_i$ *otherwise. By applying the above observation, for any* $(x,y) \in K$, *we have*

$$f(x,y) + \lambda\Phi(x) \geq f(t^x,y) + \lambda\Phi(t^x).$$

The equality holds iff $|x_i| \geq \frac{1}{\theta}\ \forall i \in \mathrm{supp}(x)$.

Therefore, if (x^*,y^*) *is a solution of* (5.22), *we have* $|x_i^*| \geq \frac{1}{\theta}\ \forall i \in \mathrm{supp}(x^*)$. *Then, for any* $(x,y) \in K$,

$$f(x,y) + \lambda\|x\|_0 \geq f(x,y) + \lambda\Phi(x) \geq f(x^*,y^*) + \lambda\Phi(x^*) = f(x^*,y^*) + \lambda\|x^*\|_0.$$

This means that (x^*,y^*) *is a solution of* (5.1).

Conversely, assume that (x^*,y^*) *is a solution of* (5.1). *Then for any* $(x,y) \in K$, *we have*

$$\begin{aligned} f(x,y) + \lambda\Phi(x) &\geq f(t^x,y) + \lambda\Phi(t^x) = f(t^x,y) + \lambda\|t^x\|_0 \\ &\geq f(x^*,y^*) + \lambda\|x^*\|_0 \geq f(x^*,y^*) + \lambda\Phi(x^*). \end{aligned}$$

Thus, (x^*,y^*) *is a solution of* (5.22). ■

For the problem of feature selection in SVM, we consider the loss function

$$f(x,b) = (1-\lambda)\left(\frac{1}{N_A}\|\max\{0, -Ax + eb + e\}\|_1 + \frac{1}{N_B}\|\max\{0, Bx - eb + e\}\|_1\right),$$

It is easy to prove that for $u \in \mathbb{R}^n$, $\iota \in \mathbb{R}$ and $i \in \{1,\ldots,n\}$, we have

$$|\max\{0, \langle u,x\rangle + \iota\} - \max\{0, \langle u,x'\rangle + \iota\}| \leq |u|_i|x_i - x_i'|,$$

for all $x, x' \in \mathbb{R}^n$ such that $x_j = x_j'\ \forall j \neq i$. Therefore, for

$$\kappa = (1-\lambda)\max_{i=1,\ldots,n}\left\{\frac{1}{N_A}\sum_{k=1}^{N_A}|A_{ki}| + \frac{1}{N_B}\sum_{l=1}^{N_B}|B_{li}|\right\},$$

we have

$$|f(x,b) - f(x',b)| \leq \kappa\|x - x'\|, \quad \forall b \in \mathbb{R}, \forall x, x' \in \mathbb{R}^n \text{ s.t. } \|x - x'\|_0 \leq 1.$$

By virtue of Proposition 5.4, in the case of feature selection in SVM, for $\theta > \theta^* := \frac{\kappa}{\lambda}$, the problems (5.1) and (5.22) are equivalent.

5.2.4 Extension to other approximations

Proposition 5.5 *i) Suppose that σ is a function on $\mathbb{R}$ satisfying*

$$r_{cap}(t) \leq \sigma(t) \leq s(t) = \begin{cases} 0, & \text{if } t = 0, \\ 1, & \text{otherwise,} \end{cases}$$

for some $\theta_{cap} > \theta_0$. Then, the problems (5.1) *and*

$$\inf\{f(x,y) + \lambda \sum_{i=1}^{n} \sigma(x_i) : (x,y) \in K\} \tag{5.31}$$

are equivalent.
ii) In particular, if $\theta_{scad} > a\theta_0$ then for all $\tau \geq \lambda$ the approximate problem

$$\inf\{f(x,y) + \lambda \sum_{i=1}^{n} r_{scad}(x_i) : (x,y) \in K\}$$

is equivalent to (5.1).

Proof 5.8 *As discussed before, since $\theta_{cap} > \theta_0$, the problems* (5.1) *and* (5.22) *are equivalent. Moreover, if (x^*, y^*) is a common solution then*

$$f(x^*, y^*) + \lambda \sum_{i=1}^{n} r_{cap}(x_i^*) = f(x^*, y^*) + \lambda \|x^*\|_0.$$

Then i) is trivial by the fact that

$$f(x,y) + \lambda \sum_{i=1}^{n} r_{cap}(x_i) \leq f(x,y) + \lambda \sum_{i=1}^{n} \sigma(x_i) \leq f(x,y) + \lambda \|x\|_0, \quad \forall (x,y).$$

ii) is a direct consequence of i) and Propositions 5.3 and (5.20). ■

5.2.5 DCA for solving the approximate problem (5.7)

In the sequel, we will omit the parameter θ when this doesn't cause any ambiguity.

Usual sparsity-inducing functions are concave, increasing on $[0, +\infty)$. Therefore, first we present three variants of DCA for solving the problem (5.7) when r is concave on $[0, +\infty)$. We also suppose that r has the right derivative at 0, denoted by $r'(0)$, so $\partial(-r)(0) = \{-r'(0)\}$.

First, we consider the approximate problem (5.7).

5.2.5.1 The first DCA scheme for solving the approximate problem (5.7)

We propose the following DC decomposition of r :

$$r(t) = \eta|t| - (\eta|t| - r(t)) \quad \forall t \in \mathbb{R}, \tag{5.32}$$

where η is a positive number such that $\psi(t) = \eta|t| - r(t)$ is convex. The next result gives a sufficient condition for the existence of such a η.

Proposition 5.6 *Suppose that r is a concave function on $[0,+\infty)$ and the (right) derivative at 0, $r'(0)$, is well-defined. Let $\eta \geq r'(0)$. Then $\psi(t) = \eta|t| - r(|t|)$ is a convex function on $\mathbb{R}$.*

Proof 5.9 *Since r is concave on $[0,+\infty)$, the function $\eta|t| - r(t)$ is convex on $(0,+\infty)$ and on $(-\infty,0)$. Hence it suffices to prove that for any $t_1 > 0, t_2 < 0$ and $\alpha, \beta \in (0,1)$ such that $\alpha + \beta = 1$, we have*

$$\psi\left(\alpha t_1 + \beta t_2\right) \leq \alpha\psi(t_1) + \beta\psi(t_2). \tag{5.33}$$

Without loss of generality, we assume that $\alpha|t_1| \geq \beta|t_2|$. Then (5.33) is equivalent to

$$\eta(\alpha|t_1| - \beta|t_2|) - 2r\left(\alpha|t_1| - \beta|t_2|\right) \leq \eta(\alpha|t_1| + \beta|t_2|) - \alpha r(|t_1|) - \beta r(|t_2|)$$

which can be equivalently written as

$$\alpha r(|t_1|) + \beta r(|t_2|) - r\left(t_0\right) \leq 2\eta\beta|t_2|, \tag{5.34}$$

where $t_0 = \alpha|t_1| - \beta|t_2| \geq 0$. Let $\mu \in \mathbb{R}$ such that $-\mu \in \partial(-r(t_0))$. Since r is concave on $[0,+\infty)$, we have

$$\alpha r(|t_1|) + \beta r(|t_2|) - r\left(t_0\right) \leq r\left(\alpha|t_1| + \beta|t_2|\right) - r\left(t_0\right) \leq 2\mu\beta|t_2|.$$

Hence (5.34) holds when $\mu \leq \eta$. By the concavity of r, we have

$$r\left(\frac{t_0}{2}\right) \leq r(0) + r'(0)\frac{t_0}{2}, \quad \textit{and} \quad r\left(\frac{t_0}{2}\right) \leq r(t_0) - \mu\frac{t_0}{2},$$

therefore

$$(z - r'(0))t_0 \leq r(0) + r(t_0) - 2r\left(\frac{t_0}{2}\right) \leq 0.$$

This and the condition $r'(0) \leq \eta$ imply that $\mu \leq r'(0) \leq \eta$. The proof is then complete. ■

With $\eta \geq r'(0)$, a DC formulation of the problem (5.7) is given by

$$\min_{x,y}\{F_r(x,y) := G_1(x,y) - H_1(x,y)\}, \tag{5.35}$$

where

$$G_1(x,y) = \chi_K(x,y) + g(x,y) + \lambda\eta\|x\|_1,\ H_1(x,y) = h(x,y) + \lambda\sum_{i=1}^{n}\left(\eta|x_i| - r(x_i)\right),$$

and g, h are DC components of f.

By the definition $\psi(t) = \eta|t| - r(t)\ \forall t \in \mathbb{R}$, we have

$$\partial\psi(t) = \eta + \partial(-r)(t) \text{ if } t > 0,\ -\eta - \partial(-r)(-t) \text{ if } t < 0,\ \left[-\eta + r'(0), \eta - r'(0)\right] \text{ if } t = 0. \tag{5.36}$$

Following the generic DCA scheme, DCA applied on (5.35) is given by Algorithm 5.1 below.

Algorithm 5.1 DCA1 - The first DCA scheme for solving (5.7)

1: **Initialize** $(x^0, y^0) \in K$, $k \leftarrow 0$
2: **repeat**
3: Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$ and $\overline{z}_i^k \in \lambda\partial\psi(x_i^k)\ \forall i = 1, \ldots, n$ via (5.36).
4: Compute

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K}\left\{g(x,y) - \langle\overline{x}^k, x\rangle - \langle\overline{y}^k, y\rangle + \lambda\eta\|x\|_1 - \langle\overline{z}^k, x\rangle\right\}$$

5: $k \leftarrow k + 1$.
6: **until** Stopping criterion

TABLE 5.2 – Choice of η and expression of $\overline{z}_i^k \in \lambda\partial\psi(x_i^k)$ in Algorithm 5.1 and related works.

r	η	$\overline{z}_i^k \in \lambda\partial\psi(x_i^k)$	Related works	Context
r_{exp}	θ	$\text{sign}(x_i^k)\lambda\theta\left(1-e^{-\theta\lvert x_i^k\rvert}\right)$	[128] [180]	Feature selection in SVMs Learning sparse classifiers
$r_{\ell_p^+}$	$\frac{\epsilon^{1/\theta-1}}{\theta}$	$\text{sign}(x_i^k)\frac{\lambda}{\theta}\left[\epsilon^{1/\theta-1}-(\lvert x_i^k\rvert+\epsilon)^{1/\theta-1}\right]$		
$r_{\ell_p^-}$	$-p\theta$	$-\text{sign}(x_i^k)\lambda p\theta\left[1-(1+\theta\lvert x_i^k\rvert)^{p-1}\right]$		
r_{log}	$\frac{\theta}{\log(1+\theta)}$	$\text{sign}(x_i^k)\frac{\lambda\theta^2\lvert x_i^k\rvert}{\log(1+\theta)(1+\theta\lvert x_i^k\rvert)}$		
r_{scad}	$\frac{2\theta}{a+1}$	$\begin{cases}0 & \lvert x_i^k\rvert\le\frac{1}{\theta}\\ \text{sign}(x_i^k)\frac{2\lambda\theta(\theta\lvert x_i^k\rvert-1)}{a^2-1} & \frac{1}{\theta}<\lvert x_i^k\rvert<\frac{a}{\theta}\\ \text{sign}(x_i^k)\frac{2\lambda\theta}{a+1} & \text{otherwise}\end{cases}$	[140]	Feature selection in SVMs
r_{cap}	θ	$\begin{cases}0 & \lvert x_i^k\rvert\le\frac{1}{\theta}\\ \text{sign}(x_i^k)\lambda\theta & \text{otherwise}\end{cases}$	[180]	Learning sparse classifiers

Instances of Algorithm 1 can be found in our previous works [128, 140, 180] using exponential concave, SCAD or Capped$-\ell_1$ approximations (see Table 5.2). Note that for usual sparse inducing functions given in Table 5.2, this DC decomposition is nothing but that given in Table 5.1, i.e. $\varphi(t)=\eta|t|$.

Now we consider the approximate problem (5.8) and introduce a DCA scheme that includes all standard algorithms of reweighted-ℓ_1-type for sparse optimization problem (5.1).

5.2.5.2 DCA2 - Relation with reweighted-ℓ_1 procedure

The problem (5.8) can be written as a DC program as follows

$$\min_{x,y,z}\{\overline{F}_r(x,y,z) := G_2(x,y,z)-H_2(x,y,z)\}, \tag{5.37}$$

where

$$G_2(x,y,z)=\chi_{\Omega_1}(x,y,z)+g(x,y),\ H_2(x,y,z)=h(x,y)+\lambda\sum_{i=1}^{n}(-r)(z_i),$$

and g,h are DC components of f as stated in (5.6).

Assume that $(x^k,y^k,z^k)\in\Omega_1$ is the current solution at iteration k. DCA applied to DC program (5.37) updates $(x^{k+1},y^{k+1},z^{k+1})\in\Omega_1$ via two steps :

- Step 1 : compute $(\overline{x}^k,\overline{y}^k)\in\partial h(x^k,y^k)$, and $\overline{z}_i^k\in\lambda\partial(-r)(z_i^k)\ \ \forall i=1,\ldots,n$.
- Step 2 : compute

$$\begin{aligned}(x^{k+1},y^{k+1},z^{k+1}) &\in \arg\min\left\{G_2(x,y,z)-\langle\overline{x}^k,x\rangle-\langle\overline{y}^k,y\rangle-\langle\overline{z}^k,z\rangle\right\}\\ &= \arg\min_{(x,y,z)\in\Omega_1}\left\{g(x,y)-\langle\overline{x}^k,x\rangle-\langle\overline{y}^k,y\rangle+\langle-\overline{z}^k,z\rangle\right\}.\end{aligned}$$

Since r is increasing, we have $-\overline{z}^k\ge 0$. Thus, updating $(x^{k+1},y^{k+1},z^{k+1})$ can be done as follows

$$\begin{cases}(x^{k+1},y^{k+1})\in\arg\min_{(x,y)\in K}\left\{g(x,y)-\langle\overline{x}^k,x\rangle-\langle\overline{y}^k,y\rangle+\langle-\overline{z}^k,|x|\rangle\right\}\\ z_i^{k+1}=|x_i^{k+1}|\ \ \forall i.\end{cases}$$

DCA for solving the problem (5.8) can be described as in Algorithm 5.2 below.

Algorithm 5.2 DCA2 - DCA for solving (5.8)

1: **Initialize** $(x^0, y^0, z^0) \in \Omega_1$, $k \leftarrow 0$
2: **repeat**
3: Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$, $\overline{z}_i^k \in -\lambda\partial(-r)(z_i^k) \ \ \forall i = 1, \ldots, n.$
4: Compute

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{ g(x,y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \langle \overline{z}^k, |x|\rangle \right\}$$
$$z_i^{k+1} = |x_i^{k+1}| \ \forall i = 1, \ldots, n.$$

5: $k \leftarrow k + 1$.
6: **until** Stopping criterion

Relation with reweighted-ℓ_1 procedure. If the function f in (5.1) is convex, we can chose DC components of f as $g = f$ and $h = 0$. Then $(\overline{x}^k, \overline{y}^k) = 0 \ \forall k$. In this case, the step 4 in Algorithm 5.2 becomes

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{ f(x,y) + \sum_{i=1}^{n} \overline{z}_i^k |x_i| \right\}. \tag{5.38}$$

We see that the problem (5.38) has the form of a ℓ_1-regularization problem but with different weights on components of $|x_i|$. So Algorithm 5.2 iteratively solves the weighted–ℓ_1 problem (5.38) with an update of the weights $\overline{z}_i^k$ at each iteration k. The expression of weights $\overline{z}_i^k$ according to approximation functions are given in Table 5.3.

The update rule (5.38) covers standard algorithms of reweighted–ℓ_1–type for sparse optimization problem (5.1) (see Table 5.3). Some algorithms such as the two–stage ℓ_1 ([256]) and the adaptive Lasso ([263]) only run in a few iterations (typically two iterations) and their reasonings bear a heuristic character. The reweighted–ℓ_1 algorithm proposed in [44] lacks of theoretical justification for the convergence.

Next, we introduce a slight perturbation of the formulation (5.7) and develop the third DCA scheme that includes existing algorithms of reweighted–ℓ_2–type for sparse optimization problem (5.1).

5.2.5.3 DCA3 - Relation with reweighted-ℓ_2 procedure

To avoid the singularity at 0 of the function $r(t^{1/2}), t \geq 0$, we add $\epsilon > 0$ and consider the perturbation problem of (5.7) which is defined by

$$\begin{cases} \min_{x,y} & \tilde{F}_r(x,y) := f(x,y) + \lambda \sum_{i=1}^{n} r((|x_i|^2 + \epsilon)^{1/2}) \\ s.t. & (x,y) \in K, \end{cases} \quad \epsilon > 0. \tag{5.39}$$

Clearly (5.39) becomes (5.7) when $\epsilon = 0$. The problem (5.39) is equivalent to

$$\min_{(x,y,z)\in\Omega_2} \hat{F}_r(x,y,z) := f(x,y) + \lambda \sum_{i=1}^{n} r((z_i + \epsilon)^{1/2}), \tag{5.40}$$

TABLE 5.3 – Expression of $\overline{z}_i^k$ in Algorithm 5.2 and relation with reweighted-ℓ_1 algorithms.

Function r	expression of $\overline{z}_i^k$	Related works	Context
r_{exp}	$\lambda\theta e^{-\theta z_i^k}$	SLA ([35])	Feature selection in SVMs
$r_{\ell_p^+}$	$\dfrac{\lambda}{\theta(z_i^k+\epsilon)^{1-1/\theta}}$	Adaptive Lasso ([263])	Linear regression
$r_{\ell_p^-}$	$-\lambda p\theta(1+\theta z_i^k)^{p-1}$		
r_{scad}	$\begin{cases} \frac{2\lambda\theta}{a+1} & \text{if } z_i^k \le \frac{1}{\theta} \\ 0 & \text{if } z_i^k \ge \frac{a}{\theta} \\ \frac{\lambda\theta(a-\theta z_i^k)}{a^2-1} & \text{otherwise} \end{cases}$	LLA (Local Linear Approximation) ([264])	
r_{cap}	$\begin{cases} \lambda\theta & \text{if } z_i^k \le 1/\theta \\ 0 & \text{otherwise} \end{cases}$	Two-stage ℓ_1 ([256])	
r_{log}	$\dfrac{\lambda\theta}{\log(1+\theta)(1+\theta z_i^k)}$	Adaptive Lasso ([263]) ; Reweighted ℓ_1 ([44])	Sparse signal reconstruction

where $\Omega_2 = \{(x,y,z) : (x,y) \in K; \quad |x_i|^2 \le z_i \ \forall i\}$. The last problem is a DC program of the form

$$\min_{x,y,z}\{\hat{F}_r(x,y,z) := G_3(x,y,z) - H_3(x,y,z)\}, \tag{5.41}$$

where

$$G_3(x,y,z) = \chi_{\Omega_2}(x,y,z) + g(x,y), \ H_3(x,y,z) = h(x,y) + \lambda\sum_{i=1}^{n}(-r)((z_i+\epsilon)^{1/2}),$$

and g, h are DC components of f as stated in (5.6). Note that, since the functions r and $(t+\epsilon)^{1/2}$ are concave, increasing on $[0,+\infty)$, $(-r)((t+\epsilon)^{1/2})$ is a convex function on $[0,+\infty)$.

Let $(x^k,y^k,z^k) \in \Omega_2$ be the current solution at iteration k. DCA applied to DC program (5.41) updates $(x^{k+1},y^{k+1},z^{k+1}) \in \Omega_2$ via two steps :

- Step 1 : compute $(\overline{x}^k,\overline{y}^k) \in \partial h(x^k,y^k)$, and $\overline{z}_i^k \in \frac{\lambda}{2(z_i^k+\epsilon)^{1/2}}\partial(-r)((z_i^k+\epsilon)^{1/2}) \ \ \forall i = 1,\ldots,n$.
- Step 2 : compute

$$\begin{aligned}(x^{k+1},y^{k+1},z^{k+1}) &\in \arg\min\Big\{G_3(x,y,z) - \langle\overline{x}^k,x\rangle - \langle\overline{y}^k,y\rangle - \langle\overline{z}^k,z\rangle\Big\} \\ &= \arg\min_{(x,y,z)\in\Omega_2}\Big\{g(x,y) - \langle\overline{x}^k,x\rangle - \langle\overline{y}^k,y\rangle + \langle-\overline{z}^k,z\rangle\Big\}\end{aligned}$$

Since r is increasing, we have $-\overline{z}^k \ge 0$. Thus, updating $(x^{k+1},y^{k+1},z^{k+1})$ can be done as follows

$$\begin{cases}(x^{k+1},y^{k+1}) \in \arg\min_{(x,y)\in K}\left\{g(x,y) - \langle\overline{x}^k,x\rangle - \langle\overline{y}^k,y\rangle + \sum_{i=1}^n(-\overline{z}_i^k)x_i^2\rangle\right\} \\ z_i^{k+1} = |x_i^{k+1}|^2 \ \ \forall i = 1,\ldots,n.\end{cases}$$

DCA for solving the problem (5.40) can be described as in Algorithm 5.3 below.

Algorithm 5.3 DCA3 - DCA for solving (5.40)

1: **Initialize** $(x^0, y^0, z^0) \in \Omega_2$, $k \leftarrow 0$
2: **repeat**
3: Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$, $\overline{z}_i^k \in \frac{-\lambda}{2(z_i^k+\epsilon)^{1/2}} \partial(-r)(z_i^k + \epsilon)^{1/2})$ $\forall i = 1, \ldots, n$.
4: Compute

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{ g(x,y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \sum_{i=1}^{n} \overline{z}_i^k x_i^2 \right\},$$
$$z_i^{k+1} = |x_i^{k+1}|^2 \quad \forall i = 1, \ldots, n.$$

5: $k \leftarrow k + 1$.
6: **until** Stopping criterion

Relation with reweighted-ℓ_2 procedure. If the function f in (5.1) is convex, then, as before, we can chose DC components of f as $g = f$ and $h = 0$. Hence, in the step 3 of Algorithm 5.3, we have $(\overline{x}^k, \overline{y}^k) = 0$ $\forall k$. In this case, the step 4 in Algorithm 5.3 becomes

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{ f(x,y) + \sum_{i=1}^{n} \overline{z}_i^k x_i^2 \right\}. \tag{5.42}$$

Thus, each iteration of Algorithm 5.3 solves a weighted-ℓ_2 optimization problem. The expression of weights $\overline{z}_i^k$ according to approximation functions are given in Table 5.4.

If $\epsilon = 0$ then the update rule (5.42) encompasses standard algorithms of reweighted-ℓ_2 type for finding sparse solution (see Table 5.4). However, when $\epsilon = 0$ the (right) derivative at 0 of $r(t^{1/2})$ is not well-defined, that is why we take $\epsilon > 0$ in our algorithm. Note also that, in LQA and FOCUSS, if at an iteration k one has $x_i^k = 0$ then $x_i^l = 0$ for all $l \geq k$, by the way these algorithms may converge prematurely to bad solutions.

5.2.5.4 Discussion on the three DCA based Algorithms 5.1, 5.2 and 5.3

Algorithm 5.1 seems to be the most interesting in the sense that it addresses directly the problem (5.7) and doesn't need the additional variable z, then the subproblem has less constraints than the one in Algorithms 5.2 and 5.3. Moreover, the DC decomposition (5.32) is more suitable since it results, in several cases, in a DC polyhedral program where both DC components are polyhedral convex (for instance, in feature selection in SVM with the approximations r_{scad}, r_{cap}) for which Algorithm 5.1 enjoys interesting convergence properties.

Algorithms 5.2 and 5.3 are based on two different formulations of the problem (5.7). In (5.8), we have linear constraints $|x|_i \leq z_i,\ i = 1, \ldots, n$ that lead to the subproblem of weighted–ℓ_1 type. Whereas, in (5.39), quadratic constraints $|x|_i^2 \leq z_i,\ i = 1, \ldots, n$ result to the subproblem of weighted–ℓ_2 type. With second order terms in subproblems, Algorithm 5.3 is, in general, more expensive than Algorithms 5.1 and 5.2. We also see that Algorithms 5.1 and 5.2 possess nicer convergence properties than Algorithm 5.3. Both Algorithms 5.1 and 5.2 have finite convergence when the corresponding DC programs are polyhedral DC. While (5.39) can't be a polyhedral DC program because the set Ω_2 and the functions $r((t+\epsilon)^{1/2})$ are not polyhedral convex.

To compare the sparsity of solutions given by the algorithms, we consider the subproblems

TABLE 5.4 – Expression of $\overline{z}_i^k$'s in Algorithm 5.3 and relation with reweighted-ℓ_2 algorithms.

Function r	weight $\overline{z}_i^k(t_i^k = (z_i^k + \epsilon)^{1/2})$	Related works	Context
r_{exp}	$\frac{\lambda\theta}{2}\frac{e^{-\theta t_i^k}}{t_i^k}$		
$r_{\ell_p^+}$	$\frac{\lambda}{2\theta(t_i^k)^{2-\frac{1}{\theta}}}$	FOCUSS ([85, 203, 202])	Sparse signal
$r_{\ell_p^-}$	$\frac{-\lambda p\theta^p}{2t_i^k(\frac{1}{\theta}+t_i^k)^{1-p}}$	IRLS ([51])	reconstruction
r_{log}	$\frac{\lambda}{2\log(1+\theta)}\frac{1}{t_i^k(\frac{1}{\theta}+t_i^k)}$		
r_{cap}	$\begin{cases}\frac{\lambda\theta}{2t_i^k} & \text{if } \lvert t_i^k\rvert \le \frac{1}{\theta}\\ 0 & \text{otherwise}\end{cases}$		
r_{scad}	$\begin{cases}\frac{\lambda\theta}{(a+1)t_i^k} & \text{if } t_i^k \le \frac{1}{\theta}\\ 0 & \text{if } t_i^k \ge \frac{a}{\theta}\\ \frac{\lambda\theta(-\theta t_i^k+a)}{(a^2-1)t_i^k} & \text{otherwise}\end{cases}$	LQA ([70, 264])	Linear regression

in Algorithms 5.1, 5.2, and 5.3 which have the form

$$\min_{(x,y)\in K}\left\{g(x,y) - \langle\overline{x}^k, x\rangle - \langle\overline{y}^k, y\rangle + \lambda\sum_{i=1}^n \nu(x_i, x_i^k)\right\}$$

where $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$,

$$\nu(x_i, x_i^k) = \begin{cases}\nu_1(x_i, x_i^k) = \eta|x_i| - \text{sign}(x_i^k)(\eta - \overline{z}_i^k)x_i + C_i^k & \text{for Algorithm 5.1}\\ \nu_2(x_i, x_i^k) = \overline{z}_i^k|x_i| + C_i^k & \text{for Algorithm 5.2}\\ \nu_3(x_i, x_i^k) = \frac{\overline{z}_i^k}{2|x_i^k|}|x_i|^2 + \frac{1}{2}\overline{z}_i^k|x_i^k| + C_i^k & \text{for Algorithm 5.3,}\end{cases}$$

with $\overline{z}_i^k \in -\partial(-r)(|x_i^k|)$, $C_i^k = r(x_i^k) - \overline{z}_i^k|x_i^k|$ and $\eta = r'(0)$.

All three functions ν_1, ν_2 and ν_3 attain minimum at 0 and encourage solutions to be zero. Denote by $\nu'_-(t)$ and $\nu'_+(t)$ the left and right derivative at t of ν respectively. We have

$$\begin{aligned}&\nu'_{1,-}(0, x_i^k) = -2\eta + \overline{z}_i^k, \quad \nu'_{2,-}(0, x_i^k) = -\overline{z}_i^k, \quad \nu'_{3,-}(0, x_i^k) = 0,\\ &\nu'_{1,+}(0, x_i^k) = \overline{z}_i^k, \quad \nu'_{2,+}(0, x_i^k) = \overline{z}_i^k, \quad \nu'_{3,+}(0, x_i^k) = 0.\end{aligned}$$

We also have $\eta \ge \overline{z}_i^k$ by the concavity of r on $[0, +\infty)$. Observe that if the range $[\nu'_-(0), \nu'_+(0)]$ is large, it encourages more sparsity. Intuitively, the values $\nu'_-(0)$ and $\nu'_+(0)$ reflect the slope of ν at 0, and if the slope is hight, it forces solution to be zero. Here we have $[\nu'_{3,-}(0, x_i^k), \nu'_{3,+}(0, x_i^k)] \subset [\nu'_{2,-}(0, x_i^k), \nu'_{2,+}(0, x_i^k)] \subset [\nu'_{1,-}(0, x_i^k), \nu'_{1,+}(0, x_i^k)]$. Thus, we expect that Algorithm 5.1 gives sparser solution than Algorithm 5.2, and Algorithm 5.2 gives sparser solution than Algorithm 5.3.

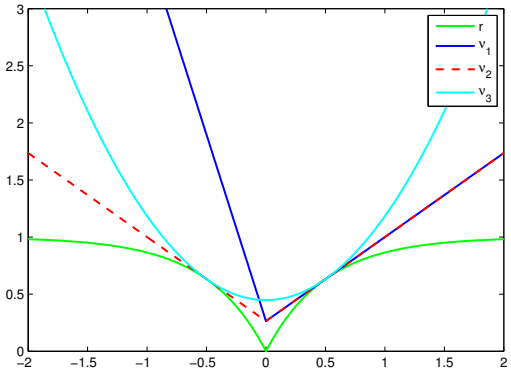


FIGURE 5.2 – Graphs of functions : $r = 1 - e^{-2|x|}$, ν_1, ν_2 and ν_3 with $x^k = 0.5$.

5.2.5.5 DCA4 - DCA applied on approximate problem (5.7) with the new DC approximation

We have proposed three DCA schemes for solving (5.7) or its equivalent form (5.8) when r is a concave function on $[0, +\infty)$. Consider now the general case where r is a DC function satisfying Assumption 1. Hence the problem (5.7) can be expressed as a DC program (5.13) for which DCA is applicable. Each iteration of DCA applied on (5.13) consists of computing

- Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$ and $\overline{z}_i^k \in \lambda \partial \psi(x_i^k)$ $\forall i = 1, \ldots, n$.
- Compute (x^{k+1}, y^{k+1}) as a solution of the following convex program

$$\min_{(x,y)\in K} \left\{ g(x,y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \lambda \sum_{i=1}^{n} \varphi(x_i) - \langle \overline{z}^k, x\rangle \right\}. \tag{5.43}$$

The new approximation function r_{PiL} is a DC function but not concave on $[0, +\infty)$. Hence we apply DCA4 for solving the problem (5.7) with $r = r_{PiL}$

$$r_{PiL} = \min\left\{1, \max\left\{0, \frac{\theta|t|-1}{a-1}\right\}\right\} = \begin{cases} 0 & \text{if } |t| \le \frac{1}{\theta}, \\ \frac{\theta|t|-1}{a-1} & \text{if } \frac{1}{\theta} < |t| < \frac{a}{\theta}, \quad a > 1. \\ 1 & \text{otherwise,} \end{cases} \tag{5.44}$$

DC components of r_{PiL} are given by

$$\varphi_{PiL}(t) := \frac{\theta}{a-1} \max\left\{\frac{1}{\theta}, |t|\right\}, \ \psi_{PiL}(t) := \frac{\theta}{a-1} \max\left\{\frac{a}{\theta}, |t|\right\} - 1 \quad \forall t \in \mathbb{R}, \tag{5.45}$$

that are polyhedral convex functions. Then, the problem (5.7) can be expressed in form of a DC program as follows

$$\min_{x,y}\{F_{r_{PiL}}(x,y) := G_4(x,y) - H_4(x,y)\}, \tag{5.46}$$

where

$$G_4(x,y) = \chi_K(x,y) + g(x,y) + \lambda \sum_{i=1}^{n} \varphi_{PiL}(x_i), \ H_4(x,y) = h(x,y) + \lambda \sum_{i=1}^{n} \psi_{PiL}(x_i),$$

and g, h are DC components of f as stated in (5.6).

At each iteration k, DCA applied to (5.46) updates (x^{k+1}, y^{k+1}) from (x^k, y^k) via two steps :
- Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$ and $\overline{z}_i^k \in \lambda \partial \psi_{PiL}(x_i^k)\ \forall i = 1, \ldots, n$.
- Compute (x^{k+1}, y^{k+1}) as a solution of the following convex program

$$\min_{(x,y)\in K} \left\{ g(x,y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \frac{\lambda\theta}{a-1} \sum_{i=1}^{n} \max\left\{\frac{1}{\theta}, |x_i|\right\} - \langle \overline{z}^k, x\rangle \right\}. \tag{5.47}$$

Calculation of $\overline{z}_i^k$ $(i = 1, \ldots, n)$ is given by

$$\overline{z}_i^k = \begin{cases} \frac{\lambda\theta}{a-1} & \text{if } x_i^k > \frac{a}{\theta} \\ \frac{-\lambda\theta}{a-1} & \text{if } x_i^k < \frac{-a}{\theta} \\ 0 & \text{otherwise.} \end{cases} \tag{5.48}$$

Furthermore, (5.47) is equivalent to

$$\min_{(x,y,t)\in\Omega_3} \left\{ g(x,y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \frac{\lambda\theta}{a-1} \sum_{i=1}^{n} t_i - \langle \overline{z}^k, x\rangle \right\}, \tag{5.49}$$

where $\Omega_3 = \big\{(x, y, t) : (x, y) \in K, \frac{1}{\theta} \le t_i, x_i \le t_i, -x_i \le t_i\ \forall i = 1, \ldots, n\big\}$.

Algorithm 5.4 DCA4 - DCA applied to (5.46)

1: Initialize $(x^0, y^0) \in K$, $k \leftarrow 0$
2: **repeat**
3: Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$ and $\overline{z}_i^k \in \lambda \partial \psi_{PiL}(x_i^k)\ \forall i = 1, \ldots, n$ via (5.48).
4: Solve the convex problem (5.49) to obtain (x^{k+1}, y^{k+1}).
5: $k \leftarrow k + 1$.
6: **until** Stopping criterion.

5.2.5.6 Updating θ procedure

According to consistency results, the larger θ is, the better approximate solution would be. However, from a computational point of view, with large values of θ, the approximate problems are difficult and the algorithms converge often to local minimums. We can overcome this bottleneck by using an update procedure for θ. Starting with a chosen value θ^0, at each iteration k, we compute (x^{k+1}, y^{k+1}) from (x^k, y^k) by applying the DCA based algorithms with $\theta = \theta^k$. The sequence $\{\theta^k\}_k$ is increasing by $\theta^{k+1} = \theta^k + \Delta\theta^k$. $\Delta\theta^k$ can be fixed or updated during the iterations.

5.3 Nonconvex exact reformulation for sparse optimization

In this section, we focus on the nonconvex exact reformulation approach. We will present some main results concerning penalty techniques related to ℓ_0-norm allowing the reformulation of (5.1) and (5.2) as nonconvex programs in the continuous framework, especially DC programs, that can be treated by DC programming and DCA.

Denote by e the vector of ones in the appropriate vector space. We suppose that f is a Lipschitz function on K and K is a polyhedral convex set and bounded in the variable x, i.e.

$K \subset \Pi_{i=1}^{n}[a_i, b_i] \times \mathbb{R}^m$ where $a_i, b_i \in \mathbb{R}$ such that $a_i \leq 0 < b_i$ for $i = 1, ..., n$. Let $c_i := \max\{|x_i| : x_i \in [a_i, b_i]\} = \max\{|a_i|, |b_i|\}$ for $i = 1, ..., n$. Define the binary variable $u_i \in \{0, 1\}$ as

$$u_i = |x_i|_0 = \begin{cases} 1 \text{ if } x_i \neq 0 \\ 0 \text{ if } x_i = 0, \end{cases} \qquad \forall i = 1...n. \tag{5.50}$$

Then (5.1) and (5.2) can be reformulated as

$$\alpha := \inf\{f(x, y) + \lambda e^T u : (x, y) \in K, u \in \{0, 1\}^n, |x_i| \leq c_i u_i,\ i = 1, ..., n\}, \tag{5.51}$$

and

$$\alpha := \inf\{f(x, y) : (x, y) \in K, u \in \{0, 1\}^n, |x_i| \leq c_i u_i, i = 1, ..., n,\ e^T u \leq k\ \}, \tag{5.52}$$

respectively.

Let $p(u)$ be the penalty function defined by

$$p(u) := \sum_{i=1}^{n} \min\{u_i, 1 - u_i\}\ . \tag{5.53}$$

Then (5.1) and (5.2) can be rewritten respectively as

$$\alpha = \inf\{f(x, y) + \lambda e^T u : (x, y) \in K, u \in [0, 1]^n, |x_i| \leq c_i u_i,\ i = 1, ..., n,\ p(u) \leq 0\}, \tag{5.54}$$

and

$$\alpha := \inf\{f(x, y) : (x, y) \in K, u \in [0, 1]^n, |x_i| \leq c_i u_i, i = 1, ..., n,\ e^T u \leq k,\ p(u) \leq 0\}. \tag{5.55}$$

It leads to the corresponding penalized problems (τ being the positive penalty parameter)

$$\alpha(\tau) := \inf\{f(x, y) + \lambda e^T u + \tau p(u) : \quad (x, y) \in K, u \in [0, 1]^n, |x_i| \leq c_i u_i,\ i = 1, ..., n\}, \tag{5.56}$$

and

$$\alpha(\tau) := \inf\{f(x, y) + \tau p(u) : \quad (x, y) \in K, u \in [0, 1]^n, |x_i| \leq c_i u_i, i = 1, ..., n, e^T u \leq k\}. \tag{5.57}$$

Proposition 5.7 *There is $\tau_0 \geq 0$ such that for every $\tau > \tau_0$ problems (5.1) (resp. (5.2)) and (5.56) (resp. (5.57)) are equivalent, in the sense that they have the same optimal value and $(x^*, y^*) \in K$ is a solution of (5.1) (resp. (5.2)) iff there is $u^* \in \{0, 1\}^n$ such that (x^*, y^*, u^*) is a solution of (5.56) (resp. (5.57)).*

Proof 5.10 *Direct consequences of Theorem 8 in [145].* ■

Since f is a DC function, (5.56) and (5.57) are DC programs.

Note that, in general, the minimal penalty parameter τ_0, if any, is not computable. In practice, upper bounds for τ_0 can be calculated in some cases, e.g. sparse eigenvalue problems ([222]).

In the sequel, we will focus on the ℓ_0- regularizer problem (5.1) and its penalized problem (5.56).

5.3.1 Link between the exact formulation (5.51) and the ℓ_1- regularization problem

It is easy to see that the linear relaxation of Problem (5.51) is a ℓ_1- regularization problem. Indeed, the linear relaxation of Problem (5.51) (which is in fact the penalized problem (5.56) when $\tau = 0$) takes the form

$$\inf\{f(x,y) + \lambda e^T u : (x,y) \in K, u \in [0,1]^n, |x_i| \leq c_i u_i, i = 1, ..., n\}. \tag{5.58}$$

Let $M = \max\{c_i : i = 1, \ldots, n\}$, problem (5.58) becomes

$$\inf\{f(x,y) + \lambda e^T u : (x,y) \in K, u \in [0,1]^n, |x_i| \leq M u_i,\ i = 1, \ldots, n\}$$

which can be rewritten as

$$\inf\left\{f(x,y) + \lambda e^T u\ : (x,y) \in K, u \in [0,1]^n, \frac{|x_i|}{M} \leq u_i \leq 1,\ i = 1, \ldots, n\right\}$$

or again

$$\inf\left\{f(x,y) + \frac{\lambda}{M}\sum_{i=1}^{n}|x_i| : (x,y) \in K\right\} = \inf\left\{f(x,y) + \frac{\lambda}{M}\ \|x\|_1 : (x,y) \in K\right\}.$$

We are going now show how to solve the continuous exact reformulation of the ℓ_0-regularization (5.1), say the penalized problem (5.56), by DC programming and DCA.

5.3.2 DCA for solving the continuous exact reformulation problem (5.56)

We consider in the sequel the problem (5.56) with a sufficient large number $\tau > \tau_0$:

$$\alpha(\tau) := \inf\{f(x,y) + \lambda e^T u + \tau p(u) : (x,y) \in K, u \in [0,1]^n, |x_i| \leq M u_i,\ i = 1, \ldots, n\}.$$

Let Δ be the feasible set of Problem (5.56), i.e. $\Delta := \{(x,y,u) : (x,y) \in K, u \in [0,1]^n, |x_i| \leq M u_i,\ i = 1, \ldots, n\}$. Since f is DC with the DC decompostion (5.6) and p is concave, the following DC formulation of (5.56) seems to be natural :

$$\inf\{G(x,y,u) - H(x,y,u) : (x,y,u) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^n\}, \tag{5.59}$$

where

$$G(x,y,u) := \chi_\Delta(x,y,u) + g(x,y), \quad H(x,y,u) := h(x,y) - \lambda e^T u - \tau p(u)$$

are clearly convex functions.

According to the standard DCA scheme, applying DCA to (5.59) amounts to computing two sequences $\{(x^l, y^l, u^l)\}$ and $\{(\bar{x}^l, \bar{y}^l, \bar{u}^l)\}$ in the way that $(\bar{x}^l, \bar{y}^l, \bar{u}^l) \in \partial h(x^l, y^l, u^l)$ and $(x^{l+1}, y^{l+1}, u^{l+1})$ solves the convex program of the form

$$\min\{g(x,y) - \langle (x,y,u,), (\bar{x}^l, \bar{y}^l, \bar{u}^l)\rangle : (x,y,u) \in \Delta\}$$

Since $(\bar{x}^l, \bar{y}^l, \bar{u}^l) \in \partial h(x^l, y^l, u^l) \Leftrightarrow \bar{x}^l = \partial_x h(x^l, y^l)$, $\bar{y}^l = \partial_y h(x^l, y^l)$ and

$$\bar{u}_i^l = \begin{cases} -\lambda + \tau & \text{if } u_i^l \geq 0.5 \\ -\lambda - \tau & \text{if } u_i^l < 0.5 \end{cases}, i = 1, ...n,$$

the algorithm can be described as follow.

Algorithm 5.5 DCAEP - DCA applied on continuous exact formulation (5.56)

1: **Initialization :** Choose an initial point $(x^0, y^0, u^0) \in \mathbb{R}^n \times \mathbb{R}^p \times [0,1]^n$, and $l \leftarrow 0$.
2: **repeat**
3: Compute $\bar{x}^l = \partial_x h(x^l, y^l)$, $\bar{y}^l = \partial_y h(x^l, y^l)$ and set $\bar{u}_i^l = -\lambda + \tau$ with if $u_i^l \geq 0.5$, $\bar{u}_i^l = -\lambda - \tau$ otherwise, for $i = 1, ...n$. .
4: Compute $(x^{l+1}, y^{l+1}, u^{l+1}) = \arg\min\{g(x,y) - \langle (x,y,u,), (\bar{x}^l, \bar{y}^l, \bar{u}^l)\rangle : (x,y,u) \in \Delta\}$
5: $k \leftarrow k+1$.
6: **until** Stopping criterion.

Remark 5.2 *In [130], we considered a special case of* (5.1) *where f is a convex function. In this case, we can simply take $g = f$ and $h = 0$ as DC components of f and H becomes a polyhedral convex function. Thus* (5.59) *is a DC polyhedral program and DCAEP enjoys a finite convergence.*

5.4 Conclusion

We have intensively studied DC programming and DCA for sparse optimization problem.Two approaches, namely nonconvex approximation and nonconvex exact reformulation, were developed with a unifying point of view in DC programming framework.

In the nonconvex approximation approach, DC approximations have been investigated from both a theoretical and an algorithmic point of view. Considering a class of DC approximation functions of the zero-norm including all usual sparse inducing approximation functions, we have proved several novel and interesting results : the consistency between global (resp. local) minimizers of the approximate problem and the original problem, the equivalence between these two problems (in the sense that, for a sufficiently large related parameter, any optimal solution to the approximate problem solves the original problem) when the feasible set is a bounded polyhedral convex set and the approximation function is concave, the equivalence between Capped-ℓ_1 (and/or SCAD) approximate problems and the original problem with sufficiently large parameter θ (in the sense that they have the same set of optimal solutions), the way to compute such parameters θ in some special cases, and a comparative analysis between usual sparse inducing approximation functions. Considering the three DC formulations for a common model to all concave approximation functions we have developed three DCA schemes and showed the link between our algorithms with standard approaches. It turns out that all standard nonconvex approximation algorithms are special versions of our DCA based algorithms. A new DCA scheme has been also investigated for the DC approximation (piecewise linear) which is not concave as usual sparse inducing functions.

In nonconvex exact reformulation approach, using the exact penalty in DC programming, we show that the sparse optimization problem can be reformulated as a continuous optimization problem which is a DC program. The resulting problem is then solved by DCA. Furthermore, we establish the link between the exact reformulation and the approximate problem using ℓ_1 or Capped-ℓ_1 regularization.

Our unified DC programming framework shed a new light on sparse nonconvex programming. It permits to establish the crucial relations among existing sparsity-inducing methods and therefore to exploit, in an elegant way, the nice effect of DC decompositions of objective functions. The four algorithms (DCA1-DCA4) can be viewed as an ℓ_1-perturbed algorithm / reweighted-ℓ_1 algorithm (intimately related to the ℓ_1-penalized LASSO approach / reweighted-ℓ_2 algorithm in case of convex objective functions. It specifies the flexibility/versatility of these theoretical and

algorithmic tools. These results should enhance deeper developments of DC programming and DCA, in order to efficiently model and solve real-world nonconvex sparse optimization problems, especially in the large-scale setting.

In future works we plan to

i) Develop global approaches such as Brand and Bound and/or interval analysis for sparse optimization problems. In deed, as DCA has been shown to be efficient and scalable, it is worthwhile to suitable combine DCA with these global approaches to possibly improve the quality of computer solution.

ii) Find tight convex underestimation of functions involving the zero norm.

Chapitre 6

Applications of Sparse Optimization

In this chapter, we will present three important applications which are formulated as a sparse optimization problem. The first considered problem is the feature selection in Support Vector Machine (SVM). The second application is an extension of the first one in context of semi-supervised learning, namely Semi-Supervised SVM (S3VM) with feature selection. In the third application, we address the sparse recovery signal in compressed sensing. For each problem, taking into account its structure, we carefully investigate the DCA based algorithms for sparse optimization presented in Chapter 5 to solve it.

Beside the three above-mentioned problems, we have addressed other problems (not presented in this chapter) that can be also formulated as a sparse optimization problem. In Chapter 3 and Chapter 4, we develop DCA-Like/ADCA-Like and SDCA for the group variables selection in multi-class logistic regression. The sparse binary logistic regression are solved by ADCA (Chapter 2 and DCA-Like/ADCA-Like (Chapter 3). In Chapter 10, we develop DCA and DCA-Like for solving the Gaussian mixture model (GMM) avec ℓ_0 regularization. It is clear that all these three problems involve the minimization of ℓ_0 norm and can also be solved by DCA based algorithms presented in Chapter 5.

6.1 Feature selection in Support Vector Machine[1]

In this section we focus on the context of feature selection in Support Vector Machines (SVM) learning with two-class linear models. Generally, the problem can be formulated as follows. Given two finite point sets $\mathcal{A}$ (with label $+1$) and $\mathcal{B}$ (with label -1) in $\mathbb{R}^n$ represented by the matrices $A \in \mathbb{R}^{N_A \times n}$ and $B \in \mathbb{R}^{N_B \times n}$, respectively, we seek to discriminate these sets by a separating hyperplane ($x \in \mathbb{R}^n, b \in \mathbb{R}$)

$$P = \{w \in \mathbb{R}^n : w^T x = b\} \tag{6.1}$$

1. The results presented in this section were published in :

- H.A. Le Thi, T. Pham Dinh, H.M. Le, X.T. Vo, DC approximation approaches for sparse optimization, European Journal of Operational Research,244(1) :26-46, 2015.
- H.A. Le Thi, H.M. Le and T. Pham Dinh, Feature Selection in machine learning : an exact penalty approach using a Different of Convex function Algorithm, Machine Learning, 101(1) :163-186, 2015.
- H.A. Le Thi, H.M. Le, V.V. Nguyen, T. Pham Dinh, A DC Programming approach for Feature Selection in Support Vector Machines learning, Journal of Advances in Data Analysis and Classification, 2(3) :259-278, 2008.

which uses as few features as possible. We adopt the notations introduced in [35] and consider the optimization problem proposed in [35] that takes the form ($e \in \mathbb{R}^n$ being the vector of ones) :

$$\min_{x,b}(1-\lambda)\left(\frac{1}{N_A}\|\max\{0,-Ax+eb+e\}\|_1+\frac{1}{N_B}\|\max\{0,Bx-eb+e\}\|_1\right)+\lambda\|x\|_0 \quad (6.2)$$

or equivalently

$$\begin{array}{ll}\min_{x,b,\xi,\zeta} & (1-\lambda)(\frac{1}{N_A}e^T\xi+\frac{1}{N_B}e^T\zeta)+\lambda\|x\|_0\\ s.t. & -Ax+eb+e\le\xi,\ Bx-eb+e\le\zeta,\ \xi\ge 0,\ \zeta\ge 0.\end{array} \quad (6.3)$$

The nonnegative slack variables $\xi_j, j=1,...,N_A$ represent the errors of classification of $a_j \in \mathcal{A}$ while $\zeta_j, j=1,...,N_B$ represent the errors of classification of $b_j \in \mathcal{B}$. More precisely, each positive value of ξ_j determines the distance between a point $a_j \in A$ (lying on the wrong side of the bounding hyperplane $w^Tx=b+1$ for $\mathcal{A}$) and the hyperplane itself. Similarly for ζ_j, $\mathcal{B}$ and $w^Tx=b-1$. The first term of the objective function of (6.3) is the average error of classification, and the second term is the number of nonzero components of the vector x, each of which corresponds to a representative feature. Further, if an element of x is zero, the corresponding feature is removed from the dataset. Here λ is a control parameter of the trade-off between the training error and the number of selected features.

In [35], the authors used the following concave approximation for $\|v\|_0$:

$$\|v\|_0 \simeq e^T(e-\varepsilon^{-\alpha v}).$$

By introducing a non-negative variable $v \ge 0$ and the constraint relaxation $-v \le x \le v$, they reformulated the problem (6.3) in the next parametric (for $\lambda \in [0,1)$) concave minimization problem which is known as Feature Selection concaVe (FSV) :

$$\begin{array}{lll}\min\ \overline{F}(x,b,\xi,\zeta,v) & := (1-\lambda)(\frac{1}{N_A}e^T\xi+\frac{1}{N_B}e^T\zeta)+\lambda e^T(e-\varepsilon^{-\alpha v})\\ \text{s.t.} & -Ax+eb+e\le\xi,\ Bx-eb+e\le\zeta,\ \xi\ge 0,\ \zeta\ge 0,\ -v\le x\le v.\end{array} \quad (6.4)$$

Denote by $\overline{K}$ the polyhedral convex set in $\mathbb{R}^{m+k+2n+1}$ defined as :

$$\overline{K} := \left\{ x,b,\xi,\zeta,v : \begin{array}{c} -Ax+eb+e\le\xi,\ Bx-eb+e\le\zeta,\\ \xi\ge 0,\ \zeta\ge 0,\ -v\le x\le v.\end{array} \right\}. \quad (6.5)$$

The SLA (Successive Linearization Algorithm) proposed in [35] consists of solving at each iteration l the linear program of the form

$$\min\left\{(\frac{1}{N_A}e^T\xi+\frac{1}{N_B}e^T\zeta)+\lambda\alpha(\varepsilon^{-\alpha v^l})^T(v-v^l) : (x,b,\xi,\zeta,v)\in\overline{K}\right\}.$$

It is easy to verify that SLA is exactly DCA applied to (6.4).

Our contributions. Observe that the problem (6.3) is a special case of (5.1) where the function f is given by

$$f(x,b,\xi,\zeta) := (1-\lambda)\left(\frac{1}{N_A}e^T\xi+\frac{1}{N_B}e^T\zeta\right) \quad (6.6)$$

and K is a polytope defined by

$$K := \left\{(x,b,\xi,\zeta)\in\mathbb{R}^n\times\mathbb{R}\times\mathbb{R}^{N_A}_+\times\mathbb{R}^{N_B}_+ : -Ax+eb+e\le\xi,\ Bx-eb+e\le\zeta\right\}. \quad (6.7)$$

Hence we will develop all the DCA based algorithms for sparse optimization presented in Chapter 5 to solve (6.3). Our experiments aim to realize a complete comparison between all these DCA based algorithm for sparse optimization.

6.1.1 DCA based algorithms for feature selection in SVM

Here the function f is simply linear, and DC components of f is taken as $g = f$ and $h = 0$. Hence, all algorithms presented in Chapter 5 can be developed to solve (6.3).

6.1.1.1 Solving (6.3) by DC approximation approach

The approximate problem of (6.3) takes the form

$$\min \left\{ F(x, b, \xi, \zeta) := f(x, b, \xi, \zeta) + \lambda \sum_{i=1}^{n} r(x_i) : (x, b, \xi, \zeta) \in K \right\}, \tag{6.8}$$

where r is one of the sparsity-inducing functions given in Table 5.1. This problem is also equivalent to

$$\min \left\{ \overline{F}(x, b, \xi, \zeta, z) := f(x, b, \xi, \zeta) + \lambda \sum_{i=1}^{n} r(z_i) : (x, b, \xi, \zeta, z) \in \overline{K} \right\}, \tag{6.9}$$

where $\overline{K} = \{(x, b, \xi, \zeta, z) : (x, b, \xi, \zeta) \in K, -z_i \leq x_i \leq z_i \ \forall i = 1, \ldots, n\}$.

Note that, since K is a polyhedral convex set, all the resulting approximate problems (6.8) with approximation functions given in Table 5.2 (except for $r = r_{PiL}$) are equivalent to the problem (6.3) in the sense of Corollary 5.1. More strongly, from Proposition 5.4, if $r = r_{cap}$ and $\theta > \theta^* := \frac{1-\lambda}{\lambda}\Delta$, where

$$\Delta := \max_{j=1,\ldots,n} \left\{ \frac{1}{N_A} \sum_{i=1}^{N_A} |A_{ij}| + \frac{1}{N_B} \sum_{i=1}^{N_B} |B_{ij}| \right\}, \tag{6.10}$$

then the problems (6.3) and (6.8) are equivalent.

For η given in Table 5.2, let $\psi(t) = \eta|t| - r(t)$. The DCA1 for solving (6.8) is described as follows.

Algorithm 6.1 DCA1-SVM : DCA1 applied on the approximate problem (6.8)

1: **Initialization :** Choose an initial point $(x^0, b^0, \xi^0, \zeta^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^{N_A}_+ \times \mathbb{R}^{N_B}_+$, and $k \leftarrow 0$.
2: **repeat**
3: Compute $\overline{z}_i^k \in \lambda \partial \psi(x_i^k) \ \forall i = 1, \ldots, n$ as given in Table 5.2.
4: Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$ by solving the linear program

$$\min \left\{ (1-\lambda) \left(\frac{1}{N_A} e^T \xi + \frac{1}{N_B} e^T \zeta \right) + \lambda\eta \sum_{i=1}^{n} z_i - \langle \overline{z}^k, x \rangle : (x, b, \xi, \zeta, z) \in \overline{K} \right\}. \tag{6.11}$$

5: $k \leftarrow k + 1$.
6: **until** Stopping criterion.

Since f is linear and K is a polyhedral convex set, the first DC component G_1 in (5.35) is polyhedral convex. Therefore, (5.35) is always a polyhedral DC program. According to the convergence property of polyhedral DC programs, DCA1 applied to (6.8) generates a sequence $\{(x^k, b^k, \xi^k, \zeta^k)\}$ that converges to a critical point $(x^*, b^*, \xi^*, \zeta^*)$ after finitely many iterations. Furthermore, if $r = r_{cap}$ and $|x_i^*| \neq \frac{1}{\theta} \ \forall i = 1, \ldots, n$, the second DC component H_1 in (5.35) is

polyhedral convex and differentiable at $(x^*, b^*, \xi^*, \zeta^*)$. Using the DCA's convergence property, we deduce that $(x^*, b^*, \xi^*, \zeta^*)$ is a local solution of (6.8).

DCA2 applied to the approximation (6.9) is given in Algorithm 6.2.

Algorithm 6.2 DCA2-SVM : DCA2 applied on the approximate problem (6.9)

1: **Initialization :** Choose an initial point $(x^0, b^0, \xi^0, \zeta^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_+^{N_A} \times \mathbb{R}_+^{N_B}$, and $k \leftarrow 0$.
2: **repeat**
3: Compute $\overline{z}_i^k \in -\lambda \partial(-r)(|x_i^k|)\ \forall i = 1, \ldots, n$ as given in Table 5.3.
4: Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$ by solving the linear program

$$\min \left\{ (1-\lambda) \left(\frac{1}{N_A} e^T \xi + \frac{1}{N_B} e^T \zeta \right) + \langle \overline{z}^k, z \rangle : (x, b, \xi, \zeta, z) \in \overline{K} \right\}.$$

5: $k \leftarrow k + 1$.
6: **until** Stopping criterion.

Similar to the case of DCA1 mentioned above, (6.9) is also a polyhedral DC program. Thus, DCA2 applied to (6.9) generates a sequence $\{(x^k, b^k, \xi^k, \zeta^k, |x^k|)\}$ that converges to a critical point $(x^*, b^*, \xi^*, \zeta^*, |x^*|)$ after finitely many iterations. Furthermore, if $r = r_{cap}$ and $|x_i^*| \neq \frac{1}{\theta}\ \forall i = 1, \ldots, n$, the second DC component H_2 in (5.37) is polyhedral convex and differentiable at $(x^*, b^*, \xi^*, \zeta^*, |x^*|)$. Then $(x^*, b^*, \xi^*, \zeta^*, |x^*|)$ is a local solution of (6.9).

DCA3 for solving (6.8) is described in Algorithm 6.3.

Algorithm 6.3 DCA3-SVM : DCA3 applied on the approximate problem (6.8)

1: **Initialization :** Choose an initial point $(x^0, b^0, \xi^0, \zeta^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_+^{N_A} \times \mathbb{R}_+^{N_B}$, and $k \leftarrow 0$.
2: **repeat**
3: Compute $\overline{z}_i^k \in \frac{-\lambda}{2(|x_i^k|^2+\epsilon)^{1/2}} \partial(-r)((|x_i^k|^2 + \epsilon)^{1/2})\ \forall i = 1, \ldots, n$ as given in Table 5.4.
4: Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$ by solving the quadratic convex program

$$\min \left\{ (1-\lambda) \left(\frac{1}{N_A} e^T \xi + \frac{1}{N_B} e^T \zeta \right) + \sum_{i=1}^{n} \overline{z}_i^k x_i^2 : (x, b, \xi, \zeta) \in K \right\}.$$

5: $k \leftarrow k + 1$.
6: **until** Stopping criterion.

Consider the case $r = r_{PiL}$. DCA4 for solving (6.8) is given in Algorithm 6.4.

Since the second DC component H_4 in (6.8) is polyhedral convex, (5.46) is a polyhedral DC program. Thus, DCA4 applied to (6.8) generates a sequence $\{(x^k, b^k, \xi^k, \zeta^k)\}$ that converges to a critical point $(x^*, b^*, \xi^*, \zeta^*)$ after finitely many of iterations. Moreover, if $|x_i^*| \neq \frac{1}{\theta}\ \forall i = 1, \ldots, n$, then H_4 is differentiable at $(x^*, b^*, \xi^*, \zeta^*)$. This implies that $(x^*, b^*, \xi^*, \zeta^*)$ is a local solution of (6.8).

We have seen in Sect. 5.2.3 that the approximate problem using Capped-ℓ_1 and SCAD approximations are equivalent to the original problem if the parameter θ is beyond a certain threshold : $\theta \geq \theta_0$ (cf. Proposition 5.3 and Proposition 5.5). However, the computation of such a value θ_0 is in general not available, hence one must take large enough values for θ_0. But, as

Algorithm 6.4 DCA4-SVM : DCA4 applied on the approximate problem (6.8) with $r = r_{PiL}$

Initialization : Choose an initial point $(x^0, b^0, \xi^0, \zeta^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_+^{N_A} \times \mathbb{R}_+^{N_B}$, and $k \leftarrow 0$.
repeat
  Compute $\overline{z}_i^k \in \lambda \partial \psi_{PiL}(x_i^k)\ \forall i = 1, \ldots, n$ via (5.48).
  Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$ by solving the linear program

$$\begin{aligned} \min \quad & \left\{ (1-\lambda)\left(\frac{1}{N_A} e^T \xi + \frac{1}{N_B} e^T \zeta\right) + \frac{\lambda\theta}{a-1} \sum_{i=1}^{n} t_i - \langle \overline{z}^k, x \rangle \right\}, \\ s.t. \quad & (x, b, \xi, \zeta, t) \in \overline{K}, \frac{1}{\theta} \leq t_i\ \forall i = 1, \ldots, n. \end{aligned}$$

  $k \leftarrow k + 1$.
until Stopping criterion.

discussed in Sect. 5.2.5.6, a large value of θ makes the approximate problem hard to solve. For the feature selection in SVM, we can compute exactly a θ_0 as shown in (6.10), but it is quite large. Hence we use an updating θ procedure. On the other hand, in the DCA1 scheme, at each iteration, we have to compute $\bar{z}^k \in \partial \lambda \psi(x^k)$ and when ψ is not differentiable at x^k, the choice of $\bar{z}^k$ can influence on the efficiency of the algorithm. For Capped-ℓ_1 approximation, based on the properties of this function we propose a specific way to compute $\bar{z}^k$. Below, we describe the updating θ procedure for DCA1 with Capped-ℓ_1 approximation.

In the above procedure, the computation of $\overline{z}^k$ is slightly different from formula given in Table 5.2. When $|x_i^k| = \alpha^{k+1}$, $\partial r(x_i^k)$ is an interval. Taking into account information of derivative of u w.r.t. the variable x_i at x_i^k helps us judge which between two extreme values of $\partial r(x_i^k)$ may give better decrease of algorithm.

At each iteration, the value of θ increases at least $\Delta\theta > 0$ as long as it does not exceed θ^* – the value from which the problems (6.3) and (6.8) are equivalent. Moreover, we know that for each fixed θ, DCA1 has finite convergence. Hence, the above procedure also possesses finite convergence property.

If $F(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1}) = F(x^k, b^k, \xi^k, \zeta^k)$ then $(x^k, b^k, \xi^k, \zeta^k)$ is a critical point of (6.8) with $r = r_{cap}$ and $\theta = \theta^{k+1}$. In addition, if $\alpha^{k+1} = \alpha^k$, which means that $|x_i^k| \geq \alpha^k \geq \frac{1}{\theta^k}$ for any $i \in supp(x^k)$, then $(x^k, b^k, \xi^k, \zeta^k)$ is a critical point of (6.8) for all $\theta \geq \theta^{k+1}$.

6.1.1.2 Exact reformulation approach for solving (6.3)

The continuous exact reformulation of (6.3) is written as :

$$\inf\{G(x, b, \xi, \zeta, u) - H(x, b, \xi, \zeta, u) : (x, b, \xi, \zeta, u) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_+^{N_A} \times \mathbb{R}_+^{N_B} \times [0,1]^n\}, \quad (6.12)$$

where

$$\begin{aligned} &G(x, b, \xi, \zeta, u) := \chi_\Delta(x, b, \xi, \zeta, u) + (1-\lambda)(\tfrac{1}{N_A} e^T \xi + \tfrac{1}{N_B} e^T \zeta), \\ &H(x, b, \xi, \zeta, u) := -\lambda e^T u - \tau p(u). \end{aligned}$$

with $\Delta := \{(x, b, \xi, \zeta, u) : (x, y, \xi, \zeta) \in K, u \in [0,1]^n, |x_i| \leq M u_i,\ i = 1, \ldots, n\}$. Since K is a polyhedral convex set, so is Δ, hence χ_Δ is a polyhedral convex function. Therefore (6.12) is a polyhedral DC program with both polyhedral DC components g and h.

DCAEP (Algorithm 5.5) applied to (6.3) is described in Algorithm 6.6.

Algorithm 6.5 updating θ procedure for DCA1 with Capped-ℓ_1 approximation

1: **Initialization :** $\Delta\theta > 0, \alpha^0 = +\infty$, $\theta^0 = 0$, $k = 0$. Let $(x^0, b^0, \xi^0, \zeta^0)$ be a solution of the linear problem (6.8).

2: **repeat**

3: $I = \{i : 0 < |x_i^k| < \alpha^k\}$, $\alpha^{k+1} = \begin{cases} \max\{|x_i^k| : i \in I\} & \text{if } I \neq \emptyset, \\ \alpha^k & \text{otherwise.} \end{cases}$

4: Compute $\theta^{k+1} = \min\left\{\theta^*, \max\left\{\frac{1}{\alpha^{k+1}}, \theta^k + \Delta\theta\right\}\right\}$.

5: Compute $\overline{z}^k$: For $i = 1, \ldots, n$

- If $|x_i^k| < \alpha^{k+1}$, $\overline{z}_i^k = 0$.
- If $|x_i^k| > \alpha^{k+1}$, $\overline{z}_i^k = \text{sign}(x_i^k)\lambda\theta$.
- If $|x_i^k| = \alpha^{k+1}$, compute F_i^- (resp. F_i^+) the left (resp. right) derivative of the function $u(x, b)$ w.r.t. the variable x_i at x_i^k, where

$$u(x,b) = \begin{array}{l}(1-\lambda)\left(\frac{1}{N_A}\|\max\{0, -Ax + eb + e\}\|_1 + \frac{1}{N_B}\|\max\{0, Bx - eb + e\}\|_1\right) \\ +\lambda\sum_{j=1}^n r(x_j).\end{array}$$

$$\text{Then } \overline{z}_i^k = \begin{cases} \text{sign}(x_i^k)\lambda\theta^{k+1} & \text{if } x_i^k(F_i^- + F_i^+) < 0 \\ 0 & \text{ortherwise.} \end{cases}$$

6: Solve the linear problem (6.11) with $\eta = \theta^{k+1}$ to obtain $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$.

7: $k \leftarrow k + 1$.

8: **until** Convergence of $\{x^k, b^k, \xi^k, \zeta^k\}$

Theorem 6.1 *(Convergence properties of DCAEP-SVM)*

(i) DCAEP-SVM generates a sequence $\{(x^l, b^l, \xi^l, \zeta^l, u^l)\}$ contained in $V(\Delta)$ such that the sequence $\{f(x^l, b^l, \xi^l, \zeta^l) + \tau p(u^l)\}$ is decreasing.

(ii) For a number τ sufficiently large, if at an iteration q we have $u^q \in \{0, 1\}^n$, then $u^l \in \{0, 1\}^n$ for all $l \geq q$.

(iii) The sequence $\{(x^l, b^l, \xi^l, \zeta^l, u^l)\}$ converges to $\{(x^*, b^*, \xi^*, \zeta^*, u^*)\} \in V(\Delta)$ after a finite number of iterations. The point $(x^*, b^*, \xi^*, \zeta^*, u^*)$ is a critical point of Problem (6.12). Moreover if $u_i^* \neq \frac{1}{2}$ for all $i = 1...n$, then $\{(x^*, b^*, \xi^*, \zeta^*, u^*)\}$ is a local solution to (6.12).

Proof 6.1 *i) is consequence of DCA's convergence Theorem for a general DC program.*

ii) Let $\tau > \tau_1 := \max\left\{\frac{f(x,b,\xi,\zeta)+\lambda e^T u-\eta}{\delta} : (x, b, \xi, \zeta, u) \in V(\Delta), p(u) \leq 0\right\}$ where $\eta := \min\{f(x, b, \xi, \zeta) + \lambda e^T u : (x, b, \xi, \zeta, u) \in V(\Delta)\}$ and $\delta := \min\{p(u) : (x, b, \xi, \zeta, u) \in V(\Delta)\}$. Let $\{(x^l, b^l, \xi^l, \zeta^l, u^l)\} \subset V(\Delta)$ $(l \geq 1)$ be generated by DCA1DCAEP-SVM. If $V(\Delta) \subset \{\Delta \cap u \in \{0, 1\}^n\}$, then the assertion is trivial. Otherwise, let $(x^l, b^l, \xi^l, \zeta^l, u^l) \in \{\Delta \cap u \in \{0, 1\}^n\}$ and $(x^{l+1}, b^{l+1}, \xi^{l+1}, \zeta^{l+1}, u^{l+1}) \in V(\Delta)$ be an optimal solution of the linear program (6.13). Then from (i) of this theorem we have

$$f(x^{l+1}, b^{l+1}, \xi^{l+1}, \zeta^{l+1}) + \lambda e^T u^{l+1} + tp(u^{l+1}) \leq f(x^l, b^l, \xi^l, \zeta^l) + \lambda e^T u^l + tp(u^l).$$

Since $p(u^l) = 0$, it follows

$$\begin{array}{rl} \tau p(u^{l+1}) & \leq f(x^l, b^l, \xi^l, \zeta^l) + \lambda e^T u^l - f(x^{l+1}, b^{l+1}, \xi^{l+1}, \zeta^{l+1}) - \lambda e^T u^{l+1} \\ & \leq f(x^l, b^l, \xi^l, \zeta^l) + \lambda e^T u^l - \eta. \end{array}$$

Algorithm 6.6 DCAEP-SVM : DCA applied on continuous exact reformulation (6.12)

1: **Initialization :** Choose an initial point $(x^0, b^0, \xi^0, \zeta^0, u^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_+^{N_A} \times \mathbb{R}_+^{N_B} \times [0,1]^n$, and $l \leftarrow 0$.
2: **repeat**
3: Set $\bar{u}^l = (\bar{u}_i^l)$ with $\bar{u}_i^l = -\lambda + \tau$ if $u_i^l \geq 0.5$, $-\lambda - \tau$ otherwise, for $i = 1, ...n$.
4: Solve the linear program

$$\min\{(1-\lambda)(\frac{1}{N_A}e^T\xi + \frac{1}{N_B}e^T\zeta) - \langle u, \bar{u}^l\rangle : (x, b, \xi, \zeta, u) \in \Delta\} \tag{6.13}$$

to obtain $(x^{l+1}, b^{l+1}, \xi^{l+1}, \zeta^{l+1}, u^{l+1})$
5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

If $p(u^{l+1}) > 0$, *then*

$$\tau \leq \frac{f(x^l, b^l, \xi^l, \zeta^l) + \lambda e^T u^l - \eta}{p(u^{l+1})} \leq \frac{f(x^l, b^l, \xi^l, \zeta^l) + \lambda e^T u^l - \eta}{\delta} \leq \tau_1$$

which contradicts the fact that $\tau > \tau_1$. *Therefore we have* $p(u^{l+1}) = 0$.

iii) Since (6.12) is a polyhedral DC program, DCAEP-SVM has a finite convergence, say, the sequence $\{(x^l, b^l, \xi^l, \zeta^l, u^l)\}$ *converges to a critical point* $(x^*, b^*, \xi^*, \zeta^*, u^*) \in V(\Delta)$ *after a finite number of iterations. If* $u_j^* \neq 1/2, \forall j \in 1..n$, *then the function* h *is differentiable at* $(x^*, b^*, \xi^*, \zeta^*, u^*)$ *and then the necessary local condition*

$$\partial h(x^*, b^*, \xi^*, \zeta^*, u^*) \subset \partial g(x^*, b^*, \xi^*, \zeta^*, u^*)$$

holds. Since h *is a polyhedral convex function, this subdifferential inclusion is also a sufficient local optimality condition, i.e.* $(x^*, b^*, \xi^*, \zeta^*, u^*)$ *is a local minimizer of (6.12). The proof is then complete.* ■

6.1.2 Numerical experiments

Dataset

Experiments were performed on several real-word datasets taken from well-known UCI data repository and from challenging feature-selection problems of the NIPS 2003 datasets. In Table 6.1, the number of features, the number of points in training and test set of each dataset are given. The full description of each dataset can be found on the web site of UCI repository and NIPS 2003.

Experiment setting

We stop all algorithms with the tolerance $\epsilon = 10^{-5}$. The non-zero elements of x are determined according to whether $|x_i|$ exceeds a small threshold (10^{-5}).

For the comparison of algorithms, we are interested in the accuracy (PWCO - Percentage of Well Classified Objects) and the sparsity of obtained solution as well as the rapidity of the algorithms. $POWC_1$ (resp. $POWC_2$) denotes the POWC on training set (resp. test set). The sparsity of solution is determined by the number (and percentage) of selected features (SF) while the rapidity of algorithms is measured by the CPU time in seconds.

TABLE 6.1 – Datasets

Data	#features	# points in training set	# points in test set
Ionosphere	34	234	117
WPBC (24 months)	32	104	51
WPBC (60 months)	32	380	189
Breast Cancer	24481	78	19
Leukemia	7129	38	34
Arcene	10000	100	100
Gisette	5000	6000	1000
Prostate	12600	102	21
Adv	1558	2458	821

6.1.2.1 Experiment 1

In this experiment, we study the effectiveness of the DCA1-SVM, DCA2-SVM and DCA3-SVM for a same approximation. Capped-ℓ_1 approximation is chosen for this experiment. For each dataset, the same value of λ is used for all algorithms. We set $\lambda = 0.1$ for first three datasets (*Ionosphere, WPBC(24), WPBC(60)*) while $\lambda = 0.001$ is used for five large datasets (*Adv, Arcene, Breast, Gisette, Leukemia*). To chose a suitable value of θ for each algorithm DCA1-SVM, DCA2-SVM and DCA3-SVM, we perform them by 10 folds cross-validation procedure on the set $\{0.001, 0.005, 0.01, 0.1, 0.5, 1, 2, 3, 5, 10, 20, 50, 100, 500\}$ and then take the value corresponding to the best results. Once θ is chosen (its value is given in Table 6.2), we perform these algorithms 10 times from 10 random starting solutions and report, in the columns 3 - 5 of Table 6.2, the mean and standard deviation of the accuracy, the sparsity of obtained solutions and CPU time of the algorithm.

We are also interested on the efficiency of Updating θ procedure. For this purpose, we compare two versions of DCA1-SVM with and without Updating θ procedure (in case of Capped-ℓ_1 approximation). For a fair comparison, we first run DCA1-SVM with Updating θ procedure and then perform DCA1 with the fixed value θ^* which is the last value of θ when the Updating θ procedure stops. Computational results are reported in the columns 6 (DCA1-SVM with fixed θ) and 7 (DCA1-SVM with Updating θ procedure) of Table 6.2.

To evaluate the globality of the DCA based algorithms we use CPLEX 12.2 for globally solving the exact formulation problem (5.51) via exact penalty techniques (Mixed 0-1 linear programming problem) and report the results in the last column of Table 6.2.

Bold values in the result tables correspond to best results for each data instance.

Comments on numerical results

— Comparison between DCA1-SVM, DCA2-SVM and DCA3-SVM (columns 3 - 5)

  — Concerning the correctness, DCA1-SVM furnishes the best solution out of the three algorithms for all datasets (with an important gain of $6,9\%$ on dataset WPBC(24)). DCA2-SVM and DCA3-SVM are comparable in terms of correctness.

  — As for the sparsity of solution, all the three DCA schemes reduce considerably the number of selected features (up to 99% on large datasets such as *Arcene*, *Breast*, *Leukemia*, ...). Moreover, DCA1-SVM gives better results than DCA2-SVM/DCA3-SVM on 6 out of 7 datasets.

  — In terms of CPU Time, DCA1-SVM and DCA2-SVM are faster than DCA3-SVM. This is natural, since at each iteration, the first two algorithms only require solving

TABLE 6.2 – Comparison of different DCA schemes for Capped-ℓ_1 approximation

		DCA1-SVM	DCA2-SVM	DCA3-SVM	DCA1-SVM with θ^*	DCA1-SVM with Updating θ	CPLEX
Ionosphere	θ	3	5	3	4,3	4,3	
	$POWC_1$	86,2 ±1,5	85,2 ±1,7	84,8 ±1,8	84,0 ±1,2	**90,2**	**90,2**
	$POWC_2$	80,3 ±1,6	75,3 ±1,3	74,3 ±1,3	80,3 ±1,4	**83,7**	**83,7**
	FS	3,5 (10,3%)	3,8 (11,2%)	3,8 (11,2%)	3,2 (9,4%)	**2 (5,9%)**	**2 (5,9%)**
	CPU	**0,2**	**0,2**	0,7	0,3	0,6	2,5
WPBC(24)	θ	1	0,1	0,1	661	661	
	$POWC_1$	**84,3 ±1,4**	75,3 ±1,3	77,4 ±1,1	75,3 ±1,2	77,4	77,4
	$POWC_2$	77,9 ±1,4	**80,2 ±1,6**	79,3 ±1,6	72,3 ±1,2	77,2	78,4
	FS	7,4 (23,1%)	8,5 (26,6%)	8,5 (26,6%)	8,4 (26,3%)	8 (25,0%)	**7 (21,9%)**
	CPU	**0,2**	0,3	0,8	**0,2**	1,1	6,4
WPBC(60)	θ	1	3	3	347	347	
	$POWC_1$	96,2 ±1,3	95,2 ±1,3	95,2 ±1,3	**98,2 ±1,3**	96	96
	$POWC_2$	92,5±1,4	92,5±1,4	90,8±1,8	**96,8±1,8**	95,3	95,3
	FS	4,7 (15,7%)	5,5 (18,3%)	5,7 (19,0%)	8,9 (29,7%)	**3 (10,0%)**	**3 (10,0%)**
	CPU	**0,4**	0,6	1,6	0,5	1	1,8
Breast	θ	5	10	2	435	435	
	$POWC_1$	95,1±1,3	94,2±1,3	95,2±1,4	93,2±1,6	**96,8**	N/A
	$POWC_2$	68,3±1,2	67,3±1,2	**70,3±1,6**	66,3±1,1	65,1	N/A
	FS	32,6 (0,1%)	47,5 (0,2%)	43,5 (0,2%)	52,3 (0,2%)	**28 (0,1%)**	N/A
	CPU	30	**25**	78	79	76	3600
Leukemia	θ	5	5	5	178	178	
	$POWC_1$	**100**	**100**	**100**	**100**	**100**	N/A
	$POWC_2$	**97,2±0,4**	97,1±0,4	96,8±0,3	94,8±0,7	**97,2**	N/A
	FS	8,2 (0,1%)	8,5 (0,1%)	8,5 (0,1%)	12,0 (0,2%)	**8 (0,1%)**	N/A
	CPU	**10**	**10**	75	14	17	3600
Arcene	θ	0,1	0,01	3	328	328	
	$POWC_1$	**100**	**100**	**100**	**100**	**100**	N/A
	$POWC_2$	80±1,6	**82±1,1**	81±1,9	61±1,1	70	N/A
	FS	78,5 (0,79%)	82,4 (0,82%)	82,4 (0,82%)	35 (0,35%)	**32 (0,32%)**	N/A
	CPU	**21**	26	273	30	118	3600
Gisette	θ	0,1	0,01	0,1	735	735	
	$POWC_1$	**92,5±1,3**	88,5±1,3	88,5±1,3	90,5±1,2	91,2	N/A
	$POWC_2$	**85,3±1,2**	83,4±1,2	83,1±1,6	84,1±1,1	83,2	N/A
	FS	339,4 (6,8%)	330,7 (6,6%)	332,2 (6,6%)	456 (9,1%)	**123 (2,5%)**	N/A
	CPU	87	**65**	253	71	387	3600
Adv	θ	0,1	0,01	0,1	321	321	
	$POWC_1$	95,5±1,5	92,3±1,5	95,3±1,5	92,3±1,2	**97,2**	N/A
	$POWC_2$	**94,2±1,1**	93,2±1,5	93,1±1,2	92,1±1,6	93,2	N/A
	FS	5,4 (0,35%)	6,2 (0,40%)	6,4 (0,41%)	6,5 (0,42%)	**5 (0,32%)**	N/A
	CPU	**2,1**	2,4	7,8	2,3	4,6	3600

one linear program while DCA3-SVM has to solve one convex quadratic program. DCA1-SVM is somehow a bit faster than DCA2-SVM on 5 out 7 datasets.

— Overall, we see that DCA1-SVM is better than DCA2-SVM and DCA3-SVM on all the three evaluation criteria. Hence, it seems to be that the first DCA scheme is more appropriate than the other two for Capped-ℓ_1 approximation.

— DCA1-SVM with and without Updating θ procedure (columns 3, 6 and 7) :

— For all datasets, Updating θ procedure gives a better solution (on both accuracy and sparsity) than DCA1-SVM with $\theta = \theta^*$.

— Except for dataset *WPBC(24)*, Updating θ procedure is better than DCA1-SVM with θ chosen by 10 folds cross-validation in terms of sparsity of solution. As for accuracy, the two algorithms are comparable.

— The choice of the value of θ defining the approximation function is very important. Indeed, the results given in columns 3 and 6 are far different, due to the fact that, the value of θ chosen by 10 folds cross-validation is much more smaller than θ^*.

These results confirm our analysis in Subsection 5.2.5.6 above : while the approximate function would be better with larger values of θ, the approximate problems become more difficult and it can be happened that the obtained solutions are worse when θ is quite large. To overcome this "contradiction" between theoretical and computational aspects, the proposed Updating θ procedure seems to be efficient.

— Comparison between DCA-SVM based algorithms and CPLEX for solving the original problem (5.51)

  — For *Ionosphere* and *WPBC(60)*, Updating θ procedure for Capped-ℓ_1 gives exactly the same accuracy and the same number of selected features as CPLEX. It means that Updating θ procedure reaches the global solution for those two datasets. For *WPBC(24)*, the two obtained solutions are slightly different (same accuracy on training set and 7 selected features for CPLEX instead of 8 for Updating θ procedure).

  — For large datasets, CPLEX can't furnish a solution with a CPU Time limited to 3600 seconds while DCA based algorithms give a good solution in a short time.

6.1.2.2 Experiment 2

In the second experiment, we study the effectiveness of different approximations of ℓ_0. We use DCA1-SVM for all approximations except PiL for which DCA4-SVM is applied (cf. Section 5.2.5.5).

In this experiment, for the trade-off parameter λ, we used the following set of candidate values $\{0.001, 0.002, 0.003, 0.004, 0.05, 0.1, 0.25, 0.4, 0.7, 0.9\}$. The value of parameter θ is chosen in the set $\{0.001, 0.005, 0.01, 0.1, 0.5, 1, 2, 3, 5, 10, 20, 50, 100, 500\}$. The second parameter a of SCAD approximation is taken from $\{1, 2, 3, 5, 10, 20, 30, 50, 100\}$. For each algorithm, we firstly perform a 10-folds cross-validation to determine the best set of parameter values. In the second step, we run each algorithm, with the chosen set of parameter values in step 1, 10 times from 10 starting random points and report the mean and standard deviation of each evaluation criterion. The comparative results are reported in Table 6.3.

We observe that :

— In terms of sparsity of solution, the quality of all approximations are comparable. All the algorithms reduce considerably the number of selected features, especially for 5 large datasets (*Adv, Arcene, Breast, Gisette, Leukemia*). For *Breast* dataset, our algorithms select only about thirty features out of 24481 while preserving very good accuracy (up to $98,7\%$ correctness on train set).

— Capped-ℓ_1 is the best in terms of accuracy : it gives best accuracy on all train sets and 4 out of 7 test sets. The quality of other approximations are comparable.

— The CPU time of all the algorithms is quite small : less than 34 seconds (except for *Gisette*, CPU time of DCAs varies from 72 to 102 seconds).

TABLE 6.3 – Comparison of different approximations

		DCA1-SVM Capped-l1	DCA1-SVM SCAD	DCA1-SVM Exp	DCA1-SVM lp+	DCA1-SVM lp-	DCA1-SVM Log	DCA4-SVM PiL	SVM-ℓ_1
Ionosphere	$POWC_1$	**86,2 ±1,5**	80,1 ±1,4	82,1 ±1,4	81,5 ±1,3	83,1 ±1,4	81,2 ±1,4	83,2 ±1,4	77,3
	$POWC_2$	80,3 ±1,6	73,5 ±1,6	**84,8 ±1,3**	75,1 ±1,1	70,3 ±1,2	73,1 ±2,1	83,5 ±1,6	75,3
	SF	3,5 (10,3%)	3,1 (9,1%)	**2,3 (6,8%)**	3,8 (11,2%)	3,1 (9,1%)	3,3 (9,7%)	2,6 (7,6%)	10 (29,4%)
	CPU	0,2	0,3	0,3	0,2	0,3	0,15	0,2	**0,01**
WPBC(24)	$POWC_1$	**84,3 ±1,4**	77 ±1,3	**84,3 ±1,5**	81,3 ±1,2	81,9 ±1,2	71,3 ±1,4	84,2 ±1,4	73,3
	$POWC_2$	77,9 ±1,4	**79,3 ±1,6**	74,3 ±1,9	78,4 ±1,2	**79,8 ±1,1**	68,4 ±1,6	78,5 ±1,4	72,1
	SF	7,4 (23,1%)	8,1 (25,3%)	**7,2 (22,5%)**	7,8 (24,4%)	7,5 (23,4%)	**7,2 (22,5%)**	7,6 (23,8%)	8 (25,0%)
	CPU	0,1	0,2	0,1	0,2	0,2	0,2	0,2	**0,01**
WPBC(60)	$POWC_1$	**97,2 ±1,3**	93,5 ±1,7	95,1 ±1,6	93 ±1,2	94,5 ±1,1	89 ±1,5	95,2 ±1,3	84,3
	$POWC_2$	**93,5±1,4**	89,1 ±1,9	92,3 ±1,9	85 ±1,2	90,6 ±1,2	80 ±1,6	88,5±1,1	85,5
	SF	5,4 (18,0%)	**5,2 (17,3%)**	**5,2 (17,3%)**	5,9 (19,7%)	5,7 (19,0%)	5,4 (18,0%)	5,4 (18,0%)	6 (20,0%)
	CPU	0,4	0,4	0,4	0,5	0,4	0,6	0,5	**0,01**
Breast	$POWC_1$	**98,7±1,3**	91,9±1,4	96,3±1,4	93,2±1,4	91,9±1,4	91,2±1,4	92,4±1,2	94,3
	$POWC_2$	68,3±1,2	69,1±1,6	**70%±1,4**	67,3±1,1	69,1±1,6	66,3±1,2	71,3±1,4	67,8
	SF	35,3 (0,1%)	37,0 (0,2%)	37,4 (0,2%)	40,3 (0,2%)	37,0 (0,1%)	45,3 (0,2%)	**26,5**	4325 (17,7%)
	CPU	30	31	25	32	31	31	31	**3,2**
Leukemia	$POWC_1$	**100**	98,3±0,2	**100**	**100**	98,3±0,2	**100**	**100**	68,3
	$POWC_2$	**97,2±0,4**	88,3±0,6	97,2±0,5	90,1±0,8	92,3±0,6	90,1±0,3	89,2±0,9	70,3
	SF	**8,2 (0,1%)**	**8,2 (0,1%)**	8,3 (0,1%)	27,9 (0,4%)	**8,2 (0,1%)**	27,3 (0,4%)	12,8 (0,2%)	2134 (29,9%)
	CPU	25	21	23	27	21	28	22	**3**
Arcene	$POWC_1$	**100**	**100**	**100**	**100**	**100**	**100**	**100**	74,3
	$POWC_2$	**80±1,6**	78,2±1,9	78,9±1,4	78,9±1,1	74,2±1,2	72,9±1,6	79±1,2	66,5
	SF	78,5 (0,79%)	72,5 (0,73%)	**69,4 (0,69%)**	71,1 (0,71%)	73,1 (0,73%)	72,3 (0,72%)	83,5 (0,84%)	2102 (21,02%)
	CPU	21	31	34	31	31	30	23	**3**
Gisette	$POWC_1$	**92,5±1,3**	87,3±1,5	87,3±2,1	88,3±2,4	86,4±1,2	86,3±2,1	89,5±1,4	74,3
	$POWC_2$	**85,3±1,2**	81,2±1,4	82,2±1,2	77,3±1,3	82,2±1,5	79,3±1,4	84,5±1,2	73,1
	SF	339,4 (6,8%)	340,1 (6,8%)	**330,1 (6,6%)**	341,5 (6,8%)	342,3 (6,8%)	354,5 (7,1%)	344,3 (6,9%)	1424 (28,5%)
	CPU	87	81	98	102	81	102	72	**12**
Adv	$POWC_1$	**95,5±1,5**	94,2±1,3	**95,5±1,1**	93,2±1,1	92,2±1,5	95,2±1,6	94,1±1,8	84,3
	$POWC_2$	94,2±1,1	94,4±1,9	**94,5±1,5**	80,2±1,5	88,1±1,2	92,2±1,5	90,2±1,1	77,8
	SF	5,4 (0,35%)	8,1 (0,52%)	**5,1 (0,33%)**	12,3 (0,79%)	6,4 (0,41%)	21,3 (1,37%)	7,4 (0,47%)	413 (26,5%)
	CPU	2,1	2,5	2,3	2,8	2,5	2,8	3,1	**12**

6.1.2.3 Experiment 3

In this experiment, we aim to compare the two approaches, continuous exact reformulation avec DC approximation. Among the three DCA based algorithm for DC approximation approach, DCA1 is chosen according to the Experiment 1. Furthermore, as we have proved the Capped-ℓ_1 is equivalent to our exact formulation with suitable parameters, we exclude it from this comparison and focus on the piecewise concave exponential and SCAD approximations. We also compare our proposed algorithms with

— ℓ_1-SVM : SVM with ℓ_1 regularization [35],

— ElasticNet-SVM : SVM with Elastic net regularization which is a combination between the ridge (ℓ_2 norm) and the LASSO regularization [231],

— CPLEX 12.2 for globally solving the exact formulation problem (5.51).

The comparative results are reported in Table 6.4.
Comments on numerical results :

— CPLEX only gives a solution for 3 small datasets. For large datasets, CPLEX can't furnish a solution with a CPU Time limited to 3600 seconds.

— Concerning the sparsity of solution (the number of selected feature), DCAEP-SVM and DCA1-SVM-Exp are the best : averagely, only 5% and 4.6% of features are selected, respectively. All DCA based algorithms perform better than ℓ_1-SVM and ElasticNet-SVM, especially on *Gisette* and *Breast*. All DCA based algorithms suppress considerably the number of features (up to 99% on large datasets such as Arcene and Leukemia) while the correctness of classification is quite good (from 77% to 100%). For WPBC(60) and Prostate, DCAEP-SVM suppresses more features than the other algorithms while furnishing a better classification accuracy. On other datasets, DCAEP-SVM selects slightly more features than DCA1-SVM-Exp (1 or 2 features, except for *Gisette*). Overall, DCAEP-SVM realizes a better trade-off between accuracy and sparsity than other algorithms.

— As for the accuracy of classification, DCAEP-SVM is the best for 6 out of 8 training sets. The gain is important on 2 datasets : *WPBC(24)* $10,4\%$ and *Gisette* $12,1\%$. The same conclusion goes for test sets, DCAEP-SVM is better on 6 datasets (with a gain up to 17.1% on *Gisette* dataset). ElasticNet-SVM is slightly better than DCA based algorithms (1.1% and 1.8% on two datasets *Breast* and *Ionosphere*. This can be explained by the fact that ElasticNet-SVM selects 6 (resp. 4) times more features than DCA based algorithms on *Breast* (resp. *Ionosphere*) dataset.

— In terms of CPU time, not surprisingly, ℓ_1-SVM is the fastest algorithm, with an average of CPU time $11,1$ seconds, since it only requires solving one linear program. The CPU time of DCA based algorithms is quite small, less than 101 seconds for the largest dataset (Gisette). *DCAEP-SVM* is somehow slightly faster with an average of CPU time $21,5$ seconds while that of *DCA-SVM-Exp* (resp. DCA1-SVM-SCAD) is $24,6$ (resp. $22,6$) seconds.

TABLE 6.4 – Comparison of convex, nonconvex approximation and exact reformulation

Dataset		ℓ_1-SVM	ElasticNet -SVM	DCA1-SVM -SCAD	DCA1-SVM -Exp	DCAEP -SVM	CPLEX
Ionosphere	$POWC_1$	85,6 ±1,2	86,3 ±1,6	81,5 ±1,4	82,1 ±3,1	85,2 ±1,2	**90,2**
	$POWC_2$	81,2 ±1,1	80,3 ±1,4	73,5 ±1,2	82,3 ±2,2	83,4 ±1,5	**8**3,7
	FS	10,9 (32,1%)	9,4 (27,6%)	3,1 (9,1%)	2,3 (6,8%)	3,1 (9,1%)	**2 (5,9%)**
	Time	**0,01**	0,04	0,3	0,3	0,2	2.5
WPBC(24)	$POWC_1$	74,8±1,2	76,5±1,2	77,8 ±1,3	78,3 ±1,5	**85,2 ±1,2**	77,4
	$POWC_2$	77,1±1,5	78,9±1,3	79,2 ±1,2	82,3 ±1,2	**83,5 ±1,2**	78,4
	FS	9,1 (28,4%)	8,3 (25,9%)	4,1 (12,8%)	**3 (9,4%)**	4,1 (12,8%)	7 (21,9%)
	Time	**0,03**	0,07	0,2	0,2	0,2	6.4
WPBC(60)	$POWC_1$	89,2 ±1,6	87,5 ±1,1	84,3 ±1,3	90,3 ±1,2	92,5 ±1,5	**96**
	$POWC_2$	82,5 ±1,1	87,4 ±1,3	89,4 ±1,4	92,3 ±1,4	94,2 ±1,2	**95,3**
	FS	8,3 (27,7%)	7,9 (26,3%)	5,2 (17,3%)	5,2 (17,3%)	4,2 (14,0%)	**3 (10%)**
	Time	**0,02**	0,05	0,4	0,4	0,3	1,8
Breast	$POWC_1$	94,2±1,2	**95,6±1,3**	91,4±1,4	91,5±1,5	93,8±1,1	N/A
	$POWC_2$	68,1±1,3	**71,9±1,2**	69,4±1,6	70,3±1,4	71,5±1,7	N/A
	FS	142,3 (0,6%)	102,4 (0,4%)	37,0 (0,2%)	**17,0 (0,1%)**	18,6 (0,1%)	N/A
	Time	**19**	45	31	25	25	3600
Leukemia	$POWC_1$	92,2±1,4	**100**	98,9±1,1	**100**	**100**	N/A
	$POWC_2$	96,3±1,4	96,2±1,3	**97,3±3,2**	96,2±2,4	**97,4±1,2**	N/A
	FS	12,0 (0,2%)	10,2 (0,1%)	**8,2 (0,1%)**	8,3 (0,1%)	15,3 (0,2%)	N/A
	Time	1	5	21	23	24	3600
Arcene	$POWC_1$	91,4±1,5	90,4±2,1	89,4±2,1	88,3±1,7	**96,5±3,7**	N/A
	$POWC_2$	72,2±1,3	76,2±1,2	78,2±3,7	**78,9±1,4**	78,3±2,6	N/A
	FS	81,3 (0,81%)	92,3 (0,92%)	32,5 (0,33%)	**32,4 (0,32%)**	32,8 (0,33%)	N/A
	Time	**10**	20	31	34	31	3600
Gisette	$POWC_1$	78,3±1,2	79,2±1,2	87,3±1,5	83,3±2,1	**90,4±1,2**	N/A
	$POWC_2$	68,5±1,2	80,1±1,3	82,2±1,4	82,2±1,2	**85,6±1,3**	N/A
	FS	1276,5 (25,5%)	1034,4 (20,7%)	**140,1 (2,8%)**	**140,1 (2,8%)**	165,5 (3,3%)	N/A
	Time	**45**	192	81	98	74	3600
Prostate	$POWC_1$	92,4±1,4	91,8±1,3	91,5±1,9	91,6±1,8	**94,4±1,3**	
	$POWC_2$	**100**	**100**	**100**	**100**	**100**	N/A
	FS	56,4 (0,45%)	45,4 (0,36%)	34,2 (0,27%)	32,1 (0,25%)	**30,8 (0,24%)**	N/A
	Time	**14**	16	16	16	18	3600

6.1.3 Conclusion

In this section, we have considered the feature selection in SVM problem. This problem is formulated as a sparse optimization problem which involves the ℓ_0 norm in the objective function. We then developed five DCA based algorithms proposed in Chapter 5 to solve the latter : DCA1-SVM, DCA2-SVM, DCA3-SVM, DCA4-SVM that belong to the DC approximation approach and DCAEP-SVM for the exact reformulation approach. Thanks to the special structure of considered problem, ours DCA algorithms enjoy interesting convergence properties. DCAEP-SVM, DCA1-SVM and DCA2-SVM converge to a critical point after a finite number of iterations. Several numerical experiments were conducted allowing to analyze the efficiency of our approaches and to show their superiority versus existing methods. The numerical results confirmed our analysis in Section 5.2.5.4, DCA1 is better than DCA2 and DCA3 in all three comparative criteria (accuracy, sparsity of solution and running time). Comparing to DCA1, DCAEP seems to realize a better trade-off between accuracy and sparsity with similar running time.

6.2 Sparse Semi-Supervised Support Vector Machines[1]

In machine learning, supervised learning is a task of inferring a predictor function (classifier) from a labeled training dataset. Each example in training set consists of an input object and a label. The objective is to build a predictor function which can be used to identify the label of new examples with highest possible accuracy. With the rapid development of various technologies, it is easy to collect a huge amount of data and it requires a huge amount of time and efforts to manually label the data. Hence, in most of real word applications, a large portion of training data are unlabeled and supervised learning can not be used in these contexts. To deal with this difficulty, there has been an attracting increasing attention in using semi-supervised learning methods that take into account a small amount of labeled data and a large amount of unlabeled data to construct prediction models. Semi-supervised methods also allow to improve the prediction model's robustness.

We are interested in semi-supervised classification, more precisely, in the so called Semi-Supervised Support Vector Machines (S3VM). Among the semi-supervised classification methods, the large margin approach S3VM, which extends the Support Vector Machine (SVM) to semi-supervised learning concept, is certainly the most popular [48, 50, 55, 82, 250, 159, 217, 216, 255]. S3VM was originally proposed by Vapnik and Sterin in 1977 [226] under the name of transductive support vector machine. Later, in 1999, Bennett and Demiriz [22] proposed the first optimization formulation of S3VM which is described as follows.

Given a training set which consists of m labeled points $\{(w_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}, i = 1, \ldots, m\}$ and p unlabeled points $\{w_i \in \mathbb{R}^n, i = (m+1), \ldots, (m+p)\}$. We are to find a separating hyperplane $P = \{w \in \mathbb{R}^n \mid w^T x = b\}$, far away from both the labeled and unlabeled points. Hence, the optimization problem of S3VM takes the form

$$\min_{x,b} \|x\|_2^2 + \alpha \sum_{i=1}^{m} L\left(y_i(\langle x, w_i\rangle + b)\right) + \beta \sum_{i=m+1}^{m+p} L\left(|\langle x, w_i\rangle + b|\right). \tag{6.14}$$

1. The results presented in this section were published in :
   - H.M. Le, H.A. Le Thi, M.C. Nguyen, Sparse semi-supervised support vector machines by DC programming and DCA, Neurocomputing, 153 :62-72, 2015.
   - H.M. Le, H.A. Le Thi and M.C. Nguyen, DCA based algorithms for feature selection in Semi-Supervised Support Vector Machines, Lecture Notes in Computer Science (LNCS), Vol. 7988, 528-542, 2013.

The first two terms in (6.14) define the standard SVM while the third one incorporates the loss function of unlabeled data points. The loss function of labeled and unlabeled data points are weighted by penalty parameters $\alpha > 0$ and $\beta > 0$. Usually, in classical SVM one uses the hinge loss function $L(u) = \max\{0, 1-u\}$ which is convex. On contrary, the problem (6.14) is nonconvex, due to the nonconvexity of the third term.

There are two broad strategies for solving the optimization problem (6.14) of S3VM : the combinatorial methods (Mixed IntegerProgramming [22], Branch and Bound algorithm [49]) and the continuous optimisation methods such as self-labeling heuristic S^3VM^{light} [115], gradient descent [48, 81], deterministic annealing [217], semi-definite programming [28], DC programming [55]. Combinatorial methods are not available for massive data sets in real applications (*high* dimension and *large* data set). Thus, major efforts have focused on efficient local algorithms. For more complete reviews of S3VM methods, the reader is referred to [50, 262, 62] and references therein.

We consider here the S3VM problem with feature selection. More precisely, we are to find a separating hyperplane far away from both the labeled and unlabeled points, that uses the least number of features. Similar to SVM, we replace the term $\|w\|_2^2$ in (6.14) by the ℓ_0-norm and then formulate the S3VM problem with features selection as follows :

$$\min_{x,b} \|x\|_0 + \alpha \sum_{i=1}^{m} L\left(y_i(\langle x, w_i\rangle + b)\right) + \beta \sum_{i=m+1}^{m+p} L\left(|\langle x, w_i\rangle + b|\right). \tag{6.15}$$

While S3VM has been widely studied, there exist few methods in the literature for feature selection in S3VM. Due to the discontinuity of the ℓ_0 term and the non convexity of the third term, we are facing "double" difficulties in solving (6.15). Even if we replace the ℓ_0 by a convex approximation, the resulting problem is still nonconvex. In [245], the author used the concave exponential approximation of ℓ_0-norm, that was proposed in [35], for (6.15) and developed DCA to solve the resulting problem. Later, in our conference paper [121], we proposed another DC decomposition for the concave exponential approximation and presented a DCA to solve it. Note that the DCA developed in [121] is the nothing but DCA1 applied to concave exponential approximation.

Our contributions. In this section, we will develop both nonconvex exact reformulation and DC approximation approaches presented in Chapter 5 for the S3VM with feature selection problem (6.15). According to the analysis of DCA1, DCA2 and DCA3 in Section 5.2.5.4, DCA1 seems to be more interesting than DCA2 and DCA3 since the corresponding subproblem has less constraints. Furthermore, DCA1 is expected to give sparser solution than DCA2 and DCA3. These analysis were confirmed by the numerical results given in the Section 6.1. Hence, we will develop DCAEP and DCA1 for all DC approximations except PiL for which DCA4 is available.

6.2.1 DCA based algorithms for solving S3VM with feature selection problem

Assume that the m labeled points and p unlabeled points are represented by the matrix $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times n}$, respectively. D is a $m \times m$ diagonal matrix where $D_{i,i} = y_i, \forall i = 1, \ldots, m$. Denote by e the vector of ones in the appropriate vector space. For each labeled point w_i $(i = 1, \ldots, m)$, we introduce a new variable ξ_i which represents the misclassification error. Similarly, for each unlabeled point w_i $(i = (m+1), \ldots, (m+p))$, we define r_i and s_i for two possible misclassification errors. Let r and s be vectors in $\mathbb{R}^p$ who i^{th} component are r_i and s_i respectively. Then, the final class of unlabeled w_i corresponds to the one that has minimal

misclassification. The feature selection in S3VM problem (6.15) can be rewritten as follows :

$$\min\left\{F(x,b,\xi,r,s) := \alpha\langle e,\xi\rangle + \beta\langle e,\min\{r,s\}\rangle + \|x\|_0 : (x,b,\xi,r,s)\in K\right\}, \tag{6.16}$$

where K is polyhedral convex set defined by

$$\left\{\begin{array}{ll}(x,b,\xi,r,s)\in\mathbb{R}^n\times\mathbb{R}\times\mathbb{R}^m\times\mathbb{R}^p\times\mathbb{R}^p: & D(Ax-eb)+\xi\geq e,\\ & Bx-eb+r\geq e,\\ & -Bx+eb+s\geq e,\\ & \xi,r,s\geq 0.\end{array}\right\}. \tag{6.17}$$

It is easy to see that (6.16) is a special case of (5.1) where $f(x,b,\xi,r,s) = \alpha\langle e,\xi\rangle + \beta\langle e,\min\{r,s\}\rangle$. Since f is a concave function, we can rewrite f as a DC function : $f = g - h$ with $g = 0$ and $h = -f$.

We will now develop DCA based algorithms presented in Chapter 5 to solve (6.16).

6.2.1.1 DC approximation approach for solving (6.16)

The approximate problem of (6.16) takes the form

$$\min\left\{F_r(x,b,\xi,r,s) := f(x,b,\xi,r,s) + \sum_{i=1}^{n} r(x_i) : (x,b,\xi,r,s)\in K\right\}, \tag{6.18}$$

where r is one of the sparsity-inducing functions given in Table 5.1.

For η given in Table 5.2, let $\psi(t) = \eta|t| - r(t)$. A DC decomposition of (6.18) is given by

$$\min\left\{F_r(x,b,\xi,r,s) := G_1(x,b,\xi,r,s) - H_1(x,b,\xi,r,s)\right\}, \tag{6.19}$$

where $G_1(x,b,\xi,r,s) = \eta\|x\|_1 + \chi_K(x,b,\xi,r,s)$ and $H_1(x,b,\xi,r,s) = f(x,b,\xi,r,s) + \sum\limits_{i=1}^{n}(\eta|x_i| + r(x_i))$.

The computation of a subgradient of $H_1(x,b,\xi,r,s)$ leads us to take $(\overline{x}^l,\overline{b}^l,\overline{\xi}^l,\overline{r}^l,\overline{s}^l) = (\bar{z}^l,0,-\alpha e,\overline{r}^l,\overline{s}^l)$ with $\bar{z}_i^l\in\partial\psi(x_i^l)$ and

$$\overline{r}_i^l = \begin{cases}-\beta & \text{if } r_i^l < s_i^l,\\ 0 & \text{if } r_i^l > s_i^l,\\ -\beta\mu & \text{if } r_i^l = s_i^l\end{cases} \qquad \forall i=1,\ldots,p,\ \mu\in[0,1] \tag{6.20}$$

$$\bar{s}_{l_i} = \begin{cases}0 & \text{if } r_i^l < s_i^l,\\ -\beta & \text{if } r_i^l > s_i^l,\\ -\beta(1-\mu) & \text{if } r_i^l = s_i^l,\end{cases} \qquad \forall i=1,\ldots,p,\ \mu\in[0,1] \tag{6.21}$$

The DCA1 for solving (6.18) is described in Algorithm 6.7.

It is easy to see that (6.18) is a DC polyhedral program. According to the convergence property of polyhedral DC programs, DCA1 applied to (6.18) generates a sequence $\{(x^k,b^k,\xi^k,r^k,s^k)\}$ that converges to a critical point (x^*,b^*,ξ^*,r^*,s^*) after finitely many iterations. Furthermore, if $r = r_{cap}$ and $|x_i^*|\neq\frac{1}{\theta}\ \forall i=1,\ldots,n$ then (x^*,b^*,ξ^*,r^*,s^*) is a local solution of (6.18).

Consider now the case $r = r_{PiL}$, DCA4 for solving (6.18) is given in Algorithm 6.8.

Similar to DCA1, DCA4 applied to (6.18) generates a sequence $\{(x^k,b^k,\xi^k,r^k),s^k)\}$ that converges to a critical point (x^*,b^*,ξ^*,r^*,s^*) after finitely many of iterations. Moreover, if $|x_i^*|\neq\frac{1}{\theta}\ \forall i=1,\ldots,n$, then (x^*,b^*,ξ^*,r^*,s^*) is a local solution of (6.18).

Algorithm 6.7 DCA1-S3VM : DCA1 applied on the approximate problem (6.18)

1: **Initialization :** Choose an initial point $(x^0, b^0, \xi^0, r^0, s^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R}^p$, and $l \leftarrow 0$.
2: **repeat**
3: Set $(\overline{x}^l, \overline{b}^l, \overline{\xi}^l, \overline{r}^l, \overline{s}^l, \overline{u}^l) = (\bar{z}^l, 0, -\alpha e, \overline{r}^l, \overline{s}^l, \overline{u}^l)$ following (6.20), (6.21) and $\bar{z}_i^l \in \partial\psi(x_i^l)$.
4: Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, r^{k+1}, s^{k+1})$ by solving the linear program

$$\min \eta \sum_{i=1}^{n} v_i - \langle (x, b, \xi, r, s), (\overline{x}^l, \overline{b}^l, \overline{\xi}^l, \overline{r}^l, \overline{s}^l) \rangle : (x, b, \xi, r, s, v) \in \bar{K} \tag{6.22}$$

5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

Algorithm 6.8 DCA4-S3VM : DCA4 applied on the approximate problem (6.18) with $r = r_{PiL}$

1: **Initialization :** Choose an initial point $(x^0, b^0, \xi^0, r^0, s^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R}^p$, and $l \leftarrow 0$.
2: **repeat**
3: Set $(\overline{x}^l, \overline{b}^l, \overline{\xi}^l, \overline{r}^l, \overline{s}^l, \overline{u}^l) = (\bar{z}^l, 0, -\alpha e, \overline{r}^l, \overline{s}^l, \overline{u}^l)$ following (6.20), (6.21) and $\bar{z}_i^l \in \partial\psi(x_i^l)$.
4: Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, r^{k+1}, s^{k+1})$ by solving the linear program

$$\min \frac{\theta}{a-1} \sum_{i=1}^{n} t_i - \langle (x, b, \xi, r, s), (\overline{x}^l, \overline{b}^l, \overline{\xi}^l, \overline{r}^l, \overline{s}^l) \rangle : (x, b, \xi, r, s, v) \in \bar{K}, \frac{1}{\theta} \leq t_i \tag{6.23}$$

5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

6.2.1.2 Exact reformulation approach for solving (6.16)

The continuous exact reformation of (6.16) takes the form

$$\inf\{G(x, b, \xi, r, s, u) - H(x, b, \xi, r, s, u) : (x, b, \xi, r, s, u) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R}^p \times [0, 1]^n\}, \tag{6.24}$$

where

$$\begin{aligned} G(x, b, \xi, r, s, u) &:= \chi_\Delta(x, b, \xi, r, s, u) \\ H(x, b, \xi, r, s, u) &:= -\alpha\langle e, \xi \rangle - \beta\langle e, \min\{r, s\} \rangle - e^T u - \tau p(u). \end{aligned}$$

with $\Delta := \{(x, b, \xi, r, s, u) : (x, b, \xi, r, s) \in K, u \in [0, 1]^n, |x_i| \leq M u_i,\ i = 1, \ldots, n\}$. Since K is a polyhedral convex set, so is Δ, hence χ_Δ is a polyhedral convex function. Therefore (6.24) is a polyhedral DC program with both polyhedral DC components g and h.

The computation of a subgradient of $H(x, b, \xi, r, s, u)$ leads us to take $(\overline{x}^l, \overline{b}^l, \overline{\xi}^l, \overline{r}^l, \overline{s}^l, \overline{u}^l) = (0, 0, -\alpha e, \overline{r}^l, \overline{s}^l, \overline{u}^l)$ with $\bar{r}^l$ and $\bar{s}^l$ being defined in (6.20) and (6.21)

$$\overline{u}_i^l = \begin{cases} +\tau & \text{if } u_i^l \geq 0.5, \\ -\tau & \text{if } u_i^l < 0.5 \end{cases} \quad \forall i = 1, \ldots n. \tag{6.25}$$

Finally, DCAEP (Algorithm 5.5) applied to (6.24) is presented in Algorithm 6.9.

Theorem 6.2 *(Convergence properties of DCAEP-S3VM)*

Algorithm 6.9 DCAEP-S3VM : DCA applied on continuous exact reformulation (6.24)

1: **Initialization :** Choose an initial point $(x^0, b^0, \xi^0, r^0, s^0, u^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R}^p \times [0,1]^n$, and $l \leftarrow 0$.
2: **repeat**
3: Set $(\overline{x}^l, \overline{b}^l, \overline{\xi}^l, \overline{r}^l, \overline{s}^l, \overline{u}^l) = (0, 0, -\alpha e, \overline{r}^l, \overline{s}^l, \overline{u}^l)$ following (6.20), (6.21), (6.25).
4: Solve the linear program

$$\min -\langle (x, b, \xi, r, s, u), (\overline{x}^l, \overline{b}^l, \overline{\xi}^l, \overline{r}^l, \overline{s}^l, \overline{u}^l) \rangle : (x, b, \xi, r, s, u) \in \Delta\} \tag{6.26}$$

to obtain $(x^{l+1}, b^{l+1}, \xi^{l+1}, \zeta^{l+1}, u^{l+1})$
5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

(i) DCAEP-S3VM generates a sequence $\{(x^l, b^l, \xi^l, r^l, s^l, u^l)\}$ contained in $V(\Delta)$ such that the sequence $\{f(x^l, b^l, \xi^l, r^l, s^l, u^l) + \tau p(u^l)\}$ is decreasing.

(ii) For a number τ sufficiently large, if at an iteration q we have $u^q \in \{0,1\}^n$, then $u^l \in \{0,1\}^n$ for all $l \geq q$.

(iii) The sequence $\{(x^l, b^l, \xi^l, r^l, s^l, u^l)\}$ converges to $\{(x^*, b^*, \xi^*, r^*, s^*, u^*)\} \in V(\Delta)$ after a finite number of iterations. The point $(x^*, b^*, \xi^*, r^*, s^*, u^*)$ is a critical point of Problem (6.24). Moreover if $u_i^* \neq \frac{1}{2}$ and $r_i^* \neq s_i^*$ for all $i = 1...n$, then $\{(x^*, b^*, \xi^*, r^*, s^*, u^*)\}$ is a local solution to (6.12).

The proof of Theorem 6.2 is similar to the one of Theorem 6.1.

6.2.2 Numerical experiments

Datasets. To illustrate the performances of algorithms, we performed numerical tests on 2 sets of data sets. The first set of data sets contains several real-world data sets taken from UCI Machine Learning Repository and NIPS Feature Selection Challenge. The second set of data sets is the *CBCL Face Database*, taken from MIT Center For Biological and Computation Learning (`http://cbcl.mit.edu/projects/cbcl/software-datasets/`). The *CBCL Face Database* is devoted to the face detection task that we will briefly present later (cf. Experiment 3). The *CBCL Face Database* contains 19×19 pixels grayscale images. As proposed in [208] and [99], we represent each image by a vector of $19 \times 19 = 361$ elements which correspond to the gray level of its pixels. The training set (resp. test set) *CBCL Face Database* contains $2,429$ face and $4,548$ non-face images (resp. 472 face and 23,573 non-face images).

The information about data sets is summarized in Table 6.5 (#Att is the number of features while #Train(resp. #Test) stands for the number of points in training set (resp. test set)). For GIS, LEU and ARC datasets, training and test sets are given. For the remaining datasets, training and test sets are randomly sampled from the original set.

Comparative algorithms. We compare our DCA based algorithms with the S3VM using ℓ_1-norm, namely

$$\min \{F_{\ell_1}(x, b, \xi, r, s) := \alpha\langle e, \xi\rangle + \beta\langle e, \min\{r, s\}\rangle + \|x\|_1 : (x, b, \xi, r, s) \in K\}. \tag{6.27}$$

Once again, problem (6.27) can be formulated as a DC program and then solved by DC pro-

TABLE 6.5 – Datasets

Dataset	#Att	#Train	#Test	#Total
W60	30	-	-	569
Ionosphere (INO)	34	-	-	351
Spambase (SPA)	57	-	-	2301
Internet Advertisements (ADV)	1558	-	-	3279
Gisette (GIS)	5000	6000	1000	7000
Leukemia (LEU)	7129	38	34	72
Arcene (ARC)	10000	100	100	200
CBCL Face Database	361	7077	24045	32122

gramming and DCA. F_{ℓ_1} is a DC function with the following DC decomposition

$$\begin{array}{ll} F_{\ell_1}(x,b,\xi,r,s) & = G_{\ell_1}(x,b,\xi,r,s) - H_{\ell_1}(x,b,\xi,r,s) \\ G_{\ell_1}(x,b,\xi,r,s) & = \|x\|_1 \\ H_{\ell_1(x,b,\xi,r,s)} & = -\alpha\langle e,\xi\rangle - \beta\langle e,\min\{r,s\}\rangle \end{array}$$

Then, DCA applied to (6.27) can be described as follows :

Algorithm 6.10 S3VM-ℓ_1 : DCA applied to S3VM with ℓ_1 regularization

1: **Initialization :** Choose an initial point $(x^0,b^0,\xi^0,r^0,s^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R}^p$, and $l \leftarrow 0$.
2: **repeat**
3: Compute $(\bar{x}^l,\bar{b}^l,\bar{\xi}^l,\bar{r}^l,\bar{s}^l) = (0,0,-\alpha e,\bar{r}^l,\bar{s}^l)$ with $\bar{r}^l$ and ,$\bar{s}^l$ being defined in (6.20) and (6.21).
4: Solve the linear program

$$\begin{array}{ll} \min & \langle e,t\rangle - \langle(\bar{x}^l,\bar{b}^l,\bar{\xi}^l,\bar{r}^l,\bar{s}^l),(x,b,\xi,r,s)\rangle, \\ s.t. & (x,b,\xi,r,s) \in K, \\ & t_i \geq x_i,\ t_i \geq -x_i \quad \forall i = 1,\ldots,n. \end{array} \tag{6.28}$$

to obtain $(x^{l+1},b^{l+1},\xi^{l+1},\zeta^{l+1})$
5: $k \leftarrow k+1$.
6: **until** Stopping criterion.

The second comparative algorithm is presented in [245]. The authors used the concave exponential approximation proposed in ([35]), named FSV, to replace the ℓ_0 norm in (6.15) and developed DCA to solve the resulting problem. The proposed algorithm, name S3VM-FSV is descripbed as follows. By introducing a non-negative variable $u \geq 0$ and the constraint relation $-u \leq x \leq u$, the resulting problem takes the form

$$\min\left\{F_{FSV}(x,b,\xi,r,s,u) := \alpha\langle e,\xi\rangle + \beta\langle e,\min\{r,s\}\rangle + \sum_{i=1}^{n}(1-e^{-\lambda u_i}) : (x,b,\xi,r,s,u) \in \tilde{K}\right\} \tag{6.29}$$

with $\tilde{K} = \{(x,b,\xi,r,s) \in K; -u \leq x \leq u; u \geq 0\}$. The problem (6.29) can be written as

$$F_{FSV}(x,b,\xi,r,s,u) = G_{FSV}(x,b,\xi,r,s,u) - H_{FSV}(x,b,\xi,r,s,u)$$

where $G_{FSV}(x,b,\xi,r,s,u) = \chi_{\tilde{K}}$ and $H_{FSV}(x,b,\xi,r,s,u) = -F_{FSV}(x,b,\xi,r,s,u)$. It is clear that $G_{FSV}(x,b,\xi,r,s,u)$ and $H_{FSV}(x,b,\xi,r,s,u)$ are convex functions. Therefore (6.29) is a DC program. Below is the description of DCA applied to (6.29) :

Algorithm 6.11 S3VM-FSV : DCA applied (6.29)

1: **Initialization :** Choose an initial point $(x^0, b^0, \xi^0, r^0, s^0, u^0) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R}^p \times \mathbb{R}^n$, and $l \leftarrow 0$.
2: **repeat**
3: Compute $(\bar{x}^l, \bar{b}^l, \bar{\xi}^l, \bar{r}^l, \bar{s}^l, \bar{u}^l) = (0, 0, -\alpha e, \bar{r}^l, \bar{s}^l, \bar{u}^l)$ with $\bar{r}^l$ (resp. $\bar{s}^l$) being defined in (6.20) (resp.(6.21)) and $\bar{u}_i = \lambda \epsilon^{-\lambda u_i} \quad \forall i = 1, \ldots, n$.
4: Solve the linear program

$$\begin{array}{ll} \min & -\langle (\bar{x}^l, \bar{b}^l, \bar{\xi}^l, \bar{r}^l, \bar{s}^l, \bar{u}^l), (x, b, \xi, r, s, u) \rangle, \\ s.t. & (x, b, \xi, r, s, u) \in \tilde{K}, \end{array} \tag{6.30}$$

to obtain $(x^{l+1}, b^{l+1}, \xi^{l+1}, \zeta^{l+1}, u^{l+1})$
5: $k \leftarrow k + 1$.
6: **until** Stopping criterion.

Setting

All algorithms were implemented in the Visual C++ 2005, and performed on a PC Intel i5 CPU650, 3.2 GHz of 4GB RAM. We stop DCA with the tolerance $\tau = 10^{-6}$. The non-zero elements of x are determined according to whether $|x_i|$ exceeds a small threshold (10^{-6}). On each given training set and test set, each algorithm is performed 10 times from 10 random starting points taken in the following way : ξ^0, r^0, s^0 are set to 0 and x^0 and b^0 are randomly chosen. Overall, we perform 100 executions on each dataset. We then report the best result, the average result and the standard deviation over the executions.

We are interested in the classification error and the sparsity of obtained solution as well as the rapidity of the algorithms. We measure the classification error via two criteria : the maximum sum (MS) and the accuracy (ACC) which are defined as follows :

$$MS = (SE + SP)/2, \quad ACC = (TP + TN)/(TP + TN + FP + FN),$$

where TP and TN denote true positives and true negatives while FP and FN represent false positives and false negatives. $SE = TP/(TP + FN)$ (resp. $SP = TN/(TN + FP)$) is the correctness rate of positive class (resp. negative class). The sparsity of solution is determined by the number of selected features (SF) while the rapidity of algorithms is measured by the CPU time in seconds.

6.2.2.1 Experiment 1

In the first experiment, we are interested in the effectiveness of all algorithms when the number of unlabeled points varies. For this purpose, we arbitrarily choose a data set (**LEU**) and then change the percentage of unlabeled points in training set from 20% to 80%. We report in Figure 6.2 (resp. Figure 6.1), the ACC (resp. SF) of all algorithms on training and test set.

We observe that :

— In most cases, the DCA based algorithms on the ℓ_0 model give better accuracy while choosing much less features than S3VM-ℓ_1 (the gain on number of selected features is up to $5,6$ times).

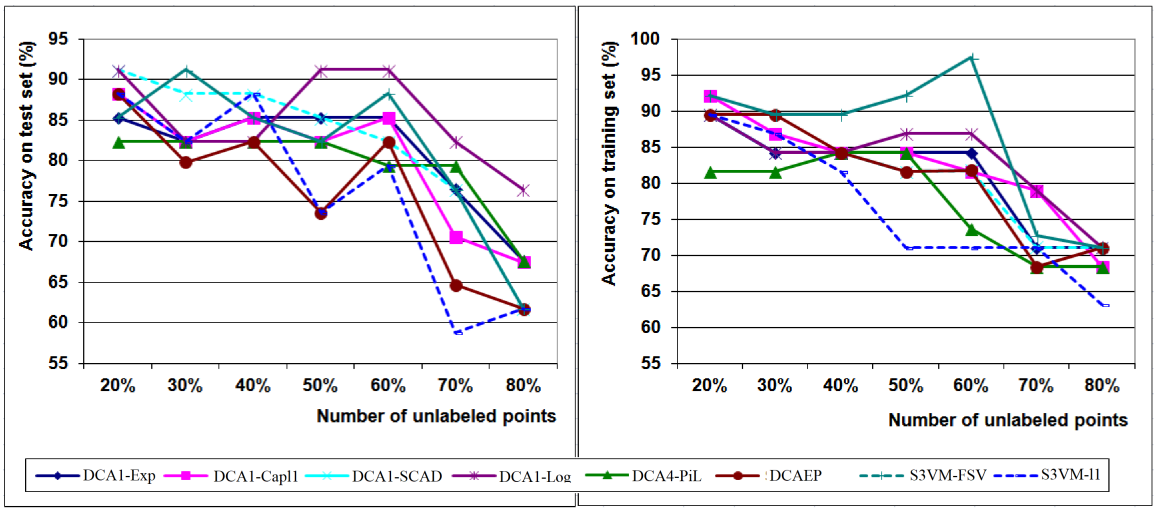


FIGURE 6.1 – Accuracy of classifiers (ACC) on training (right) and test (left) of dataset **LEU** with different number of unlabeled points.

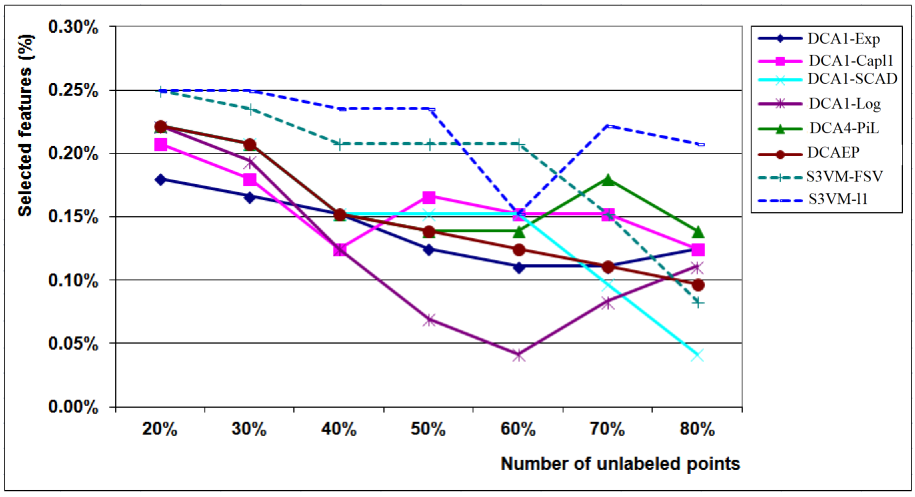


FIGURE 6.2 – Percentage of selected features on dataset **LEU** with different number of unlabeled points.

— When the number of unlabeled points exceeds 70%, the accuracy of all algorithms decrease dramatically, especially for S3VM-ℓ_1.

6.2.2.2 Experiment 2

In the second experiment, we compare the effectiveness of all the algorithms with a fixed percentage of unlabeled points . According to the second remark of experiment 1, 60% of training set will be set to be unlabeled points. The comparative results are reported in Table 6.6, Table 6.7, Table 6.8, Table 6.9.
From the computational results we observe that :

— All DCA based algorithms on ℓ_0 model reduce considerably the number of features (from 72.11% to 99.94%) while the accuracy of classifier is quite good (from 61.80% to 94.94%). In comparison with S3VM-ℓ_1, naturally the DCA based algorithms on ℓ_0 model suppress much more features while they always furnish better accuracy (MS/ACC). The gain of percentage of selected features (SF) with respect to S3VM-ℓ_1 is up to 123.29 times (DCA1-S3VM-Exp comparing to S3VM-ℓ_1 on dataset ADV). Averagely on all datasets, DCA1-S3VM-Exp is the best on term of selected features. The number of selected features of S3VM-ℓ_1 is higher than that of DCA1-S3VM-Exp (resp. DCA1-S3VM-Cappedℓ_1, DCA4-S3VM-PiL, DCA1-S3VM-SCAD, DCA1-S3VM-Log and S3VM-DCAEP) 3.39 (resp. $2.80, 2.78, 3.06, 3.19$ and 2.98) times.

— In term of ACC/MS, the quality of all five DCA based algorithms on ℓ_0 model are comparable. DCA1-S3VM-Log is better than other on 5 out of 7 datasets which can be explained by the fact that DCA1-S3VM-Log selects slightly more features than other algorithms in these datasets.

— Concerning the computation time, the CPU time of all DCA based algorithms is quite small : less than 22 seconds (except for dataset **GIS**).

6.2.2.3 Experiment 3

In this experiment, we are interested in the face detection problem which has been one of the most studied topics in the computer vision in the past few decades. The goal of face detection is to locate an unknown number (if any) of human faces in a digital image. Face detection can be exploited in numerous application areas : security, surveillance, smart card, . . . For more complete reviews of face detection approaches and its applications, the reader is referred to [104] and references therein. Face detection involves segmentation, extraction, and verification of faces ([104]). In this experiment, we are only interested in the last step of face detection, i.e. verification of faces. Classification algorithms can be applied for this task. Among them, SVMs have been successfully developed for this purpose ([99, 181]). Motivated by the good results of ours proposed approaches on numerics data sets in experiments 1 and 2, we intend to apply them for face detection problem. Our experiment is realized on *CBCL Face Database*. The comparative results are reported in Table 6.10. We observe that :

— Overall, all DCA based algorithms on l_0 model give better results than S3VM-l_1, not only for number of selected features but also the accuracy of classifiers (ACC/MS).

— Concerning the sparsity of solution, DCA4-S3VM-PiL gives the best result. DCA4-S3VM-PiL only selects 40 out of 361 features (11.08%) while giving a good accuracy (81.14% in term of ACC and 86.06% in term of MS). It does mean that, using the solution given by DCA4-S3VM-PiL, one only needs 40 out of 361 pixels to determine whether a new image

TABLE 6.6 – Selected features (and corresponding percentages) of the algorithms

Dataset	DCA1-Exp		DCA1-S3VM-capped-ℓ_1		DCA4-S3VM-PiL		DCA1-S3VM-SCAD		DCA1-S3VM-Log		DCAEP-S3VM		S3VM-SFV		S3VM-l_1	
	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD
W60(#)	**2**	2.92±0.6	4	4.12±0.3	5	5.24±0.4	**2**	**2.44**±0.5	3	3.40±0.8	3	4.27±1.1	4	4.28±0.5	6	6.61±0.6
(%)	**6.67**	9.73±2.1	13.33	13.73±1.1	16.67	17.47±1.4	**6.67**	**8.13**±1.6	10.00	11.33±2.7	10.00	14.23±3.7	13.33	14.27±1.5	20.00	22.03±2.1
INO(#)	**4**	**5.5**±0.7	5	8.21±1.8	7	7.31±0.3	5	6.01±0.6	5	8.11±2.9	6	7.02±0.4	6	6.64±0.9	28	29.10±0.6
(%)	**11.76**	**16.18**±2.0	14.71	24.15±5.3	20.59	21.50±1.0	14.71	17.68±1.6	14.71	23.85±8.5	17.65	20.65 ±1.3	17.65	19.53±2.6	82.35	85.59±1.9
SPA(#)	4	9.69±3.5	**3**	6.54±2.5	5	12.35±4.5	5	9.65±4.1	4	6.41±1.0	5	12.31±3.5	**3**	**6.26**±1.7	15	23.71±5.2
(%)	7.02	17.00±6.2	**5.26**	11.47±4.4	8.77	21.67±7.8	8.77	16.93±7.2	7.02	11.25±1.8	8.77	21.60 ±6.2	**5.26**	**10.98**±3.0	26.32	41.60±9.2
ADV(#)	**2**	**3.97**±1.6	14	30.93±13.8	24	45.60±14.3	15	25.28±6.6	3	4.32±0.6	3	27.24±14.9	32	38.17±6.9	265	489.4±109.7
(%)	**0.13**	**0.25**±0.1	0.90	1.99±0.9	1.54	2.93±0.9	0.96	1.62±0.4	0.19	0.28±0.0	0.19	1.75±0.9	2.05	2.45±0.4	17.01	31.42±7.0
GIS(#)	1082	1112.8±27	1344	1394.4±34	**525**	**808.3**±279	1334	1389.0±37	1122	1144.8±14	**525**	**808.3**±279	1135	1237.6±51	1918	2073.5±122
(%)	21.64	22.26±0.5	26.88	27.89±0.7	**10.5**	**16.17**±5.6	26.68	27.78±0.7	22.44	22.90±0.3	**10.5**	**16.17**±5.6	22.70	24.75±1.0	38.36	41.47±2.4
LEU(#)	8	14.40±2.9	11	16.10±2.8	10	12.60±3.0	11	17.50±3.6	**3**	**4.50**±1.3	9	13.20±2.4	15	20.40±12.9	11	18.40±5.5
(%)	0.11	0.20±0.0	0.15	0.23±0.0	0.14	0.18±0.0	0.15	0.25±0.0	**0.04**	**0.06**±0.0	0.13	0.19±0.0	0.21	0.29±0.2	0.15	0.26±0.1
ARC(#)	**10**	**10.30**±0.4	14	21.80±6.2	18	42.10±15.2	46	62.9±12.6	15	15.70±0.8	13	20.20±4.4	20	29.60±16.1	51	71.20±13.4
(%)	**0.10**	**0.10**±0.0	0.14	0.22±0.1	0.18	0.42±0.1	0.46	0.63±0.1	0.15	0.16±0.0	0.13	0.20±0.0	0.20	0.30±0.2	0.51	0.71±0.1
Avg.(%)	6.78	**9.39**±1.57	8.8	11.38±1.8	8.34	11.48±2.4	8.34	10.43±1.7	7.79	9.98±1.9	**6.77**	10.68±2.5	8.77	10.37±1.3	26.39	31.87±3.3

TABLE 6.7 – Accuracy of classifiers of the algorithms on : (1) - test set, (2) - training set

Dataset	DCA1-S3VM -Exp		DCA1-S3VM -Capped-ℓ_1		DCA4-S3VM -PiL		DCA1-S3VM -SCAD		DCA1-S3VM -Log		DCAEP-S3VM		S3VM-SFV		S3VM-l_1	
W60(1)	**94.69**	90.94±2.6	93.36	90.11±3.5	**94.69**	**93.66**±0.9	92.06	91.88±0.4	92.92	90.09±3.1	92.92	90.53±0.8	89.42	84.78±2.7	84.13	84.01±0.1
(2)	93.29	91.35±1.3	**94.46**	**92.40**±3.4	**94.46**	90.41±3.3	92.90	91.93±1.9	91.25	89.91±2.0	93.88	91.65±1.1	86.05	83.42±2.9	85.66	83.66±1.2
INO(1)	**85.05**	76.31±2.2	84.25	74.27±1.7	78.81	75.96±1.7	75.32	74.66±0.4	77.78	**76.68**±1.1	76.92	75.87±1.5	77.78	75.47±2.8	72.89	71.69±0.3
(2)	78.52	73.89±2.4	84.77	74.98±7.5	82.24	80.88±0.7	80.55	80.04±0.1	85.90	82.56±2.4	82.24	80.86±0.4	81.2	77.66±3.2	**88.46**	**88.26**±0.4
SPA(1)	**65.36**	**63.56**±0.4	63.72	63.38±0.3	63.72	62.88±0.5	63.88	62.87±0.4	63.72	62.87±0.3	63.78	62.89±0.4	63.89	63.02±0.7	63.55	62.11±0.5
(2)	**65.68**	**63.17**±0.4	63.36	62.89±0.4	64.49	62.36±0.7	63.97	62.99±0.5	63.97	62.98±0.5	64.41	62.55±0.6	64.7	62.56±0.5	64.56	62.74±0.6
ADV(1)	95.50	93.17±1.8	95.04	**93.81**±0.8	95.04	93.48±3.2	94.36	92.44±1.2	95.04	93.61±1.2	**95.88**	93.20±1.9	92.81	91.77±0.8	93.88	92.14±0.6
(2)	95.63	93.60±1.8	**96.44**	**94.94**±1.0	95.63	94.18±0.9	95.22	93.22±1.1	95.22	93.89±0.9	96.24	93.96±1.6	95.04	93.21±0.9	95.32	94.08±1.0
GIS(1)	68.40	67.34±0.8	70.10	68.24±1.3	68.60	66.48±1.1	**73.09**	69.32±1.8	70.70	**70.00**±0.4	68.60	66.48±1.1	69.90	69.07±0.8	65.10	62.94±1.6
(2)	69.65	68.98±0.5	69.08	67.94±0.9	67.87	65.61±1.1	69.07	68.18±0.9	**70.9**	**69.73**±0.6	67.87	65.61±1.1	69.38	69.12±0.3	64.60	63.49±0.5
LEU(1)	85.29	74.41±4.6	85.29	75.88±4.9	79.41	72.06±3.5	82.35	76.77±2.8	**91.18**	**84.41**±3.4	82.35	67.65±5.1	88.24	76.47±4.9	79.41	71.18±4.3
(2)	84.21	67.89±9.8	81.58	73.95±5.9	73.68	61.84±6.9	81.79	71.84±7.3	**94.97**	**85.53**±5.2	73.68	66.05±4.1	97.37	80.00±7.8	71.06	63.68±6.6
ARC(1)	75.00	75.00±0.0	76.00	72.50±2.2	71.00	66.80±2.2	68.00	64.00±1.6	**79.00**	**75.30**±1.7	71.00	67.50±1.5	72.00	70.80±0.7	75.00	66.40±3.2
(2)	76.00	75.30±0.9	69.00	66.67±1.7	73.00	61.80±5.8	78.00	69.90±4.4	**81.00**	**78.90**±1.0	73.00	68.00±3.8	81.00	75.80±3.3	71.00	62.80±4.2
Avg.(1)	78.97	77.25±1.9	81.11	76.88±2.1	78.75	75.90±1.8	78.44	75.99±1.2	**81.48**	**78.99**±1.6	78.78	74.87±1.8	79.15	75.92±1.9	76.28	72.92±1.5
(2)	80.43	76.31±2.5	79.81	76.25±2.9	78.77	73.87±2.8	80.21	76.87±2.3	**83.32**	**80.50**±1.8	78.76	75.53±1.8	82.11	77.40±2.7	77.24	74.10±2.1

TABLE 6.8 – Maximum Sum of the algorithms on : (1) - test set, (2) - training set

Dataset	DCA1-S3VM-Exp		DCA1-S3VM-Capped-ℓ_1		DCA4-S3VM-PiL		DCA1-S3VM-SCAD		DCA1-S3VM-Log		DCAEP-S3VM		S3VM-SFV		S3VM-l_1	
	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD	Best	Mean±SD
W60(1)	**94.47**	90.97±2.3	91.96	91.49±0.3	94.08	**93.66**±1.2	91.74	91.44±0.6	92.47	91.24±2.3	93.15	91.47±0.8	91.6	87.23±1.7	86.81	86.63±0.5
(2)	94.02	91.49±1.9	94.95	92.06±2.1	94.02	**92.77**±2.7	89.52	89.37±0.2	91.07	90.29±1.0	**94.49**	92.67±0.9	88.72	85.35±1.2	85.57	85.21±0.3
INO(1)	**82.32**	75.01±2.7	78.89	73.71±1.9	77.54	72.91±0.3	73.33	72.66±0.1	82.14	**75.11**±2.3	75.55	72.85±0.3	76.62	74.21±2.0	69.68	69.18±0.2
(2)	78.36	76.39±2.2	**85.67**	76.17±5.8	81.54	81.11±0.1	81.87	81.55±0.6	85.66	82.72±3.1	82.24	81.22±0.1	83.5	80.36±2.6	83.94	**83.02**±0.3
SPA(1)	54.69	53.88±0.4	**54.88**	53.78±0.3	54.65	53.18±0.4	53.88	53.24±0.3	54.14	**53.91**±0.4	54.55	53.84±0.7	54.39	**53.91**±0.5	54.12	52.99±0.7
(2)	53.89	53.12±0.5	54.52	**53.57**±0.4	54.88	53.56±1.2	53.98	52.19±0.7	53.89	52.87±0.4	54.96	53.11±1.0	55.15	52.55±1.1	**54.98**	52.81±0.8
ADV(1)	92.13	90.41±1.7	**92.52**	89.35±1.7	**92.52**	**91.31**±0.8	89.62	87.42±1.1	89.06	83.00±4.5	92.40	85.45±7.1	88.91	87.19±0.8	89.06	86.69±2.9
(2)	**94.65**	92.24±0.9	94.17	92.17±1.3	94.24	93.66±0.5	93.04	89.96±1.7	88.49	83.64±3.7	94.18	87.50±7.4	94.44	93.12±0.6	94.36	**93.84**±0.4
GIS(1)	68.40	67.34±0.8	70.10	68.24±1.3	68.60	66.48±1.1	**74.95**	69.67±2.4	70.70	**70.00**±0.4	68.60	66.48±1.1	69.90	69.07±0.8	65.10	62.94±1.6
(2)	69.65	68.98±0.5	69.08	67.94±0.9	67.87	65.61±1.1	69.07	68.18±0.9	**70.9**	**69.73**±0.6	67.87	65.61±1.1	69.38	69.12±0.3	64.60	63.49±0.5
LEU(1)	83.21	70.86±5.8	84.29	73.29±5.8	80.36	72.18±4.1	78.57	74.46±2.9	**90.36**	**81.29**±4.6	81.79	69.93±4.3	85.71	71.86±5.8	80.36	68.32±5.5
(2)	86.20	65.56±11.6	75.25	69.82±6.3	66.84	58.60±6.1	87.04	69.68±8.6	90.91	**75.00**±8.9	78.79	67.22±5.6	**90.16**	66.12±12.5	68.86	58.01±7.7
ARC(1)	74.51	**74.51**±0.0	74.68	72.16±1.6	70.94	66.17±3.5	67.78	64.79±1.2	**77.11**	73.81±1.5	70.7	66.62±1.5	71.10	69.03±1.5	74.76	66.32±3.3
(2)	75.41	74.78±0.8	70.13	66.86±2.2	73.21	61.14±7.1	78.9	71.06±4.7	**80.36**	**78.26**±1.1	72.48	68.00±3.8	78.90	74.47±2.4	72.89	63.13±4.2
Avg.(1)	78.53	74.71±1.9	78.19	74.57±1.8	76.96	73.70±1.6	75.70	73.38±1.2	**79.43**	**75.48**±2.3	76.68	72.38±2.7	76.89	73.21±1.9	74.27	70.44±2.1
(2)	78.88	74.65±2.6	77.68	74.08±2.7	76.09	72.35±2.7	79.06	74.62±2.5	**80.18**	**76.07**±2.7	77.86	73.62±2.8	80.04	74.44±2.9	75.03	71.36±2.0

TABLE 6.9 – Comparisons of the algorithms in terms of CPU time

Data	DCA1-S3VM-Exp	DCA1-S3VM-Capped-ℓ_1	DCA4-S3VM-PiL	DCA1-S3VM-SCAD	DCA1-S3VM-Log	DCAEP-S3VM	S3VM-SFV	S3VM-l_1
W60	0.133±0.020	0.199±0.009	0.181±0.009	0.229±0.008	0.174±0.006	**0.087**±0.011	0.255±0.006	0.149±0.017
ING	0.125±0.103	0.181±0.011	**0.101**±0.003	0.157±0.08	0.139±0.066	0.113±0.016	0.219±0.004	0.169±0.03
SPA	**0.258**±0.150	0.445±0.055	0.484±0.054	0.437±0.06	0.451±0.053	0.499±0.054	0.468±0.089	0.491±0.026
ADV	7.411±1.29	7.247±0.084	7.57±0.019	7.028 ±0.092	**5.105**±0.146	6.878±0.169	6.785±0.177	7.695±0.207
GIS	**551.55**±41.98	630.92±169.79	819.66±431.71	566.34±10.41	598.32±18.17	819.65±431.70	599.87±49.73	553.77±6.94
LEU	6.01±0.26	5.980±0.455	6.253±1.047	**5.788**±1.00	5.937±0.225	6.348±1.569	6.059±0.144	6.265±0.227
ARC	**11.68**±0.34	13.254±0.816	18.64±2.050	13.228±1.19	12.157±0.548	21.739±4.677	11.022±0.602	14.439±0.596
Average	**82.45**±6.31	94.03±24.46	121.84±62.13	84.74±1.83	88.89±2.75	122.19±62.60	89.24±7.25	83.28±1.15

is face image or not. The gain of percentage of selected features with respect to S3VM-l_1 is 3.375 times.

— In term of accuracy of classifiers, DCA1-S3VM-Exp (resp. S3VM-DCAEP) furnishes the best ACC on the test set (resp. on the training set) with 83.20% (resp. 86.17%) in average. Similarly, S3VM-DCAEP gives the best MS on the training set with 89.16% whereas, on the test set, DCA1-S3VM-Cappedℓ_1 achieves the best result with 77.61% in average.

— For the CPU time, DCA1-S3VM-Log is the fastest algorithm with 52.69 seconds in average.

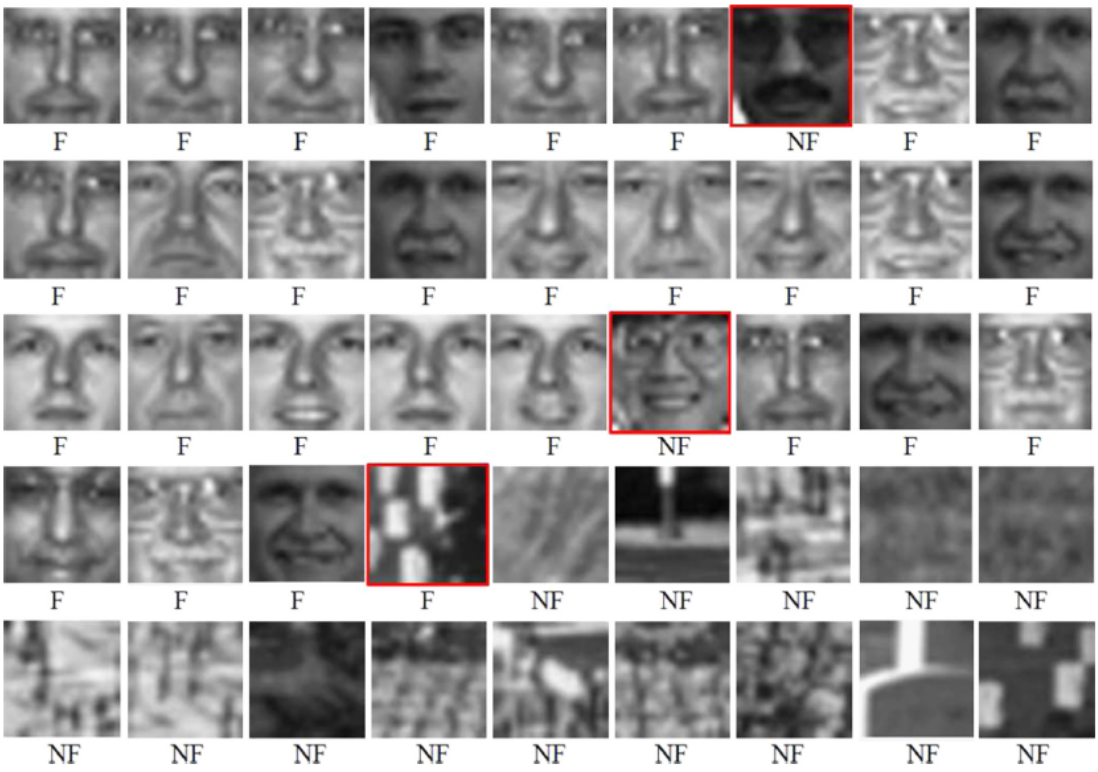


FIGURE 6.3 – Some image detection results given by DCA4-PiL. F :Face,NF :Non-Face.

6.2.3 Conclusion

Using 5 different approximations and the continuous exact reformulation via an exact penalty technique, we have investigated 6 DCA based algorithms for the S3VM with feature selection problem. All algorithms converge to a critical point after finitely many iterations. Numerical results on several real datasets showed the robustness, the effectiveness of the our algorithms. Our algorithms select much less features than the S3VM using ℓ_1 regularization, the gain is up to 123.29 times. The test performed on the face detection problem is promising. Our algorithm only needs 40 out of 361 pixels to determine whether a new image is face image or not with an accuracy of 86.06%.

TABLE 6.10 – Numerical results on CBCL Face Database

		DCA1-S3VM-Exp	DCA1-S3VM-Capped-ℓ_1	DCA4-S3VM-PiL	DCA1-S3VM-SCAD	DCA1-S3VM-Log	DCAEP-S3VM	S3VM-SFV	S3VM-l_1
SF	Best	81 (22.44%)	98 (27.15%)	**40 (11.08%)**	103 (28.53%)	108 (29.92%)	95 (26.32%)	99 (27.42%)	135 (37.40%)
	Mean	93.30±6.54	105.60±4.25	**48.50±8.50**	109.80±2.44	109.70±1.35	98.20±2.18	103.2±3.03	144.90±5.94
ACC on test set	Best	**83.84**	80.37	81.14	82.58	84.57	81.14	77.35	73.89
	Mean	**83.20±0.43**	80.13±0.33	79.06±1.93	79.77±1.84	80.62±3.30	81.02±0.07	75.15±0.98	71.89±1.13
ACC on training set	Best	86.34	82.34	82.49	82.44	80.05	**86.37**	83.4	73.73
	Mean	84.96±0.76	82.06±0.21	80.84±1.54	80.27±1.40	77.00±2.14	**86.17±0.08**	82.95±0.30	72.66±0.54
MS on test set	Best	76.30	**77.85**	77.23	75.89	78.81	75.84	72.73	67.75
	Mean	70.60±2.11	77.61±0.35	73.35±3.42	70.74±3.27	74.78±4.81	75.72±0.06	71.48±0.66	65.88±0.90
MS on training set	Best	89.27	86.16	86.06	86.15	84.45	**89.31**	87.06	79.74
	Mean	88.21±0.58	85.94±0.21	84.86±1.08	84.56±1.02	82.17±2.14	**89.16±0.06**	86.72±0.23	78.94±0.41
CPU	Mean	119.54±15.31	121.18±17.43	213.10±15.37	121.37±33.56	**52.69±9.46**	179.79±18.25	127.63±9.28	172.72±32.26

6.3 Sparse Signal Recovery[1]

Compressed Sensing (CS) (or compressive sensing or compressive sampling) which was introduced by Donoho [64] and Candes et al. [38], is an emerging area having significant interest in data analysis. CS is considered as a new framework for signal acquisition and sensor design to remove the deficiencies in the classical approach. CS suggests that one can represent many signals by using only a few non-zero coefficients in a suitable basis or dictionary and then, one can recover these signals from a few non-adaptive, linear measurements. Since CS was introduced, it is applied in various fields including radar imaging, signal extraction, medical imaging, geophysics, oil-exploration, landmine detection, civil engineering, etc.

Let us firstly give some basic definitions and notations in CS. For a complete study of CS the reader is referred to [68] and the references therein. We rely on a signal representation in a given basis $\{\psi_i\}_{i=1}^n$ for $\mathbb{R}^n$. Every signal $x \in \mathbb{R}^n$ is representable in terms of n coefficients $\{\theta_i\}_{i=1}^n$ as $x = \sum_{i=1}^{n} \psi_i \theta_i$. Arranging the ψ_i as columns into the $n \times n$ matrix Ψ and the coefficients θ_i into the $n \times 1$ coefficient vector θ, we can write that $x = \Psi\theta$, with $\theta \in \mathbb{R}^n$. In a general setting, we refer to Ψ as the sparsifying dictionary [68].

A vector $x \in \mathbb{R}^n$ is called $k - sparse$ in the basis or frame Ψ if there exists a vector $\theta \in \mathbb{R}^n$ with only $k \ll n$ nonzero entries such that $x = \Psi\theta$. The set of indices nonzero entries is called the *support* of θ and denote it by *supp(θ)*.

If a signal is not sparse itself, we can sparsify it by choosing an appropriate representation system. There exist various well-known transformations to sparsify a signal, for example Fourier, Cosine, wavelet, curvelet,... Signals are known to be very nearly sparse when represented using these transforms, for example wavelet transform. The wavelet transform consists of recursively dividing the image into its low- and high-frequency components. The lowest frequency components provide a coarse scale approximation of the image, while the higher frequency components fill in the detail and resolve edges ([59]).

A signal is called *compressible signal* if it has the vector of coefficients in certain basis is composed of few large coefficients and other coefficients with small value. This kind of signal is not really sparse signal. However, if the small coefficients are set to zero, the remaining large coefficients can represent the original signal almost exactly.

The research in CS can be classified into two major contribution areas. The first one consists of the theory and applications related to finding a sensing matrix A to ensure that it preserves the information of the signal x. The second area includes reconstruction techniques for recovering the original sparse signal x from its measurement $y = Ax$ via a sensing matrix A.

In this work, we focus on the problem of sparse signal recovery. Suppose that the signal x is already sparse. The problem can be stated as follows. Given a sensing matrix $A \in \mathbb{R}^{m\times n}$ $(m << n)$ and a measurement vector $y = Ax \in \mathbb{R}^m$. We are to recover the sparse signal $x \in \mathbb{R}^n$.

Since the linear system $y = Ax$ is highly underdetermined, and has therefore infinitely many solutions, the recovery sparse signal can be seen as finding mutually the sparsest signal x being consistent with its measurement. This leads to solving the ℓ_0-minimization problem :

$$(P) \qquad \alpha := \min \{||x||_0 : Ax = y, x \in \mathbb{R}^n\}. \tag{6.31}$$

1. A part of the results presented in this section was published in :

- H.A. Le Thi, T.B.T. Nguyen, H.M. Le, Sparse Signal Recovery by Difference of Convex Functions, Lecture Notes in Computer Science (LNCS) 7803, 387-397, 2012.

An alternative version of (P), due to the hight underdetermination of the linear system $y = Ax$, can be formulated as follows : finding a sparse signal vector x which is as consistent with y as possible according to the square error criterion. Then the resulting optimization problem, named Regularization Least Square model (RLS), is written as

$$(RLS) \qquad \alpha_\lambda = \min \left\{ \frac{1}{2}\|Ax - y\|^2 + \lambda \|x\|_0 : x \in \mathbb{R}^n \right\}. \tag{6.32}$$

where $\lambda > 0$, called the regularized parameter, represents a trade-off between error and sparsity.

Intuitively, if λ decreases to zero, we attach more importance in $\|Ax - b\|_2^2$ and it seems normal that, if x_λ is a solution of (RLS), $\|Ax_\lambda - b\|_2^2$ decreases and x_λ becomes a good approximation of a solution of (P). The following proposition expresses it in a rigorous way.

Proposition 6.1 *Assume that the linear system $Ax = y$ admits a solution. Then*

1. *$\alpha_\lambda \leq \lambda\alpha$, $\forall \rho > 0$,*
2. *there exists $\lambda_0 > 0$, such that $\alpha_\lambda = \lambda\alpha$ and (P) and (RLS) have the same solution set, $\forall 0 < \lambda \leq \lambda_0$.*

Proof. The property 1) comes from the fact that any feasible point x of (P) is feasible for (RLS) and satisfies $Ax = y$.
The proof of 2) can be found in [220].

Many efficient algorithms have been developed in literature to deal with the problem (P) or (RLS). The readers are referred to [97, 197, 201] for complete reviews on existing work on compressed sensing. As the minimization of ℓ_0 is NP-Hard, alternates have been proposed. The most commonly-used is to use the ℓ_1 norm, as given in [67], that leads to a convex optimization problem. Another common alternate in CS consists in using the nonconvex regularization $l_p(0 < p < 1)$ norm. In [253, 252], the authors proposed to replace ℓ_0 by a nonconvex Lipschitz continuous regularization, that is $\ell_1 - \ell_2$. This hybrid norm model is proved to be closer to ℓ_0 than ℓ_1 and can improve the robustness of ℓ_1. Later, inspired by this idea, Zhou and Yu [261] introduced the $\ell_p - \ell_1$ (0<p<1) regularization. It is obvious that $\ell_1 - \ell_2$ and $\ell_p - \ell_1$ are DC functions and DCA was developed in [253, 252, 261] to solve the resulting optimization problems.

Our contributions. In this work, we deal with the RLS model (6.32). Clearly, (6.32) is a particular case of the sparse optimization model (5.1). We will develop DCA1 for 3 approximations Capped-ℓ_1, concave exponential, SCAD and DCA4 for PiL (recall that for PiL approximation, only DCA4 is available (c.f. Section 5.2.5.5). This choice is justified by the performance of DCA1 and these approximations in our numerical experiments on sparse SVM (Section 6.1) and sparse S3VM (Section 6.2).

6.3.1 DCA for sparse signal recovery

It is easy to see that (6.32) is a special case of (5.1) with $f(x) := \frac{1}{2}\|Ax - y\|^2$ and $K = \mathbb{R}^n$. Here, f is convex, and the DC components of f can simply be $g = f$ and $h = 0$.

The approximate problem of (6.32) takes the form :

$$\min \left\{ F_r(x) := f(x) + \lambda \sum_{i=1}^{n} r(x_i) : (x) \in \mathbb{R}^n \right\}, \tag{6.33}$$

where r is one of the sparsity-inducing functions given in Table 5.1. For η given in Table 5.2, let $\psi(t) = \eta|t| - r(t)$. A DC decomposition of (6.33) is given by

$$\min \{F_r(x) := G_1(x) - H_1(x)\}, \tag{6.34}$$

where $G_1(x) = \frac{1}{2}\|Ax - y\|^2 + \lambda\eta\|x\|_1$ and $H_1(x) = \lambda \sum_{i=1}^{n} \psi(x_i)$. The DCA1 for solving (6.33) is described as in Algorithm 6.12.

Algorithm 6.12 DCA1-RLS : DCA1 applied on the approximate problem (6.33)

1: **Initialization :** Choose an initial point $x^0 \in \mathbb{R}^n$, and $l \leftarrow 0$.
2: **repeat**
3: Set $\bar{x}_i^l \in \partial\psi(x_i^l)$.
4: Compute x^{k+1} by solving the linear program

$$\min \frac{1}{2}\|Ax - y\|^2 + \eta \sum_{i=1}^{n} v_i - \langle x, \overline{x}^l \rangle : v_i \geq x_i, v_i \geq -x_i \tag{6.35}$$

5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

Consider now $r = r_{PiL}$. DCA4 for solving (6.33) is given in Algorithm 6.13.

Algorithm 6.13 DCA4-RLS : DCA4 applied on the approximate problem (6.33) with $r = r_{PiL}$

1: **Initialization :** Choose an initial point $x^0 \in \mathbb{R}^n$, and $l \leftarrow 0$.
2: **repeat**
3: Set $\bar{x}_i^l \in \partial\psi(x_i^l)$.
4: Compute x^{k+1} by solving the linear program

$$\min \frac{1}{2}\|Ax - y\|^2 + \frac{\theta}{a-1} \sum_{i=1}^{n} t_i - \langle x, \overline{x}^l \rangle : v_i \geq x_i, v_i \geq -x_i, \frac{1}{\theta} \leq t_i \tag{6.36}$$

5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

6.3.2 Numerical experiments

Dataset. Our experiments are performed on 11 datasets (Prob1-Prob11) taken from Sparco Toolbox, a well-known testing framework for sparse reconstruction problem.

Comparative algorithms

We compare our algorithms with 5 other ones :

— *GPSR* ([72]) - a gradient projection algorithm for solving

$$\min\{\frac{1}{2}\|Ax - y\|^2 + \lambda\|x\|_1\} \tag{6.37}$$

— *L1eq* ([40]) also known as basis pursuit, solves the problem

$$\min\{\|x\|_1 : Ax = y\} \tag{6.38}$$

— *L1qc* ([39]) that addresses the following problem

$$\min\{\|x\|_1 : \|Ax - y\|_2 \leq \epsilon\} \tag{6.39}$$

— *L1dantzig* ([41]) which deals with the Dantzig selector constraint

$$\min\{\|x\|_1 : \|A^*(Ax - y)\|_\infty \leq \gamma\} \tag{6.40}$$

where $\gamma > 0$ is a regularization parameter.

— *SL0* ([168]) which uses the concave exponential approximation to replace ℓ_0 in (6.31).

Setting

We use two criteria for evaluation : the value $\|x\|_0$ and the MSE (Mean Square Error) values. The first criterion presents the sparsity of solution. The second criterion MSE quantify the difference between the value estimated and the true value. MSE is defined by : $MSE = \|x - x_0\|^2/n$.

All our algorithms were developed in Visual C++ 2008, and performed on a PC Intel Core(TM)2 Quad CPU Q9505, 2.83 GHz and 4GB RAM. CPLEX 12.3 was used for solving quadratic sub-problems.

Numerical results

The numerical results are reported in Table 6.11. The sparsity of the original signal $\|x_0\|_0$ is given in the first column.

We observe that :

— As for the sparsity of solution, DCA1-SCAD is the best, following by our three other algorithms. DCA1-SCAD recovers the exact value of $\|x_0\|_0$ in 9 out of 11 datasets. Even if DCA1-SCAD fails to give the exact value of $\|x_0\|_0$ on *Prob3* and *Prob4* datasets, the difference is small. Our algorithms are clealy better than SL0 which also uses a nonconvex aproximation. SL0 recovers the true value of $\|x_0\|_0$ on only 2 datasets while the difference between the sparsity of its solutions and $\|x_0\|_0$ is huge for the rest. l1dz seems to be the worst algorithm in this experiment, it fails to give a sparse solution on all datasets. In some cases, GPSR gives a much more sparser solution than the other algorithms. However, its MSE is much more bigger, meaning that GPSR recovers the wrong components.

— Concerning the MSE criteion, our algorithms are clearly better than the others existing methods. They gives best value of MSE on all 11 datasets.

TABLE 6.11 – Comparison of $\|x\|_0$ and MSE

Dataset		DCA1 -Exp	DCA1 -SCAD	DCA1 -Capped-ℓ_1	DCA4 -Pil	l1dz	l1eq	l1qc	GPSR	SL0
Prob1	$\|x\|_0$	4	4	4	4	2048	505	1040	35	1651
($\|x_0\|_0$=4)	MSE	3.69E-07	1.43E-06	1.33E-06	2.30E-06	1.24E-04	6.53E-06	1.93E-03	1.4937	1.2199
Prob2	$\|x\|_0$	71	71	71	71	733	71	71	67	71
($\|x_0\|_0$=71)	MSE	2.56E-05	1.07E-05	1.00E-05	1.03E-05	1.00E-05	1.03E-05	3.12E-05	0.0656	1.03E-05
Prob3	$\|x\|_0$	119	114	119	128	2048	252	257	38	1255
($\|x_0\|_0$=121)	MSE	3.41E-05	6.41E-05	3.93E-05	3.24E-06	2.73E-06	3.09E-06	1.57E-05	0.0302	3.34E-06
Prob4	$\|x\|_0$	116	112	119	119	2048	119	274	35	644
)$\|x_0\|_0$=119)	MSE	3.12E-05	2.50E-05	0	1.45E-06	5.23E-07	1.95E-15	1.42E-05	0.0292	2.13E-07
Prob5	$\|x\|_0$	113	63	118	127	2048	280	302	37	1256
($\|x_0\|_0$=63)	MSE	8.66E-06	9.06E-07	9.87E-06	3.82E-05	1.54E-05	1.18E-05	5.58E-05	0.0199	0.0037
Prob6	$\|x\|_0$	191	191	191	191	2048	595	598	594	1569
($\|x_0\|_0$=191)	MSE	1.84E-08	8.16E-09	9.40E-08	1.05E-07	0.0058	0.0057	0.0057	0.0226	0.0031
Prob7	$\|x\|_0$	20	20	20	47	2560	20	20	0	327
($\|x_0\|_0$=20)	MSE	9.49E-07	9.68E-07	9.68E-07	2.38E-05	1.39E-06	3.90E-16	2.99E-05	0.0078	1.16E-07
Prob8	$\|x\|_0$	20	20	20	20	2560	20	20	0	184
($\|x_0\|_0$=20)	MSE	1.06E-06	0	1.09E-06	1.42E-05	1.57E-06	3.90E-16	3.13E-05	0.0078	7.33E-08
Prob9	$\|x\|_0$	12	12	12	12	128	12	54	16	12
($\|x_0\|_0$=12)	MSE	2.86E-05	3.51E-06	0	0	1.47E-06	0	3.93E-05	0.0726	0
Prob10	$\|x\|_0$	12	12	12	12	1024	12	496	151	12
($\|x_0\|_0$=12)	MSE	0.0001	0.0001	0.0001	0.0001	0.0023	0.00017	0.0140	78.836	0.00017
Prob11	$\|x\|_0$	32	32	32	32	1024	35	37	34	191
($\|x_0\|_0$=32)	MSE	1.22E-08	4.55E-09	1.23E-08	3.16E-06	1.56E-08	1.67E-08	9.19E-08	6.97E-07	1.91E-07

6.3.3 Conclusion

We considered the sparse signal recovery in compressed sensing which involves the minimization of ℓ_0 norm. Four DCA based schemes based on DC approximation approach were investigated for the RLS (Regularization Least Square) model. The numerical experiments on well-known Sparco toolbox datasets have showed the efficiency of our algorithms. In most of cases, they successfully recover a signal with the same sparsity of the original one and small MSE (Mean Square Error).

Troisième partie

DC Programming and DCA for Clustering

In this part, we develop DCA based algorithms to address several problems in unsupervised classification, commonly known as clustering.

In Chapter 7, we deal with the minimum sum-of-squares clustering (MSSC in short). Among many criteria used in clustering, MSSC is one of the most popular since it expresses both homogeneity and separation. MSSC consists in partitioning the set $\mathcal{X} := \{x_1, x_2, ..., x_n\}$ into c clusters in order to minimize the sum of squared distances from the entities to the centroid of their cluster. This problem may be formulated mathematically in several ways, which suggest different possible algorithms. The two most widely used models are a bilevel programming problem and a mixed integer program. We first consider the mixed integer formulation of MSSC and equivalently reformulate it as a DC program thanks to the exact penalty technique. Secondly, we introduce a Gaussian Kernel version of the bilevel programming formulation of MSSC (named GKMSSC) and then reformulate GKMSSC as a DC program. We develop two DCA based algorithms to solve the two resulting optimization problems. It fortunately turns out that the corresponding DCA consists in computing, at each iteration, the projection of points onto a simplex and/or a ball, that all are given in the explicit form. For initializing DCA, a local search procedure VNS (Variable Neighborhood Search) is proposed.

In Chapter 8, we address the MSSC model using weighted dissimilarity measure (each feature is assigned a continuous value in the interval $[0, 1]$ which represents the relevant degree of the feature). We consider the bilevel programming and mixed integer formulations of MSSC with weighted dissimilarity measures. Both problems can be reformulated as DC programs for which efficient DCA algorithms are developed.

The Chapter 9 deals with the so-called block clustering that consists in simultaneous clustering on the set of samples (objects) and on the set of their features in order to find homogeneous blocks. Based on exact penalty technique, the block clustering problem is recast as a DC program in its elegant matrix formulation. A very nice DC decomposition is proposed of which the corresponding DCA is simple, elegant and inexpensive : it consists in computing, at each iteration, the projection of points onto a simplex and/or a box, that all are given in the explicit form.

In Chapter 10, we consider Gaussian Mixture Model (GMM) clusetering, the most popular model-based clustering model in the literature. We address three fundamental issues in GMM clustering : the choice of appropriate number of clusters, the over-parameterization and the selection of useful features. Although all these three issues are linked together, most of existing works deal with them separately. For the first time, we present an unified optimization formulation that takes into account all three above-mentioned issues. It turns out that the corresponding optimization problem involves the minimization of ℓ_0-norm. To tackle the resulting problem, we develop DCA-Like and then a Two-Step DCA-Like to further improve the performance of DCA-Like.

In Chapter 11, we deal with the time-series data clustering. We tackle three crucial issues in high-dimensional time-series data clustering for pattern discovery : appropriate similarity measures, efficient procedures for high-dimensional setting, and fast/scalable clustering algorithms. Our approach incorporates several advanced techniques : t-distributed stochastic neighbor embedding (t-SNE) method to transform the high-dimensional time-series data into a lower dimensional space, efficient algorithm for finding the number of clusters (DCA-Modularity), and fast and scalable clustering algorithm (DCA-MSSC). The innovative character intervenes in all stages of the proposed approach : the data transformation via t-SNE with the DTW measure in the original data space and the Euclidean distance in the transformed space is original. As application, we applied our approach for customer clustering of French transmission system operator (RTE) based on their electricity consumption.

Chapitre 7

Minimum Sum-of-Squares Clustering[1]

Abstract: We consider the two most widely used models for the so called Minimum Sum-of-Squares Clustering (MSSC in short) that are a bilevel programming problem and a mixed integer program. Firstly, the mixed integer formulation of MSSC is carefully studied and is reformulated as a continuous optimization problem via the exact penalty technique in DC programming. DCA is then investigated to the resulting problem. Secondly, we introduce a Gaussian Kernel version of the bilevel programming formulation of MSSC, named GKMSSC. The GKMSSC problem is formulated as a DC program for which a simple and efficient DCA scheme is developed. A regularization technique is investigated for exploiting the nice effect of DC decomposition and a simple procedure for finding good starting points of DCA is developed. The proposed DCA schemes are original and very inexpensive because they amount to computing, at each iteration, the projection of points onto a simplex and/or onto a ball, and/or onto a box, that are all determined in the explicit form. Numerical results on real word datasets show the efficiency, the scalability of DCA and its great superiority with respect to k-means and Kernel k-means, standard methods for clustering.

7.1 Introduction

An instance of the partitional clustering problem consists of a data set $\mathcal{X} := \{x_1, x_2, ..., x_n\}$ of n entities in $\mathbb{R}^d$, a measured distance, and an integer c $(2 \leq c \leq n)$; we are to choose c members v_i $(i = 1, ...c)$ and assign each member of $\mathcal{X}$ to its closest centroid (or center).

Let $U = (u_{ik}) \in \mathbb{R}^{c\times n}$ with $i = 1, \ldots, c$ and $k = 1, \ldots, n$ be the matrix defined by :

$$u_{ik} := \begin{cases} 1 & \text{if} \quad x_k \in C_i \\ 0 & \text{otherwise.} \end{cases}$$

Then a straightforward mixed integer formulation of MSSC is

$$\text{(IP-MSSC)} \qquad \begin{cases} \min F(U,V) := \sum\limits_{k=1}^{n}\sum\limits_{i=1}^{c} u_{ik}||x_k - v_i||^2 \\ s.t. \quad \sum\limits_{i=1}^{c} u_{ik} = 1, \ k = 1, .., n \\ \qquad u_{ik} \in \{0,1\} \ \ i = 1, .., c, \ k = 1, .., n \end{cases} \tag{7.1}$$

1. The results presented in this chapter were published in :
   - H.A. Le Thi, H.M. Le, T. Pham Dinh, New and efficient DCA based algorithms for Minimum Sum-of-Squares, Pattern Recognition, 47(1) :388-40, 2014.
   - H.M. Le, H.A. Le Thi, T. Pham Dinh, Gaussian Kernel Minimum Sum-of-Squares Clustering and solution method based on DCA, Lecture Notes in Artificial Intelligence (LNAI) 719, 331-340, 2012.

where $\|.\|$ is, in this chapter, the Euclidean norm, and V the $(c \times d)$ - matrix whose ith row is $v_i \in \mathbb{R}^d$, the center of cluster C_i. In the sequel to simplify related computations we will work on the vector space $\mathbb{R}^{c\times n} \times \mathbb{R}^{c\times d}$ of real matrices. The variables are then $(U, V) \in \mathbb{R}^{c\times n} \times \mathbb{R}^{c\times d}$, where $U \in \mathbb{R}^{c\times n}$ whose k^{th} column is denoted U^k and $V \in \mathbb{R}^{c\times d}$ whose i^{th} row is denoted V_i or v_i (v_i is a row-vector in $\mathbb{R}^d$).

The last constraint of (7.1) ensures that each point x_k is assigned to one and only one group. It means that the intersection of any pairwise clusters is empty, hence we are faced with *hard clustering*. Problem (7.1) is a mixed integer program where the objective function is nonconvex. It has been shown in [9] that the problem is NP-hard with possibly many local minima. Several works in optimization approaches have been developed for the MSSC with the mixed integer programming formulation. Exact solution methods are difficult but a variety of approaches have been explored, among them the branch and bound or cutting methods via reformulation-linearization techniques [215, 195], 0-1 SDP formulation [37, 187], interior point method [65]. However, exact solution methods are not available for massive data sets in real applications (*high* dimension and *large* data set, i.e. d and n are very large). Most of the methods for the model (7.1) are heuristics that can only locate a "good" local solution.

Another formulation of MSSC is based on bilevel programming and was first introduced by Vinod [228] :

$$\text{(B-MSSC)} \quad \min \quad \left\{ \sum_{k=1}^{n} \min_{i=1,\ldots,c} \|x_k - v_i\|^2 : v_i \in \mathbb{R}^d, i = 1, \ldots, c \right\}. \tag{7.2}$$

This is a nonconvex nonsmooth optimization problem and is very hard to solve. While several heuristic methods based on k-means algorithm have been proposed for solving (7.2), there are few deterministic approaches that address directly this bilevel problem. Two efficient works in this direction can be found in Le Thi et al. [123] that is based on DC programming and DCA and in Xavier et al. [241] which uses a hyperbolic smoothing technique.

Our contributions. We develop *efficient* and *scalable* DCA based algorithms for both formulations of MSSC : the mixed integer programming problem (7.1) and the bilevel programming problem (7.2).

Firstly, we propose an efficient continuous approach based on DCA for solving the mixed integer programming problem (7.1). Using an exact penalty result developed in [145] we equivalently reformulate the combinatorial problem (7.1) in term of a continuous optimization problem which is in fact a DC program. We then investigate a DCA scheme for solving the resulting problem. From the construction of this algorithm we are able to interpret *why DCA is better than k-means algorithm.* It is worthy to note that minimizing a nonconvex functions under $0 - 1$ variables is a very hard problem and there is no continuous approach in the literature for solving the combinatorial problem (7.1). The preliminary version of this method has been published in the conference paper [143] where the authors presented a DC reformulation and its corresponding DCA for solving (7.1) as well as some preliminary numerical experiments. In this chapter we carefully explore and exploit this approach from both a theoretical and an algorithmic point of view. Related key questions are investigated : the choices of DC decomposition, some convexification techniques to construct appropriate DC components, a procedure for finding good starting points for DCA. Numerical experiments on several test problems and comparative results between different approaches are reported.

Secondly, we introduce a Gaussian kernel version of (7.2), called GKMSSC, and investigate DCA for solving it. Kernel clustering methods are known to be efficient in the situations where

the sets to be discriminated in the original space (for example, a data set composed of two rings of points) can not be separabled by standard partitioning methods using centroids (k-means, Fuzzy C-Means...). The basic idea is to implicitly transform data to a higher dimensional space and to find a classifier without having to perform any computations in the high dimensional space. Among several kernel functions we have chosen voluntarily in this chapter the Gaussian kernel. This choice is very favorable to the developement of DCA for the corresponding GKMSSC problem : we get a DCA scheme very simple and inexpensive in term of time consuming. In addition, exploiting the effect of DC decompositions, we use a regularization technique for the GKMSSC problem which allows DCA to well separate the clustering centers in the projection space.

For both (IP-MSSC) and (GKMSSC), the two proposed DCA schemes are simple, fast and scalable. We also investigate a Variable Neighborhood Search (VNS) algorithm to find good starting points for DCA.

The remainder of the chapter is organized as follows.The solution of the mixed integer programming problem (7.1) by DCA is developed in Section 7.2 while Section 7.3 deals with DCA for solving the Gaussian kernel MSSC problem. The procedure of finding a good starting point by VNS algorithm is discussed in Section 7.4 and the computational results are reported in Section 7.5.

7.2 DCA for solving the mixed integer (IP-MSSC) problem

For applying DCA we reformulate (IP-MSSC) as a continuous optimization problem. First of all, since $u_{ik} \in \{0,1\}$ we can replace u_{ik} by u_{ik}^2 and rewrite the objective function of (IP-MSSC) by

$$F(U,V) := \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^2 ||x_k - v_i||^2.$$

We will see on the next that using u_{ik}^2 instead to u_{ik} is useful for getting a good DC decomposition and the resulting DCA is interesting.

In the problem (IP-MSSC) the variable U is a priori bounded in $\mathbb{R}^{c\times n}$. One can also find a constraint for bound the variable V by determining a ball of radius r and center $0 \in \mathbb{R}^d$ containing necessarily the optimum centers v_i. Indeed, let Δ be the $(c-1)$-simplex in $\mathbb{R}^c$ defined by

$$\Delta = \{y \in [0,1]^c : \sum_{i=1}^{c} y_i = 1\}$$

and Δ^n be the Cartesian product of n simplices Δ. Hence, $U^k \in \Delta \cap \{0,1\}^c, \forall k = 1,\ldots,n$ and $U \in \Delta^n \cap \{0,1\}^{c\times n}$.

The problem (IP-MSSC) can be rewritten as :

$$\min_{U\in\Delta^n\cap\{0,1\}^{c\times n}} \min_{V\in\mathbb{R}^{c\times d}} F(U,V). \tag{7.3}$$

For each fixed $U \in \Delta^n \cap \{0,1\}^{c\times n}$, the necessary first order optimality condition for the problem $\min_{V\in\mathbb{R}^{c\times d}} F(U,V)$ implies that $\nabla_V F(U,V) = 0$, i.e.,

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

or

$$v_i \sum_{k=1}^{n} u_{ik} = \sum_{k=1}^{n} u_{ik} x_k.$$

On the other hand, the condition that cluster should not be empty imposes that $\sum_{k=1}^{n} u_{ik} > 0$ for $i = 1, ..., c$. Hence

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

Let R_i $(i = 1, ..., c)$ be the Euclidean ball centered at the origin and of radius r in $\mathbb{R}^d$, and let $\mathcal{C} := \Pi_{i=1}^{c} R_i$. We can rewritte the problem (IP-MSSC) as :

$$\min \left\{ F(U,V) : \ (U,V) \in \Delta^n \cap \{0,1\}^{c\times n} \times \ \mathcal{C} \ \right\}. \tag{7.4}$$

7.2.1 A continuous reformulation

Our reformulation technique is based on the following results developed in [145].

Theorem 7.1 ([145]) *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 :*

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

For applying this result in the reformulation of (IP-MSSC) we first show that $F(U,V)$ is a DC function. Using the equation $2g_1g_2 = (g_1+g_2)^2 - (g_1^2+g_2^2)$ we can express $F(U,V)$ as

$$F(U,V) \ = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{c} \left[\left(u_{ik}^2 + \|x_k - v_i\|^2\right)^2 - \left(u_{ik}^4 + \|x_k - v_i\|^4\right) \right].$$

Hence the following DC decomposition of $F(U,V)$ seems to be natural :

$$F(U,V) := G_0(U,V) - H_0(U,V) \tag{7.5}$$

where

$$G_0(U,V) = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{c} \left(u_{ik}^2 + \|x_k - v_i\|^2\right)^2 \tag{7.6}$$

and

$$H_0(U,V) = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{c} \left(u_{ik}^4 + \|x_k - v_i\|^4\right) \tag{7.7}$$

are clearly convex functions.

Let T_i be a box containing the ball R_i $(i = 1, ..., c)$ and let $\mathcal{T} := \Pi_{i=1}^{c} T_i$. Hence $\mathcal{C} \subset \mathcal{T}$. Consider the function p defined on $\mathbb{R}^{c\times n}$ by

$$p(U) := \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}(1 - u_{ik}).$$

Clearly, p is finite concave on $\mathbb{R}^{c\times n}$, nonnegative on Δ^n, and

$$\Delta^n \cap \{0,1\}^{c\times n} = \{U \in \Delta^n :\ p(U) = 0\} = \{U \in \Delta^n :\ p(U) \leq 0\}.$$

Using the above theorem we can now write (7.3) in the form of the following nonconvex program in continuous variables :

$$\min\left\{F_1(U,V) := F(U,V) + tp(U) : (U,V) \in \Delta^n \times \mathcal{T}\right\}, \tag{7.8}$$

where $t > t_0$ is called penalty parameter. Since $\mathcal{C} \subset \mathcal{T}$ and $\mathcal{C}$ containing necessarily the optimum centers v_i,the following problem is equivalent to (7.8) :

$$\min\left\{F_1(U,V) := F(U,V) + tp(U) : (U,V) \in \Delta^n \times \mathcal{C}\right\}. \tag{7.9}$$

In the sequel we will consider the MSSC problem via the continuous formulation (7.9) with t being sufficiently large. We will develop DC programming and DCA for solving (7.9).

7.2.2 A DC formulation for the resulting MSSC problem (7.9)

We first remark that, if F is a DC function with DC components G_0 and H_0 then the function $F_1(U,V) := F(U,V) + tp(U)$ is DC too with DC components G_0 and $H_0 - tp$ (remember that p is concave function). Hence, the natural DC decomposition (7.5) of F involves the next DC decomposition of F_1 :

$$F_1(U,V) \ := G_0(U,V) - [H_0(U,V) - tp(U)]. \tag{7.10}$$

However, from numerical point of views, the DCA scheme corresponding to this DC decomposition is not interesting because it requires an iterative algorithm for solving a convex program at each iteration. We will propose another DC decomposition of $F_1(U,V)$ for which the resulting DCA is explicitly determined via a very simple formula. We rewrite $F_1(U,V)$ as follows

$$\begin{array}{rl} F_1(U,V) & = \left[\frac{\rho}{2}\|(U,V)\|^2\right] - \left[\frac{\rho}{2}\|(U,V)\|^2 - F_1(U,V)\right] \\ & \left[\frac{\rho}{2}\|(U,V)\|^2\right] - \left[\frac{\rho}{2}\|(U,V)\|^2 - F(U,V) - tp(U)\right] \end{array} \tag{7.11}$$

Since $-tp(U)$ is convex, it suffices to compute ρ such that $H(U,V) := \frac{\rho}{2}\|(U,V)\|^2 - F(U,V)$ is convex to get a DC decomposition of $F_1(U,V)$. We have

$$\begin{array}{rcl} H(U,V) & = & \frac{\rho}{2}\|(U,V)\|^2 - \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^2 ||x_k - v_i||^2 \\ & = & \sum_{k=1}^{n}\sum_{i=1}^{c} \frac{\rho}{2}\ u_{ik}^2 + \frac{\rho}{2n}\|x_k - v_i\|^2\ - u_{ik}^2\|x_k - v_i\|^2 + \frac{\rho}{n}\langle x_k, v_i\rangle - \frac{\rho}{2n}\|x_k\|^2. \end{array}$$

Let us consider the function $\theta_{i,k}: \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ defined by

$$\theta_{i,k}(u_{ik},v_i) = \tfrac{\rho}{2}\,u_{ik}^2 + \tfrac{\rho}{2n}\|x_k-v_i\|^2 - u_{ik}^2\|x_k-v_i\|^2\ . \tag{7.12}$$

The Hessian of $\theta_{i,k}$ is given by :

$$H_{\theta_{i,k}}(u_{ik},v_i) = \begin{pmatrix} \rho-2\|x_k-v_i\|^2 & -4u_{ik}\|x_k-v_i\|^2 \\ -4u_{ik}\|x_k-v_i\|^2 & \frac{\rho}{n}-2u_{ik}^2 \end{pmatrix}. \tag{7.13}$$

The function $\theta_{i,k}(u_{ik},v_i)$ is convex on $\{0\le u_{ik}\le 1, \|v_i\|\le r\}$ if the determinant of $H_{\theta_{i,k}}(u_{ik},v_i)$ is positive and $\rho-2\|x_k-v_i\|^2>0, \frac{\rho}{n}-2u_{ik}^2>0$ for all $0\le u_{ik}\le 1, \|v_i\|\le r$. Hence, we deduce that for all

$$\rho \ \ge\ n\left[\tfrac{1}{n}\,\xi^2+1+\sqrt{\left[\tfrac{1}{n}\,\xi^2+1\right]^2+\tfrac{12}{n}\xi^2}\right],\ \text{with } \xi = r+\max_{1\le k\le n}\|x_k\|\,, \tag{7.14}$$

$\theta_{i,k}$ is convex on $\{0\le u_{ik}\le 1, \|v_i\|\le r\}$.
As a consequence, the function $h_{i,k}$ defined by

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

is convex. Finally, since

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

the function $H(U,V)$ is convex on $\Delta^n\times\mathcal{C}$ with ρ satisfying (7.14). ■

Assume in the sequel that the function H is defined with a ρ satisfying the condition (7.14). We can express our second DC decomposition of F_1 as follows :

$$F_1(U,V) := G_1(U,V) - H_1(U,V) \tag{7.15}$$

with

$$G_1(U,V) := \frac{\rho}{2}\,\|(U,V)\|^2\,,\ H_1(U,V) := H(U,V) - tp(U)$$

being clearly convex functions. Now, the optimization problem (7.4) can be written as

$$\min\left\{\chi_{\Delta^n\times\mathcal{C}}(U,V) + \frac{\rho}{2}\,\|(U,V)\|^2 - H_1(U,V) : (U,V)\in(U,V)\in\mathbb{R}^{c\times n}\times\mathbb{R}^{c\times d}\right\}. \tag{7.16}$$

7.2.3 DCA applied to (7.16)

The function H_1 is differentiable and its gradient at the point (U^l,V^l) is given by :

$$(Y^l,Z^l) = \nabla H_1(U^l,V^l) \quad \text{where}$$

$$\begin{aligned} Y_{k,i}^l &= \left(\left(\rho u_{ik}^l - 2u_{ik}^l\right)\|x_k-V_i^l\|^2 + 2tu_{ik}^l - t\right)_{i=1,\dots c}^{k=1,\dots n}, \\ Z_i^l &= \left(\rho V_i^l - 2\sum_{k=1}^{n}(V_i^l-x_k)(u_{ik}^l)^2\right)_{i=1,\dots c}. \end{aligned} \tag{7.17}$$

The projection of (U, V) on $\Delta^n \times \mathcal{C}$ can be separably computed as

$$\left(U^{l+1}\right)^k = \text{Proj}_\Delta\left(\left(Y^l\right)^k\right) \; k = 1, ...n, \quad V_i^{l+1} = \text{Pr oj}_{R_i}\left(\tfrac{1}{\rho}(Z^l)_i\right) \; i = 1, ...c. \tag{7.18}$$

For computing the projection of points onto a simplex Δ, some efficient algorithms are available among them we use the very inexpensive algorithm developed in [116]. The projection of points onto a ball is explicit. The algorithm can be described as follows.

Algorithm 7.1 IP-DCA : DCAP2 applied to (7.16)

1: **Initialization :** Choose the memberships U^0 and the cluster centers V^0. $l \leftarrow 0$.
2: **repeat**
3: Compute Y^l and Z^l via (7.17).
4: Define $(U^{(l+1)}, V^{(l+1)})$ by setting :

$$\begin{aligned} \left(U^{(l+1)}\right)^k &= \text{Proj}_\Delta\left(\left(Y^l\right)^k\right) \text{ for } k = 1, ...n, \\ V_i^{(l+1)} &= \text{Pr oj}_{R_i}\left(\frac{1}{\rho}(Z^l)_i\right) = \begin{cases} \frac{(Z^l)_{i,\cdot}}{\rho} \text{ if } \|(Z^l)_{i,\cdot}\| \le \rho r \\ \frac{(Z^l)_{i.}r}{\|(Z^l)_i\|} \text{ otherwise} \end{cases}. \\ (i &= 1, .., c) \end{aligned}$$

5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

Remark 7.1 *a) With the DC decomposition (7.15) the resulting DCA is simple :each iteration of DCA consists of computations of the projection of points onto a simplex ([116]) and/or onto a ball, that all are explicitly computed. So DCA do not require an iterative method at each iteration as in the DCA scheme applied to the first DC decomposition (7.10).*

b) Theoretically, (7.14) *gives a sufficient condition to obtain a value ρ_0 such that the functions $\frac{1}{2}\rho\|(U, V)\|^2 - F(U, V)$ is convex for all $\rho \ge \rho_0$. But this is not a necessary condition, that means, when $0 < \rho < \rho_0$ the function $\frac{1}{2}\rho\|(U, V)\|^2 - F(U, V)$ may be still convex. In such a case the DC decompostion (7.15) is still valid.*

c) On a numerical point of view, from the description of DCA we observe that, among the values ρ such that $\frac{1}{2}\rho\|(U, V)\|^2 - F(U, V)$ is convex, the smaller ρ is, the less important the concave part $-h(x) := -\frac{1}{2}\rho\|(U, V)\|^2 - F(U, V)$ will be, and then the more efficient IP-DCA is. Hence it is interesting to find a tight lower bound of ρ and choose ρ as small as possible. In practice, based on the "descent" property of DCA, such values can be determined via a dynamic computation of ρ during DCA's steps as shown in the procedure "Update ρ" below.

For finding a good value of ρ we star DCA with ρ_0 computed by (7.14), and then, at each iteration l, reduce ρ while the sequence $\{f(x^l)\}$ decreases. The procedure can be described as follows.

Procedure Update ρ : let x^0 be given in $\mathbb{R}^d$ and let τ be a number such that $0 < \tau < 1$. Set $l := 0$, STOP :=false

While not STOP **do**

— Set $\rho := \tau\rho_l$.

— Calculate $x^{l+1} = Proj_C((\rho x^l - \nabla f(x^l)/\rho)$.

— If $f(x^{l+1}) > f(x^l)$ then $STOP := true$, else set $l \leftarrow l+1$, $\rho_l \leftarrow \rho$

Ouput : set $\rho := \rho_l$, $x^0 := x^l$.
We then continue to apply IP-DCA with this value of ρ and from the point x^0.
Note that we can again refine this procedure by modifying τ at each iteration.

7.2.4 An interpretation of DCA : why DCA is better than k-means ?

In the MSSC formulation (7.1) the variable U corresponds to the affectation of objects to clusters while the variable V stands for centers of clusters. The computation of U^{l+1} at iteration l of DCA can be interpreted as the affectation of objects to clusters with centers V_i^l, $i = 1, ..., c$ (that are defined at iteration $l-1$) while the calculation of V^{l+1} is nothing but the determination of the new center of clusters. There are actually some similarities between DCA and the k-means algorithm. However DCA enjoys the main advantage that might explain why DCA is better than k-means : in DCA U^{l+1} and V^{l+1} are separately but simultaneously computed in the way that U^{l+1} as well as V^{l+1} depend on both U^l and V^l, while k-means determines them alternatively and V^{l+1} depends on U^{l+1}. In other words DCA determines at the same time the clusters and their centers.

7.3 Solving a Kernel version of MSSC problems by DCA

We introduce a Gaussian Kernel version of bilevel formulation of MSSC (7.2), called (GKMSSC), and develop a new DCA based algorithm for solving it.

7.3.1 Kernel Minimum Sum-of-Square Clustering

Let $\phi : \mathbb{R}^m \longrightarrow \mathcal{H}$ be a transformation that maps each data x_i from the input space $\mathbb{R}^m$ to a new space $\mathcal{H}$, being a Hilbert space, where the given algorithm can be used. Such transformation is done implicitly by means of a kernel function [100] $\mathcal{K}$, satisfying :

$$\mathcal{K}(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle.$$

Mercer's theorem guarantees that as long as the corresponding matrix $\mathcal{K}$ of kernel function is positive definite, the algorithm implicitly operates in a higher dimensional space. This kernel trick saves the algorithm from the computational expense of explicitly representing all of the features in a higher-dimensional space.

With a kernel function ϕ the objective function of (7.2) becomes

$$f_K(V) := \sum_{k=1}^{n} \min_{i=1,...c} \|\phi(x_k) - \phi(v_i)\|^2 . \tag{7.19}$$

Since

$$\begin{aligned} \|\phi(x_k) - \phi(v_i)\|^2 &= \langle \phi(x_k), \phi(x_k) \rangle - 2\langle \phi(x_k), \phi(v_i) \rangle + \langle \phi(v_i), \phi(v_i) \rangle \\ &= \mathcal{K}(x_k, x_k) - 2\mathcal{K}(x_k, v_i) + \mathcal{K}(v_i, v_i), \end{aligned} \tag{7.20}$$

the kernel version of (MSSC) takes the form

$$\text{(KMSSC)} \quad \left\{ \min_V \left\{ f_K(V) := \sum_{k=1}^{n} \min_{i=1,...c} \Big(\mathcal{K}(x_k, x_k) - 2\mathcal{K}(x_k, v_i) + \mathcal{K}(v_i, v_i) \Big) \right\} \right.$$

This is named "Kernel Minimum Sum-of-Square Clustering" problem or KMSSC in short.

Using a given kernel function we will get its corresponding KMSSC model. Obviously, from numerical points of view, the degree of difficulty of the KMSSC problem varies from each to other model. Among several kernel functions we consider in this chapter the Gaussian kernel. We will see in the next that this choice is interesting : we can investigate simple and efficient DCA based algorithm for solving the corresponding problem which is so named GKMSSC .

By replacing the Gaussian kernel function $k(x,y) := \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$ in the objective function of (KMSSC) we have :

$$\begin{aligned} f_{GK}(V) &= \sum_{k=1}^{n} \min_{i=1,\dots c} \Big[\exp\Big(-\tfrac{\|x_k-x_k\|^2}{2\sigma^2}\Big) - 2\exp\Big(-\tfrac{\|x_k-v_i\|^2}{2\sigma^2}\Big) \\ &\qquad + \exp\Big(-\tfrac{\|v_i-v_i\|^2}{2\sigma^2}\Big)\Big] \\ &= \sum_{k=1}^{n} \min_{i=1,\dots c} \Big[2 - 2\exp\Big(-\tfrac{\|x_k-v_i\|^2}{2\sigma^2}\Big)\Big]. \end{aligned}$$

Then the optimization model of the Gaussian kernel MSSC takes the form

$$\text{(GKMSSC)} \quad \min \quad \left\{ F_{GK}(V) := \sum_{k=1}^{n} \min_{i=1,\dots c} \left[-2\exp\left(-\frac{\|x_k - v_i\|^2}{2\sigma^2}\right)\right]\right\}. \tag{7.21}$$

We will show in the next section how to investigate DCA for solving the GKMSSC problem.

7.3.2 DC formulation of the GKMSSC problem

First of all, by bounding the variable V in the box $\mathcal{T} := \Pi_{i=1}^{c}\mathcal{T}_i$ with $\mathcal{T}_i := \Pi_{j=1}^{d}[\zeta_j, \beta_j]$ and $\zeta_j := \min_{k=1,\dots,n} x_{k,j}$, $\beta_j := \max_{k=1,\dots,n} x_{k,j}$, we can express the unconstrained optimization problem (KMSSC) as a constrained optimization problem whose feasible set is $\mathcal{T}$.

Our DC decomposition of $F_{GK}(V)$ is based on the similar DC decomposition in (7.11). Define

$$f_{ki}(V) = -2\exp\left(-\frac{\|x_k-v_i\|^2}{2\sigma^2}\right). \tag{7.22}$$

Denote by $H_f(x)$ the Hessian matrix of f at x and $\lambda_n(H_f(x))$ the largest eigenvalue of the $H_f(x)$. We have, for all $\rho \geq \frac{2}{\sigma^2} \geq \lambda_n(H_{f_{ki}}(V)) = \frac{2}{\sigma^2}\exp\left(-\frac{\|x_k-v_i\|^2}{2\sigma^2}\right)$, the function $\frac{\rho}{2}\|v_i\|^2 - f_{ki}(V)$ is convex on $\mathcal{T}_i$. Then with $\rho \geq \frac{2}{\sigma^2}$, we get the following DC decomposition of $f_{ki}(V)$:

$$f_{ki}(V) := g_{ki}(V) - h_{ki}(V), \quad g_{ki}(V) = \tfrac{\rho}{2}\|v_i\|^2, h_{ki}(V) = \tfrac{\rho}{2}\|v_i\|^2 - f_{ki}(V). \tag{7.23}$$

It follows that, for $k = 1 \dots n$,

$$\begin{aligned} f_k(V) &:= \min_{i=1..c} f_{ki}(V) = \min_{i=1..c} g_{ki}(V) - h_{ki}(V) \\ &= \min_{i=1..c} \left(\textstyle\sum_{l=1}^{c} g_{kl}(V) - \sum_{l=1..c, l\neq i} g_{kl}(V)\right) - h_{ki}(V) \\ &= \textstyle\sum_{l=1}^{c} g_{kl}(V) - \max_{i=1..c} \left(\sum_{l=1..c, l\neq i} g_{kl}(V) + h_{ki}(V)\right) \\ &= \tfrac{\rho}{2}\sum_{l=1}^{c}\|v_l\|^2 - \max_{i=1..c}\left(\sum_{l=1..c, l\neq i} \tfrac{\rho}{2}\|v_l\|^2 + \tfrac{\rho}{2}\|v_i\|^2 - f_{ki}(V)\right) \\ &= \tfrac{\rho}{2}\sum_{i=1}^{c}\|v_i\|^2 - \max_{i=1..c}\Big(\tfrac{\rho}{2}\sum_{i=1}^{c}\|v_i\|^2 - f_{ki}(V)\Big). \end{aligned} \tag{7.24}$$

Consequently, the objective function of (GKMSSC) can be now written as

$$F_G(V) = \sum_{k=1}^{n} f_k(V) = G_2(V) - H_2(V), \tag{7.25}$$

where $G_2(V) := \frac{n\rho}{2} \sum_{i=1}^{c} \|v_i\|^2$ and $H_2(V) := \sum_{k=1}^{n} \max_{i=1..c} \left(\frac{\rho}{2} \sum_{i=1}^{c} \|v_i\|^2 - f_{ki}(V) \right)$ are convex functions. Hence, we can recast Problem (GKMSSC) as a DC program

$$\min \{G_2(V) - H_2(V) : V \in \mathcal{T}\}. \tag{7.26}$$

7.3.3 DCA applied to DC program (7.26)

The DCA scheme applied to (7.26) amounts to computing the two sequences $\{V^l\}$ and $\{W^l\}$ in $\mathbb{R}^{c\times d}$ such that

$$W^l \in \partial \left(H_2(V^l) \right) \tag{7.27}$$

and

$$V^{(l+1)} \in \arg\min \left\{ \frac{n\rho}{2} \sum_{i=1}^{c} \|V_i\|^2 - \langle W^l, V \rangle : V \in \mathcal{T} \right\} \tag{7.28}$$

or again

$$(V_i)^{(l+1)} = \mathrm{Proj}_{\mathcal{T}_i} \left((w_i)^{(l)}/(n\rho) \right), \text{for } i = 1..c. \tag{7.29}$$

Note that the projection of points onto a box is explicitly computed.

A gradient of $H_2(X)$ is computed as follows :

$$W \in \partial H_2(V) \Leftrightarrow W = \sum_{k=1}^{n} \partial h_k(V), \tag{7.30}$$

where $h_k(V) := \max_{i=1..c} h_{ki}(V)$ and $h_{ki}(V) = \frac{\rho}{2} \sum_{i=1}^{c} \|v_i\|^2 - f_{ki}(V)$.

Let $I_k(V) := \{i = 1, \ldots, c : h_{ki}(V) = h_k(V)\}$. We have :

$$\partial h_k(V) = co \left\{ \cup_{i\in I_k(V)} \partial h_{ki}(V) \right\},$$

where co stands for the convex hull. Hence $\partial h_k(V)$ is a convex combination of $\{\nabla(h_{ki})(V) : i \in I_k(V)\}$, i.e.,

$$\partial h_k(V) = \sum_{i\in I_k(V)} \lambda_i^{[j]} \nabla(h_{ki})(V) \text{ with } \lambda_i^{[k]} \geq 0 \text{ for } i \in I_k(V) \text{ and } \sum_{j\in I_k(V)} \lambda_i^{[k]} = 1. \tag{7.31}$$

The gradient of $h_{ki}(V)$ is computed as (for $q = 1, ..., c$)

$$[\nabla(h_{ki})(V)]_q = \rho v_i - \frac{2}{\sigma^2}. \exp\left(-\frac{\|x_k - v_i\|^2}{2\sigma^2} \right) (v_i - x_k) \text{ if } q = i, 0 \text{ otherwise.} \tag{7.32}$$

Finally, $W \in \partial H_2(V)$ is determined via the formulas (7.30), (7.31) and (7.32). From the above computations, the DCA applied to problem (7.26) can be described as follows :

Algorithm 7.2 GK-DCA : DCA applied to (7.26)

Initialization : Let V^0 be given. Set $l \leftarrow 0$.
repeat
  Compute $W^l \in \partial H_2(V^l)$ via the formulas (7.30), (7.31) and (7.32).
  Compute $x^{k+1} = arg\min_{x \in X} \left\{ G(x) - \langle y^k, x \rangle \right\}$.
  Compute $V^{(l+1)}$ via (7.29).
  $k \leftarrow l + 1$.
until Stopping criterion.

7.3.4 A regularized version of GK-DCA

Since we work on the projection space, it can happen, due the computation of V_i^l in *GK-DCA*, that some V_i^l are coincide. It imply that there are some empty clusters. To avoid this situation we propose a regularization technique by adding the convex term $\frac{\tau}{2}\|V\|^2$ in each DC component $G_2(V)$ and $H_2(V)$ to get the following DC program (τ is a positive number) :

$$F_{GK}(V) = \left(G_2(V) + \frac{\tau}{2}\|V\|^2 \right) - \left(H_2(V) + \frac{\tau}{2}\|V\|^2 \right), \tag{7.33}$$

DCA applied to the regularized problem (7.33), called **GK-RDCA** is nothing else *GK-DCA* with a little modification : in (7.30) W is replaced by $W + \tau V$, and in (7.28) and (7.29) $n\rho$ is replaced by $n\rho + \tau$.

7.4 VNS for initializing DCA

Finding a good starting point is a challenge in designing DCA schemes for DC programs. The search of such a point depends on the structure of the problem being considered. Generally, a good starting point for DCA must not be a local minimizer, because DCA is stationary from such a point. For this problem, we use a simple and effective metaheuristic (or framework for heuristics) called Variable Neighborhood Search (VNS) [167].

The principle of VNS is to change and randomly explore neighborhoods with an appropriate local search routine. Contrary to other metaheuristics, e.g., simulated annealing or Tabu search , VNS does not follow a trajectory but explores increasingly distant neighborhoods of the current incumbent solution, and jumps from there to a new one if and only if an improvement has been made, through a local search.

Let us denote a finite set of pre-selected neighborhood structures with $\mathcal{N}_l (l = 1, \ldots, l_{max})$ and with $\mathcal{N}_l(x)$ (and preferably such as $\mathcal{N}_l \subset \mathcal{N}_{l+1}$) the set of solutions in the l^{th} neighborhood of x.

The basic VNS heuristic, applied to the problem $\min\{f(x) : x \in S\}$, is summarized in Algorithm 7.3.

Note that (VNS) uses only one parameter l_{max} which can often be disposed of, e.g., by setting it equal to the size of the vector x considered.

For all the problems considered the neighborhood structure $\mathcal{N}_l(x)$ is defined by the Hamming distance γ between solutions X and x' (i.e., the number of components in which these vectors differ) :

$$\gamma(x, x') = l \Longleftrightarrow x' \in \mathcal{N}_l(x).$$

The local search routine is two step of K-Means algorithm on the neighborhood $\mathcal{N}_l(x)$.

Algorithm 7.3 VNS

Initialization. Select a set of neighborhood structures $\mathcal{N}_l$, that will be used in the search ; choose an initial solution x.
repeat
  $l \leftarrow 1$
  repeat
    Shaking. Generate randomly a point x' in the neighborhood of x ($x' \in \mathcal{N}_l(x)$).
    Local search. Apply some local search method with x' as initial solution ; denote with x'' the obtained solution.
    Move or not. If x'' is better than x, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1(l \leftarrow 1)$; otherwise, set $l \leftarrow l + 1$.
  until $l = l_{max}$
until Stopping criterion.

7.5 Numerical experiments

Datasets. Numerical experiments were performed on several real world datasets taken from UCI Machine Learning Repository. The information about datasets is summarized in Table 7.1. We have implemented the algorithms in the V.S C++ v6.0 environment and performed the experiments on a Intel Duo Core 3.06GHz, with 4Go of RAM.

TABLE 7.1 – Datasets

Dataset	Points	Dimension	Number of classes
Pima	768	8	2
Yeast	1484	8	10
ADN	3186	60	3
Vote	435	16	2
Lympho	148	18	4
Waveform	5000	40	3
Papillon	23	4	4
Iris	150	4	3
Ionosphere	315	34	3
Wine	178	13	3
Breast	683	9	2
Statlog	4435	36	6
Glass	214	10	6
Comp	3891	10	3

Comparative algorithms. Three DCA based algorithms have been implemented : *IP-DCA*, *GK-RDCA* and the DCA applied on the bilevel formulation developed in [123] named *B-DCA*. For each DCA based algorithm, two variants have been considered that differ from one of other by the choice of initial point :

— **IP-DCA-Rand** / **GK-RDCA-Rand** / **B-DCA-Rand** : apply DCA from randomly chosen k centers in $\mathcal{T}$.

— **IP-DCA-VNS** / **GK-RDCA-VNS** / **B-DCA-Rand** : apply DCA after performing some iterations of the VNS algorithm from k centers randomly chosen.

Setting. In the Gaussian Kernel regularized algorithm *GK-RDCA* the regularized parameter τ is chosen as 0.1 for the first ten data and as 0.2 for others. The value of σ in the Gaussian kernel function is taken in the interval $[0.5, 10]$.

For computing ρ, we use the procedure Update ρ in *IP-DCA* while in **GK-RDCA** we take $\rho = \frac{2}{\sigma^2}$.

The two main criteria used for comparing the performances of algorithms are the percentage of well classified objects (PWCO), and the running time in seconds. For giving more informations about the quality of clustering algorithms we also report, in the last experiment, three other criteria that are the Rand index, the inertia interclass and intraclass inertia.

The Rand index (named by 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 :

$$RandI = \frac{a+d}{a+b+c+d}, \tag{7.34}$$

where

$$\begin{array}{rcl} a = & \mid \{i,j \mid I_{ref}(x_i) = I_{ref}(x_j) & \& \quad I_{class}(x_i) = I_{class}(x_j)\} \mid, \\ b = & \mid \{i,j \mid I_{ref}(x_i) = I_{ref}(x_j) & \& \quad I_{class}(x_i) \neq I_{class}(x_j)\} \mid, \\ c = & \mid \{i,j \mid I_{ref}(x_i) \neq I_{ref}(x_j) & \& \quad I_{class}(x_i) = I_{class}(x_j)\} \mid, \\ d = & \mid \{i,j \mid I_{ref}(x_i) \neq I_{ref}(x_j) & \& \quad I_{class}(x_i) \neq I_{class}(x_j)\} \mid . \end{array} \tag{7.35}$$

The intraclass inertia is a measure of how compact each cluster is when the number of cluster is fixed. The dispersion of classes from the center of gravity of the points is called the interclass inertia. They are defined as follows :

$$\begin{array}{rcl} I_{Intra} = & \frac{1}{n} \sum\limits_{j=1}^{c} \sum\limits_{x_i \in C_j} \|x_i - v_j\|^2, \\ I_{Inter} = & \frac{1}{n} \sum\limits_{j=1}^{c} n_j \|v_j - v^*\|^2, \end{array} \tag{7.36}$$

where n_j is the number of points in cluster C_j, $\bar{v_j}$ is the center of gravity of cluster C_j and v^* is the center of gravity of all points x_i.

Our experiments are composed of five parts.

Experiment 1

In the first experiment we are interested in the effectiveness of the first approach, say DCA applied to the integer nonconvex programming (IP-MSSC). For this purpose we compare the two DCA schemes applied to the two formulations of MSSC : the bilevel formulation (B-MSSC) and the mixed integer programming (IP-MSSC). In Figure 7.1, the best PWCO of *IP-DCA-VNS* and *B-DCA-VNS* over 10 executions are reported. We also report the CPU time of the execution that achieved the best PWCO. The mean and the standard deviation are given in Table 7.2.

We observe that the quality of two DCA based algorithms applied on the bilevel formulation and the integer programming formulation are comparable, and moderately *IP-DCA* is slightly better. *IP-DCA* gives better results, with an important gain of PWCO, on three datasets (Yeast 5.89%, Lympo 5.80%, Wave 9.7%) while *B-DCA* is better on one dataset (Stalog, with the gain 4.9%). For other datasets, the difference of PWCO given by the two algorithms is quite small, and it is less than 2% in all most cases. The average of PWCO of *IP-DCA* is 81.17 and that of *B-DCA* is 80.47 (cf. Table 7.2). Both algorithms are very fast and *B-DCA* is somewhat faster.

Experiment 2

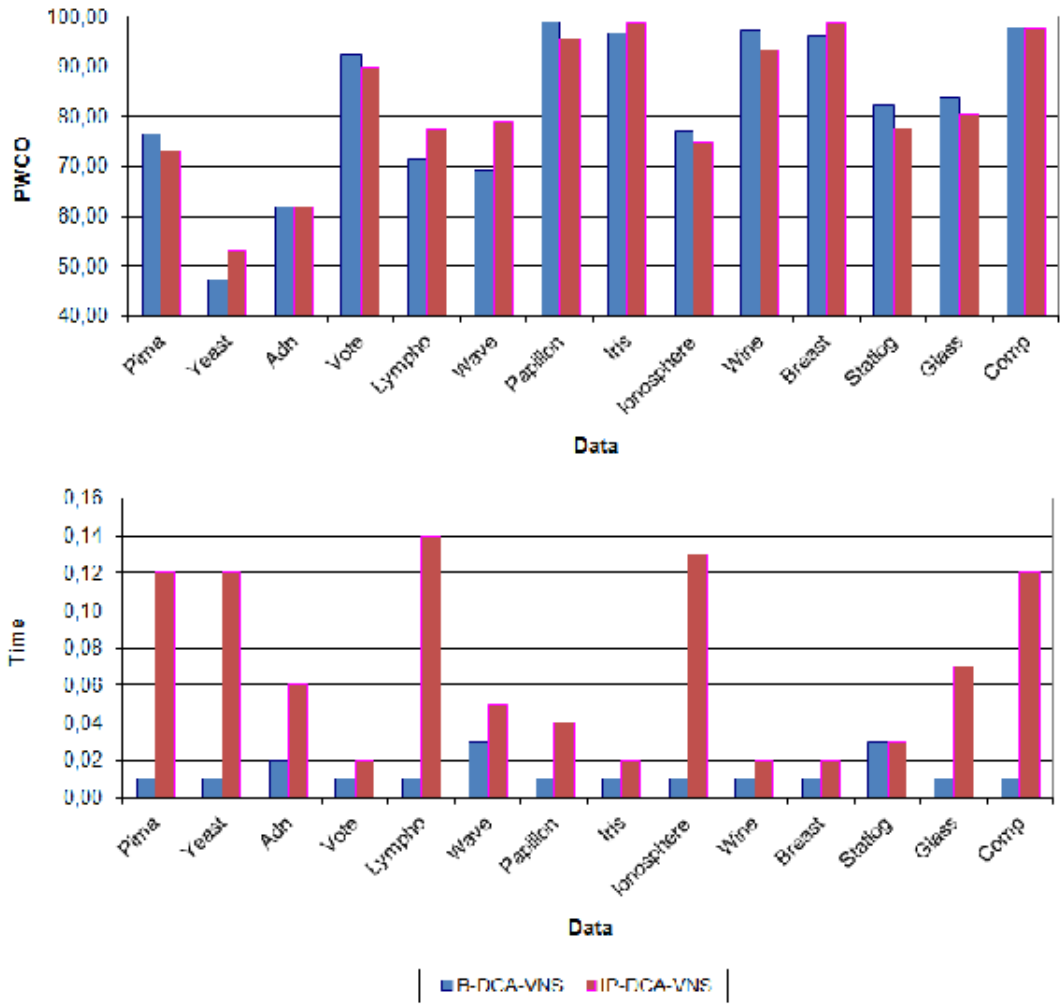


FIGURE 7.1 – Comparative results of *IP-DCA-VNS* and *B-DCA-VNS*

In the second experiment, we are interested in the impact of kernel method. We compare the two DCA based algorithms applied on the bilevel formulation, with and without kernel, say *B-DCA-VNS* and *GK-RDCA-VNS*. The best results out of 10 executions for each algorithm are given in Figure 7.2.

We can see that the kernel algorithm *GK-RDCA* is very efficient. In most of cases *GK-RDCA* gives the best results (except for Statlog and Glass), with a very slight increase in the running time.

Experiment 3

The third experiment deals with the effect of starting points for DCA. We compare two variants (with and without VNS procedure) of two new DCA based algorithms (*IP-DCA* and *GK-RDCA*). The comparative results (best result over 10 executions) are presented in Figure 7.3 and 7.4.

From the computational results we observe that using VNS for the initial point improves the result, in all most of cases.

FIGURE 7.2 – Comparative results of *B-DCA-VNS* and *GK-DCA-VNS*

Experiment 4

In this experiment, we are interested in the effectiveness of *IP-DCA* when the value of parameter ρ varies. Note that the *Update* ρ proceduce can also be used for *GK-DCA* but the ρ_0 in GKMSSC is small and a slightly smaller value do not influence on the quality of *GK-DCA* .

For this purpose, we arbitrarily choose a data set (Iris) and then vary the value of ρ from 178 (the value obtained by applying (7.14)) to 25 (the value computed by the procedure *Update* ρ). We report in Figure 7.5 the PWCO and the number of iterations of *IP-DCA* for each value of ρ.

We observe that, not surprisingly, the smaller ρ is, the more efficient *IP-DCA* is (in term of PWCO). The best value of ρ is 30 (on both PWCO and number of iterations) which is much more smaller than the theoretical value given by (7.14).

Experiment 5

In the last experiment we compare the performance between six variants of DCA based algorithms as well as those with k-means, Gaussian kernel k-means algorithms (*GK-k-mean*) and

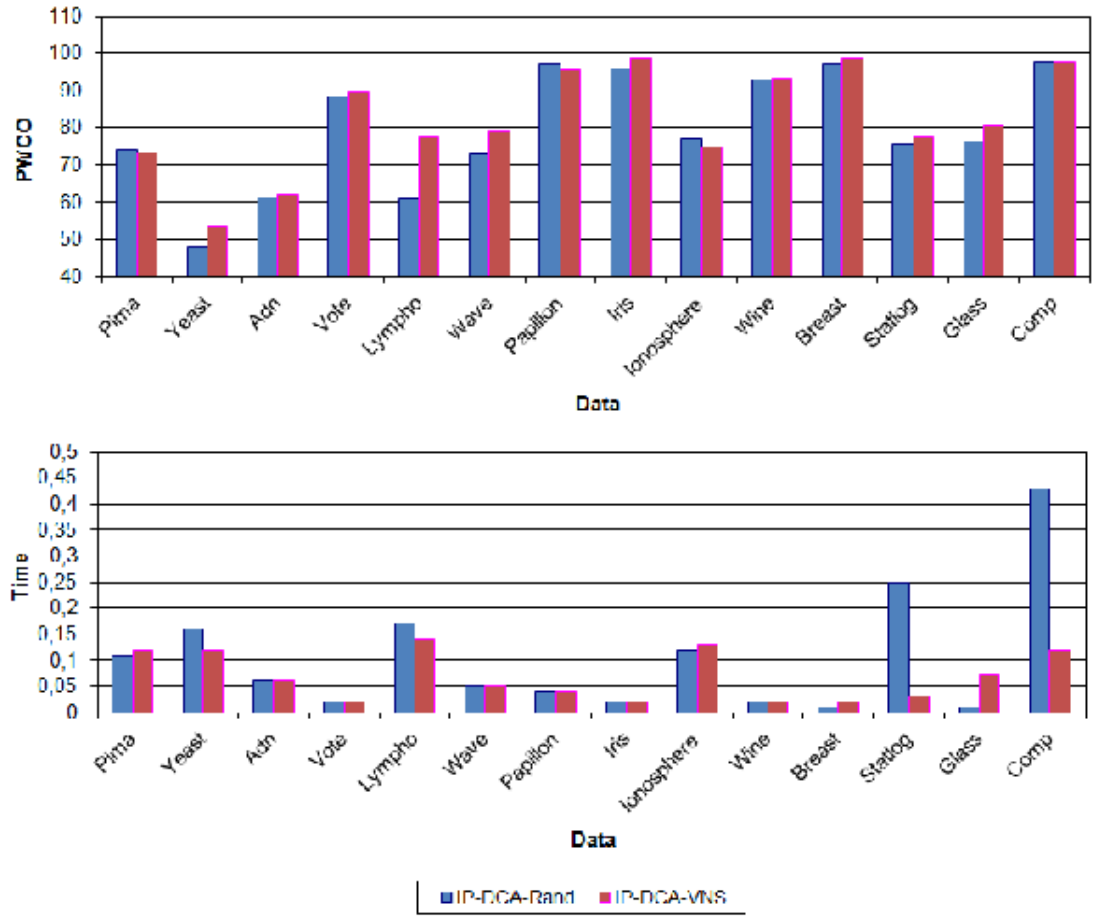


FIGURE 7.3 – IP-DCA with and without VNS for starting point

VNS algorithm(*VNS*).

For every data instance, we perform each algorithm 10 times from 10 random starting points and report in Table 7.2 the mean and the standard deviation of each criterion.

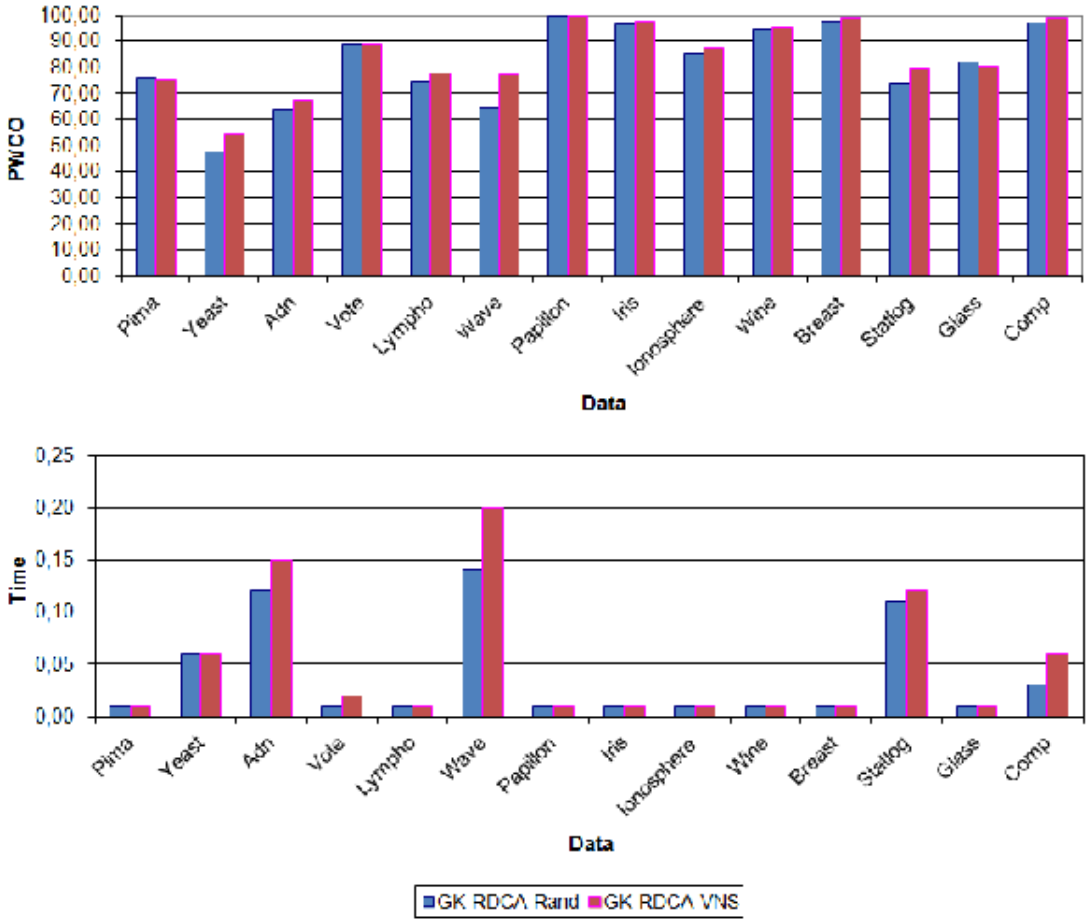


FIGURE 7.4 – GK-RDCA with and without VNS for starting point

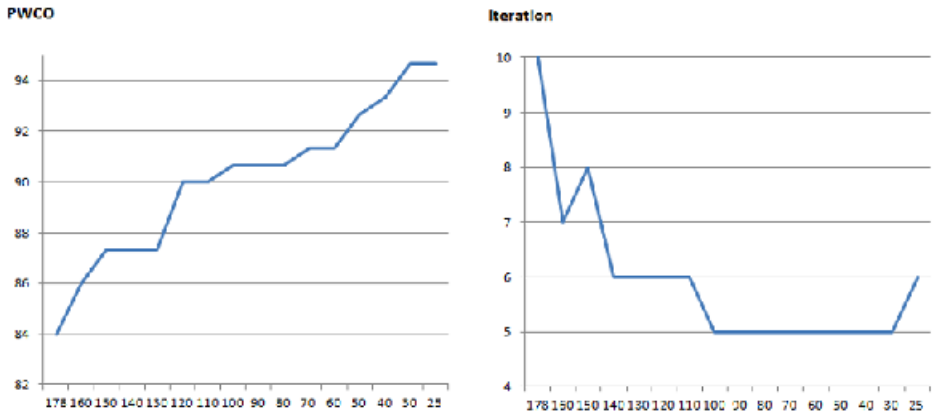


FIGURE 7.5 – Results of *IP-DCA* when the value of parameter ρ varies

TABLE 7.2 – Comparative results of DCA based algorithms, k-mean based algorithms and VNS

Data	Results	IP-DCA-Rand	IP-DCA-VNS	B-DCA-Rand	B-DCA-VNS	GK-DCA-Rand	GK-DCA-VNS	k-means	GK-k-means	VNS
Pima	POWC	73,1±4,1	72,5±3,3	73,4±3,4	74,2±3,6	76,2±3,4	73,2±3,4	70,3±4,6	68,4±4,9	69,2±5,2
	RandI	61,4±3,6	59,6±4,1	60,7±4,1	60,1±3,5	62.3±3,2	61,2±3,4	58,2±4,6	60,3±4,6	60,4±4,2
	I_{Intra}	0,19±0,04	0,17±0,04	0,16±0,04	0,17±0,04	2,53±0,04	0,38±0,03	0,15±0,05	0,15±0,04	0,15±0,04
	I_{Inter}	0,009±0,002	0,036±0,005	0,026±0,004	0,016±0,03	2,514±0,5	0,369±0,05	0,032±0,008	0,5±0,06	0,4±0,1
	Time	0,11±0,05	0,12±0,05	0,01±0,005	0,01±0,005	0,01±0,005	0,01±0,01	0,01±0,01	86±5	0,5±0,1
Yeast	POWC	47,2±2,5	51,2±2,4	43,3±2,6	45,2±2,6	46,5±3,2	52,3±2,7	47,84±2,5	50,32±3,8	44,4±2,5
	RandI	73,4±3,6	69,2±3,4	73,2±3,5	71,4±4,1	63,5±4,3	68,1±4,1	70,9±4,5	40,2±4,1	66,9±3,5
	I_{Intra}	0,04±0,01	0,049±0,01	0,05±0,01	0,052±0,01	1,25±0,1	0,08±0,01	0,03±0,01	0,04±0,01	0,05±0,01
	I_{Inter}	0,053±0,05	0,066±,.05	0,025±0,008	0,020±0,007	1,327±0,30	0,15±0,04	0,054±0,009	0,34±0.1	0,05±0,01
	Time	0,16±0,04	0,17±0,04	0,01±0.003	0,01±0,004	0,06±0,002	0,06±0,002	0,03±0,02	194±32	0,09±0,02
Adn	POWC	61,5±5,1	61,9±4,1	61,7±3,5	62.1±3,7	63,5±4,1	65,2±4,5	58,4±5,1	60,3±4,6	60,4±3,5
	RandI	60,5±4,5	60,1±4,1	60,1±3,1	60,5±3,5	63,6±3,7	64,4±3,9	60,3±4,7	45,3±4,1	64,2±3,1
	I_{Intra}	8,1±0,4	8,1±0,3	8,2±0,4	8,5±0,3	10,1±0,4	9,9±0,5	7,8±0,6	1,9±0,3	8,5±0,8
	I_{Inter}	0,255±0,07	0,593±0,1	0,592±0,1	0,492±0,1	4,09±1,1	3,839±0,5	0,427±0,1	8,61±1,3	0,325±0,12
	Time	0,06±0,02	0,06±0,02	0,02±0,01	0,03±0,01	0,12±0,03	0,15±0,03	0,1±0,04	216±25	0,4±0,6
Vote	POWC	88,5±2,3	87,5±2,4	88,1±1,5	89,3±2,5	88,9±2,6	88,1±2,1	87,1±2,6	85,8±2,6	85,3±3,6
	RandI	79,6±2,5	78,2±2,6	78,8±1,7	80,8±2,2	80,3±2,7	78,9±2,4	77,5±2,6	80,2±2,3	75,2±2,6
	I_{Intra}	0,96±0,3	0,998±0,3	0,97±0,3	1,05±0,3	1,15±0,4	2,81±0,5	0,94±0,4	2,55±0,9	0,74±0,6
	I_{Inter}	0,30±0,06	0,32±0,05	0,33±0,06	0,25±0,04	0,92±0,02	3,33±0,5	0,4±0,1	4,1±0,9	0,8±0,2
	Time	0,02±0,01	0,02±0,01	0,01±0,01	0,01±0,01	0,01±0,01	0,02±0,01	0,01±0,005	15±2	0,1±0,2
Lympho	POWC	58,7±4,1	74,3±3,5	70,1±3,1	70,5±3,1	73,6±3,7	75,6±3,9	65,54±4,2	56,54±4,5	68,13±3,2
	RandI	61,3±3,6	66,7±3,1	64,1±4,1	67,5±3,5	66,7±4,1	70,2±3,5	64,1±3,8	53,2±3,5	68,1±3,2
	I_{Intra}	0,60±0,2	0,89±0,2	0,65±0,12	0,7±0,14	1,98±0,4	0,99±0,2	0,58±0,2	0,88±0,4	0,52±0,2
	I_{Inter}	0,16±0,05	0,45±0,1	0,22±0,03	0,19±0,03	1,73±0,3	0,68±0,2	0,25±0,1	0,9±0,3	0,55±0,7
	Time	0,17±0,04	0,19±0,04	0,01±0,005	0,01±0,003	0,01±0,003	0,01±0,002	0,01±0,003	9±3	0,4±0,1
Wave	POWC	71,9±3,4	76,9±4,3	65,9±4,3	67,6±4,5	64,1±3,6	76,8±4,3	58,5±4,3	58,2±4,5	62,4±3,4
	RandI	71,4±3,6	75,9±4,6	68,8±3,1	69,5±3,6	69,5±4,2	75,9±3,7	69,9±4,1	52,1±3,2	72,3±4,1
	I_{Intra}	2,3±0,4	2,3±0,4	2,3±0,3	2,9±0,3	2,4±0,3	2,2±0,2	2,2±0,4	2,01±0,3	2,3±0,4
	I_{Inter}	0,27±0,05	0,35±0,05	0,37±0,04	0,27±0,03	0,08±0,02	0,35±0,05	0,38±0,05	0,65±0,15	0,33±0,05
	Time	0,05±0,01	0,05±0,01	0,03±0,01	0,04±0,01	0,14±0,07	0,2±0,06	0,08±0,03	2,91±1	0,5±0,2
Papillon	POWC	96,1±4,1	95,6±3,4	96,1±4,5	97,1±4,1	99,5±0,4	100±0	99±1	88,2±4,6	99±1
	RandI	95,02±4,3	94,4±4,1	95,7±3,1	95,4±3,4	99,3±0,3	100±0	99±1	85,1±3,2	99±1
	I_{Intra}	0,015±0,003	0,013±0,003	0,011±0,003	0,011±0,002	0,016±0,003	0,018±0,003	0,01±0,002	0,01±0,003	0,01±0,003
	I_{Inter}	0,023±0,004	0,036±0,004	0,021±0,004	0,021±0,005	0,01±0,002	0,009±0,002	0,03±0,004	0,17±0,004	0,03±0,006

	Time	0,04±0,01	0,04±0,01	0,01±0,005	0,01±0,005	0,01±0,003	0,01±0,005	0,01±0,003	2±0,2	0,15±0,005
Iris	POWC	96±3,5	96,7±3,4	96,1±3,5	96,6±3,1	96,7±4,3	96,8±4,5	96±4,3	83,2±3,8	94,5±3,3
	RandI	94,95±3,2	95,75±3,4	95,1±3,6	96±3,1	95,75±3,5	95,75±3 ;7	94,95±3,4	80,2±3	92,9±3,4
	I_{Intra}	0,023±0,003	0,058±0,005	0,023±0,002	0,023±0,002	0,0279±0,005	0,059±0,001	0,019±0,007	0,03±0,015	0,021±0,006
	I_{Inter}	0,102±003	0,138±0,04	0,131±0,04	0,131±0,04	0,195±0,04	0,138±0,04	0,159±0,04	0,17±0,05	0,149±0,05
	Time	0,02±0,01	0,02±0,01	0,01±0,005	0,01±0,005	0,01±0,005	0,01±0,005	0,01±0	2±0,4	0,2±0,04
Ionosphere	POWC	77,1±3,4	74,6±3,5	74,1±2,5	75,1±2,9	85,4±3,1	85,2±2,6	71,2±2,4	80,9±2,8	72,4±2,6
	RandI	63,4±3,1	62,1±2,8	61,7±2,2	64,2±3,4	75,1±3,4	74,6±2,8	58,8±4,1	77,4±4,1	59,8±3,1
	I_{Intra}	8,6±2,1	14,5±2,4	8,94±1,7	9,24±1,6	22,9±3,1	28,4±2,5	6,8±1,6	20,4±2,3	6,9±1,2
	I_{Inter}	0,549±0,2	6,88±1,4	0,443±0,07	0,343±0,05	18,29±1,5	20,78±2,1	2,34±0,7	23,13±2,4	2,7±0,7
	Time	0,12±0,08	0,13±0,04	0,01±0,003	0,02±0,006	0,01±0,005	0,01±0,005	0,01±0,005	3±1	0,4±0,1
Wine	POWC	92,7±2,6	93,2±2,1	92,8±1,6	94,2±2,1	94,3±2,4	94,3±1,7	93,2±3,1	76,45±3,1	92,4±2,4
	RandI	90,3±01,6	91,2±2,5	90,5±1,8	92,4±2,5	92,4±3,1	92,4±2,1	91,0±2,5	75,1±2,8	91,0±2,2
	I_{Intra}	0,155±0,06	0,299±0,04	0,159±0,04	0,16±0,03	0,4102±0,05	0,167±0,03	0,154±0,05	0,04±0,02	0,16±0,05
	I_{Inter}	0,149±0,04	0,42±0,08	0,144±0,01	0,12±0,01	0,569±0,03	0,216±0,04	0,154±0,03	0,45±0,05	0,14±0,04
	Time	0,02±0,01	0,02±0,01	0,01±0,004	0,02±0,001	0,01±0,01	0,01±0,01	0,01±0,01	2±0,5	0,3±0,1
Breast	POWC	96,9±3,1	97,5±2,4	96,6±2,3	97±2,1	97,6±1,6	97,5±2,1	96±2,4	79,36±4,5	94,5±2,1
	RandI	94,1±2,3	95,1±2,4	93,6±2,7	94,8±2,9	95,4±1,9	95,1±1,7	92,4±2,6	77,1±3,8	91,3±2,2
	I_{Intra}	0,497±0,1	0,712±0,2	0,318±0,11	0,38±0,12	0,337±0,08	0,543±0,11	0,283±0,09	0,28±0,09	0,23±0,1
	I_{Inter}	0,097±0,013	0,914±0,021	0,325±0,011	0,23±0,09	0,534±0,09	0,052±0,014	0,426±0,07	0,75±0,19	0,45±0,06
	Time	0,01±0,005	0,02±0,007	0,01±0,01	0,02±0,008	0,01±0,006	0,01±0,006	0,01±0,004	168±25	0,3±0,03
Statlog	POWC	75,45±2,1	76,4±1,9	76,4±1,8	79,4±1,2	74,1±3,2	77,3±2,3	71,5±4,2	62,41±4,5	73,5±3,4
	RandI	86,7±2,4	86,3±2,2	87,4±1,9	86,4±1,4	85,7±2,7	88,4±1,7	82,8±3,8	59,1±3,6	84,2±2,8
	I_{Intra}	0,215±0,06	0,964±0,12	0,207±0,08	0,307±0,08	0,0450,01	0,236±0,04	0,147±0,04	0,19±0,05	0,17±0,04
	I_{Inter}	0,378±0,07	0,464±0,14	0,381±0,11	0,311±0,007	0,621±0,12	0,323±0,008	0,537±0,13	0,4±0,15	0,39±0,11
	Time	0,25±0,08	0,03±0,01	0,03±0,01	0,03±0,01	0,11±0,003	0,12±0,004	0,08±0,001	192±16	0,3±0,1
Glass	POWC	76,2±2.3	80,4±2,1	79,8±3,2	80,8±3,1	81,8±2,5	80,3±2,3	74,3±3,4	68,6±5,3	74,9±2,5
	RandI	85,9±2 ;1	89,8±1,6	90,2±2,6	89,2±27	91,4±2,1	89,8±1,8	86±3,1	63,2±4,1	86,4±3,3
	I_{Intra}	0,099±0,01	0,09±0,02	0,11±0,05	0,14±0,04	0,1445±0,03	0,089±0,02	0,08±0,02	0,06±0,02	0,084±0,02
	I_{Inter}	0,166±0,02	1,329±0,03	0,149±0,03	0,149±0,02	0,067±0,01	0,203±0,03	0,208±0,05	0,46±0,05	0,18±0,05
	Time	0,01±0,005	0,07±0,02	0,01±0,004	0,02±0,008	0,01±0,002	0,01±0,003	0,01±0,002	54±4	0,2±0,007
Comp	POWC	97,7±3,5	97,6±2,9	96,4±3,5	97,7±2,6	97,2±2.9	97,6±2,7	94,5±3,4	83,4±3,8	95,2±2,5
	RandI	97,05±3,2	96,9±2,5	97±2,5	97,4±2,9	96,4±3,2	96,9±3,1	96,8±3,5	82,5±4,2	97,4±2,9
	I_{Intra}	0,445±0,08	0,431±0,08	0,442±0,09	0,47±0,11	0,665±0,15	1,069±0,34	0,431±0,14	0,91±0,22	0,44±0,12
	I_{Inter}	0,264±0,06	0,323±0,08	0,289±0,07	0,219±0,04	0,01±0,02	1,8±0,5	0,32±0,04	2±0,4	0,30±0,04
	Time	0,43±0,1	0,12±0,04	0,01±0,003	0,02±0,004	0,03±0,003	0,06±0,008	0,02±0,002	189±11	0,5±0,1
Average	PWCO	79,19±3,4	81,17±3,2	79,39±2,9	80,47±2,7	81,43±2,8	82,95±2,7	77,70±3,6	71,52±3,7	77,50±3,2
	RandI	79,68±3,2	80,13±2,9	79,79±2,4	80,38±2,6	81,39±2,7	82,27±2,4	78,86±3,5	66,52±3,7	79,2±2,9

I_{Intra}	1,59±0,2	2,08±0,3	1,61±0,2	1,72±0,3	3,14±0,2	3,37±0,2	1,42±0,4	2,11±0,6	1,21±0,3
I_{Inter}	0,20±0,05	0,88±0,04	0,25±0,05	0,20±0,04	2,21±0,3	2,31±0,5	0,40±0,05	3,05±0,5	1,56±0,4
Time	0,11±0,02	0,08±0,02	0,01±0,006	0,02±0,01	0,04±0,01	0,05±0,01	0,03±0,01	81,07±8	0,3±0,1

Overall, all DCA based algorithms are efficient : the PWPO given by these algorithms is greater than 70% in 12/15 datasets and the average of PWCO on all datasets is greater than 79%, while running times are less than 0.2 second. Morerover, DCA based algorithms are much better than k-means and GK-k-means on the quality of solutions. The rapidity of DCA and k-means are comparable while GK-k-means is much slower : the running time of GK-k-means varies from 2 to 216 seconds while the one of other algorithms is less than 0.2 second.

7.6 Conclusion

We have proposed two new and efficient based DCA algorithms for solving the MSSC problem. The hard combinatorial optimization MSSC model has been recast as a DC program in its elegant matrix formulation and with a nice DC decomposition, in order to make simpler and so much less expensive the computations in the resulting DCA. It fortunately turns out that the corresponding DCA consists in computing, at each iteration, the projection of points onto a simplex and/or a ball, that all are given in the explicit form. In addition, a kernel version of the bilevel formulation of the MSSC has been considered. The choice of the Gaussian kernel method is to facilitate not only the formulation of the problem, but also the implicit projection of the data in a space where the representation is richer and the resolution of the problem is simpler. For both DCA schemes, the two main features of DCA were carefully studied. The choice of DC decomposition is judicious, and the regularization technique is very useful in the context of hard clustering. Like *IP-DCA*, the resulting *GK-DCA* is very simple, it requires, at each iteration, the explicit projection of points onto a box. For initializing DCA, a local search procedure VNS is proposed. Numerical results on several real datasets showed the robustness, the effectiveness and the superiority of the DCA based schemes with respect to the k-means based algorithms. We are convinced that DCA is an efficient, fast and scalable approach for clustering.

Chapitre 8

Minimum Sum-of-Squares Clustering using weighted dissimilarity measures[1]

Abstract: In this chapter, we address the MSSC (Minimum Sum-of-Squares Clustering) using weighted dissimilarity measures. Two most widely used models of MSSC(bilevel program and mixed integer program) are studied. It turns out that both optimization problems can be reformulated as a DC program and then efficient DCA algorithms are developed. Experimental results on real world datasets have illustrated the efficiency of our proposed algorithms and its superiority with respect to standard algorithms in terms of quality of solution.

8.1 Introduction

Feature selection is one of the techniques to deal with irrelevant or redundant features. Feature selection methods aim to select a subset of features that minimize redundancy while preserving or improving the classification rate of algorithm. On the other hand, feature weighting that can be seen as an extension of feature selection, has attracted the attention of many researchers. 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. The difference between feature selection and weighting feature is that weighting feature does not aim to reduce the number of used features in clustering. The main objective of feature weighting is to improve the quality of classification algorithm.

Feature weighting has been applied successfully in many classification algorithms. Feature weighting in SVMs [63], in K-Means type clustering [47, 109, 156], in Fuzzy classification [101], etc to name a few. In this chapter, we deal with the MSSC (Minimum Sum of Squares Clustering) using weighted features. Recall that, in Chapter 7, we have presented two most used models that are the mixed integer program (7.1) and the bilevel programming problem (7.2). We now introduce the feature weighting to these two models. In feature weighing, the dissimilarity measure

1. The results presented in this chapter were published in :
   - H.M. Le, M.T. Ta, DC Programming and DCA for Solving Minimum Sum-of-Squares Clustering Using Weighted Dissimilarity Measures, Transactions on Computational Intelligence XIII, 113-131, 2014.

between a center v_i and a data point x_k is now defined by d weighted features, namely

$$d^2_{WF}(v_i, x_k) = \sum_{j=1}^{d} \lambda_{ij}^{\beta}(v_{ij} - x_{kj})^2,$$

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

$$\begin{cases} \min F_1(V, \Lambda) := \sum\limits_{k=1}^{n} \min\limits_{i=1,\ldots,c} \sum\limits_{j=1}^{d} \lambda_{ij}^{\beta}(v_{ij} - x_{kj})^2 \\ s.t : \sum\limits_{j=1}^{d} \lambda_{ij} = 1, i = 1..c, \\ \quad \lambda_{ij} \in [0, 1], i = 1..c, j = 1..d. \end{cases} \tag{8.1}$$

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

$$\begin{cases} \min F_2(U, V, \Lambda) := \sum\limits_{i=1}^{c} \sum\limits_{k=1}^{n} \sum\limits_{j=1}^{d} u_{ik}\lambda_{ij}^{\beta}(v_{ij} - x_{kj})^2 \\ s.t : \sum\limits_{i=1}^{c} u_{ik} = 1, k = 1..n, \\ \quad \sum\limits_{j=1}^{d} \lambda_{ij} = 1, i = 1..c, \\ \quad u_{ik} \in \{0, 1\}, i = 1..c, k = 1..n, \\ \quad \lambda_{ij} \in [0, 1], i = 1..c, j = 1..d. \end{cases} \tag{8.2}$$

where β is an exponent greater than 1.

While several heuristic and deterministic approaches have been investigated to MSSC, there is a few deterministic methods dealing with weighted MSSC models. In [47], the authors considered a K-means type algorithm, named **WF-KM**, to solve the problem (8.2). At first, **WF-KM** fixes V, Λ and finds U to minimize $F(U, ., .)$. Then U, Λ are fixed for finding V minimizing $F(., V, .)$. Finally, Λ is obtained by minimizing $F(., ., \Lambda)$ with U and V fixed. The process is repeated until no more improvement in the objective function can be made. In [156], the authors proposed a variance of (8.2) by adding the entropy of dimensions, namely $\gamma \sum_{k=1}^{d} \lambda_{ij} \log \lambda_{ij}$, 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 ([109]), a simplified version of (8.2) was considered where the matrix of weights Λ becomes a vector $\bar{\Lambda}$. More precisely, $\bar{\Lambda}_k$ defines the relevance of k-th feature to all cluster C_i $(i = 1..c)$. The proposed algorithms in [156] and [109] are similar to the **WF-KM** developed in [47].

Based on the algorithms we have developed in Chapter 7 for the mixed integer program (7.1) and the bilevel programming problem (7.2), we develop similar algorithm to solve (8.1) and (8.2).

The remainder of the chapter is organized as follows. In Section 8.2, we present DCA based algorithms for solving (8.1) and (8.2). Numerical results on real datasets and some remarks will be presented in the Section 8.4.

8.2 DCA for solving MSSC using weighted features

We will adapt the techniques developed in Chapter 7 for getting DC formulations and corresponding DCA for solving (8.1) and (8.2).

8.2.1 DCA for solving the bilevel formulation (8.1)

In the problem (8.1) the variable Λ are a priori bounded. One can also find a constraint to bound the variable V. Let $\alpha_i := \min_{k=1,...,n} x_{kj}$, $\gamma_i := \max_{k=1,...,n} x_{kj}$. Hence $v_i \in \mathcal{T}_l := \Pi_{j=1}^{d} [\alpha_i, \gamma_i]$ for all $i = 1, ..., c$. Finally, $V \in \mathcal{T} := \Pi_{i=1}^{c} \mathcal{T}_i$. Let Δ_i be the $(m-1)$-simplex in $\mathbb{R}^m$, for each $i \in \{1, ..., c\}$, defined by :

$$\Delta_i := \left\{ \Lambda_i := (\lambda_{ij})_i \in [0,1]^d : \sum_{j=1}^{d} \lambda_{ij} = 1 \right\}$$

and $\mathcal{T} := \Pi_{i=1}^{c} \mathcal{T}_i, \Delta := \Pi_{i=1}^{c} \Delta_i$.

Then the problem (8.1) can be rewritten as :

$$\min \{F_1(V, \Lambda) : V \in \mathcal{T}, \Lambda \in \Delta\}. \tag{8.3}$$

Denote $f_i(v_{ij}, \lambda_{ij}) = \sum_{j=1}^{d} \lambda_{ij}^{\beta} (v_{ij} - x_{kj})^2$. Then

$$F_1(V, \Lambda) = \sum_{k=1}^{n} \min_{i=1,...,c} f_i(v_{ij}, \lambda_{ij}). \tag{8.4}$$

We can see that f_i can be decomposed as follows

$$f_i(v_{ij}, \lambda_{ij}) = g_i(v_{ij}, \lambda_{ij}) - h_i(v_{ij}, \lambda_{ij})$$

where $g_i(v_{ij}, \lambda_{ij}) = \sum_{j=1}^{d} (\frac{\rho_1}{2} v_{ij}^2 + \frac{\rho_1}{2} \lambda_{ij}^2)$ and $h_i(v_{ij}, \lambda_{ij}) = \sum_{j=1}^{d} \Big[(\frac{\rho_1}{2} v_{ij}^2 + \frac{\rho_1}{2} \lambda_{ij}^2) - \lambda_{ij}^{\beta} (v_{ij} - x_{kj})^2 \Big]$.
On another hand, one has

$$\min_{i=1,...,c} f_i = \min_{i=1,...,c} (g_i - h_i) = \sum_{i=1}^{c} g_i - \max_{i=1,...,c} \Big\{ \sum_{p=1, p \neq i}^{c} g_p + h_i \Big\} \tag{8.5}$$

By applying the above formula to the objective function of (8.4), we obtain :

$$\begin{aligned} F_1(V, \Lambda) &= \sum_{k=1}^{n} \sum_{i=1}^{c} g_i - \sum_{k=1}^{n} \max_{i=1,...,c} \Big\{ \sum_{p=1, p \neq i}^{c} g_p + h_i \Big\} \\ &= G_1(V, \Lambda) - H_1(V, \Lambda) \end{aligned} \tag{8.6}$$

where

$$G_1(V, \Lambda) = \sum_{k=1}^{n} \sum_{i=1}^{c} g_i$$

and

$$H_1(V, \Lambda) = \sum_{k=1}^{n} \max_{i=1,...,c} \Big\{ \sum_{p=1, p \neq i}^{c} \sum_{j=1}^{d} \frac{\rho_1}{2} (v_{pj}^2 + \lambda_{pj}^2) + \sum_{j=1}^{d} \Big[\frac{\rho_1}{2} (v_{ij}^2 + \lambda_{ij}^2) - \lambda_{ij}^{\beta} (v_{ij} - x_{kj})^2 \Big] \Big\}.$$

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

Proposition 8.1 *There exists $\rho_1 > 0$ such that the function H_1 is a convex function on $\{V \in \mathcal{T}, \Lambda \in \Delta\}$.*

Proof 8.1 *We consider the function $f_1 : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ defined by :*

$$f_1(v, y) = y^\beta (v - a)^2. \tag{8.7}$$

The Hessian of f_1 is given by :

$$J(v, y) = \begin{pmatrix} 2y^\beta & 2\beta(v-a)y^{\beta-1} \\ 2\beta(v-a)y^{\beta-1} & \beta(\beta-1)(v-a)^2 y^{\beta-2} \end{pmatrix}. \tag{8.8}$$

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

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

Hence :

$$\lambda_{1,2} = \frac{1}{2} * \Big\{ [2y^\beta + \beta(\beta-1)(v-a)^2 y^{\beta-2}] \pm sqrt(\triangle) \Big\} \tag{8.9}$$

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

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

Hence, the function :

$$h_1(v, y) = \frac{\rho_1}{2}\left(v^2 + y^2\right) - y^\beta (v-a)^2 \tag{8.10}$$

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

$$\rho_1 \geq \frac{1}{2} * \left(2 + \beta(\beta-1)\gamma^2 + \sqrt{4 + \beta^2(\beta-1)^2\gamma^4 + 12\beta^2\gamma^2 + 4\beta\gamma^2}\right) \tag{8.11}$$

where $\gamma = \sigma - \alpha$ and $\beta > 1$.
As a consequence, for $v \leftarrow v_{ij}, y \leftarrow \lambda_{ij}$, the function

$$h_{li}(v_{ij}, \lambda_{ij}) = \frac{\rho_1}{2}\left(v_{ij}^2 + \lambda_{ij}^2\right) - \lambda_{ij}^\beta (v_{ij} - x_{kj})^2 \tag{8.12}$$

is convex on $\{v_{ij} \in [\alpha_i, \gamma_i], \lambda_{ij} \in [0, 1]\}$
Hence, the function $H_1(V, \Lambda)$ is convex on $\{V \in \mathcal{T}, \Lambda \in \Delta\}$. ■

DCA applied to (8.6). According the generic DCA scheme, at each iteration, we have to compute $(\bar{Z}^l, \bar{\Lambda}^l) \in \partial H_1(Z^l, \Lambda^l)$ and then solve the convex program :

$$\min \left\{ \sum_{k=1}^{n} \sum_{i=1}^{c} \sum_{j=1}^{d} \left(\frac{\rho_1}{2} v_{ij}^2 + \frac{\rho_1}{2} \lambda_{ij}^2 \right) - \langle (V, \Lambda), (, \bar{Z}^l, \bar{\Lambda}^l) \rangle : V \in \mathcal{T}, \Lambda \in \Delta \right\}. \tag{8.13}$$

We have

$$H_1(V, \Lambda) = \sum_{k=1}^{n} H_k(V, \Lambda) \tag{8.14}$$

where $H_k(V, \Lambda) = \max\limits_{i=1,\ldots,c} H_{ki}(v_{ij}, \lambda_{ij})$ and

$$H_{ki}(v_{ij}, \lambda_{ij}) = \sum_{p=1, p\neq i}^{c} \sum_{j=1}^{d} \frac{\rho_1}{2}(v_{pj}^2 + \lambda_{pj}^2)\ + \sum_{j=1}^{d} \Big[\frac{\rho_1}{2}(v_{ij}^2 + \lambda_{ij}^2) - \lambda_{ij}^\beta (v_{ij} - x_{kj})^2 \Big]. \tag{8.15}$$

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

$$\partial H_1(V,\Lambda) = \sum_{k=1}^{n} \partial H_k(V,\Lambda) \tag{8.16}$$

where (co denotes convex hull)

$$\partial H_k(V,\Lambda) = co\{\partial H_{ki} : H_{ki} = H_k\}. \tag{8.17}$$

H_{ki} is differentiable and

$$\begin{aligned} \frac{\nabla H_{ki}}{\nabla v_{rj}} &= \begin{cases} \rho_1 v_{rj} - 2\lambda_{ri}^{\beta}(v_{rj} - x_{kj}) \text{ if } r = i \\ \rho_1 v_{rj} \text{ if } r \neq i \end{cases} \quad \forall r = 1,...,c, \\ \frac{\nabla H_{ki}}{\nabla \lambda_{rj}} &= \begin{cases} \rho_1 \lambda_{rj} - \beta\lambda_{rj}^{\beta-1}(v_{rj} - x_{kj})^2 \text{ if } r = i \\ \rho_1 v_{rj} \text{ if } r \neq i \end{cases} \quad \forall r = 1,...,c. \end{aligned} \tag{8.18}$$

On another hand, the solution of the auxiliary problem (8.13) is explicitly computed as ($Proj_D$ stands for the orthogonal projection on D) :

$$\begin{aligned} &(V^{l+1})_{li} = \text{Proj}_{[\alpha_i,\gamma_i]}\left(\tfrac{1}{n\rho_1}(\bar{Z}^l)_{li}\right) \ i = 1,...,c, j = 1,...,d; \\ &(\Lambda^{l+1})_l = \text{Proj}_{\Delta_l}\left(\tfrac{1}{n\rho_1}(\bar{\Lambda}^l)_l\right) \ i = 1,...,c. \end{aligned} \tag{8.19}$$

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 ([116]).

The algorithm can be described in Algorithm 8.1.

Algorithm 8.1 BI-WF-DCA : DCA applied to (8.6)

1: **Initialization :** Choose an initial point (V^0, Λ^0) and $l \leftarrow 0$.
2: **repeat**
3: Compute $(\bar{V}^l, \bar{\Lambda}^l)$ via (8.14)-(8.18).
4: Compute (V^{l+1}, Λ^{l+1}) via (8.19).
5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

8.2.2 DCA for solving the mixed integer formulation (8.2)

In this section, we deal with the mixed integer programming problem 8.2. Since $u_{ik} \in \{0,1\}$ we can replace u_{ik} by u_{ik}^2 and rewrite the objective function of (8.2) by $F_2(U,V,\Lambda) := \sum_{i=1}^{c}\sum_{k=1}^{n}\sum_{j=1}^{d} u_{ik}^2\lambda_{ij}^{\beta}(v_{ij} - x_{kj})^2$. In the problem (8.2) the variables U and Λ are a priori bounded. By the same way as in Section 3.1, the variable V can be bounded. Let $\mathcal{C}_j$ be the $(k-1)$-simplex in $\mathbb{R}^k$, for each $j \in \{1,...,n\}$, defined by :

$$\mathcal{C}_j := \left\{ W_j := (u_{ik})_j \in [0,1]^k : \sum_{i=1}^{c} u_{ik} = 1 \right\}$$

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

The problem (8.2) can be rewritten as :

$$\min \left\{ F_2(U,V,\Lambda) : U \in \mathcal{C} \cap \{0,1\}^{c\times n}, V \in \mathcal{T}, \Lambda \in \Delta \right\}. \tag{8.20}$$

8.2.2.1 A continuous reformulation

Our reformulation technique is based on the following new results developed in [145]. We first show that $F_2(U,V,\Lambda)$ is a DC function. Clearly, $F_2(U,V,\Lambda)$ can be reformulated as :

$$F_2(U,V,\Lambda) = G_2(U,V,\Lambda) - H_2(U,V,\Lambda) \tag{8.21}$$

where

$$\begin{aligned} G_2(U,V,\Lambda) &:= \sum_{i=1}^{c}\sum_{k=1}^{n}\sum_{j=1}^{d} \tfrac{\rho_2}{2}\left(u_{ik}^2 + v_{ij}^2 + \lambda_{ij}^2\right), \\ H_2(U,V,\Lambda) &:= \sum_{i=1}^{c}\sum_{k=1}^{n}\sum_{j=1}^{d} \left[\tfrac{\rho_2}{2}\left(u_{ik}^2 + v_{ij}^2 + \lambda_{ij}^2\right) - u_{ik}^2\lambda_{ij}^{\beta}(v_{ij} - x_{kj})^2\right]. \end{aligned} \tag{8.22}$$

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

Proposition 8.2 *There exists $\rho_2 > 0$ such that the function $H_2(U,V,\Lambda)$ is a convex function on $\{U \in \mathcal{C}, V \in \mathcal{T}, \Lambda \in \Delta\}$.*

Proof 8.2 *First, we consider the function $f_2 : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ defined by :*

$$f_2(u,v,y) = u^2 y^{\beta}(v-a)^2. \tag{8.23}$$

The Hessian of f_2 is given by :

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

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

$$\begin{aligned} |J(u,v,\ \ y)|_1 = \max\Big\{ & 2y^{\beta}(v-a)^2 + 4uy^{\beta}(v-a) + 2\beta u y^{\beta-1}(v-a)^2; \\ & 4uy^{\beta}(v-a) + 2u^2y^{\beta} + 2\beta u^2 y^{\beta-1}(v-a); \\ & 2u\beta y^{\beta-1}(v-a)^2 + 2\beta u^2 y^{\beta-1}(v-a) + \beta(\beta-1)u^2y^{\beta-2}(v-a)^2\Big\}. \end{aligned} \tag{8.24}$$

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

$$h_2(u,v,y) = \frac{\rho_2}{2}\left(u^2+v^2+y^2\right) - u^2 y^\beta (v-a)^2 \tag{8.25}$$

is convex on $\{u \in \{0,1\}, v \in [\alpha,\sigma], y \in [0,1]\}$.
Hence, for $u \leftarrow u_{ik}, v \leftarrow v_{ij}, y \leftarrow \lambda_{ij}$, the function :

$$h_{lij}(u_{ik},v_{ij},\lambda_{ij}) = \frac{\rho_2}{2}\left(u_{ik}^2+v_{ij}^2+\lambda_{ij}^2\right) - u_{ik}^2\lambda_{ij}^\beta (v_{ij}-x_{kj})^2 \tag{8.26}$$

is convex on $\{u_{ik} \in [0,1], v_{ij} \in [\alpha_i,\gamma_i], \lambda_{ij} \in [0,1]\}$.

As a result, the function $H_2(U,V,\Lambda)$ is convex on $\{U \in \mathcal{C}, V \in \mathcal{T}, \Lambda \in \Delta\}$. ■
Then $F_2(U,V,\Lambda)$ is a DC function with DC decomposition (8.21).

We will now reformulate (8.21) as a continuous optimization problem thanks to an exact penalty technique ([145]) (c.f. Theorem 7.1).
Let us consider the function p defined on $\mathbb{R}^{c\times n}$ by :

$$p(U) := \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}(1-u_{ik}).$$

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

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

By using Theorem 7.1, we obtain the following problem which is equivalent to problem (8.2) :

$$\min\left\{\bar{F}_2(U,V,\Lambda) := F_2(U,V,\Lambda) + tp(U) : U \in \mathcal{C}, V \in \mathcal{T}, \Lambda \in \Delta\right\}, \tag{8.27}$$

where $t > t_0$ is called penalty parameter.

We will now develop DC programming and DCA for solving (8.27). Remark that, if F_2 is a DC function with DC components G_2 and H_2 then the function $\bar{F}_2(U,V,\Lambda)$ is also DC :

$$\bar{F}_2(U,V,\Lambda) \ := G_2(U,V,\Lambda) - \bar{H}_2(U,V,\Lambda) \tag{8.28}$$

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

8.2.2.2 DCA applied to (8.28)

For designing a DCA applied to (8.28), we first need to compute $(\bar{W}^l,\bar{Z}^l,\bar{\Lambda}^l) \in \partial\bar{H}_2(W^l,Z^l,\Lambda^l)$ and then solve the following convex program :

$$\min\Big\{\frac{\rho_2}{2}\sum_{i=1}^{c}\sum_{k=1}^{n}\sum_{j=1}^{d}\left(u_{ik}^2+v_{ij}^2+\lambda_{ij}^2\right) - \langle(U,V,\Lambda),(\bar{W}^l,\bar{Z}^l,\bar{\Lambda}^l)\rangle :$$
$$U \in \mathcal{C}, V \in \mathcal{T}, \Lambda \in \Delta\Big\}. \tag{8.29}$$

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

$$\begin{aligned}
\bar{W}^l &= \nabla_W\bar{H}_2(U,V,\Lambda) = \left(m\rho_2 u_{ik} - \sum_{j=1}^{d} 2u_{ik}\lambda_{ij}^\beta (v_{ij}-x_{kj})^2 + t(2u_{ik}-1)\right)_{k=1,\ldots,n}^{i=1,\ldots,c},\\
\bar{Z}^l &= \nabla_Z\bar{H}_2(U,V,\Lambda) = \left(n\rho_2 v_{ij} - \sum_{k=1}^{n} 2u_{ik}^2\lambda_{ij}^\beta (v_{ij}-x_{kj})\right)_{i=1,\ldots,c}^{j=1,\ldots,d},\\
\bar{\Lambda}^l &= \nabla_\Lambda\bar{H}_2(U,V,\Lambda) = \left(n\rho_2\lambda_{ij} - \sum_{k=1}^{n} \beta u_{ik}^2\lambda_{ij}^{\beta-1}(v_{ij}-x_{kj})^2\right)_{i=1,\ldots,c}^{j=1,\ldots,d}.
\end{aligned} \tag{8.30}$$

Furthermore, the solution of the auxiliary problem (8.29) is explicitly computed as :

$$\begin{aligned}
&(W^{l+1})_j = \text{Proj}_{\mathcal{C}_j}\left(\tfrac{1}{m\rho_2}(\bar{W}^l)_j\right) \;\; j = 1, ...n;\\
&(Z^{l+1})_{li} = \text{Proj}_{[\alpha_i,\gamma_i]}\left(\tfrac{1}{n\rho_2}(\bar{Z}^l)_{li}\right) \;\; i = 1, ..., c, j = 1, ..., d;\\
&(\Lambda^{l+1})_l = \text{Proj}_{\Delta_l}\left(\tfrac{1}{n\rho_2}(\bar{\Lambda}^l)_l\right) \;\; i = 1, ..., c.
\end{aligned} \tag{8.31}$$

Finally, DCA scheme applied to (8.28) can be described as follows :

Algorithm 8.2 IP-WF-DCA : DCA applied to (8.28)

1: **Initialization :** Choose an initial point (U^0, V^0, Λ^0) and $l \leftarrow 0$.
2: **repeat**
3: Compute $(\bar{W}^l, \bar{Z}^l, \bar{\Lambda}^l)$ via (8.30).
4: Compute $(W^{l+1}, Z^{l+1}, \Lambda^{l+1})$ via (8.31).
5: $l \leftarrow l + 1$.
6: **until** Stopping criterion.

8.2.3 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 and can be done by, for example, a heuristic procedure. As proposed in [123], we use an alternative KMeans - DCA procedure for finding a starting point. The procedure is described as follows.

KM - DCA procedure

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

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

8.3 Numerical experiments

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 (`http://archive.ics.uci.edu/ml/datasets.html`), *SVM guide1* taken from LibSVM-Dataset Repository (`http://www.csie.ntu.edu.tw/cjlin/libsvm`), *ItalyPowerDemand* taken from KENT-Dataset Repository (`www.cs.ucr.edu/$\sim$eamonn/time_series_data/`) and *Mamographic* taken from KEEL-Dataset Repository (`http://sci2s.ugr.es/keel/category.php?cat=clas&order=name#sub2`). The information about datasets is summarized in Table 8.1.

Setting. 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.

TABLE 8.1 – 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

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 :

$$RandI = \frac{a+d}{a+b+c+d} \tag{8.32}$$

where

$$\begin{array}{rl} a = & \mid \{i,j \mid I_{ref}(x_i) = I_{ref}(x_j) \quad \& \quad I_{class}(x_i) = I_{class}(x_j)\} \mid, \\ b = & \mid \{i,j \mid I_{ref}(x_i) = I_{ref}(x_j) \quad \& \quad I_{class}(x_i) \neq I_{class}(x_j)\} \mid, \\ c = & \mid \{i,j \mid I_{ref}(x_i) \neq I_{ref}(x_j) \quad \& \quad I_{class}(x_i) = I_{class}(x_j)\} \mid, \\ d = & \mid \{i,j \mid I_{ref}(x_i) \neq I_{ref}(x_j) \quad \& \quad I_{class}(x_i) \neq I_{class}(x_j)\} \mid . \end{array} \tag{8.33}$$

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 $\{1.1; 1.5; 2.0; 2.5; 3.0; 3.5; 4.0; 4.5; 5.0\}$.

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 four algorithms

— *BI-WF-DCA*,

— *IP-WF-DCA*,

— *BI-DCA* : DCA for bilevel formulation of MSSC (7.2), ([123])

— *IP-DCA* : DCA of the mixed interger formulation (7.1) (Algorith 7.1).

We report in Table 8.2 the mean and standard deviation of PWCO. The comparative results of Rand Index(resp. CPU Time) are reported in Table 8.3 (resp. Table 8.4).

From the numerical results, we observe that :

— *BI-WF-DCA* and *IP-WF-DCA* give better PWCO than *BI-DCA* and *IP-DCA* on all datasets except *Wave form*. The gain *BI-WF-DCA*(resp. *IP-WF-DCA*) over *BI-DCA* is more than 10% for 5 (resp. 6) our of 11 datasets. The gain can go up to 30.74% (*Italy*

TABLE 8.2 – Comparison of CPU Time between *BI-DCA*,*BI-WF-DCA* and *IP-WF-DCA*

Data	PWCO			
	BI-DCA	IPDCA	BI-WF-DCA	IP-WF-DCA
Stalog Shuttle	43.987%±5.143%	75.411±2.132	62.884%±7.093%	**79.166%±0.000%**
Wave form	48.116%±15.855%	**71.42±3.65**	64.514%±7.118%	64.514%±7.118%
Breast Cancer Wiscosin	96.032%±0.044%	96.12±0.53	96.076%±0.471%	**96.354%±0.406%**
Ecoli	57.798%±8.311%	58.52±0.98	**61.012%±9.658%**	**61.012%±9.658%**
Column_3C	59.613%±5.652%	63.533±3.13	**72.032%±5.107%**	71.677%±4.921%
Magic	64.516%±0.580%	62.231±1.12	65.419%±2.581%	**65.991%±7.136%**
Breast Tissue	54.906%±4.401%	55.313±3.212	56.604%±5.236%	**57.547%±5.660%**
Madelon	54.433%±4.096%	54.123±3.21	55.867%±3.900%	**57.033%±2.845%**
Svmguide1	75.073%±0.013%	83.562±3.45	**87.355%±6.062%**	86.818%±6.427%
Italy Power Demand	51.448%±0.078%	73.315±4.12	**82.187%±15.543%**	**82.187%±15.543%**
Mamographic	68.687%±0.266%	67.423±0.912	68.831%±0.607%	**79.880%±0.258%**
Average	61.328%±4.040%	63.241±2.41	70.253%±5.761%	**72.925%±5.452%**

TABLE 8.3 – Comparison of Rand index Time between *BI-DCA*,*BI-WF-DCA* and *IP-WF-DCA*

Data	Rand Index			
	BI-DCA	IP-DCA	BI-WF-DCA	IP-WF-DCA
Stalog Shuttle	0.525±0.037	0.59±0.045	**0.617±0.072**	0.515±0.058
Wave form	0.686±0.038	**0.714±0.043**	0.688±0.029	0.688±0.029
Breast Cancer Wiscosin	0.924±0.001	0.921±0.001	0.925±0.009	**0.930±0.007**
Ecoli	0.821±0.027	0.822±0.031	**0.823±0.043**	**0.823±0.043**
Column_3C	0.669±0.009	0.712±0.014	**0.731±0.019**	0.726±0.017
Magic	0.542±0.003	0.554±0.019	0.549±0.015	**0.561±0.042**
Breast Tissue	0.811±0.007	0.811±0.012	0.813±0.020	**0.822±0.015**
Madelon	0.506±0.009	0.509±0.002	0.509±0.009	**0.510±0.010**
Svmguide1	0.626±0.001	0.723±0.021	**0.786±0.083**	0.779±0.088
Italy Power Demand	0.500±0.000	0.71±0.003	**0.755±0.141**	**0.755±0.141**
Mamographic	0.569±0.002	0.623±0.032	0.570±0.004	**0.678±0.003**
Average	0.653±0.012	0.69±0.192	0.706±0.040	**0.708±0.041**

TABLE 8.4 – Comparison of CPU Time between *BI-DCA*,*BI-WF-DCA* and *IP-WF-DCA*

Data	Running time			
	BI-DCA	IP-DCA	BI-WF-DCA	IP-WF-DCA
Stalog Shuttle	**0.984±0.008**	15.312±1.311	8.539±1.323	43.972±2.808
Wave form	**0.401±0.029**	1.2±0.021	0.689±0.021	1.441±0.036
Breast Cancer Wiscosin	**0.016±0.007**	0.314±0.012	0.053±0.072	0.506±0.038
Ecoli	0.030±0.008	0.03±0.021	**0.022±0.007**	0.045±0.008
Column_3C	**0.010±0.002**	0.032±0.014	0.089±0.006	0.242±0.017
Magic	**0.539±0.143**	1.10±0.312	4.944±1.093	61.779±2.087
Breast Tissue	**0.010±0.001**	0.12±0.003	0.088±0.009	0.206±0.026
Madelon	**0.404±0.048**	0.9±0.01	0.788±0.019	1.153±0.028
Svmguide1	**0.094±0.012**	0.14±0.001	0.490±0.081	0.268±0.019
Italy Power Demand	**0.039±0.020**	0.02±0.001	0.072±0.008	0.131±0.016
Mamographic	**0.013±0.008**	0.1±0.002	0.016±0.003	0.573±0.107
Average	**0.231±0.026**	1.7±0.212	1.435±0.240	10.029±0.472

TABLE 8.5 – Comparison of PWCO between *WF-KM*, *BI-WF-DCA* and *IP-WF-DCA*

Data	PWCO		
	WF-KM	BI-WF-DCA	IP-WF-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%**

Power Demand dataset) with *BI-WF-DCA* and 35.18% (*Stalog Shuttle* dataset) with *IP-WF-DCA*. The average of PWCO of *BI-WF-DCA* is 70.25% and that of *IP-WF-DCA* is 72.93% which a much more better than *BI-DCA* (61, 33%).

— The quality of our two algorithms *BI-WF-DCA* and *IP-WF-DCA* are comparable. However, *IP-WF-DCA* furnishes better PWCO than *BI-WF-DCA* with a big gain on *Stalog Shuttle* (16.29%) and *Mamographic* (11.05%).

— Concerning the Rand index criterion, except for *Stalog Shuttle* where *BI-DCA* is better than *IP-WF-DCA*, *BI-WF-DCA* and *IP-WF-DCA* alway furnish better result.

— Undoubtedly *DCA-KM* is fastest algorithm out of three and *IP-WF-DCA* is the most time consuming, especially *Magic* datasets where *DCA-KM* is somehow 44 and 116 times faster than *IP-WF-DCA*. Except for *Ecoli*, *DCA-KM* is faster than *BI-WF-DCA*, the gain varies from 1.2 times to 9, 8 times.

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 *BI-WF-DCA*, *IP-WF-DCA* and *WF-KM* ([47]). We also reported the PWCO, Rand Index and CPU Time of each algorithm.

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 *BI-WF-DCA* and 28, 27% with *IP-WF-DCA*. *BI-WF-DCA* is faster than *WF-KM* for 6 out of 11 datasets while *IP-WF-DCA* is the slowest algorithm.

8.4 Conclusion

We have studied two widely used models bilevel program MSSC and mixed integer program MSSC using weighted feature. Based on the reformulation technique and exact penalty in DC programming, two optimization models were recast as a DC program. 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 experiments, we can conclude that the introduction of weighted feature allows to improve the performance

TABLE 8.6 – Comparison of Rand index between *WF-KM*, *BI-WF-DCA* and *IP-WF-DCA*

Data	Rand Index		
	WF-KM	BI-WF-DCA	IP-WF-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 8.7 – Comparison of CPU time between *WF-KM*, *BI-WF-DCA* and *IP-WF-DCA*

Data	Running time		
	WF-KM	BI-WF-DCA	IP-WF-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

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.

Chapitre 9

Block Clustering[1]

Abstract: We address the Block clustering problem in the continuous framework, which traditionally requires solving a hard combinatorial optimization problem. DC reformulation techniques and exact penalty in DC programming are developed to build an appropriate equivalent DC program of the Block clustering problem. They lead to an elegant explicit DCA scheme for the resulting DC program. Computational experiments show the robustness and efficiency of the proposed algorithm and its superiority over standard algorithms such as Two-mode K-means, Two-mode Fuzzy Clustering, Block Classification EM.

9.1 Introduction

In this chapter, we are interested in the so-called block clustering, or bi-clustering or again two-mode clustering that consists in simultaneous clustering on the set of samples (objects) and on the set of their features in order to find homogeneous blocks. Bi-clustering has a great significance in different fields, in particular for biomedical applications. When bi-clustering is performed with high reliability, we are able not only to diagnose conditions represented by sample classes, but also to identify features (e.g., genes or proteins) responsible for them, or serving as their markers.

Let a data set of n objects (samples), each of them is composed of m features (variables). This data set is presented by a $n \times m$ real-valued rectangular matrix $X = (x_{ij})_{n\times m}$ where $i \in I := \{1, ...n\}$, the set of n rows, and $j \in J := \{1, ...m\}$, the set of m columns, and the value x_{ij} is the expression of j^{th} feature in i^{th} object. More precisely, each row of X refers to an object while each column of X corresponds to a feature.

A bi-clustering of a data set is a collection of pairs of object and feature subsets $B = \{(O_1, F_1), ... (O_k, F_l, ...)\}$ such that the collection $\{O_1, ..., O_k, ..., O_K\}$ forms a partition of the set of objects on K clusters (row clusters), and the collection $\{F_1, ..., F_l, ..., F_L\}$ forms a partition of the set of features on L clusters (column clusters). A pair (O_k, F_l) will be called a bi-cluster. In other words, bi-clustering (two-mode clustering) assigns each element of X to a row cluster and a column cluster. If $L = m$, two-mode clustering reduces to one-mode clustering (partitioning the set of rows), and the same is true if $K = n$ (partitioning the set of columns).

1. The results presented in this chapter were published in :

- H.M. Le, H.A. Le Thi, T. Pham Dinh, V.N. Huynh, Block Clustering based on DC programming and DCA, Neural Computation, 25(10) :2776-2807, 2013.

Depending on the definition of the distance measure, bi-clustering covers a large variety of problems. In this paper, we focus on bi-clustering using the squared Euclidean metric as the distance measure. The problem can then be stated as follows. Let

— $P_{n\times K} = (p_{ik})_{I\times K}$ be the binary matrix containing row cluster membership values (called row-classification matrix), i.e. $p_{ik} = 1$ if row i belongs to row cluster k, and $p_{ik} = 0$ otherwise ;

— $Q_{m\times L} = (q_{jl})_{J\times L}$ be the binary matrix containing column cluster membership values (called column-classification matrix), i.e. $q_{jl} = 1$ if column j belongs to column cluster l, and $q_{jl} = 0$ otherwise ;

— $V_{K\times L} = (v_{kl})_{K\times L}$ be the matrix representing prototype values for the entries in a block, i.e. x_{ij} is in the block with prototype v_{kl} when $p_{ik}q_{jl} = 1$.

We have to search for the block cluster prototype matrix V and the two cluster membership values matrices P, Q which minimize the sum of squared Euclidean distances from the entries to their respective prototype values.

Define the inner product $\langle X, Y\rangle$ in $\mathcal{M}_{n,m}(\mathbb{R})$ (the vector space of $n\times m$ real-valued matrices) as the trace of the matrix X^TY which is denoted by $Tr(X^TY)$, say

$$\langle X, Y\rangle_{\mathcal{M}_{n,m}(\mathbb{R})} = \sum_{i=1}^{n} X_i Y_i^T = \sum_{j=1}^{m} (X^j)^T Y^j = Tr(X^TY),$$

where X_i and X^k are, respectively ith row and kth column of the matrix X. Denote by $\|\cdot\|$ the corresponding Euclidean norm on $\mathcal{M}_{n,m}(\mathbb{R})$.Then the function to be minimized in bi-clustering is given by

$$f(P,Q,V) := \sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ik}q_{jl}(x_{ij}-v_{kl})^2 = \left\|X - PVQ^T\right\|^2. \tag{9.1}$$

As usual, the classification matrices must satisfy the following constraints :

— The cluster membership values of each row and column object must sum to one :

$$\sum_{k=1}^{K} p_{ik} = 1 \quad \forall i = 1,\dots,n, \ \sum_{l=1}^{L} q_{jl} = 1 \quad \forall j = 1,\dots,m.$$

— None of the row or column clusters is empty :

$$\sum_{i=1}^{n} p_{ik} \geq 1 \quad \forall k = 1,\dots,K, \ \sum_{j=1}^{m} q_{jl} \geq 1 \quad \forall l = 1,\dots,L.$$

— All cluster membership values must be either zero or one :

$$\begin{aligned} p_{ik} &\in \{0,1\} \quad \forall i = 1,\dots,n, k = 1,\dots,K, \\ q_{jl} &\in \{0,1\} \quad \forall j = 1,\dots,m, l = 1,\dots,L. \end{aligned}$$

Finally, the mathematical block clustering problem can be written as :

$$\begin{cases} \min_{(P,Q,V)} f(P,Q,V) = \left\|X - PVQ^T\right\|^2 \\ s.t \\ \quad \sum_{k=1}^{K} p_{ik} = 1; \sum_{l=1}^{L} q_{jl} = 1; \quad \forall i = 1,\ldots,n;\ j = 1,\ldots,m \\ \quad \sum_{i=1}^{n} p_{ik} \geq 1; \sum_{j=1}^{m} q_{jl} \geq 1; \quad \forall k = 1,\ldots,K;\ l = 1,\ldots,L) \\ \quad P \in \{0,1\}^{n\times K}, \quad Q \in \{0,1\}^{m\times L}, \quad V \in I\!R^{K\times L}. \end{cases} \tag{9.2}$$

Problem (9.2) is a nonconvex nonlinear mixed 0-1 program, known to be NP-hard and for which no efficient global algorithm is available. Even for small n and m, the number of possible partitions can become extremely large. There are several optimization methods for two-mode clustering. All of these algorithms are heuristics (often based on meta-heuristic, multistart procedures) that are not guaranteed to find the global optimum and often get stuck in local minima. An extensive overview of two-mode clustering methods can be found in [165].

In [207], the authors have given an overview of existing optimization methods based on meta-heuristics and introduced new Fuzzy algorithms (Two-mode Fuzzy) for solving Problem (9.2). A simulation study has been performed to compare these methods and to determine how effective they are at finding the optimal clustering. It turns out that the application of the Multistart heuristic in combination with the K-means algorithm usually has the best average performance.

Our contributions. We aim is to develop a new *efficient* and *scalable* DCA based algorithm for block clustering problem (9.2). The investigation of DC programming and DCA to the hard optimization block clustering problem (9.2) requires a rigorous study for reformulating it in terms of a DC program. Using an interesting exact penalty result developed in [145] we first transform (9.2) into an equivalent continuous optimization problem. Afterwards we carefully study DC reformulation techniques for the continuous optimization problem to get finally a nice DC program. A simple and elegant DCA scheme for solving the resulting DC program is then developed. The proposed DCA scheme is original and very inexpensive because it amounts to computing, at each iteration, the projection of points onto a simplex and/or onto a box, that are all determined in the explicit form. Numerical experiments on several data sets have shown the efficiency of the proposed method and its superiority over the Two-mode K-means, Two-mode Fuzzy (two best algorithms presented in [207]) and Block Classification EM (Expection Maximization) called BCEM [86] see below in Section 4.

The rest of the chapter is organized as follows. In Section 9.2, we introduce a continuous formulation of problem (9.2) by using an exact penalty result in DC Programming. The Section 9.3 is devoted to DC programming and DCA for solving the block clustering problem in the continuous optimization form. Comparative experimental results with Two-mode K-Means, Two-mode Fuzzy [207] and BCEM [86] are reported in Section 9.4.

9.2 A continuous formulation of block clustering problem

First, let us show that the variable V in Problem (9.2) can be bounded in a box. Let (P^*, Q^*, V^*) be a solution of Problem (9.2). For every k, l there exist $i,\ j$ such that $p_{ik} = q_{jl} = 1$. Thus

$$(x_{ij} - v_{kl})^2 \leq f(P^*, Q^*, V^*) \leq \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}^2.$$

Hence

$$x_{ij} - a \leq v_{kl} \leq x_{ij} + a, \quad \text{where } a^2 := \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}^2.$$

Denote by $\alpha = \min\limits_{i=1,...,n;j=1,...,m}\{x_{ij} - a\}$, $\beta = \max\limits_{i=1,...,n;j=1,...,m}\{x_{ij} + a\}$, then $v_{kl} \in [\alpha, \beta],\ \forall\ k, l$. By this observation, it suffices to consider (9.2) with $V \in \mathcal{V} = [\alpha, \beta]^{K\times L}$.
Set

$$\mathcal{P} = \Big\{P = (p_{ik})_{I\times K} : \sum_{k=1}^{K} p_{ik} = 1;\ p_{ik} \in \{0,1\}, i = 1,...n, k = 1,...K\ ; \sum_{i=1}^{n} p_{ik} \geq 1\Big\},$$

$$\mathcal{Q} = \Big\{Q = (q_{jl})_{J\times L} : \sum_{l=1}^{L} q_{jl} = 1;\ q_{jl} \in \{0,1\}, j = 1,...m, l = 1,...L\ ; \sum_{j=1}^{m} q_{jl} \geq 1\Big\}.$$

Problem (9.2) is equivalent to

$$\min\{f(P,Q,V)\ :\quad (P,Q,V) \in \mathcal{P}\times\mathcal{Q}\times\mathcal{V}\}. \tag{9.3}$$

Lemma 9.1 *The function $f(\cdot,\cdot,V)$ is uniformly Lipschitz on $\mathcal{P}\times\mathcal{Q}$ for $V \in \mathcal{V}$ with uniform Lipschitz constant $\mathcal{L}$ given by*

$$\mathcal{L} = \sqrt{2}\max\{\alpha^2, \beta^2\}\max\{nK, mL\}. \tag{9.4}$$

Proof 9.1 *For $x, y, z;\ x', y' \in R$, by the Cauchy -Schwarz inequality, we have the following bound*

$$xyz^2 - x'y'z^2 = z^2y(x - x') + z^2x'(y - y') \leq z^2\sqrt{y^2 + x'^2}\sqrt{(x - x')^2 + (y - y')^2}.$$

So by taking $z = x_{ij} - v_{kl}$, $x = p_{ik}$, $x' = p'_{ik}$, $y = q_{jl}$, $y' = q'_{jl}$ into account, it follows directly that $f(\cdot,\cdot,V)$ is uniformly Lipschitz on $\mathcal{P}\times\mathcal{Q}$ for $V \in \mathcal{V}$ with uniformly Lipschitzian constant

$$\mathcal{L} = \sqrt{2}\max\{\alpha^2, \beta^2\}\max\{nK, mL\}. \tag{9.5}$$

■

For reformulating (9.2) in a continuous optimization form, first let us recall the following exact penalty result. For a point $x \in \mathbb{R}^n$ and a set $S \subset \mathbb{R}^n$, denote by $d(x, S)$ the distance from x to S, i.e.,

$$d(x, S) := \inf_{z\in S} \|x - z\|).$$

We have

Lemma 9.2 *([145]) Suppose that f is Lipschitz on $\mathcal{X} \subset \mathbb{R}^n$ with constant $\mathcal{L}_0$. Let $\varphi : \mathcal{X} \to \mathbb{R}^n \cup \{+\infty\}$ be a nonnegative function defined on $\mathcal{X}$ and*

$$S := \{x \in \mathcal{X} :\ \varphi(x) = 0\}.$$

If $d(x, S) \leq \varphi(x)$ for all $x \in \mathcal{X}$, then for all $\mathcal{L} > \mathcal{L}_0$, the two problems

$$\inf\{f(x) :\ x \in S\} \quad \text{and } \inf\{f(x) + \mathcal{L}\varphi(x) :\ x \in \mathcal{X}\}$$

are equivalent insofar as they have the same optimal value and the same solution set.

Denote by

$$S_p = \Big\{P = (p_{ik})_{I\times K} : \sum_{k=1}^{K} p_{ik} = 1;\ p_{ik} \in [0,1], i = 1,..,n;\ k = 1,..,K\Big\};$$

$$S_q = \Big\{Q = (q_{jl})_{J\times L} : \sum_{l=1}^{L} q_{jl} = 1;\ q_{jl} \in [0,1],\ j = 1,..,m;\ l = 1,..,L\Big\};$$

$$S_p^1 = \Big\{P \in S_p : p_{ik} \in \{0,1\},\ i = 1,..,n;\ k = 1,..,K\Big\};$$

$$S_q^1 = \Big\{Q \in S_q : q_{jl} \in \{0,1\},\ j = 1,..,m;\ l = 1,..,L\Big\}.$$

For using Lemma 9.2 we find an upper bound for $d((P,Q), \mathcal{P}\times\mathcal{Q})$ with $(P,Q) \in S_p \times S_q$.

Step 1. *Estimation Bounds for* $d((P,Q), S_p^1 \times S_q^1)$ *with* $(P,Q) \in S_p \times S_q$.

The following lemma will be needed.

Lemma 9.3 *Let*

$$K := \{x \in \mathbb{R}^n :\ \sum_{i=1}^{n} x_i = 1,\ x_i \in [0,1], i = 1,...n\},$$

and

$$C := \{x \in \mathbb{R}^n :\ \sum_{i=1}^{n} x_i = 1,\ x_i \in \{0,1\}, i = 1,...n\}.$$

Then for all $x = (x_1, \cdots, x_n) \in K$, *one has*

$$d(x,C) \le 2(1 - \max_{1\le i\le n}\{x_i\}).$$

Proof 9.2 *For* $x = (x_1, ..., x_n) \in K$, *one has*

$$\begin{aligned} d^2(x,C) &= \min\left\{\sum_{i=1}^{n}(x_i - y_i)^2 :\ y = (y_1,...,y_n) \in C\right\} \\ &= \min\left\{\sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i + 1 :\ y = (y_1,...,y_n) \in C\right\} \\ &= \sum_{i=1}^{n} x_i^2 + 1 - 2\max_{1\le i\le n}\{x_i\}. \end{aligned}$$

Let $r \in \{1, \cdots, n\}$ *be such that* $\max_{1\le i\le n}\{x_i\} = x_r$. *Then*

$$\begin{aligned} d^2(x,C) &= \sum_{i\ne r} x_i^2 + (1 - x_r)^2 \le \Big(\sum_{i\ne r} x_i + 1 - x_r\Big)^2 \\ &= 4(1 - x_r)^2 = 4(1 - \max_{1\le i\le n}\{x_i\})^2. \end{aligned}$$

The proof is complete. ■

Remark 9.1 *Let* $x = (x_1, ..., x_n) \in K$ *and* $\varphi(x) = 2(1 - \max_{1\le i\le n}\{x_i\})$. *Then*

$$d(x,C) \le \varphi(x)\ \forall x \in K \text{ and } x \in C \quad \textit{iff} \quad \varphi(x) = 0.$$

From this lemma, one obtains the following bound

$$d((P,Q),S_p^1\times S_q^1)\le 2\Big(\sum_{i=1}^{n}(1-\max_{1\le k\le K}\{p_{ik}\})+\sum_{j=1}^{m}(1-\max_{1\le l\le L}\{p_{jl}\})\Big) \tag{9.6}$$

for all $(P,Q)\in S_p\times S_q$.

Step 2. *Bound for $d((P,Q),\mathcal{P}\times\mathcal{Q})$ with $(P,Q)\in S_p^1\times S_q^1$.*

Let $(P,Q)\in(S_p^1\times S_q^1)\setminus(\mathcal{P}\times\mathcal{Q})$. It is easy to see that

$$d(P,\mathcal{P})\le\sqrt{2n};\quad d(Q,\mathcal{Q})\le\sqrt{2m}.$$

On the other hand, since $(P,Q)\notin\mathcal{P}\times\mathcal{Q}$, there exists indices k_0, l_0 such that

$$\sum_{i=1}^{n}p_{ik_0}=0\quad\text{and}\quad\sum_{j=1}^{m}p_{jl_0}=0.$$

Therefore,

$$d((P,Q),\mathcal{P}\times\mathcal{Q})\le\sqrt{2n}\sum_{k=1}^{K}\max\{0,1-\sum_{i=1}^{n}p_{ik}\}+\sqrt{2m}\sum_{l=1}^{L}\max\{0,1-\sum_{j=1}^{n}q_{jl}\}. \tag{9.7}$$

By using the relation $\max\{0,x\}=|x|-\max\{0,-x\}$ and observing that $1-\sum_{i=1}^{n}p_{ik}$, $1-\sum_{j=1}^{m}q_{jl}$ are integer for $(P,Q)\in\mathcal{P}\times\mathcal{Q}$, one has the following inequalities :

$$\begin{aligned}\max\Big\{0,1-\sum_{i=1}^{n}p_{ik}\Big\}&=\Big|1-\sum_{i=1}^{n}p_{ik}\Big|-\max\Big\{0,\sum_{i=1}^{n}p_{ik}-1\Big\}\\&\le\Big(\sum_{i=1}^{n}p_{ik}-1\Big)^2-\Big|1-\sum_{i=1}^{n}p_{ik}\Big|\max\Big\{0,\sum_{i=1}^{n}p_{ik}-1\Big\}\\&\le\Big(\sum_{i=1}^{n}p_{ik}-1\Big)^2-\max\Big\{0,\sum_{i=1}^{n}p_{ik}-1\Big\}^2.\end{aligned} \tag{9.8}$$

Similarly, one has

$$\max\Big\{0,1-\sum_{j=1}^{n}q_{jl}\Big\}\le\Big(\sum_{j=1}^{n}q_{jl}-1\Big)^2-\max\Big\{0,\sum_{j=1}^{n}q_{jl}-1\Big\}^2. \tag{9.9}$$

Combining the preceding inequalities, one obtains

$$\begin{aligned}d((P,Q),\mathcal{P}\times\mathcal{Q})\le\sqrt{2n}\Big(\sum_{k=1}^{K}\big(\sum_{i=1}^{n}p_{ik}-1\big)^2-\sum_{k=1}^{K}\max\Big\{0,\sum_{i=1}^{n}p_{ik}-1\Big\}^2\Big)+\\+\sqrt{2m}\Big(\sum_{l=1}^{L}\big(\sum_{j=1}^{m}q_{jl}-1\big)^2-\sum_{l=1}^{L}\max\{0,\sum_{j=1}^{m}q_{jl}-1\}^2\Big).\end{aligned} \tag{9.10}$$

Denote by

$$\varphi(P,Q):=2\Big(\sum_{i=1}^{n}(1-\max_{1\le k\le K}\{p_{ik}\})+\sum_{j=1}^{m}(1-\max_{1\le l\le L}\{p_{jl}\})\Big), \tag{9.11}$$

and

$$\psi(P,Q) := \sqrt{2n}\Big(\sum_{k=1}^{K}\big(\sum_{i=1}^{n} p_{ik}-1\big)^2 - \sum_{k=1}^{K}\max\{0,\sum_{i=1}^{n} p_{ik}-1\}^2\Big)+ \\ + \sqrt{2m}\Big(\sum_{l=1}^{L}\big(\sum_{j=1}^{m} q_{jl}-1\big)^2 - \sum_{l=1}^{L}\max\{0,\sum_{j=1}^{m} q_{jl}-1\}^2\Big). \tag{9.12}$$

Obviously, φ, ψ are nonnegative on $S_p \times S_q$ and

$$S_p^1 \times S_q^1 = \{x \in S_p \times S_q : \varphi(P,Q) = 0\};$$

$$\mathcal{P} \times \mathcal{Q} = \{x \in S_p^1 \times S_q^1 : \psi(P,Q) = 0\}.$$

The following lemma gives an upper bound for $d((P,Q), \mathcal{P} \times \mathcal{Q})$ in terms of the function φ and ψ.

Lemma 9.4 *For all $(P,Q) \in S_P \times S_q$, one has*

$$d((P,Q), \mathcal{P} \times \mathcal{Q}) \leq \tau\varphi(P,Q) + \psi(P,Q), \tag{9.13}$$

where $\tau = t + 1 = \max\{4n\sqrt{2n}, 4m\sqrt{2m}\} + 1$.

Proof 9.3 *Observe that ψ is Lipschitz on $S_p \times S_q$ with constant t. Let $(P,Q) \in S_p \times S_q$. Let $(P_1,Q_1) \in S_p^1 \times S_q^1$ be the projection of (P,Q) into $S_p^1 \times S_q^1$. One has*

$$\begin{aligned} d((P,Q), \mathcal{P} \times \mathcal{Q}) &\leq d((P,Q), S_p^1 \times S_q^1) + d((P_1,Q_1), \mathcal{P} \times \mathcal{Q}) \\ &\leq d((P,Q), S_p^1 \times S_q^1) + \psi(P_1,Q_1) \\ &\leq (t+1)d((P,Q), S_p^1 \times S_q^1) + \psi(P,Q) \leq \tau\varphi(P,Q) + \psi(P,Q). \end{aligned}$$

The lemma is then proved. ■

From Lemma 9.2 and Lemma 9.4, we derive that the problem (9.2) is equivalent to the following continuous optimization program :

$$\min\{F(P,Q,V) := f(P,Q,V) + \gamma\tau\varphi(P,Q) + \gamma\psi(P,Q) : (P,Q,V) \in S_p \times S_q \times \mathcal{V}\}, \tag{9.14}$$

for all $\gamma > \mathcal{L}$, where $\mathcal{L}$ is defined by (9.5). In the next section, we will consider the block clustering problem in the form (9.14) with a sufficiently large value γ, say $\gamma > \sqrt{2}\max\{\alpha^2, \beta^2\}\max\{nK, mL\}$.

9.3 DCA for solving block clustering problem

We now develop DCA for solving the equivalent block clustering problem (9.14).

9.3.1 DC formulation of problem (9.14)

By directly checking, for

$$C := \max\{\alpha^2 + 2\alpha, \beta^2 + 2\beta, 4\alpha + 2, 4\beta + 2\}, \tag{9.15}$$

the function

$$(x, y, z) \in \mathbb{R}^3 \mapsto \frac{C}{2}(x^2 + y^2 + z^2) - xyz^2$$

is convex on $[0, 1] \times [0, 1] \times [\alpha, \beta]$. Therefore, for

$$\rho_{1,P} \geq CmL, \quad \rho_{1,Q} \geq CnK, \quad \rho_V \geq Cmn, \tag{9.16}$$

the function

$$\frac{\rho_{1,P}}{2}\|P\|^2 + \frac{\rho_{1,Q}}{2}\|Q\|^2 + \frac{\rho_V}{2}\|V\|^2 - f(P, Q, V) \quad \text{is convex on} \quad S_p \times S_q \times \mathcal{V}.$$

Likewise, for $\rho_2 \geq 4\sqrt{2n}\gamma$ and $\rho_3 \geq 4\sqrt{2m}\gamma$, the following functions are convex :

$$\rho_2/2\|P\|^2 - \sqrt{2n}\gamma \sum_{k=1}^{K} \big(\sum_{i=1}^{n} p_{ik} - 1\big)^2;$$

$$\rho_3/2\|Q\|^2 - \sqrt{2m}\gamma \sum_{l=1}^{L} \big(\sum_{j=1}^{m} q_{jl} - 1\big)^2.$$

Hence, we can recast Problem (9.14) as a DC program as follows.

$$\min\{F(P, Q, V) = G(P, Q, V) - H(P, Q, V) : (P, Q, V) \in S_p \times S_Q \times \mathcal{V}\}, \tag{9.17}$$

where,

$$G(P, Q, V) = \rho_P/2\|P\|^2 + \rho_Q/2\|Q\|^2 + \rho_V/2\|V\|^2;$$

$$\rho_P \geq CmL + 4\sqrt{2n}\gamma; \quad \rho_Q \geq CnK + 4\sqrt{2m}\gamma; \quad \rho_V \geq Cmn; \tag{9.18}$$

$$\begin{aligned} H(P, Q, V) = {} & \rho_P/2\|P\|^2 + \rho_Q/2\|Q\|^2 + \rho_V/2\|V\|^2 - f(P, Q, V) - \\ & - \sqrt{2n}\gamma \sum_{k=1}^{K} \big(\sum_{i=1}^{n} p_{ik} - 1\big)^2 - \sqrt{2m}\gamma \sum_{l=1}^{L} \big(\sum_{j=1}^{m} q_{jl} - 1\big)^2 + \\ & + \sqrt{2n}\gamma \sum_{k=1}^{K} \max\{0, \sum_{i=1}^{n} p_{ik} - 1\}^2 + \sqrt{2m}\gamma \sum_{l=1}^{L} \max\{0, \sum_{j=1}^{m} q_{jl} - 1\}^2 + \\ & + 2\gamma\tau \sum_{i=1}^{n} \max_{1 \leq k \leq K} \{p_{ik}\} + 2\gamma\tau \sum_{j=1}^{m} \max_{1 \leq l \leq L} \{q_{jl}\}. \end{aligned}$$

9.3.2 DCA for solving DC program (9.17)

According to the general DCA scheme described in Subsection 3.1, applying DCA to (9.17) amounts to computing two sequences $\{(Y^r, Z^r, W^r)\}$ and $\{(P^r, Q^r, V^r)\}$ in the way that$(Y^r, Z^r, W^r) \in \partial H(P^r, Q^r, V^r)$ and $(P^{r+1}, Q^{r+1}, V^{r+1})$ solves the convex program of the form (P_r).

Computing $\partial H(P, Q, V)$

For simplicity of notations, let us set

$$P_i = (p_{ik})_{1\leq k\leq K},\ Q_j = (q_{jl})_{1\leq l\leq L},\ i = 1, \cdots, n,\ j = 1, \cdots, m;$$
$$P = (P_1, P_2, \cdots, P_n)^T,\ Q = (Q_1, Q_2, \cdots, Q_m)^T;$$
$$h_{ip}(P_i) := \max_{1\leq k\leq K}\{p_{ik}\};\ K_{ip} = \big\{k \in \{1, \cdots K\} : p_{ik} = \max_{1\leq k\leq K}\{p_{ik}\}\big\};$$
$$h_{jq}(Q_j) := \max_{1\leq l\leq L}\{q_{jl}\};\ L_{jp} = \big\{l \in \{1, \cdots L\} : q_{jl} = \max_{1\leq l\leq L}\{q_{jl}\}\big\}.$$

Then,

$$\partial h_{ip}(P_i) = \{P_i' = (p_{ik}') : p_{ik}' \geq 0;\ p_{ik}' = 0 \text{ if } k \notin K_{ip},\ \sum_{k=1}^{K} p_{ik}' = 1\}.$$

Similarly,

$$\partial h_{jq}(Q_j) = \{Q_j' = (q_{jl}') : q_{jl}' \geq 0;\ q_{jl}' = 0 \text{ if } l \notin L_{jq},\ \sum_{l=1}^{L} q_{jl}' = 1\}.$$

Hence $(Y^r, Z^r, W^r) \in \partial H(P^r, Q^r, V^r)$ can be determined as :

$$Y^r = (Y_i^r), Z^r = (Z_j^r), W^r = (W_{kl}^r),$$

where

$$W_{kl}^r = \rho_V v_{kl}^r - 2\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ik}^r q_{jl}^r (v_{kl}^r - x_{ij}); \tag{9.19}$$

$$\begin{aligned} Y_i^r &= \rho_P P_i^r - [\nabla_P f(P^r, Q^r, V^r)]_i - 2\sqrt{2n}\gamma\big(\textstyle\sum_{i=1}^{n}(p_{ik}^r - 1)\big)_{1\leq k\leq K} \\ &\quad +2\sqrt{2n}\gamma\big(\max\{0, \textstyle\sum_{i=1}^{n} p_{ik}^r - 1\}\big)_{1\leq k\leq K} + 2\gamma\tau\partial h_{ip}(P_i^r); \end{aligned} \tag{9.20}$$

$$\begin{aligned} Z_j^r &= \rho_Q Q_j^r - [\nabla_Q f(P^r, Q^r, V^r)]_j - 2\sqrt{2m}\gamma\big(\textstyle\sum_{j=1}^{m}(q_{jl}^r - 1)\big)_{1\leq l\leq L} \\ &\quad +2\sqrt{2m}\gamma\big(\max\{0, \textstyle\sum_{j=1}^{m} q_{jl}^r - 1\}\big)_{1\leq l\leq L} + 2\gamma\tau\partial h_{jq}(Q_j^r). \end{aligned} \tag{9.21}$$

Computing $(P^{r+1}, Q^{r+1}, V^{r+1})$

This is equivalent to solving the convex program of the form (P_r), say

$$\min\{G(P, Q, V) - \langle (P, Q, V), (Y^r, Z^r, W^r)\rangle : (P, Q, V) \in S_p \times S_q \times \mathcal{V}\}.$$

It is easy to see that ($Proj_C$ denotes the projection on the set C)

$$\begin{aligned} &P_i^{r+1} = Proj_{\Delta_K}(Y_i^r/\rho_P); Q_j^{r+1} = Proj_{\Delta_L}(Z_j^r/\rho_Q); \\ &V^{r+1} = Proj_C(W^r/\rho_V), \end{aligned}$$

where, Δ_K, Δ_L, are the $(K-1)-$simplex, the $(L-1)-$simplex defined as

$$\Delta_K = \{x \in \mathbb{R}^K : \sum_{k=1}^{K} x_k = 1,\ x_k \in [0,1]\}; \Delta_L = \{x \in \mathbb{R}^L : \sum_{l=1}^{L} x_l = 1,\ x_l \in [0,1]\},$$

and $C = [\alpha, \beta]^{K\times L}$. Since the projections on a simplex and/or on a box are explicitly determined, the computation of the sequence $(P^{r+1}, Q^{r+1}, V^{r+1})$ is explicit too. The proposed DCA applied to the DC program (9.17) then is explicit and, consequently, inexpensive. It can be summarized in Algorithm 9.1.

Algorithm 9.1 ADCA for solving the standard DC program

1: **Initialization :** Choose the two matrices of cluster memberships (P^0, Q^0) and the cluster centers V^0. $r \leftarrow 0$.
2: **repeat**
3: Set (Y^r, Z^r, W^r) according to (9.19), (9.20), (9.21).
4: Set
$$P_i^{r+1} = Proj_{\Delta_K}(Y_i^r/\rho_P); Q_j^{r+1} = Proj_{\Delta_L}(Z_j^r/\rho_Q);$$
$$V^{r+1} = Proj_C(W^r/\rho_V),$$
5: $r \leftarrow r + 1$.
6: **until** Stopping criterion.

9.4 Numerical experiments

9.4.1 Experiment setting

To our knowledge, there is no algorithm for globally solving the problem (9.2). Since the approaches developed in [207] considered the same clustering criterion (the squared error (9.1)), we compare our algorithm DCA with Two-mode K-means and Two-mode Fuzzy Clustering, the two best algorithms studied in [207]. On the other hand , we also compare DCA with the BCEM algorithm [86] (an EM based algorithm for block clustering) which is an efficient standard method for block clustering. BCEM is based on probabilistic mixture models but it has been shown in several works (see e.g. [45, 46, 86]) that some of the most popular heuristic clustering approaches including K-mean can be viewed as approximate estimations of probability models.

Data

To illustrate the performances of algorithms, we performed numerical tests on 3 types of data : real data sets, simulated continuous data sets (Data1-Data12) and simulated binary data sets (Data13-Data17). The real data sets are available online `http://algorithmics.molgen.mpg.de/Static/Supplements/CompCancer/datasets.htm`. Thereby only the number of row clusters (K) is assumed to be known. We give below a brief description of real datasets :

— "Brain cancer 1" contains 50 gliomas : 28 glioblastomas and 22 anaplastic oligodendrogliomas. This dataset was used to build a two-class prediction of malignant gliomas [179].

— "Brain cancer 2" is a dataset containing 42 patient samples (10 medulloblastomas, 5 CNS AT/RTs, 5 renal and extrarenal rhabdoid tumours, and 8 supratentorial PNETs, as well as 10 non-embryonal brain tumours (malignant glioma) and 4 normal human cerebella). In [196], the authors used this dataset for the problem of distinguishing different central nervous system embryonal tumour from each other.

— "Multi Tissue" contains 190 tumor samples, spanning 14 common tumor types. This dataset was used in [200], in order to determine whether the diagnosis of multiple common adult malignancies could be achieved purely by molecular classification.

— "Lung" consists of 186 lung tumor samples and 17 normal lung tissues (NL). The lung tumors included 139 adenocarcinoma (AD), 6 small-cell lung cancer (SCLC), 20 pulmonary carcinoids (COID) and 21 squamous cell lung carcinomas (SQ). In [26], the authors used this dataset for studying a new molecular taxonomy of tumors and demonstrate the potential power of gene expression profiling in lung cancer diagnosis.

TABLE 9.1 – Simulated continuous datasets.

Name	Size					Name	Size				
	n	m	K	L	E		n	m	K	L	E
Data1	20	20	5	4	0.5	Data2	20	20	5	4	1
Data3	20	20	5	4	2	Data4	60	60	6	6	0.5
Data5	60	60	6	6	1	Data6	60	60	6	6	2
Data7	100	100	10	10	0.5	Data8	100	100	10	10	1
Data9	100	100	10	10	2	Data10	150	100	7	7	0.5
Data11	150	100	7	7	1	Data12	150	100	7	7	2

We use the same way as proposed in [207] to generate continuous datasets (Data1-Data12). For each of four data dimensions (n, m, K, L) $((20, 20, 5, 4)$, $(60, 60, 6, 6)$, $(100, 100, 10, 10)$, $(150, 100, 7, 7))$, the procedure for generating continuous datasets is described as follows :

1. Randomly generate the row-classification matrix $P^* \in \{0, 1\}^{n \times K}$.
2. Randomly generate the column-classification matrix $Q^* \in \{0, 1\}^{m \times L}$.
3. Generate the matrix of prototypes $V^* \in I\!R^{K \times L}$ according to $K \times L$ independent normal distribution with mean 0 and standard deviation equal to 1.
4. Generate 3 noise matrices $E \in I\!R^{n \times m}$ according to $n \times m$ independent normal distribution with mean 0 and standard deviation equal to 0.5, 1 and 2. Note that standard deviations of 0.5 and 1 give a reasonable amount of noise in the simulated data, whereas a standard deviation of 2 can make clustering difficult.
5. Finally, the data matrix X is computed by $X = P^* V^* Q^{*T} + E$.

The information on these data sets is summarized in Table 9.1.

For generating binary datasets (Data13-Data17), we use a procedure that is similar to the one given in [86] and can be described as follows.

1. Generate the row-classification matrix $P^* \in \{0, 1\}^{n \times K}$ according to the mixing proportions $\pi_1 = \pi_2 = \ldots = \pi_K = 1/K$.
2. Generate the column-classification matrix $Q^* \in \{0, 1\}^{m \times L}$ according to the mixing proportions $\eta_1 = \eta_2 = \ldots = \eta_L = 1/L$.
3. Randomly generate the matrix of prototypes $V^* \in \{0, 1\}^{K \times L}$.
4. Randomly generate $\epsilon_{kl} \in [0, \frac{1}{2}]$ for $k = 1, \ldots, K, l = 1, \ldots, L$.
5. Compute $\alpha_{kl} = \epsilon_{kl}$ if $\epsilon_{kl} \in [0, 1/2]$ and $\alpha_{kl} = 1 - \epsilon_{kl}$ if $\epsilon_{kl} \in [1/2, 1]$, for $k = 1, \ldots, K, l = 1, \ldots, L$.
6. Generate x_{ij} for $i = 1, \ldots, n$, $j = 1, \ldots, m$ according to the Bernoulli distribution parameterized by α_{kl}, where (k, l) is the bi-cluster containing x_{ij}.

Set up experiments and Parameters

All clustering algorithms were implemented in the Visual C++ 2008, and performed on a PC Intel i5 CPU650, 3.2 GHz of 4GB RAM. We stop all algorithms with the tolerance $\epsilon = 10^{-4}$.

In [207], the authors stated that if the initial value of s is too high, the Two-mode Fuzzy algorithm often reaches a saddle point. To prevent from this drawback, in our experiments, we set the fuzzy step size γ equal to 0.9 and the initial value of s equal to 1.2. The threshold value s_{min} is set to 1.01.

Our experiments are composed of two parts : on continuous data and on binary data. In the first experiment, we compare the performance of DCA, Two-mode K-Means and Two-mode Fuzzy on the simulated continuous data sets containing 12 problems (Data1-Data12) and on 4 real data sets.

The following criteria were used to compare the performance of three algorithms DCA, Two-mode K-means and Two-mode Fuzzy : "VAF" (Variance Accounted For criterion), the error rate (in comparing with true clusters in case of simulated data where true clusters are known) and the CPU time in seconds.

The "VAF" is defined as

$$VAF = 1 - \frac{\sum\limits_{k=1}^{K}\sum\limits_{l=1}^{L}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m} p_{ik}q_{jl}(x_{ij} - v_{kl})^2}{\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}(x_{ij} - \bar{x})^2} = 1 - \frac{f(P,Q,V)}{\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}(x_{ij} - \bar{x})^2},$$

where $\bar{x} = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m} x_{ij}$. It is clear that maximizing VAF corresponds to minimizing $f(P,Q,V)$.

The error rate of the partitions (P,Q) given by an algorithm in comparing with true partitions (P^*,Q^*) in block clustering is defined from one-way clustering as shown in [87] :

$$e\big((P,Q),(P^*,Q^*)\big) = e(P,P^*) + e(Q,Q^*) - e(P,P^*)e(Q,Q^*),$$

$$\text{where} \quad e(P,P^*) = 1 - \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} p_{ik}{p^*}_{ik},\ e(Q,Q^*) = 1 - \frac{1}{m}\sum_{j=1}^{m}\sum_{l=1}^{L} q_{jl}{q^*}_{jl}.$$

For real data sets, we only know the true row clusters but the true column clusters are not available. Hence, the evaluation is done by the VAF criterion. For each real data set, we performed the tests with five values of L : $L \in \{5, 10, 15, 20, 25\}$.

In the second experiment, the BCEM algorithm is added. Here we compare the error rate and CPU times in seconds of the four algorithms on binary data.

For every data instance, we perform each algorithm 20 times from 20 random starting points and report the best, the mean and the standard deviation of each criterion. Bold values in result tables are best value for each data instance.

9.4.2 Experimental results and comments

Experiment 1 : continuous data

The computational results on simulated data sets (resp. real data sets) are reported in Table 9.2 (resp. Table 9.3).

From Table 9.2 we observe that in all experiments, among three algorithms, DCA is the best in terms of VAF and Error rate. The results of DCA are quite stable with different starting points while Two-mode K-means and Two-mode Fuzzy are sensitive to them. Moreover, not surprisingly, the larger the standard deviation of the noise matrix E is, the more difficult problem is, and then the larger error rate given by each algorithm would be.

Similarly, for real data sets DCA is always the best in terms of VAF criterion. For each algorithm and in all real data sets, the maximal value of VAF when L varies corresponds to the case $L = 10$. In fact, the VAF value increases at first, reaches the best value at $L = 10$ and then decreases. According to this phenomenon, we can conclude that $L = 10$ is the optimal number of column clusters in these data sets.

Concerning the computation time, DCA is faster than Two-mode Fuzzy and slightly slower than Two-mode K-means : the average CPU time of Two-mode K-means, DCA, Two-mode Fuzzy is, respectively, 1.3, 2.7, 9.9 seconds in simulated data sets and 12.5, 12.8, 14 seconds in real data sets.

Experiment 2 : binary data

The computational results (error rate and CPU Time) are reported in Table 9.4.

We observe from Table 9.4 that DCA is the best in terms of error rate : DCA is better than Two-mode K-means and Two-mode Fuzzy for all 5 data instances, DCA is better than BCEM on 3/5 data sets and on the 2 other ones DCA and BCEM give the same results. The average CPU time of Two-mode K-means, Two-mode Fuzzy, BCEM and DCA is, respectively, 2.28, 3.02, 2.66 and 2.68 seconds.

9.5 Conclusion

We have rigorously studied the DC programming and DCA for bi-clustering. Based on an interesting result of exact penalty, the hard combinatorial optimization model of bi-clustering has been recast as a DC program in its elegant matrix formulation. A very nice DC decomposition is proposed of which the corresponding DCA is simple, elegant and inexpensive : it consists in computing, at each iteration, the projection of points onto a simplex and/or a box, that all are given in the explicit form. Computational experiments show the efficiency and the superiority of DCA with respect to the Two-mode K-means, Two-mode Fuzzy clustering, and BCEM.

TABLE 9.2 – Comparative results on simulated continuous data sets.

Data	**Two-mode K-Means**					**Two-mode Fuzzy**					**DCA**				
Name	VAF		Error rate		Time	VAF		Error rate		Time	VAF		Error rate		Time
	Mean±SD	Best	Mean±SD	Best		Mean±SD	Best	Mean±SD	Best		Mean±SD	Best	Mean±SD	Best	
Data1	0.73±0.05	0.76	0.17±0.04	0.16	0.02	0.75±0.05	0.78	0.17±0.04	0.16	0.90	**0.80±0.02**	**0.81**	**0.08±0.02**	**0.07**	0.04
Data2	0.53±0.04	0.54	0.26±0.04	0.24	0.02	0.44±0.03	0.45	0.29±0.03	0.28	0.47	**0.54±0.02**	**0.55**	**0.18±0.02**	**0.17**	0.04
Data3	0.24±0.05	0.26	0.45±0.03	0.43	0.29	0.22±0.07	0.26	0.41±0.05	0.39	0.22	**0.27±0.02**	**0.28**	**0.31±0.04**	**0.30**	0.04
Data4	**0.75±0.04**	**0.77**	**0.9±0.03**	**0.07**	0.65	0.42±0.05	0.43	0.10±0.01	0.09	1.60	**0.75±0.02**	**0.76**	**0.09±0.03**	**0.07**	0.50
Data5	0.43±0.05	0.45	0.15±0.03	0.16	0.91	0.32±0.04	0.34	0.19±0.04	0.20	1.10	**0.41±0.02**	**0.42**	**0.12±0.02**	**0.11**	0.40
Data6	0.18±0.05	0.20	0.41±0.04	0.40	0.75	0.12±0.04	0.14	0.49±0.03	0.47	2.37	**0.19±0.02**	**0.20**	**0.34±0.02**	**0.33**	0.60
Data7	0.72±0.03	0.74	0.05±0.02	0.04	1.9	0.59±0.03	0.61	0.10±0.02	0.09	39	**0.77 ± 0.02**	**0.78**	**0.03 ± 0.01**	**0.02**	5.7
Data8	0.44±0.05	0.46	0.10±0.04	0.09	2	0.32±0.03	0.33	0.12±0.04	0.11	12	**0.47±0.02**	**0.48**	**0.08±0.02**	**0.07**	6.8
Data9	0.12±0.04	0.15	0.43±0.04	0.41	2	0.12±0.05	0.15	0.41±0.03	0.39	8	**0.13**±0.01	**0.13**	**0.31**±0.01	**0.30**	4.7
Data10	0.74±0.04	0.77	0.07±0.03	0.06	2	0.57±0.04	0.59	0.10±0.05	0.09	25	**0.80±0.02**	**0.81**	**0.03±0.01**	**0.02**	4.5
Data11	**0.50±0.03**	**0.52**	**0.12±0.03**	**0.10**	2	0.40±0.05	0.41	0.18±0.04	0.17	20	**0.51±0.02**	**0.52**	**0.12±0.03**	**0.10**	3.9
Data12	0.19±0.05	0.20	0.48±0.05	0.46	3	0.19±0.03	0.20	0.50±0.05	0.49	8	**0.21±0.02**	**0.22**	**0.41±0.03**	**0.40**	4.8

TABLE 9.3 – Comparative results on real data sets

Data					Two-mode K-means			Two-mode Fuzzy			DCA		
Name	Size				VAF		Time	VAF		Time	VAF		Time
	n	m	K	L	Mean±SD	Best		Mean±SD	Best		Mean±SD	Best	
Brain Cancer 1	50	1377	4	5	0.65±0.07	0.68	5	0.60±0.04	0.63	6	**0.71±0.04**	**0.73**	6
				10	0.75±0.08	0.77	6	0.64±0.07	0.66	8	**0.82±0.04**	**0.84**	6
				15	0.73±0.08	0.76	8	0.63±0.08	0.65	10	**0.81±0.04**	**0.82**	12
				20	0.66±0.05	0.68	13	0.60±0.08	0.62	18	**0.75±0.06**	**0.77**	15
Brain Cancer 2	42	1379	5	5	0.68±0.02	0.69	4	0.60±0.04	0.63	4	**0.76±0.03**	**0.77**	4
				10	0.77±0.04	0.79	4	0.65±0.04	0.67	4	**0.82±0.02**	**0.83**	5
				15	0.75±0.03	0.76	6	0.74±0.05	0.76	8	**0.84±0.03**	**0.85**	10
				20	0.70±0.04	0.72	14	0.68±0.06	0.70	12	**0.80±0.04**	**0.82**	15
Multi Tissue	190	1363	14	5	0.75±0.03	0.76	8	0.61±0.02	0.62	9	**0.82±0.03**	**0.83**	7
				10	0.75±0.04	0.76	9	0.67±0.03	0.69	13	**0.86±0.02**	**0.87**	7
				15	0.75±0.03	0.76	14	0.67±0.04	0.69	16	**0.81±0.02**	**0.82**	15
				20	0.71±0.04	0.72	24	0.61±0.03	0.62	21	**0.81±0.04**	**0.83**	22
Lung	203	1543	5	5	0.75±0.01	0.76	15	0.70±0.02	0.71	15	**0.76±0.03**	**0.77**	14
				10	0.67±0.02	0.68	19	0.72±0.02	0.73	22	**0.76±0.02**	**0.77**	14
				15	0.73±0.03	0.74	23	0.71±0.03	0.72	28	**0.84±0.02**	**0.85**	24
				20	0.77±0.03	0.79	28	0.71±0.03	0.72	30	**0.85±0.03**	**0.86**	29

TABLE 9.4 – Computational results on simulated binary data sets.

Data					Two-mode K-means			Two-mode Fuzzy			BCEM			DCA		
Name	Size				Error rate		Time	Error rate		Time	Error rate		Time	Error rate		Time
	n	m	K	L	Mean+SD	Best		Mean+SD	Best		Mean+SD	Best		Mean+SD	Best	
Data13	60	90	2	3	0.09±0.02	0.07	0.8	0.12±0.03	0.10	0.9	0.05±0.02	0.03	1	**0.04±0.01**	**0.03**	1.2
Data14	50	60	5	2	0.13±0.02	0.12	1.2	0.15±0.01	0.14	1.5	**0.10±0.01**	**0.09**	1.3	**0.10±0.01**	**0.09**	1.3
Data15	100	60	4	3	0.12±0.02	0.11	1.5	0.18±0.03	0.16	2.5	0.12±0.02	0.11	1.9	**0.09±0.03**	**0.08**	2.1
Data16	300	200	5	4	0.09±0.02	0.08	3.7	0.20±0.02	0.19	4.9	**0.07±0.02**	**0.05**	4.6	**0.07±0.02**	**0.05**	4.3
Data17	400	180	8	5	0.20±0.04	0.18	4.2	0.25±0.04	0.23	5.3	0.15±0.03	0.13	4.5	**0.12±0.02**	**0.11**	4.5

Chapitre 10

Gaussian Mixture Models clsutering with Sparse Regularization[1]

Abstract: We address three fundamental issues in Gaussian Mixture Models (GMM) clustering that are the model selection, the feature selection and the over-parameterization. Although all these three issues are linked together, most of existing works deal with them separately. For the first time, we present an unified optimization formulation that takes into account all three above-mentioned issues. It turns out that the corresponding optimization problem involves the minimization of ℓ_0-norm. We choose the non-convex approximation approach to deal with the l_0-norm and develop DCA-Like to tackle the resulting optimization problem. Exploiting the specific structure of the considered optimization problem, DCA-Like proposes a new and efficient way to approximation the DC objective function without knowing a DC decomposition. Furthermore, we propose a Two-Step DCA-Like in order to improve the performance of DCA-Like. Numerical experiments on several benchmark and synthetic datasets illustrate the efficiency of our algorithms comparing to existing EM methods.

10.1 Introduction

In this chapter, we address the Gaussian mixture model (GMM) clustering, the most popular model-based clustering model in the literature. GMM has been largely developed and successfully applied to several applications thanks to its interesting properties on both theoretical and computational aspect [32]. Moreover, it was shown that any continuous distribution can be approximated by a finite mixture of normal densities [164]. The GMM clustering problem can be described as follows. Given a data set $X = \{x_i \in \mathbb{R}^D : i \in \{1, \ldots, N\}\}$ of N data points that are assumed to be generated from a finite mixture of K probability distributions :

$$p(x_i \mid \theta_1, \ldots, \theta_K) = \sum_{k=1}^{K} \pi_k p\left(x_i \mid \theta_k\right), \tag{10.1}$$

where θ_k, π_k are respectively the parameter and the mixing proportion of the k-th probability distribution $p\left(\cdot \mid \theta_k\right)$, with $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$ for all $k \in \{1, \ldots, K\}$. In GMM, each component probability distribution is a Gaussian distribution with mean μ_k and inverse covariance

1. The results presented in this chapter were published/submitted in
   - V.A. Nguyen, H.A. Le Thi, H.M. Le, A DCA based algorithm for Feature Selection in Model-Based Clustering, Lecture Notes in Artificial Intelligence (LNAI) 12033, 404-415, 2020.
   - H.A. Le Thi, H.M. Le, V.A. Nguyen, DCA-Like for GMM Clsutering with Sparse Regularization, submitted.

matrix (precision matrix) W_k :

$$p\left(x_i \mid \theta_k = (\mu_k, W_k)\right) = \sqrt{\frac{\det\left(W_k\right)}{\left(2\pi\right)^D}} \exp\left(-\frac{1}{2}\left(x_i - \mu_k\right)^T W_k \left(x_i - \mu_k\right)\right). \tag{10.2}$$

The parameters θ_k and the mixing proportion π_k can be estimated by solving the following maximum log-likelihood problem :

$$\max_{\Theta \in \mathcal{D}} \left\{ L\left(\Theta\right) := \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k p\left(x_i \mid \theta_k\right)\right)\right\}, \tag{10.3}$$

where

$$\Theta = (\pi, \theta) = \left(\pi_1, ..., \pi_K, \mu_1, ..., \mu_K, W_1, ...W_K\right), \tag{10.4}$$

and

$$\mathcal{D} = \left\{ (\pi, \mu, W) : \sum_{k=1}^{K} \pi_k = 1,\ W_k \succ 0,\ \forall k \in \{1, ..., K\} \right\}. \tag{10.5}$$

The above optimization problem can be solved by the Expectation-Maximisation (EM) algorithm. Although the development of GMM has been active in last decades, there are still several open questions, especially in applications with high dimensional data.

We focus on three fundamental issues in GMM clustering : the choice of appropriate number of clusters, the over-parameterization and the selection of useful features. In many clustering applications, the number of clusters is not known a priori. If we choose a too high number of clusters is too large, the mixture model may over-fit the data while the mixture model may not be good enough to cover the data structure if the number of clusters is too low. Determining the optimal number of clusters is a very difficult issue since it is somehow subjective and depends on the clustering method. Furthermore, the second and the third above-mentioned issues often occur when working with high-dimensional data. On the one hand, it is well-known that when the number of dimensions D increases, the number of parameters of GMM increases with a fast rate, leading to a high computational cost and the over-fitting problem. On the other hand, high-dimensional data potentially contains redundant and non-informative features, which can badly affect the clustering result. Hence, it is important to choose and use only useful features. Note that the number of parameters in GMM is a quadratic function of D. Thus removing redundant and non-informative features also helps to reduce the over-parameterization problem. To the best of our knowledge, although several methods have been developed to address these above-mentioned issues, there is no existing work that considers together these three issues.

Existing works. For determining the number of clusters, also called the model selection problem in mixture model, most conventional methods are based on a model selection criterion. These methods consist of three steps : a) define an upper bound K_{max} of the number of clusters ; b) solve the maximum log-likelihood problem (10.3) to estimate the parameter $\hat{\Theta}_K$ of mixture model for all $K \in [1, \ldots, K_{max}]$ and then c) choose the best value of K with respect to a specific model selection criterion $\mathcal{P}(k)$, that is

$$K_{best} = arg\max_{K} \{L(\hat{\Theta}_K) - \mathcal{P}(k)\}$$

where $\mathcal{P}(k)$ is an increasing function penalizing higher value of K. Several model selection criteria have been proposed. The most used criteria are based on information theoretic such as the

Bayesian information criterion (BIC) [211] or Akaike information criterion (AIC) [5]. Another class of criteria is based on the distance between the fitted model and the nonparametric estimate of the population distribution, for instance Kullback-Leibler distance [114] or Hellinger distance [237]. However, these conventional methods have some majors drawbacks. Firstly, one needs to define the upper bound of number of clusters K_{max} which is not an easy problem. Secondly, the computational cost is usually high since these methods require to solve the maximum log-likelihood problem (10.3) several times. Recently, another approach, which attempts to estimate K simultaneously with the mixture parameter Θ, has been proposed. The main idea is to start with a large enough value of K and then reduce it iteratively during the estimation of Θ. In this approach, the objective function of the maximum log-likelihood problem (10.3) is penalized by regularization term of the mixing proportion π

$$\max_{\Theta \in \mathcal{D}} L\left(\Theta\right) - \lambda \mathcal{P}^1(\pi), \tag{10.6}$$

where $\lambda > 0$ is the trade-off parameter. [52] defined the penalization as $\mathcal{P}^1_{log}(\pi) = -\sum_{k=1}^{K} \log \pi_k$ which forces the value of π_k away from 0. Furthermore, they considered a second regularization term on θ to prevent that a component density is close to one another. With the assumption that $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_K$, the problem is written as

$$\max_{\Theta \in \mathcal{D}} L\left(\Theta\right) - \lambda \left(-\sum_{k=1}^{K} \log \pi_k + \sum_{k=1}^{K-1} p(\theta_{k+1} - \theta_k) \right), \tag{10.7}$$

where $p(.)$ is the SCAD penalty function [70]. The above problem is solved by a modified EM algorithm in which one reduces the number of clusters by merging clusters that have the same mean. However, it happens that this procedure merges Gaussian components with the same mean but different variance, which is not correct. In another approach, [108] proposed to remove empty clusters (e.g. clusters having π_k equals to 0) instead of merging clusters as in [52]. Hence, to promote the sparsity of π, they defined the penalization term as $\mathcal{P}^1_{\epsilon}(\pi) = \log \frac{\epsilon + \pi_k}{\epsilon}$ where ϵ is a very small positive number. Furthermore, to prevent that $\log \frac{\epsilon + \pi_k}{\epsilon}$ over-penalizes large values of π_k and consequently yields a biased estimator, they presented the following second model

$$\max_{\Theta \in \mathcal{D}} L\left(\Theta\right) - \lambda \sum_{k=1}^{K} \log \frac{\epsilon + p(\pi_k)}{\epsilon}, \tag{10.8}$$

where $p(.)$ is the SCAD penalty function. In [108], a modified EM algorithm was proposed to solve both formulations. In this procedure, whenever a mixing proportion π_k is shrunken to 0 the corresponding cluster is deleted and K is reduced by 1 for the remaining iterations. Comparing to conventional methods that need to solve several times the maximum log-likelihood problem, this new approach is undeniably less computational costly. However, the modified EM proposed in [52] and [108] still remain the limitations of EM, for instance EM breaks down when a covariance matrix becomes ill-conditioned [73].

The feature selection problem in model-based clustering has been studied in several works. A review of existing approaches can be found in [2, 32]. The idea behind the feature selection in model-based clustering is that if the means of a feature d on each component are equal to 0, i.e. $\mu_{kd} = 0$ for all $k = 1, \ldots, K$, then this feature is considered as irrelevant and can be consequently removed. The cluster means μ_{kd} can be driven toward 0 by penalizing the objective function of maximum log-likelihood problem (10.3) as follows

$$\max_{\Theta \in \mathcal{D}} L\left(\Theta\right) - \lambda \mathcal{P}^2\left(\mu\right), \tag{10.9}$$

where $\mathcal{P}^2(\mu) := \sum_{k=1}^{K}\sum_{d=1}^{D}|\mu_{kd}|_0$ and $\lambda > 0$ is the trade-off parameters between the two terms. As we can see, (10.9) involves the minimization of the l_0-norm which is known to be NP-hard. Several methods have been developed in the literature for the minimization of l_0-norm (sparse optimization). The readers are referred to Chapter 5 for an overview and our works in sparse optimization. Most of existing work in literature for feature selection in model-based clustering belong to the convex approximation approach. In [182] and [260], the authors replaced $\mathcal{P}^2(\mu)$ by the l_1 approximation defined as $\mathcal{P}_{l_1}^2(\mu) = \sum_{k=1}^{K}\sum_{d=1}^{D}|\mu_{kd}|$. Later, [27] introduced a variant of l_1 regularization function with the presence of proportion of each cluster $\mathcal{P}_{\pi}^2(\mu) = \sum_{k=1}^{K}\pi_k\sum_{d=1}^{D}|\mu_{kd}|$. In the same manner, [232] replaced the l_0 norm by the l_∞ norm. In another work, [92] introduced the pairwise fusion penalty defined by $\mathcal{P}_{fusion}(\mu) = \sum_{d=1}^{D}\sum_{1\leq k<k'\leq K}|\mu_{kd} - \mu_{k'd}|$. Instead of shrinking the means μ_{kd} to 0, the pairwise fusion penalty shrinks the means towards each other. Thus, dimension d is considered non-informative if all the values of means at that dimension are equal, i.e. $\mu_{kd} = \mu_{k'd}$ for all $k \neq k'$. For all above-mentioned works, EM algorithm was developed. It is important to note that, in [182, 232, 92], the authors assumed that all clusters have a common diagonal covariance matrix. This assumption restricts the correlations between features, therefore highly reduces the flexibility of the mixture model. Recently, for the first time, in our conference paper, a nonconvex approximation, was used for the feature selection in GMM clustering in [176]. The numerical results showed that our algorithm DCA outperforms the EM method in [260] which uses l_1-norm approximation.

To handle the third issue in model-based clustering, i.e. the over-parametrization, one thinks naturally to impose a sparse condition on the precision matrices W_k since they are the highest dimensional variables among π_k, μ_k and W_k. The majority of the existing literature assume that all clusters have a common diagonal covariance matrix, implying the same orientation for all clusters [182, 243, 232, 92]. However, it has been shown that this assumption is too stringent and can consequently badly affect the mixture models. On the other hand, the sparsity of W_k can be achieved by adding a regularization term of W_k to the objective function of the maximum log-likelihood problem (3)

$$\max_{\Theta\in\mathcal{D}} L(\Theta) - \lambda\mathcal{P}^3(W), \tag{10.10}$$

where $\mathcal{P}^3(W) := \sum_{k=1}^{K}\sum_{d=1}^{D}\sum_{j=1;j\neq d}^{D}|W_k(d,j)|_0$ ($W_k(d,j)$ is the element with index (d,j) of matrix W_k). Similarly to the feature selection problem, the l_0-norm in $\mathcal{P}^3(W)$ makes the problem (10.10) hard to solve. Most of existing works have focused on using a convex regularization to replace $\mathcal{P}^3(W)$. The l_1 regularization was used in [260] while weighted l_1 was developed in [102] and [240]. Later, [94] also considered the similarity among precision matrices W_k by combining the l_1 regularization with the l_2-norm, i.e. $\sum_{d=1}^{D}\sum_{j=1;j\neq d}^{D}(\sum_{k=1}^{K}|W_k(d,j)|^2)^{1/2}$. In another work, [78] replaced $\mathcal{P}^3(W)$ by a non-convex regularization, named truncated LASSO, to get the following optimization problem

$$\max_{\Theta\in\mathcal{D}} L(\Theta) - \lambda_1\sum_{k=1}^{K}\sum_{d=1}^{D}\sum_{j=1;j\neq d}^{D} r_\alpha(W_k(d,j)) + \lambda_2\sum_{k<k'}\sum_{d=1}^{D}\sum_{j=1;j\neq d}^{D} r_\alpha(W_k(d,j) - W_{k'}(d,j)), \tag{10.11}$$

where $r_\alpha(s) = \min\{|s|, \alpha\}$. An EM based algorithm was developed for solving (10.11). Note that, in this EM algorithm, the computation of W at step M requires to solve a DC program and the proposed method coincides with the idea of DCA.

In summary, all three above-presented issues in model-based clustering, i.e. the model selection, the features selection and the over-parameterization are related to the sparse optimization. The model selection can be done through a sparse inducing regularization of the mixing pro-

portion π_k. Similarly, feature selection (resp. over-parameterization) can be formulated as the minimization of the l_0-norm of μ_k (resp. the l_0-norm of W_k). Although all these three issues are linked together, most of existing works deal with them separately and there is not an unified model which considers all three issues simultaneously. As for resolution method, most of existing works replaced the l_0-norm by an convex approximation and then proposed an EM based algorithm to solve the resulting optimization problem.

Our contribution. In this work, for the first time, we introduce an unified optimization formulation that considers at the same time the model selection, the feature selection and the over-parameterization problem in GMM clustering. Our maximum sparse penalized log-likelihood problem takes the form

$$\max_{\Theta \in \mathcal{D}} L(\Theta) - \mathcal{P}_\lambda(W), \tag{10.12}$$

where

$$\begin{aligned} \mathcal{P}_\lambda(\Theta) &= \lambda_1 \mathcal{P}^1(\pi) + \lambda_2 \mathcal{P}^2(\mu) + \lambda_3 \mathcal{P}^3(W) \\ &= \lambda_1 \sum_{k=1}^{K} \log \frac{\epsilon + |\pi_k|_0}{\epsilon} + \lambda_2 \sum_{k=1}^{K} \sum_{d=1}^{D} |\mu_{kd}|_0 + \lambda_3 \sum_{k=1}^{K} \sum_{d=1}^{D} \sum_{j=1; j \neq d}^{D} |W_k(d,j)|_0, \end{aligned} \tag{10.13}$$

with the trade-off parameters $\lambda_1, \lambda_2, \lambda_3 \geq 0$. Lets recall that in [108], the regularization term of π is defined as $\sum_{k=1}^{K} \log \frac{\epsilon + p(\pi_k)}{\epsilon}$ where $p(.)$ is the SCAD penalty function. Since SCAD penalty function is known as an approximation of the l_0-norm, in our work, we define $\mathcal{P}^1(\pi)$ in a more general way, i.e. $\mathcal{P}^1(\pi) := \sum_{k=1}^{K} \log \frac{\epsilon + |\pi_k|_0}{\epsilon}$. Note that we do not impose any assumption on the variables π, μ and W.

On the contrary to the existing works, we adopt the non-convex approximations approach to deal with l_0-norm as they are more efficient than convex approximation approach. The resulting optimization is non-convex for which we investigate DCA based algorithms to solve.

We will firstly show that standard DCA can solve the optimization problem (10.12). It turns out that with this DC decomposition the corresponding standard DCA can be computationally expensive for high dimensional datasets. Inspired by the idea of DCA-Like presented in Chapter 3, we will exploit the specific structure of (10.12) to develop an tailored DCA-Like algorithm to solve it. Furthermore, we propose a Two-Step DCA-Like in order to improve DCA-Like.

The remainder of the chapter is organized as follows. In Section 10.2, we develop a standard DCA for solving (10.12). DCA-Like is presented in Section 10.3 while Section 10.4 is devoted to Two-Step DCA-Like. In Section 10.5, we realize several numerical experiments and provide a comparison of our algorithms with existing methods on benchmark and synthetic datasets. Finally, Section 10.6 concludes the chapter.

10.2 DCA for solving Gaussian Mixture Models clsutering with sparse regularization

We will now reformulate the problem (10.12) by replacing the l_0-norm by a non-convex approximation. Among the well-known existing non-convex approximations, we choose the exponential concave function approximation function which has been successfully applied in several machine learning problems. Note that, the algorithms developed in this paper work with other non-convex approximations of l_0-norm such as SCAD, capped-l_1, logarithm function, piecewise linear function, l_p with $0 < p < 1$ or l_p with $0 < p$. Let $s \in \mathbb{R}$, the exponential concavec function

is defined by $r_\alpha(s) = 1 - \exp(-\alpha|s|)$, where $\alpha > 0$ controls the tightness of the approximation. As showed in Chapter 5, r_α is a DC function with following DC decomposition

$$r_\alpha(s) = \gamma|s| - (\gamma|s| - r_\alpha(s)) \tag{10.14}$$

where $\gamma > 0$ is a positive real number such that $\gamma|s| - r_\alpha(s)$ is a convex function. Through out this chapter, we choose $\gamma = \alpha$ as proposed in Chapter 5. Substituting l_0 by r_α in (10.13), we obtain the new regularization term

$$\begin{aligned}\mathcal{P}_{\lambda,\alpha}(\Theta) &:= \lambda_1 \mathcal{P}^1_{\lambda,\alpha}(\pi) + \lambda_2 \mathcal{P}^2_{\lambda,\alpha}(\mu) + \lambda_3 \mathcal{P}^3_{\lambda,\alpha}(W) \\ &= \lambda_1 \sum_{k=1}^{K} \log \frac{\epsilon + r_\alpha(\pi_k)}{\epsilon} + \lambda_2 \sum_{k=1}^{K}\sum_{d=1}^{D} r_\alpha(\mu_{kd}) + \lambda_3 \sum_{k=1}^{K}\sum_{d=1}^{D}\sum_{j=1;j\neq d}^{D} r_\alpha(W_k(d,j)).\end{aligned} \tag{10.15}$$

Hence, we can reformulate the maximum sparse penalized log-likelihood problem (10.12) in minimization form as follows

$$\min_{\Theta\in\mathcal{D}} \mathcal{F}(\Theta) := -L(\Theta) + \mathcal{P}_{\lambda,\alpha}(\Theta), \tag{10.16}$$

with $L(\Theta)$ and $\mathcal{D}$ defined in (10.3) and (10.5). In the sequence, we consider the problem (10.16) instead of (10.12). We now present the standard DCA for solving (10.16).

On the one hand, since $-L(\Theta)$ is at least twice differentiable, it is a DC function, for instance, with the following DC decomposition $-L(\Theta) = G_\rho(\Theta) - H_\rho(\Theta)$ where :

$$\begin{aligned}G_\rho(\Theta) &= \frac{\rho}{2}\|\Theta\|^2, \\ H_\rho(\Theta) &= \frac{\rho}{2}\|\Theta\|^2 + L(\Theta).\end{aligned} \tag{10.17}$$

There exists a positive value ρ_0 such that for all $\rho > \rho_0$ the function $H_\rho(\Theta)$ is convex. In deed, denote by $\lambda_n(\Theta)$ the largest eigenvalue of the Hessian matrix of $-L$ at Θ. It is easy to prove that for all $\rho > max\{0, \lambda_n(\Theta)\}$ the function $H_\rho(\Theta)$ is convex.

On the other hand, $r_\alpha(s)$ is a DC function. Hence $\mathcal{P}_{\lambda,\alpha}(\Theta)$ is a DC function

$$\begin{aligned}\mathcal{P}_{\lambda,\alpha}(\Theta) &= G_{\lambda,\alpha}(\Theta) - H_{\lambda,\alpha}(\Theta), \\ G_{\lambda,\alpha}(\Theta) &= \frac{\lambda_1\alpha}{\epsilon}\|\pi\|_1 + \lambda_2\alpha\|\mu\|_1 + \lambda_3\alpha\textstyle\sum_{k=1}^{K}\|W_k\|_1, \\ H_{\lambda,\alpha}(\Theta) &= \frac{\lambda_1\alpha}{\epsilon}\|\pi\|_1 + \lambda_2\alpha\|\mu\|_1 + \lambda_3\alpha\textstyle\sum_{k=1}^{K}\|W_k\|_1 - \mathcal{P}_{\lambda,\alpha}(\Theta).\end{aligned} \tag{10.18}$$

We note here the l_1-norm for matrices, which is $\|W_k\|_1$, is off-diagonal, i.e. the diagonal elements are not considered.

Since $-L(\Theta)$ and $\mathcal{P}_{\lambda,\alpha}(\Theta)$ are DC functions, $\mathcal{F}(\Theta) = -L(\Theta) + \mathcal{P}_{\lambda,\alpha}(\Theta)$ is a DC function with $\mathcal{F}(\Theta) = \mathcal{G}(\Theta) - \mathcal{H}(\Theta)$ where $\mathcal{G}(\Theta) := G_\rho(\Theta) + G_{\lambda,\alpha}(\Theta)$ and $\mathcal{H}(\Theta) := H_\rho(\Theta) + H_{\lambda,\alpha}(\Theta)$. Thus, DCA can be developed for solving the optimization problem (10.16). According to general DCA scheme, at each iteration t we compute $\bar{\Theta}^t = (\bar{\pi}^t, \bar{\mu}^t, \bar{W}^t) \in \partial\mathcal{H}(\pi^t, \mu^t, W^t)$ and then compute $\Theta^{t+1} = (\pi^{t+1}, \mu^{t+1}, W^{t+1})$ as the solution to the following convex problem

$$\min_{\pi,\mu,W\in\mathcal{D}} \left\{\mathcal{G}(\pi,\mu,\mathcal{W}) - \langle(\bar{\pi}^t, \bar{\mu}^t, \bar{W}^t), (\pi,\mu,W)\rangle\right\}. \tag{10.19}$$

The optimization problem (10.19) is separable on all three variables. Thus, we have

$$\pi^{t+1} \in \arg\min \left\{ \frac{\rho}{2}\|\pi\|^2 + \frac{\lambda_1 \alpha}{\epsilon}\|\pi\|_1 - \langle \bar{\pi}^t, \pi \rangle : \sum_{k=1}^{K} \pi_k = 1 \right\}, \tag{10.20}$$

$$\mu^{t+1} \in \arg\min \left\{ \frac{\rho}{2}\|\mu\|^2 + \lambda_2 \alpha \|\mu\|_1 - \langle \bar{\mu}^t, \mu \rangle \right\}, \tag{10.21}$$

$$W^{t+1} \in \arg\min \left\{ \frac{\rho}{2}\|W\|^2 + \lambda_3 \alpha \|W\|_1 - \langle \bar{W}^t, W \rangle : W_k \succ 0,\ \forall k \in \{1, ..., K\} \right\}. \tag{10.22}$$

The solution of (10.20) is nothing else but the projection onto a simplex $\Delta := \{\pi \in (0,1)^K : \sum_{k=1}^{K} \pi_k = 1\}$ for which several efficient algorithms are available, for instance the work of [118]. The problem (10.21) can be efficiently solved using the soft thresholding operator [21]. The resolution of problem (10.22) requires a projection onto the positive definite cone $\mathcal{W}_k := \{W_k :\ W_k \succ 0\}$. Unfortunately, this projection is computationally expensive for high dimensional matrices.

Remark 10.1 *Finding an DC decomposition for $-L(\Theta)$ is challenging due to its mathematical definition as a logarithm function of a sum. In the above-presented DCA scheme for solving* (10.16)*, the DC decomposition of $-L(\Theta)$ is only based on the fact that $-L(\Theta)$ is twice differentiable. With this DC decomposition, $-L(\Theta)$ is fully integrated to $H_\rho(\Theta)$ and its structure is not well exploited. This leads to a standard DCA scheme that may require high effort in computation for large-scale data.*

In literature, the GMM clustering problem (10.3) *has been efficiently solved by EM algorithm. The effectiveness of EM comes from the fact that one can easily obtain a convex upper bound Q of $-L$ and the problem of minimization of Q can be explicitly solved. Inspired by the idea of DCA-Like presented in Chapter 3, we aim to utilize the convex upper bound Q to develop a DCA-Like algorithm for solving* (10.16) *in the next section.*

10.3 DCA-Like

Lets first recall briefly the EM algorithm for solving the GMM clustering problem (10.3). Let $z_{ik} = \dfrac{\pi_k p\left(x_i \mid \theta_k\right)}{\sum_l \pi_l p\left(x_i \mid \theta_l\right)}$. We have

$$-L(\Theta) = -\sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k p\left(x_i \mid \theta_k\right)\right) = \sum_{i,k} z_{ik}^{(t)} \log\left(z_{ik}\right) - \sum_{i,k} z_{ik}^{(t)} \log\left(\pi_k p\left(x_i \mid \theta_k\right)\right).$$

Let

$$z_{ik}^{(t)} = \frac{\pi_k^{(t)} p\left(x_i \mid \theta_k^{(t)}\right)}{\sum_l \pi_l^{(t)} p\left(x_i \mid \theta_l^{(t)}\right)} \geq 0.$$

Since $-\log$ is convex and $\sum_{k=1}^{K} z_{ik}^{(t)} = 1$, we have

$$\begin{aligned} -L(\Theta) &= -\sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k p\left(x_i \mid \theta_k\right)\right) = -\sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} z_{ik}^{(t)} \frac{\pi_k p\left(x_i \mid \theta_k\right)}{z_{ik}^{(t)}}\right) \\ &\leq \sum_{i,k} z_{ik}^{(t)} \log\left(z_{ik}^{(t)}\right) - \sum_{i,k} z_{ik}^{(t)} \log\left(\pi_k p\left(x_i \mid \theta_k\right)\right). \end{aligned}$$

Let $Q(\Theta, \Theta^{(t)}) = -\sum_{i,k} z_{ik}^{(t)} \log\left(\pi_k p\left(x_i \mid \theta_k\right)\right)$. Since $0 \leq z_{ik} \leq 1$, we obtain :

$$-L(\Theta) \leq -\sum_{i,k} z_{ik}^{(t)} \log\left(\pi_k p\left(x_i \mid \theta_k\right)\right) = Q(\Theta, \Theta^{(t)}). \tag{10.23}$$

The EM algorithm for maximimum log-likelihood problem, e.g. $\max_{\Theta \in \mathcal{D}} L(\Theta)$ or equivalently $\min_{\Theta \in \mathcal{D}} -L(\Theta)$, consists on computing the posterior probability z_{ik} in the E step and then minimizing $Q(\Theta, \Theta^{(t)})$ to update Θ in the M step. Fortunately, the solution of the optimization problem in step M can be obtained explicitly and there is no projection needed. This is the main reason why EM algorithm is efficient for GMM clustering. However, when dealing with penalized likelihood problem (10.16) where the regularization term is non-convex, the resulting upper bound of the objective function is non-convex. Therefore, we need a more appropriate method for solving (10.16). On the other hand, standard DCA can effectively handle the non-convex penalty term when the log-likehood term $-L$ is DC decomposed. This observation gives rise to a incorporated method between upper-bounding $-L$ by Q and treating the regularization term by DCA.

We propose to decompose $-L(\Theta)$, using the $Q(\Theta, \Theta^{(t)})$, as follows

$$\begin{array}{ll} -L(\Theta) & = G_L^{(t)} - H_L^{(t)}, \\ G_L^{(t)}(\Theta) & = Q(\Theta, \Theta^{(t)}), \\ H_L^{(t)}(\Theta) & = Q(\Theta, \Theta^{(t)}) + L(\Theta). \end{array} \tag{10.24}$$

Hence, with the DC decompsition of $\mathcal{P}_{\lambda,\alpha}(\Theta)$ in (10.18), $\mathcal{F}(\Theta)$ can be decomposed as

$$\mathcal{F}(\Theta) = \left[G_L^{(t)}(\Theta) + G_{\lambda,\alpha}(\Theta)\right] - \left[H_L^{(t)}(\Theta) + H_{\lambda,\alpha}(\Theta)\right]. \tag{10.25}$$

To solve (10.16) with this decomposition, we develop a DCA-Like algorithm. Note that (10.25) is not a DC decomposition since $H_L^{(t)}(\Theta) + H_{\lambda,\alpha}(\Theta)$ is not necessarily convex. DCA-Like is "like" DCA in the sense that they iteratively approximate the DC program (10.16) by a sequence of convex ones. However, DCA-Like is "unlike" DCA as it relaxes a key requirement of DCA in the manner to decompose $\mathcal{F}(\Theta)$. The DCA-Like algorithm for solving (10.16) is described in Algorithm 10.1.

Algorithm 10.1 DCA-Like scheme for solving (10.16)

1: **Initialization** Let $\left(\pi^{(0)}, \mu^{(0)}, W^{(0)}\right) \in \mathcal{D}$ be a best guess.
Choose $\alpha > 0, \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, t \leftarrow 0$.
2: **repeat**
3: Calculate $G_L^{(t)}(\Theta)$ and $\bar{\Theta}^t = (\bar{\pi}^t, \bar{\mu}^t, \bar{W}^t) \in \partial H_{\lambda,\alpha}(\pi^t, \mu^t, W^t)$.
4: Calculate $\Theta^{(t+1)}$ as a solution of

$$\min_{\Theta \in \mathcal{D}} \left[G_L^{(t)}(\Theta) + G_{\lambda,\alpha}(\Theta)\right] - \left\langle \bar{\Theta}^t, \Theta \right\rangle. \tag{10.26}$$

5: $t = t + 1$.
6: **until** stopping criterion.

Remark 10.2 *Lets consider a special case of* (10.16) *where* $\mathcal{P}_{\lambda,\alpha} = 0$*, e.g. the classical GMM clustering* (10.3)*. Hence the sub-problem* (10.26) *in DCA-Like scheme becomes*

$$\min_{\Theta \in \mathcal{D}} G_L^{(t)}(\Theta) = \min_{\Theta \in \mathcal{D}} Q(\Theta, \Theta^{(t)}).$$

This coincides with the M step of EM algorithm. Thus, DCA-Like is reduced to EM algorithm when $\mathcal{P}_{\lambda,\alpha} = 0$.

Below we give more details of the calculation in each step of Algorithm 10.1.

Computation of $(\bar{\pi}^t, \bar{\mu}^t, \bar{W}^t) \in \partial H_{\lambda,\alpha}(\pi^t, \mu^t, W^t)$. Recall that

$$\begin{aligned} H_{\lambda,\alpha}(\Theta) &= \frac{\lambda_1\alpha}{\epsilon}\|\pi\|_1 + \lambda_2\alpha\|\mu\|_1 + \lambda_3\alpha\sum_{k=1}^K \|W_k\|_1 \\ &-\lambda_1 \sum_{k=1}^K \log\frac{\epsilon + r_\alpha(\pi_k)}{\epsilon} - \lambda_2 \sum_{k=1}^K\sum_{d=1}^D r_\alpha(\mu_{kd}) - \lambda_3 \sum_{k=1}^K\sum_{d=1}^D\sum_{j=1;j\neq d}^D r_\alpha(W_k(d,j)). \end{aligned}$$

The function $H_{\lambda,\alpha}(\Theta)$ is differentiable and its gradient can be computed as

$$\nabla H\lambda, \alpha(\Theta) = (\bar{\pi}, \bar{\mu}, \bar{W}) \in \mathbb{R}^K \times \mathbb{R}^{K\times D} \times \mathbb{R}^{K\times D\times D}$$

where

$$\begin{aligned} \bar{\pi}_k &= \tfrac{\lambda_1\alpha}{\epsilon} - \lambda_1\tfrac{\alpha e^{-\alpha\pi_k}}{\epsilon + r_\alpha(\pi_k)} \qquad \forall k = 1,\dots,K \\ \bar{\mu}_{kd} &= \begin{cases} \lambda_2\alpha\left(1 - e^{-\alpha\mu_{kd}}\right) & \text{if } \mu_{kd} > 0, \\ -\lambda_2\alpha\left(1 - e^{\alpha\mu_{kd}}\right) & \text{otherwise,} \end{cases} \qquad \forall k = 1,\dots,K,\ \forall d = 1,\dots,D \\ \bar{W}_k(d,j) &= \begin{cases} \lambda_3\alpha\left(1 - e^{-\alpha W_k(d,j)}\right) & \text{if } W_k(d,j) > 0 \\ -\lambda_3\alpha\left(1 - e^{\alpha W_k(d,j)}\right) & \text{otherwise} \end{cases} \qquad \forall k = 1,\dots,K,\ \forall j,d = 1,\dots,D, \text{ and } j \neq d. \end{aligned} \tag{10.27}$$

Solution to the sub-problem (10.26). Recall that in DCA-Like scheme (Algorithm 10.1), we have to solve the following convex sub-problem at each iteration

$$\min_{\pi,\mu,W\in\mathcal{D}} G_L^{(t)}(\pi,\mu,W) + G_{\lambda,\alpha}(\pi,\mu,W) - \left\langle(\bar{\pi}^t,\bar{\mu}^t,\bar{W}^t),(\pi,\mu,W)\right\rangle.$$

We observe that in (10.26), the variable π can be separated from variables μ and W.
Sub-problem for mixing proportions π. By removing terms that do not contain π in (10.26), we obtain

$$\pi^{(t+1)} \in \arg\min\left\{-\sum_{i=1}^N\sum_{k=1}^K z_{ik}^{(t)}\log(\pi_k) + \frac{\lambda_1\alpha}{\epsilon}\|\pi\|_1 + \left\langle\bar{\pi}^t,\pi\right\rangle : \sum_{k=1}^K \pi_k = 1\right\}. \tag{10.28}$$

According to [108], we can obtain the close-form solution of (10.28) as follows :

$$\pi_k^{(t+1)} = \frac{1}{\gamma_k}\sum_{i=1}^n z_{ik}^{(t)}, \tag{10.29}$$

with

$$\gamma_k = N - \frac{\lambda_1\alpha}{\epsilon}\sum_{l=1}^K \nabla r_\alpha\left(\pi_l^{(t)}\right)\pi_l^{(t)} + \frac{\lambda_1\alpha}{\epsilon}\nabla r_\alpha\left(\pi_k^{(t)}\right).$$

At iteration t, if the mixing proportion π_k of component k is equal to 0, we consider that component non-relevant and eliminate it. In practice, π_k rarely equals 0 exactly due to numerical instability [108]. Hence, we can choose a small threshold η and if $\pi_k < \eta$, we shrink it to 0.

Sub-problem for means μ and inverse covariance matrices W. By neglecting terms not containing μ and W in (10.26), we have :

$$\begin{aligned}\left(\mu^{(t+1)}, W^{(t+1)}\right) \in \arg\min \quad & -\frac{1}{2}\sum_{i,k} z_{ik}^{(t)}\left[\log\left(\det\left(W\right)\right) - \left(x_i-\mu_k\right)^T W\left(x_i-\mu_k\right)\right] \\ & +\lambda_2\alpha\|\mu\|_1 + \lambda_3\alpha\|W\|_1 - \left\langle\bar{\mu}^t, \mu\right\rangle - \left\langle\bar{W}^t, W\right\rangle,\end{aligned} \tag{10.30}$$

To solve (10.30), we alternatively solve for μ and W as follows.
Means μ. The sub-problem for μ reads :

$$\min_{\mu}\sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}^{(t)}\left(\frac{1}{2}\left(x_i-\mu_k\right)^T W\left(x_i-\mu_k\right)\right) + \lambda_2\alpha\|\mu\|_1 - \left\langle\bar{\mu}^t, \mu\right\rangle \tag{10.31}$$

Consider μ_{kd}, by using KKT conditions, we obtain

$$\begin{aligned}&\sum_i z_{ik}^{(t)} W_k(d,:)\left(x_i-\mu_k\right) - \operatorname{sign}(\mu_{kd})\lambda_2\alpha + \bar{\mu}^t = 0, \text{ when } \mu_{kd}\neq 0 \\ &\left|\sum_i z_{ik}^{(t)}\left[\sum_{j\neq d} W_k(d,j)\left(x_{ij}-\mu_{kj}\right) + W_k(d,d)x_{id}\right] + \bar{\mu}^t\right| \leq \lambda_2\alpha, \text{ when } \mu_{kd}=0\end{aligned} \tag{10.32}$$

where $W_k(d,:)$ is the d-th row of W_k matrix and x_{id} is the d-th element of x_i. Put

$$\begin{cases}\psi_1 &= \sum_i z_{ik}^{(t)} W_k^{(t)}(d,d) \\ \psi_2 &= \sum_i z_{ik}^{(t)}\left[\sum_{j\neq d} W_k^{(t)}(d,j)\left(x_{ij}-\mu_{kj}^{(t)}\right) + W_k^{(t)}(d,d)x_{id}\right] + \bar{\mu}^t\end{cases} \tag{10.33}$$

From (10.32), we obtain

$$\begin{cases}\mu_{kd} = \dfrac{\psi_2 - \operatorname{sgn}(\psi_2)\lambda_2\alpha}{\psi_1}, & \text{when } \psi_2 > \lambda_2\alpha \\ \mu_{kd} = 0, & \text{when } \psi_2 \leq \lambda_2\alpha\end{cases} \tag{10.34}$$

Inverse covariance matrices W. The sub-problem of W_k has the form :

$$\begin{aligned}\min_{W_1,\ldots,W_K\succ 0} \frac{1}{2}\sum_{i,k} z_{ik}^{(t)}&\left[-\log\left(\det\left(W_k\right)\right) + \left(x_i-\mu_k^{(t+1)}\right)^T W_k\left(x_i-\mu_k^{(t+1)}\right)\right] \\ &+ \lambda_3\alpha\sum_{k=1}^{K}\|W_k\|_1 - \left\langle\bar{W}^t, W\right\rangle.\end{aligned}$$

This can be simplified as

$$\min_{W_1,\ldots,W_K\succ 0}\frac{1}{2}\sum_{k=1}^{K} z_k\left[-\log\left(\det\left(W_k\right)\right) + \operatorname{tr}\left(S_k W_k\right)\right] + \lambda_3\alpha\|\Sigma^{-1}\|_1 - \left\langle\bar{W}^t, W_k\right\rangle \tag{10.35}$$

where

$$\begin{aligned}z_k &= \sum_{i=1}^{N} z_{ik}^{(t)}, \\ S_k &= \frac{\sum_{i=1}^{N} z_{ik}^{(t)}\left(x_i-\mu_k^{(t-1)}\right)\left(x_i-\mu_k^{(t-1)}\right)^T}{z_k}.\end{aligned}$$

Problem (10.35) can be solved effectively via convex solvers [75, 107].

Remark 10.3 *When we choose the penalty coefficient $\lambda_i = 0, i = 1, \ldots, 3$, that means we do not penalize the variable corresponding to λ_i. Then, that variable is updated via EM algorithm as follows :*

$$\pi_k^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} z_{ik}^{(t)}, \tag{10.36}$$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{N} z_{ik}^{(t)} x_i}{\sum_{i=1}^{N} z_{ik}^{(t)}}, \tag{10.37}$$

$$W_k^{(t+1)} = \frac{1}{N}\sum_{i,k} z_{ik}^{(t)} \left(x_i - \mu_k\right)\left(x_i - \mu_k\right)^T. \tag{10.38}$$

Finally, the detailed DCA-Like for solving (10.16) is presented in Algorithm 10.2.

Algorithm 10.2 DCA-Like for (10.16)

1: **Initialization** Choose an initial number of clusters $K_e^{(0)}$.
Let $\left(\pi^{(0)}, \mu^{(0)}, W^{(0)}\right) \in \mathcal{D}$ be a best guess, $\alpha > 0, \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, t \leftarrow 0$.
2: **repeat**
3: Calculate $G_L^{(t)}(\Theta)$ and $\bar{\Theta}^t = (\bar{\pi}^t, \bar{\mu}^t, \bar{W}^t) \in \partial H_{\lambda,\alpha}(\pi^t, \mu^t, W^t)$ via (10.27).
4: **for** $k = 1 : K_e^{(t)}$ **do**
5: If $\lambda_1 > 0$, update $\pi^{(t+1)}$ by (10.29). Else, update $\pi^{(t+1)}$ by (10.36).
6: If $\lambda_2 > 0$, update $\mu^{(t+1)}$ by (10.34). Else, update $\mu^{(t+1)}$ by (10.37).
7: If $\lambda_3 > 0$, update $W^{(t+1)}$ by solving sub-problem (10.35). Else, update $W^{(t+1)}$ by (10.38).
8: If $\pi_k^{(t+1)} = 0$, eliminate component k and set $K_e^{(t+1)} = K_e^{(t)} - 1$.
9: **end for**
10: $t \leftarrow t + 1$
11: **until** stopping criterion.

10.4 Two-Step DCA-Like

In this section, we propose a strategy to improve the clustering results of Algorithm 10.2. Recall that at $\Theta^{(t)} = \left(\pi^{(t)}, \mu^{(t)}, W^{(t)}\right)$, Algorithm 10.2 aims at minimizing the objective function $-L(\Theta) + \mathcal{P}_{\lambda,\alpha}(\Theta)$ by first exploiting the DC-Like structure of $-L$ and $\mathcal{P}_{\lambda,\alpha}$, then forms an appropriate upper bound of $-L(\Theta) + \mathcal{P}_{\lambda,\alpha}(\Theta)$ at $\Theta^{(t)}$ and minimize it. As in Algorithm 10.2, if the mixing proportion $\pi_k^{(t)}$ is equals to 0, then we delete component k for the remaining iterations. This changes the feasible set of Θ since the dimension of π is reduced. After some iterations, the algorithm will reach a point where the number of clusters estimated becomes stable, i.e. its value does not decrease anymore until convergence. Lets call that number K_e. For a fixed K_e, the feasible set of Θ is also fixed. If we call that stable feasible set $\mathcal{D}^*$, then the problem becomes :

$$\min_{\Theta \in \mathcal{D}^*} -L(\Theta) + \mathcal{P}_{\lambda,\alpha}(\Theta) = -L(\Theta) + \lambda_1 \mathcal{P}^1_{\lambda,\alpha}(\pi) + \lambda_2 \mathcal{P}^2_{\lambda,\alpha}(\mu) + \lambda_3 \mathcal{P}^3_{\lambda,\alpha}(W).$$

As we can see, the penalty term $\mathcal{P}^1_{\lambda,\alpha}(\pi)$ of π is now unnecessary since the number of clusters is stable. If we continue keeping that penalty term, it could badly affect the quality of Algorithm 10.2. Therefore, when the number of clusters becomes stable, we should fix K_e and remove

the term $\mathcal{P}^1_{\lambda,\alpha}(\pi)$ from the regularization term $\mathcal{P}_{\lambda,\alpha}$. Based on this observation, we propose a Two-Step DCA-Like scheme as follows. In the Step 1, we apply the steps of Algorithm 10.2 until K_e becomes stable. Then we remove $\mathcal{P}^1_{\lambda,\alpha}(\pi)$ by setting $\lambda_1 = 0$ and continue the Step 2 with fixed K_e. All the computations of Step 2 are similar to Step 1 except for the update of π_k. The Two-Step DCA-Like is presented in Algorithm 10.3.

Algorithm 10.3 Two-Step DCA-Like for (10.16)

Initialization : Choose an initial number of clusters $K_e^{(0)}$.
Let $\left(\pi^{(0)}, \mu^{(0)}, W^{(0)}\right) \in \mathcal{D}$ be a best guess.
Choose $\alpha > 0, \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, t \leftarrow 0$.
Step 1 :
repeat
  Calculate $G_L^{(t)}(\Theta)$ and $\bar{\Theta}^t = (\bar{\pi}^t, \bar{\mu}^t, \bar{W}^t) \in \partial H_{\lambda,\alpha}(\pi^t, \mu^t, W^t)$ via (10.27).
  for $k = 1 : K_e^{(t)}$ **do**
    If $\lambda_1 > 0$, update $\pi^{(t+1)}$ by (10.29). Else, update $\pi^{(t+1)}$ by (10.36).
    If $\lambda_2 > 0$, update $\mu^{(t+1)}$ by (10.34). Else, update $\mu^{(t+1)}$ by (10.37).
    If $\lambda_3 > 0$, update $W^{(t+1)}$ by solving sub-problem (10.35). Else, update $W^{(t+1)}$ by (10.38).
    If $\pi_k^{(t+1)} = 0$, eliminate component k and set $K_e^{(t+1)} = K_e^{(t)} - 1$.
  end for
  $t \leftarrow t + 1$
until stopping criterion on stability of $K_e^{(t)}$
$K_e = K_e^{(t)}$
Step 2 :
repeat
  Calculate $G_L^{(t)}(\Theta)$ and $\bar{\Theta}^t = (\bar{\pi}^t, \bar{\mu}^t, \bar{W}^t) \in \partial H_{\lambda,\alpha}(\pi^t, \mu^t, W^t)$ via (10.27)
  for $k = 1 : K_e$ **do**
    Update $\pi^{(t+1)}$ by (10.36)
    If $\lambda_2 > 0$, update $\mu^{(t+1)}$ by (10.34). Else, update $\mu^{(t+1)}$ by (10.37)
    If $\lambda_3 > 0$, update $W^{(t+1)}$ by solving sub-problem (10.35). Else, update $W^{(t+1)}$ by (10.38)
  end for
  $t \leftarrow t + 1$
until stopping criterion.

10.5 Numerical experiments

10.5.1 Experiment settings

Notations. In DCA-Like (Algorithm 10.2), for each choice (either zero or non-zero) of the tuple $(\lambda_1, \lambda_2, \lambda_3)$, we have a different version of Algorithm 10.2. To differentiate different version of Algorithm 10.2, we choose the notation $DCA\overline{abc}$, with $a, b, c \in \{0, 1\}$, where value 0 means the related λ is set to 0. For instance, *DCA100* means $\lambda_1 \neq 0, \lambda_2 = 0, \lambda_3 = 0$. To denote the class of algorithms with some common λ_i, we denote by x the variations. For example, DCA1xx contains four algorithms : *DCA100*, *DCA110*, *DCA101* and *DCA111*. For Two-Step DCA-Like (Algorithm 10.3), we use the notation *DCA2bc*, where 2 stands for Two-Step. Note that for Two-Step DCA-Like, λ_1 is always non-zero.

Regarding the number of clusters, we denote by K_e the number of clusters estimated by an algorithm. K^* is the true number of clusters given from the benchmark datasets. K_{hf} is K_e with highest frequency over 100 runs of an algorithm.

Comparative criteria To evaluate the performance of algorithms, we consider the following criteria

— Adjusted Rand Index (ARI) : a well-known measure of the similarity between two clusterings [110].

— Selected Feature percentage (SF) : the percentage of selected features over the total number of features.

— True Positive Rate (TPR) : the percentage of features selected which is informative over all informative features.

— False Positive Rate (FPR) : the percenrage of non-informative features selected over all non-informative features.

— Computation time.

Datasets. Benchmark datasets are taken from UCI Machine Learning Repository [66]. Information of datasets is given in Table 10.1.

Dataset	Instances (N)	Dimension (D)	Number of classes (K*)
comp	3891	10	3
dermatology	358	34	6
glass	214	9	6
ionosphere	351	32	2
iris	150	4	3
thyroid	215	5	3
wine	178	27	3
zoo	101	16	7

TABLE 10.1 – Datasets

Experiment setting The parameter α is chosen from the set $\{1, 5, 10\}$ while λ_1, λ_3 belong to $\{0.001, 0.0002, ..., 1\}$ and λ_2 in $\{0.1, 0.1099, ..., 10\}$. We use a grid search procedure for choosing the best value of α and λ. To reduce the effects of initial values and local minima, we repeat the search multiple times. The grid point with highest average BIC value, as defined below, is chosen as the optimal tuning parameters.

$$BIC(\alpha, \lambda) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K_e} \widetilde{\pi_k} p \left(x_i \mid \widetilde{\theta_k} \right) \right) - \frac{1}{2} \log(N) \sum_{k=1}^{K_e} P_k.$$

We run each algorithm 100 times with the optima parameters and report the mean and standard deviation of each criteria. K-means is used for finding an initial point for all algorithms. Convex sub-problems on sparse covariance selection are solved by QUIC software [107].

The experiments are performed on a Intel Core i7 3.60 GHz PC with 16 GB of RAM and the codes were written in MATLAB.

10.5.2 Experiment 1 - Estimating the number of clusters

This experiment aims to evaluate the ability of our algorithms in estimating the correct number of clusters.

Comparative algorithm. We benchmark the class of four *DCA1xx* algorithms, e.g. $\lambda_1 \neq 0$ (c.f. Section 10.5.1) and compare them with the algorithm proposed by [108] and the well-known package *mclust* [212]. Recall that *mclust* uses BIC as the model selection criterion while [108] deals with the optimization (10.8), that is

$$\max_{\Theta \in \mathcal{D}} L(\Theta) - \lambda \sum_{k=1}^{K} \log \frac{\epsilon + p(\pi_k)}{\epsilon}.$$

Experiment setting. The lower bound and upper bound for the number of clusters are 1 and 9, respectively (as given default in *mclust*). Hyperparameters $\lambda_1, \lambda_2, \lambda_3$ and α are chosen as in Section 10.5.1. Let K_e be the estimated number of clusters by an algorithm. As we choose the bounds for the number of clusters above, the set of all possible outcomes of K_e is limited to $\{1, 2, ..., 9\}$. As mentioned in Section 10.3, the threshold η to shrink the mixing proportion π_k is set to 10^{-6}. We run each algorithm 100 times and record the frequency of each outcome. Let K_{hf} be the outcome of K_e with highest frequency. Adjusted Rand Index of cases where $K_e = K_{hf}$ are reported along with the running time in the Table 10.2.

TABLE 10.2 – Experiment 1 - Estimating the number of clusters of the algorithms on real datasets. Bold values correspond to the correct number of clusters estimated for each dataset ($K_{hf} = K^*$). K_{hf} is the estimated number of clusters with the highest frequency.

Algorithms	**Frequency**									K_{hf}	**ARI**	**Time(s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**			
DCA100	1	28	71	0	0	0	0	0	0	**3**	0.9533 ± 0.0095	0.5727 ± 0.01
DCA110	2	27	70	1	0	0	0	0	0	**3**	0.9317 ± 0.0018	7.9523 ± 0.18
DCA101	0	21	79	0	0	0	0	0	0	**3**	0.9496 ± 0.015	1.9554 ± 0.65
DCA111	0	16	81	3	0	0	0	0	0	**3**	0.9378 ± 0.0098	39.0379 ± 2.41
Huang	0	4	86	10	0	0	0	0	0	**3**	0.9666 ± 0	0.9047 ± 0.14
mclust	0	0	0	0	0	0	0	14	86	9	0.4372 ± 0.0413	57.384 ± 5.1
Data set comp : $N = 3891, D = 10, K^* = 3$.												
Algorithms	**Frequency**									K_{hf}	**ARI**	**Time(s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**			
DCA100	0	48	39	8	5	0	0	0	0	**2**	0.7752 ± 0.0296	0.7011± 0.0843
DCA110	3	49	40	8	0	0	0	0	0	**2**	0.7438 ± 0.022	14.9491± 1.0355
DCA101	24	74	0	0	0	0	0	0	0	**2**	0.7636 ± 0.0302	3.2247± 0.4231
DCA111	13	79	7	1	0	0	0	0	0	**2**	0.7584 ± 0.0133	18.8638± 0.6758
Huang	0	13	23	45	19	0	0	0	0	4	0.6374 ± 0.0304	0.1008± 0.0604
mclust	0	100	0	0	0	0	0	0	0	**2**	0.3461 ± 0	42.7856± 1.58
Data set ionosphere : $N = 351, D = 32, K^* = 2$.												
Algorithms	**Frequency**									K_{hf}	**ARI**	**Time(s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**			
DCA100	0	21	78	1	0	0	0	0	0	**3**	0.9094 ± 0.0293	0.0307± 0.0153
DCA110	0	16	36	35	4	0	0	0	0	**3**	0.8524 ± 0.0458	0.3596± 0.0786
DCA101	0	2	91	7	0	0	0	0	0	**3**	0.8783 ± 0.0041	0.1912± 0.0589
DCA111	0	3	97	0	0	0	0	0	0	**3**	0.8321 ± 0.0062	0.8766± 0.0421
Huang	0	74	26	0	0	0	0	0	0	2	0.7763 ± 0.0011	0.0465± 0.0293
mclust	0	100	0	0	0	0	0	0	0	2	0.5681 ± 0	0.0352± 0.0108
Data set iris : $N = 150, D = 4, K^* = 3$.												
Algorithms	**Frequency**									K_{hf}	**ARI**	**Time(s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**			
DCA100	0	2	88	9	1	0	0	0	0	**3**	0.9331 ± 0.0045	0.0607± 0.0197
DCA110	0	4	91	5	0	0	0	0	0	**3**	0.9215 ± 0.0467	0.5288± 0.0124
DCA101	0	2	98	0	0	0	0	0	0	**3**	0.8535 ± 0.003	0.1429± 0.0186

DCA111	0	1	90	9	0	0	0	0	0	**3**	0.8331 ± 0.005	1.2473± 0.1115
Huang	0	0	69	29	2	0	0	0	0	**3**	0.9317 ± 0.0045	0.0531± 0.0192
mclust	0	0	100	0	0	0	0	0	0	**3**	0.8925 ± 0	0.2822± 0.0083

Data set thyroid : $N = 215, D = 5, K^* = 3$.

Algorithms	**Frequency**									K_{hf}	**ARI**	**Time(s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**			
DCA100	0	7	70	23	0	0	0	0	0	**3**	0.9228 ± 0.037	0.2078± 0.0352
DCA110	2	14	65	13	6	0	0	0	0	**3**	0.8767 ± 0.0467	3.8266± 0.1783
DCA101	0	6	83	11	0	0	0	0	0	**3**	0.9125 ± 0.0356	0.8944± 0.0103
DCA111	0	2	81	17	0	0	0	0	0	**3**	0.8936 ± 0.0234	8.4474± 0.4546
Huang	2	16	47	35	0	0	0	0	0	**3**	0.8912 ± 0.0135	0.2311± 0.0085
mclust	0	0	100	0	0	0	0	0	0	**3**	0.9306 ± 0	12.4456± 1.06

Data set wine : $N = 178, D = 27, K^* = 3$.

Algorithms	**Frequency**									K_{hf}	**ARI**	**Time(s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**			
DCA101	25	4	0	0	2	69	0	0	0	**6**	0.9184 ± 0.0486	0.8671± 0.0427
DCA111	17	0	0	0	4	72	7	0	0	**6**	0.8966 ± 0.0815	43.1283± 4.6154
mclust	0	0	0	0	0	0	0	0	100	9	0.6362 ± 0	0.4264± 0.0091

Data set dermatology : $N = 358, D = 34, K^* = 6$.

Algorithms	**Frequency**									K_{hf}	**ARI**	**Time(s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**			
DCA101	0	0	0	7	37	56	0	0	0	**6**	0.559 ± 0.0166	0.497± 0.0232
DCA111	0	0	0	0	27	61	12	0	0	**6**	0.5213 ± 0.0191	3.5362± 0.7323
mclust	0	0	0	0	100	0	0	0	0	5	0.147 ± 0	0.1608± 0.0418

Data set glass : $N = 214, D = 9, K^* = 6$.

Algorithms	**Frequency**									K_{hf}	**ARI**	**Time(s)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**			
DCA101	0	0	0	0	1	37	42	18	2	**7**	0.9282 ± 0.0183	0.0465± 0.0031
DCA111	0	0	0	0	4	30	51	12	3	**7**	0.9008 ± 0.0286	12.4918± 1.4299
mclust	0	0	0	100	0	0	0	0	0	4	0.2633 ± 0	0.0964± 0.016

Data set zoo : $N = 101, D = 16, K^* = 7$.

In *dermatology, glass* and *zoo* datasets, the algorithms using full covariance structures (*DCA100*, *DCA110* and [108]) fail to give a result. The error is caused by ill-conditioned precision matrices W_k.

Comments on numerical results. In terms of estimating the true number of clusters, all algorithms of class *DCA1xx* ($\lambda_1 \neq 0$) are able to estimate the correct number of clusters with high frequency, in all experimented datasets. The algorithm of [108] overestimates the number of clusters in *ionosphere* dataset ($K_{hf} = 4, K^* = 2$) whereas underestimates it in *iris* dataset ($K_{hf} = 3, K^* = 2$). Although *mclust* does not encounter the ill-conditioned covariance matrices problem in *dermatology*, *glass* and *zoo*, it misestimates the number of clusters in those datasets as well as in *comp* and *iris* datasets.

As for the frequency of correct estimation, *DCA1xx* detect the correct K^* with higher frequency than [108]. In *thyroid* and *wine* datasets, *DCA1xx* correctly detects K^* with notably higher frequencies than [108]. In *thyroid* dataset, the frequencies of *DCA1xx* vary from 88 to 98, while the frequency of [108] is 69. In *wine* dataset, the smallest frequency in all *DCA1xx* algorithms is 65 and the frequency of [108] is 47. In *comp* dataset, although [108] has higher frequency (86/100) than all *DCA1xx* algorithms, the difference is not significant compared to *DCA111* (81/100). *mclust* is quite stable in term of frequency (correct estimation or not). Except for dataset *comp*, *mclust* always gives the same K over 100 runs.

We observe that the presence of feature selection and sparsity penalization on precision matrices, e.g. $\lambda_2, \lambda_3 > 0$, helps to improve the frequency of correct detection of K^* in DCA algorithms. *DCA111* has highest frequencies of correctly detecting K^* among all *DCA1xx* algorithms in most of datasets (except for *wine* and *thyroid*). Algorithms with covariance sparsity (*DCA1x1*) not only prevent ill-conditioned matrices problem (*dermatology*, *glass* and *zoo* datasets), but also increase the frequency of correct detection of K^*. Naturally, dealing with multiple penalizations increases the computational time, *DCA111* is the slowest among *DCA1xx*.

Concerning ARI, all *DCA1xx* algorithms furnish significant better results than [108] and *mclust* for all datasets, except for *comp* dataset where [108] is slightly better.

Overall, the class of DCA1xx algorithms can detect the correct number of clusters in all considered datasets with high frenquency.

10.5.3 Experiment 2 - Feature selection

In this experiment, we evaluate the capacity of algorithms in detecting the informative features to provide high classification accuracy. We will perform the experiment on synthetic datasets whose informative features are known a priori and also on real datasets.

Synthetic dataset. The synthetic dataset is generated as follows. First, we create three vectors of dimension 20, namely m_1, m_2, m_3. Three first elements of m_1 are set to 1 and the rest are 0. Similarly, the 4th, 5th and 6th elements of m_2 are 1 and the rest are 0. The element 7th, 8th and 9th of m_3 are equal to 1 and the rest are 0. Using these three vectors, we generate three corresponding multivariate Gaussians, each possesses m_i as its mean and the co-variance matrix is unit diagonal matrix. From each Gaussian, we randomly generate 100 data points. Thus, the synthetic dataset contains 300 data points evenly divided into 3 classes; each data point is represented by 20 features but only first 9 features are informative.

Comparative algorithm. For this experiment, we compare our algorithm *DCA010* ($\lambda_1 = \lambda_3 = 0, \lambda_2 > 0$) with *Zhou-010*, a modified EM algorithm presented in [260] for solving the GMM with l_1 regularization

$$\min_{\Theta \in \mathcal{D}} -L(\Theta) + \lambda_2 \sum_{k=1}^{K} \sum_{d=1}^{D} |\mu_{kd}| .$$

Experimental setting. We choose hyperparameter λ_2 as presented in Section 10.5.1. For each algorithm, we measure the Selected Feature percentage (SF), the True Positive Rate (TPR) and the False Positive Rate (FPR). TPR is the percentage of features selected which is informative over all informative features while FPR is the percentage of non-informative features selected over all non-informative features. As in previous experiment, we also report the ARI and the computational time. The results are recorded over 100 runs.

Numerical results of synthetic and benchmark dataset are given in Table 10.3a and Table 10.3b, respectively.

Comments on numerical results. In synthetic dataset, *Zhou-010* selects slightly less features than *DCA010* (54% vs 55%). However, *DCA010* selects more informative features (91.34%) than *Zhou-010* (71.11%). Consequently, *DCA010* selects less non-informative than *Zhou-010* (25.45% vs 40%). Concerning the ARI, *DCA010* gives 0.6723 which is significantly higher than 0.0663 of *Zhou-010*. Furthermore, *DCA010* is 25 faster than *Zhou-010* (4.1 vs 107.6 seconds).

In all benchmark datasets, *Zhou-010* fails to select features since it keeps all the features. We observe that, except for the *iris*, *DCA010* gives better ARI than *Zhou-010* why selecting less

TABLE 10.3 – Experiment 2- Selecting informative features. Bold values indicate best results.

Results	**DCA010**	**Zhou-010**
SF(%)	55 ± 7.4	**54 ± 15.57**
TPR(%)	**91.34 ± 7.46**	71.11 ± 18.59
FPR(%)	**25.45 ± 11.18**	40 ± 13.79
ARI	**0.6723 ± 0.0311**	0.0663 ± 0.0183
Time(s)	**4.1211 ± 0.03**	106.718 ± 1.4935

(a) Synthetic dataset

Datasets	**Results**	**DCA010**	**Zhou-010**
comp	SF(%)	**82 ± 6.01**	100 ± 0
	ARI	**0.973 ± 0.0021**	0.9279 ± 0.0015
	Time(s)	**6.1102 ± 1.5407**	641.808 ± 185.2588
ionosphere	SF(%)	**45.25 ± 4.13**	100 ± 0
	ARI	**0.7913 ± 0.009**	0.4089 ± 0
	Time(s)	**4.4182 ± 0.0945**	78.002 ± 0.0455
iris	SF(%)	**75 ± 0**	100 ± 0
	ARI	0.7432 ± 0.0031	**0.9039 ± 0**
	Time(s)	**0.3935 ± 0.0941**	24.618 ± 0.0638
thyroid	SF(%)	**60 ± 0**	100 ± 0
	ARI	**0.9387 ± 0.0032**	0.8933 ± 0
	Time(s)	**0.2049 ± 0.0096**	18.18 ± 0.0235

(b) Benchmark datasets

features. As for the computation time, *DCA-GMM* is significantly faster than *EM-GMM*, e.g. up to 105 times faster on *comp* dataset.

In summary, *DCA010* is better than *Zhou-010* in all three comparative criteria : percentage of selected feature, ARI and computation time. The results confirm again that non-convex approximation is better than convex approximation to deal with the sparse optimization.

10.5.4 Experiment 3 - Sparsity on precision matrices

In this experiment, we are interested on the sparsity of the obtained precision matrices. For this purpose, we set $\lambda_1 = 0$ (since we are not interested on model selection) and $\lambda_3 > 0$. We consider two cases where feature selection is used ($\lambda_2 > 0$) or not ($\lambda_2 = 0$). Hence, the two versions of DCA that we use for this experiment are *DCA001* and *DCA011*.

Comparative algorithms. The comparison of our algorithms are realized with two algorithms presented in [260] for solving the GMM with l_1 regularization, namely *Zhou-001* and *Zhou-011*. *Zhou-011* solves the following problem

$$\min_{\Theta \in \mathcal{D}} -L(\Theta) + \lambda_2 \sum_{k=1}^{K} \sum_{d=1}^{D} |\mu_{kd}| + \lambda_3 \sum_{k=1}^{K} \sum_{d=1}^{D} \sum_{j=1}^{D} |W_k(d,j)|$$

where $\lambda_2, \lambda_3 > 0$. *Zhou-001* deals with the above problem in the case where $\lambda_2 = 0$ and $\lambda_3 > 0$.

The comparative results are given in Table 10.4. Beside the percentage of selected feature (SF(%)), the ARI, the computation time, we report the the percentage of non-zero elements in precision matrices (Sparse(%)).

TABLE 10.4 – Experiment 3 - Evaluating sparsity on precision matrices. Bold values indicate best results.

Datasets	**Results**	**DCA001**	**Zhou-001**	**DCA011**	**Zhou-011**
comp	SF(%)	NA	NA	**70 ± 1.8**	100 ± 0
	Sparse(%)	71.5 ± 4.24	54.67 ± 1.25	65.33 ± 2.51	**53.73 ± 0.37**
	ARI	**0.9673 ± 0.0064**	0.8024 ± 0.1203	0.9653 ± 0.0063	0.8885 ± 0.0562
	Time(s)	**1.8356 ± 0.7638**	763.2 ± 181.6157	35.0387 ± 2.52	781.568 ± 392.2629
ionosphere	SF(%)	NA	NA	**42.19 ± 4.1**	93.75 ± 0
	Sparse(%)	36.87 ± 5.75	**24.12 ± 0**	25.79 ± 4.6	**24.12 ± 0**
	ARI	**0.8741 ± 0.0069**	0.0316 ± 0	0.8559 ± 0.0157	0.0257 ± 0
	Time(s)	**0.058 ± 0.0041**	43.828 ± 0.1501	14.582 ± 1.42	49.096 ± 0.0483
iris	SF(%)	NA	NA	**100 ± 0**	**100 ± 0**
	Sparse(%)	57.29 ± 0.87	**25 ± 0**	62.5 ± 0	**25 ± 0**
	ARI	**0.9686 ± 0.0027**	0.868 ± 0	0.9575 ± 0.0019	0.8341 ± 0
	Time(s)	**0.1535 ± 0.031**	13.68 ± 0.0141	0.7813 ± 0.0145	12.19 ± 0.0543
thyroid	SF(%)	NA	NA	**60 ± 2.12**	100 ± 0
	Sparse(%)	63.33 ± 12.83	**22.67 ± 0**	73.33 ± 7.29	**22.67 ± 0**
	ARI	**0.9446 ± 0.0152**	0.8763 ± 0	0.9436 ± 0.0181	0.8763 ± 0
	Time(s)	**0.1121 ± 0.0213**	8.09 ± 0.3365	0.8824 ± 0.0124	11.05 ± 0.0843
wine	SF(%)	NA	NA	**33.33 ± 1.71**	100 ± 0
	Sparse(%)	61.01 ± 3.42	**3.98 ± 0**	54.53 ± 0.64	**3.98 ± 0**
	ARI	0.9756 ± 0.0083	0.8951 ± 0	**0.9759 ± 0.0092**	0.8783 ± 0
	Time(s)	**0.7385 ± 0.0242**	9.936 ± 0.4991	17.3725 ± 2.6253	17.456 ± 0.0365
dermatology	SF(%)	NA	NA	**39.71 ± 2.2**	NA
	Sparse(%)	**14.63 ± 1.92**	NA	23.85 ± 1.32	NA
	ARI	0.9591 ± 0.0049	NA	**0.9632 ± 0.0084**	NA
	Time(s)	**0.6241 ± 0.0557**	NA	37.2428 ± 3.1934	NA
glass	SF(%)	NA	NA	**75 ± 7.86**	NA
	Sparse(%)	18.31 ± 20.14	NA	**16.05 ± 11**	NA
	ARI	**0.7036 ± 0.0318**	NA	0.7031 ± 0.0287	NA
	Time(s)	**0.4138 ± 0.0829**	NA	3.1917 ± 0.5829	NA
zoo	SF(%)	NA	NA	**45.31 ± 16.2**	NA
	Sparse(%)	**19.85 ± 1.02**	NA	67.76 ± 5.03	NA
	ARI	**0.9872 ± 0.0005**	NA	0.9833 ± 0.003	NA
	Time(s)	**0.0637 ± 0.0024**	NA	12.5823 ± 2.1449	NA

Zhou-001 and *Zhou-011* fail to furnish a result on three datasets *dermatology*, *glass* and *zoo* datasets. Note that we do not report the SF for*DCA001* and *Zhou-001* since they do not deal with feature selection.

Comments on numerical results. In term of feature selection, *Zhou-011* selects all features in most of datasets, except for *ionosphere* where 93.75% of features are selected while *DCA011* only use 42.19%. Overall, *DCA011* selects relatively small number of features in *wine* (33.33%), *dermatology* (39.71%) and zoo (45.31%), *ionosphere* (42.19%).

Regarding the sparsity of precision matrices, *Zhou-001* and *Zhou-011* are quite similar with the same values of precision matrices sparsity in almost datasets. The percentages of non-zero elements in precision matrices of *Zhou-001* and *Zhou-011* are notably smaller than *DCA001* and *DCA011*. Despite of that, ARIs of *DCA001* and *DCA011* are comparable and significantly higher than both *Zhou-001* and *Zhou-011*. For instance, in *wine* dataset, ARI of *DCA011* is 0.9759 and ARI of *Zhou-001* is 0.8951. In *iris* dataset, ARI of *DCA001* is 0.9686 while ARI of *Zhou-001* is 0.868. The differences in ARI and sparsity on precision matrices shows that *Zhou-001* and *Zhou-011* algorithms neglected the correlated features, therefore having small sparsity but also reducing ARI. On the other hand, DCA-Like algorithms do not eliminate too much elements on precision matrices, maintaining an appropriate balance between ARI and sparsity.

10.5.5 Experiment 4 : Evaluating Two-Step DCA-Like

In this experiment, we study the efficiency of Two-Step DCA-Like. We benchmark Two-Step DCA-Like algorithms *DCA201* and *DCA211*. For comparison, we take into account *mclust* package. We choose hyperparameters as in section 10.5.1. Each algorithm is performed 100 times and we report the average results in Table 10.5.

TABLE 10.5 – Experiment 4 - Evaluating Two-Step DCA-Like. Bold values indicate best results.

Datasets	Results	DCA201	DCA211	mclust
comp	ARI	0.9634 ± 0.0059	**0.9641 ± 0.0075**	0.9285 ± 0.001
	Time(s)	**2.6356 ± 0.8018**	43.6922 ± 2.9183	7.1656 ± 0.909
ionosphere	ARI	0.8694 ± 0.0077	**0.9062 ± 0.0385**	0.3461 ± 0
	Time(s)	**0.0787 ± 0.0089**	15.5862 ± 1.2646	42.7856 ± 1.58
iris	ARI	**0.9672 ± 0.0029**	0.9576 ± 0.0018	0.9039 ± 0
	Time(s)	0.1718 ± 0.0283	0.9769 ± 0.0139	**0.0922 ± 0.0097**
thyroid	ARI	**0.9472 ± 0.0157**	0.9437 ± 0.0189	0.8925 ± 0
	Time(s)	**0.17 ± 0.0199**	1.5208 ± 0.0121	0.2822 ± 0.0083
wine	ARI	**0.9801 ± 0.0079**	0.9782 ± 0.0093	0.9306 ± 0
	Time(s)	**0.9642 ± 0.0235**	21.7878 ± 2.5063	12.4456 ± 1.06
dermatology	ARI	0.9619 ± 0.0053	**0.9742 ± 0.0084**	0.8996 ± 0
	Time(s)	0.9382 ± 0.0556	49.631 ± 3.2456	**0.2833 ± 0.0071**
glass	ARI	**0.7107 ± 0.0283**	0.7004 ± 0.0309	0.131 ± 0
	Time(s)	0.7338 ± 0.0855	4.9222 ± 0.5923	**0.1489 ± 0.0078**
zoo	ARI	0.9867 ± 0.0005	**0.9874 ± 0.003**	0.5238 ± 0
	Time(s)	0.0953 ± 0.0026	14.3507 ± 2.0404	**0.0256 ± 0.0073**

Comments on numerical results. Both *DCA201* and *DCA211* have higher ARI than *mclust* in all datasets. The ARIs of Two-Step algorithms are also significantly higher than their One-step versions that we experimented in previous sections. This shows that removing the penalty term of π in the second step improves the quality of the solution. *DCA201* and *DCA211* are comparable in term of ARIs and by far better than *mclust*. As for running time of *DCA201* is notably better than *DCA211* since the feature selection procedure of *DCA211* adds considerable time to the algorithm.

Overall, we can conclude that Two-Step DCA-Like (Algorithm 10.3) allows improve the efficiency of DCA-Like (Algorithm 10.2).

10.6 Conclusion

In this chapter, we have studied three fundamental issues GMM clustering : the model selection, the feature selection and the over-parameterization. For the first time, we presented an unified model that considers all these three issues at the same time. The model selection is done though a sparse regularization of the mixing proportion π_k. Similarly, a sparse regularization of μ (resp. of W) is used for the feature selection (resp. the over-parameterization).

We approximated the l_0-norm by the concave exponential function. The resulting problem can be then recast as a DC program. However, the corresponding standard DCA seems to not be efficient since the sub-problem requires high effort in computation.

On the one hand, it is well-known that a convex upper bound of $-L(\Theta)$ ($L(Theta)$ is the log-likelihood function) can be easily computed. On the other hand, DCA can effectively handle the non-convex regularization term. This observation gives rise to a incorporated method, namely DCA-Like, between upper-bounding $-L$ by the convex function and treating the regularization term by DCA. DCA-Like can work even when we can not highlight a DC decomposition. Furthermore, we propose a Two-Step DCA-Like in order to improve the performance of DCA-Like. The first step aims to find the correct number of clusters and then we remove the regularization of π in the second step to improve the clustering results.

We have carefully conducted several numerical experiments on both synthetic and Benchmark datasets to illustrate the effectiveness of our algorithms. The results have shown that our algorithms are efficient and outperformed existing methods.

Chapitre 11

Time-series clustering. Application on customer clustering of French transmission system operator (RTE) based on their electricity consumption[1]

Abstract: We tackle three crucial issues in high-dimensional time-series data clustering for pattern discovery : appropriate similarity measures, efficient procedures for high-dimensional setting, and fast/scalable clustering algorithms. For that purpose, we use the DTW (Dynamic Time Warping) distance in the original time-series data space, the t-distributed stochastic neighbor embedding (t-SNE) method to transform the high-dimensional time-series data into a lower dimensional space, and DCA based clustering algorithms. As application, we applied our approach for customer clustering of French transmission system operator (RTE) based on their electricity consumption. The ultimate goal of customer clustering is to automatically detect patterns for understanding the behaviors of customers in their evolution. It will allow RTE to better know its customers and consequently to propose them more adequate services, to optimize the maintenance schedule, to reduce costs, etc. The numerical results on real-data of RTE's customer have shown that our clustering result is coherent : customers in the same group have similar consumption curves and the dissimilarity between customers of different groups are quite clear. Furthermore, our method is able to detect whether or not a customer changes his way of consuming.

11.1 Introduction

In recent years, due to an exponential growth of the time-series data applications in emerging areas such as sale data, finance, weather, ..., there have been considerable research and developments in time-series clustering. Time-series clustering is a hard task due to the following two main difficulties. The first lies in the nature of temporal information in time-series. More precisely, while evaluating the similarity between time-series objects, the chosen similarity measure should be able to take into account the temporal information of the considered time-series data. The second main difficulty concerns the high-dimensional nature of time-series

1. The results presented in this chapter were published in :
   - G. Da Silva, H.M. Le, H.A. Le Thi, V. Lefieux, B. Tran, Customer Clustering of French Transmission System Operator (RTE) Based on Their Electricity Consumption, Advances in Intelligent Systems and Computing 991, 893-905, 2019.

data. A high number of dimensions leads to great increases in the computation time of clustering algorithms. Furthermore, clustering techniques often suffer from the "curse of dimensionality" phenomenon, say the quality of clustering algorithms is degraded as the dimension of data increases. Hence, for developing an efficient time-series clustering algorithm, one has to deal with three important issues : appropriate similarity measures for time-series data, efficient procedures for high-dimensional setting, and fast/scalable clustering algorithms. That is the purpose of our work.

We will apply our time series clustering method for the clustering of customers of RTE (French transmission system operator) based on their electricity consumption. RTE is in charge of the high voltage grid for electricity in France. As a smart grid, RTE is responsible for the balance of production and consumption, the safety of transportation and the quality of the delivered services. The main objective of customer clustering task is to automatically detect patterns and find casualties of customers in their evolution. The results will help RTE to better know its customers and to propose them more adequate services. To speak in marketing terms, they will allow a better adaption of maintenance schedule, a smooth preparation for real-time operations and a cost reduction. Understanding more precisely the behaviors of customers on the grid is one more step towards a smart grid.

Each RTE's customer is characterized by his electricity consumption curves which contain the electricity consumption of each 10 minutes over two years. Hence, each customer is represented by a time-series sequence of $105,120$ points. We are undoubtedly facing a very large-scale time-series clustering problem.

The remainder of the chapter is organized as follows. The proposed approach is developed in Section 11.2 while the experiment and the result analysis are reported in Section 11.3. Finally, Section 11.4 concludes the chapter.

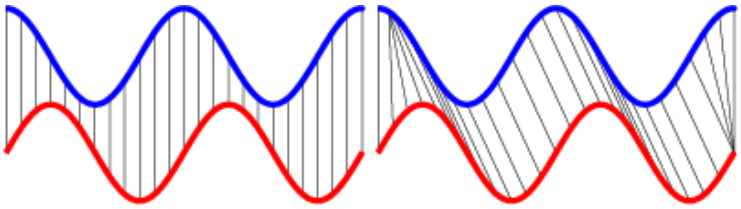
11.2 The proposed high-dimensional time-series data clustering approach

In the development of our solution method, the following questions are crucial (to which the answers are not independent) : which similarity measure to be considered ? which clustering algorithm should be investigated ? and how to deal with high-dimensional data ?

11.2.1 DTW distance : a suitable similarity measure for time-series data

Similarity measure is a major challenge in time-series clustering. A suitable choice of distance measure depends mainly on the objective of clustering task, and on the characteristic as well as the length of time-series. In this section, we will study an appropriate similarity measure in high-dimensional time-series data clustering for pattern discovery. A large number of similarity measures for time-series data have been proposed in the literature. The readers are referred to the surveys [3] and [234] for a more complete list of similarity measures for time-series data. Generally speaking, they can be divided into two main categories : lock-step measures (one-to-one) like ℓ_p-distance and elastic measures (one-to-many/one-to-none) including DTW distance, Longest Common Sub-sequence distance, Probability-based distance [234], etc. The one-to-one distances, as the name suggests, compare the two time-series point by point. However, in problems where one wants to catch similar patterns of time-series that do not occur at the same moment, the one-to-one distances are not adapted. The Figure 11.1 (a) perfectly illustrates the drawback of one-to-one distance in this case. As we can see, the Euclidean distance of two time-series is

high despite the fact that the two time-series have the same shape (the second time-series is nothing else but a horizontally shifted transformation of the first one).



(a) Euclidean Distance (b) DWT Distance

FIGURE 11.1 – Euclidean and DTW distance of two very-similar time-series (red and blue curves). The black line shows the matching $\{p_l = (n_l, m_l)\}_{l=1,\dots,L}$ between points between two time-series.

In contrast to one-to-one measures, the main difference and superiority of elastic measures lie in their ability to handle temporal drift/shifting in time-series. Among all the existing elastic measures, Dynamic Time Warping (DTW) distance has been proved to be appropriate in several applications involving time-series data [251, 1, 54]. The DTW distance of two time-series $x \in r^{T_x}$ and $y \in r^{T_y}$ can be defined as follows [166]. Let (N, M)-warping path be a sequence $p = (p_1, \dots, p_L)$ with $p_l = (n_l, m_l) \in [1, N] \times [1, M]$ for $l = 1, \dots, L$. A valid DTW path (N, M) also needs to satisfy (1) the boundary condition ($p_1 = (1, 1)$ and $p_L = (N, M) = (T_x, T_y)$); (2) the monitonicity condition ($n_1 \leq n_2 \leq \dots \leq n_L$ and $m_1 \leq m_2 \leq \dots \leq m_L$); and (3) the step-size condition ($p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in \{1, \dots, L - 1\}$). Hence, the DTW distance between two time-series x and y is given by

$$d_{\mathrm{DTW}}(x, y) = \min \left\{ \sum_{l=1}^{L} |x_{n_l} - y_{m_l}| \;\middle|\; p \text{ is a valid (N, M)-warping path} \right\}. \tag{11.1}$$

DTW minimizes the differences between two times-series by aligning each point from one time-series to the best corresponding points in the other time-series by a warping path. Hence DTW is enough flexible to handle the shifting between time-series, and is able to catch the similar patterns of two time-series sequences. Therefore, the DTW distance is chosen for our time-series clustering approach.
However, the main drawback of DTW distance is its computation time. The algorithm for computing the DTW distance is a recursive algorithm with complexity of $\mathcal{O}(T^2)$ where T is the length of time-series. Hence, for high-dimensional time-series data, it is very slow, even impossible to use directly DTW distance in clustering algorithms involving the computation of DTW at each iteration (e.g. k-means and its variants). To overcome this drawback of DTW, we adopt a feature-based clustering approach. Clustering time-series is usually tackled by two approaches : the raw-data-based, where clustering is directly applied over time-series vectors without any space-transformation previous to the clustering phase, and the feature based approach which does not directly perform clustering on the time-series raw data.

11.2.2 A feature-based clustering approach for high-dimensional time-series data

Feature-based clustering approach consists of two main steps : (a) transform the time-series into feature vectors in a smaller dimensional space, then (b) perform clustering algorithm on the transformed data. We will describe below an efficient data transformation algorithm to the first step.

11.2.2.1 Data transformation by t-SNE

In the literature, several data transformation algorithms for time-series data have been developed. The most well-known one is certainly the Fourier and Wavelet transformation. The Fourier transformation decomposes the time-series into a sum of sinusoids, which allows us working in frequency-domain instead of raw data ; whereas Wavelet transformation uses a different basic function (not necessarily sinusoids). Recently, Schafer et al. [209] proposed Bag-of-SFA-Symbol (BOSS), an advanced feature extraction algorithm for time-series, which combines Fourier transformation on sliding windows with Bag-of-words to extract the characteristic from the time-series. The authors have shown that BOSS performed better than existing transformations [209, 17].

In this work, we will consider t-distributed stochastic neighbor embedding (t-SNE), a relative new algorithm based on a completely different idea than other classical algorithms like Principal Component Analysis (PCA), Non-negative Matrix Factorization (NMF) and above-mentioned algorithms. t-SNE was first introduced by Maateen et al. [161] as a dimensional reduction algorithm for data visualization. The t-SNE transforms data-points from the original space into a new space (normally lower-dimensional space) such that the probability of two data-points be in the same cluster in the original space is equal to the probability that their transformations be in the same cluster in the new space. The reader are referred to Chapter 3 Section 3.4 for a description and mathematical formulation of t-SNE. Recall that in Section 3.4, we have developed DCA-Like to solve t-SNE and showed that DCA-Like outperformed all existing methods.

t-SNE offers the liberty to choose the similarity measures d_{original} (resp. d_{new}) in original (resp. new space). In the original work [161], the authors applied t-SNE to an application in data visualization where the number of dimensions in the new space is low (2 or 3), with both d_{original} and d_{new} are Euclidean distance. In our case, we consider d_{original} as the DTW distance while the Euclidean distance is chosen for d_{new}. On the one hand, it is obvious that d_{original} should be DTW distance since it is well adapted for time-series data as we have shown in Sub-Section 11.2.1. On the other hand, the choice of Euclidean distance for d_{new} is motivated by the existence of several efficient, scalable and robust clustering algorithms based on DCA via the Euclidean distance (e.g. DCA-MSSC [123], DCA-KMSSC [129]). To the best of our knowledge, this is the first time t-SNE is used with DTW distance.

The time-series data are now transformed into a new space by t-SNE. In the next Section, we will study some efficient clustering algorithms for the transformed data.

11.2.2.2 Fast and scalable DCA based clustering algorithms

As we have mentioned previously, in our problem, except the time series curves, we do not have any information on the clusters, nor the number of clusters. Finding the number of clusters is challenging for clustering tasks. Generally, there exist two approaches. The first approach consists in finding firstly the number of clusters with a "simple" procedure then apply a clustering algorithm with the number of clusters found previously. In the second approach, one simultaneously determines the number of clusters and clustering assignment.

In the literature, several algorithms have been developed for finding the number of clusters. For instance, Elbow algorithm which uses the WSS criterion ("total within-cluster sum of square") to determine the number of clusters. The number k^* of clusters is optimal if the corresponding WSS does not change significantly when increases the number of clusters by 1. Silhouette Average algorithm is similar to Elbow algorithm. This algorithm varies the number of clusters and chooses the one that maximizes the Silhouette criterion. Gap Statistic algorithm [224] is another variant of Elbow algorithm. Gap Statistic algorithm maximize "gap statistic" criterion,

which is defined by the difference between the measured WSS and its expected value under some conditions.

Assuming that the number of clusters is known, there exists a variety of Euclidean-based clustering algorithms such as k-means, k-medoids, fuzzy c-means, etc. Among many models for clustering, the Minimum Sum-of-Squares (MSSC) is one of the most popular since it expresses both homogeneity and separation (c.f. Chapter 7).

Le Thi et al. [123] have developed DCA-MSSC, an efficient algorithm based on DCA for solving the MSSC model. DCA-MSSC have shown its superior in comparison with state-of-the-art algorithms : performance, robustness, and adaptation to different types of data. We refer to the original paper [123] for more details of DCA-MSSC algorithm.

On the other hand, among the algorithms that simultaneously determines the number of cluster and clustering assignment, mclust [212] is a well-known one. mclust uses the Gaussian Mixture Model. The optimal number of segments K^* is determined by the Bayesian Information Criterion (BIC) and Integrated Complete-data Likelihood (ICL). In a different direction, Le Thi et al. [138] have proposed the DCA-Modularity algorithm. DCA-Modularity transforms the data-points into a graph then segments vertices of the graph using the modularity criterion as a measure of clustering quality. The problem of maximization of graph modularity is summarized as follows. Consider an undirected unweighted network $G = (V, E)$ with N nodes ($V = \{1, \ldots, N\}$) and M edges ($M = Card(E)$). Denote by the adjacency matrix $A : a_{i,j} = 1$ if $(i, j) \in E$, 0 otherwise. The degree of node i is denoted ω_i ($\omega_i = \sum_{j=1}^{N} a_{i,j}$), and ω stands for the vector whose components are ω_i. Let P be a partition of V, and K is the number of communities in P. Define the binary assignment $U = (u_{i,k})_{i=1,\ldots,N}^{k=1,\ldots,K}$, says $u_{i,k} = 1$ if vertex i belongs to community k and 0 otherwise. Then, the modularity maximization problem can be written as

$$\begin{aligned} \max_{U} \quad & Q(U) := \tfrac{1}{2M} \textstyle\sum_{i,j=1}^{N} b_{i,j} \sum_{k=1}^{K} u_{i,k} u_{j,k}, \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^{K} u_{i,k} = 1, \text{ for } i = 1, \ldots, N; \\ & u_{i,k} \in \{0, 1\}, \text{ for } i = 1, \ldots, N, \qquad k = 1, \ldots, K; \end{aligned} \tag{11.2}$$

where $B := (b_{i,j})_{i,j=1,\ldots,N} = A - \frac{1}{2M} \omega \omega^T$ is a constant matrix, called the modularity matrix. The problem (11.2) is a mixed-binary optimization problem for which Le Thi et al. [138] have proposed DCA-Modularity. Le Thi et al. [138] have proved that DCA-Modularity is able to give the right number of clusters as well as a good clustering assignment on several benchmark datasets. The readers are referred to the original paper [138] for more details of DCA-Modularity algorithm.

Motivated by the success of DCA-MSSC and DCA-Modularity, we will adopt both of them for our clustering method. Precisely, DCA-Modularity will be used to determine the number of clusters and a good staring clustering assignment. Based on the clustering assignment given by DCA-Modularity, DCA-MSSC focus on improving the clustering assignment. This combination allows us to take advantage of both DCA-MSSC and DCA-Modularity. According to all above in-depth studies, we are going to describe below our solution method for clustering of RTE's customers based on their electricity consumption.

11.2.3 Description of the main algorithm

Our proposed method (c.f. Figure 11.2) consists of several steps : data processing, data transformation and clustering. The first step, data processing, deals with the noisy and outlier data caused by erroneous measurement at electric meter. In the second step, data transformation, we transform the high-dimensional time-series data into a lower dimensional space using the

t-SNE algorithm with DTW distance. As for the clustering algorithm, we combine the DCA-Modularity [138] and DCA-MSSC [123]. DCA-Modularity [138] is used to compute the number of clusters as well as a good starting point for the clustering algorithm DCA-MSSC. Since we are

FIGURE 11.2 – The proposed time-series clustering method's pipeline.

interested in detecting similar patterns of time-series data, the transformation must be *scaling and translation invariant*. Hence, the z-normalization [83] is employed for normalizing time-series data. The effectiveness of this well-known transformation has been proved in several works [83, 199, 183]. Given a time-series a of length $T_a : (a_t)_{t=1,\dots,T_a}$, the z-normalization of a is computed as

$$a_t^{\text{norm}} = \frac{a_t - \bar{a}}{\sigma(a)} \text{ for } t \in \{1, \dots, T_a\} \tag{11.3}$$

where $\bar{a}$ and $\sigma(a)$ are mean and standard-deviation of a : $\bar{a} = \frac{1}{T_a}\sum_{t=1}^{T_a} a_t$ and $\sigma(a) = \sqrt{\frac{1}{T_a}\sum_{t=1}^{T_a}(a_t - \bar{a})}$.

The proposed method for customer clustering is summarized in Algorithm 11.1.

Algorithm 11.1 Proposed algorithm for clustering time-series

1: Input : N time-series $a_i, \dots, a_N$.
2: Output : Clustering assignment $p^* = (p_i^*)_{i=1,\dots,N}$ where p_i^* is the cluster of a_i.
3: **Step 1 :** Z-Normalization transformation.
4: Input : N time-series $a_1, \dots, a_N$ in $\mathbb{R}^T$.
5: Output : N normalized time-series $\bar{a_1}, \dots, \bar{a_N}$ in $\mathbb{R}^T$.
6: **Step 2 :** t-SNE DTW-Euclidean transformation.
7: Input : N normalized time-series $\bar{a_i}, \dots, \bar{a_N}$ in $\mathbb{R}^T$.
8: Output : N vectors $x_1, \dots, x_N$ in new space $\mathbb{R}^D$ where x_i is the corresponding transformation of a_i in the new space.
9: **Step 3 :** Clustering algorithm
10: **Step 3.1 :** DCA-Modularity clustering [138].
11: Input : N vectors $x_1, \dots, x_N$ in $\mathbb{R}^D$.
12: Output : The number of clusters K^*, a clustering assignment $p^0 = (p_i^0)_{i=1,\dots,N}$.
13: **Step 3.2 :** DCA-MSSC clustering [123].
14: Input : N vectors $x_1, \dots, x_N$ in $\mathbb{R}^D$, the number of clusters K^*, the clustering assignment $p^0 = (p_i^0)_{i=1,\dots,N}$.
15: Output : Clustering assignment $p^* := (p_i^*)_{i=1,\dots,N}$.

11.3 Numerical experiments

Our experiments are realized on a dataset which contains 462 customer's electricity consummation curves of RTE. For a confidential reason, each client is named by a randomly generated

number. The consumption curve contains the electricity consumption of each 10 minutes over two years (from 01 January 2016 to 31 December 2017).

The code was written in C# 4.7.1. All experiments are conducted on an Intel(R) Xeon(R) E5-2630v4 (40 CPUs) with 32 GB of RAM.

Experiment 1 : We first analyze the relevance of our clustering result. For this purpose, we perform the Algorithm 11.1 on the whole dataset. It is worth to note that the computation time of our method is only 42 minutes in total. It comes out that the number of clusters determined by our algorithm is 19. In Figure 11.3, we report the number of customers in each cluster.

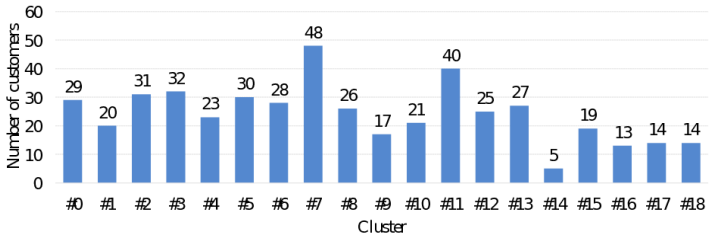


FIGURE 11.3 – Number of customers by clusters.

We will analyze the result from three clusters. The choice is solely based on the number of customers in each cluster : the biggest cluster (cluster C7), followed by a medium-size one (cluster C3, which is 25% smaller than cluster C7) and one small-size cluster (cluster C17, which is the 4th smallest cluster). The consumption curve of four arbitrarily chosen customers from each cluster are presented in Figure 11.4 (cluster C3), Figure 11.5 (cluster C7) and Figure 11.6 (cluster C17).

We observe that customers in each cluster clearly have similar shapes. For cluster C7, customers tend to have a "regular" consumption pattern : the consumption is high and followed by a short "drop", i.e. the consumption suddenly tumbles to a small value, in comparison with the consumption level of previous period), which repeats during the year. Further analysis reveals that this is a typical "weekly" consumption pattern, where the drops often happen during the weekend. In addition, they also have a long drop in consumption for $2-3$ weeks around the middle of August, and a shorter drop at the end of the year. For cluster C17, as we can see, all four customers have a low electricity consumption (all are around 0), despite the differences in their maximum consumption. They frequently generate very high peaks in a short duration during the whole year. Customers in cluster C3 have a stable consumption during the whole year (mostly varies around a base) and rarely have long "drop" during the year. They often have short drops in consumption (as oppose to short "peaks" in cluster C17).

From the Figure 11.4, Figure 11.5 and Figure 11.6, we can conclude that (1) the consumption curve of customers in the same clusters are coherent and (2) the differences of customers between clusters are quite clear.

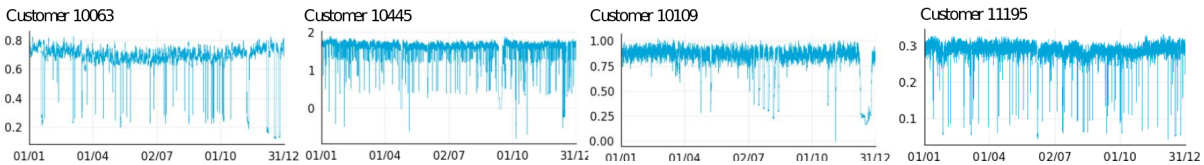


FIGURE 11.4 – Some consumption curves of customers from cluster C3.

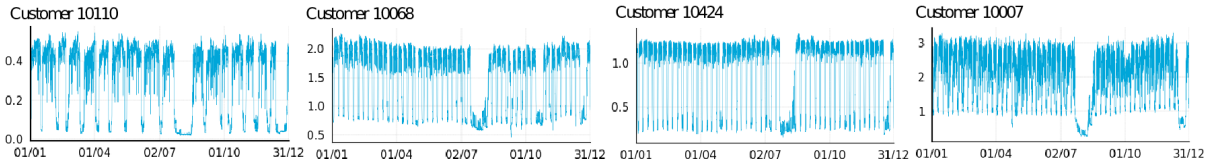


FIGURE 11.5 – Some consumption curves of customers from cluster C7.

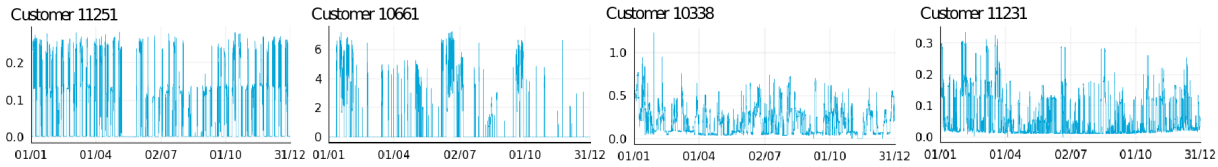
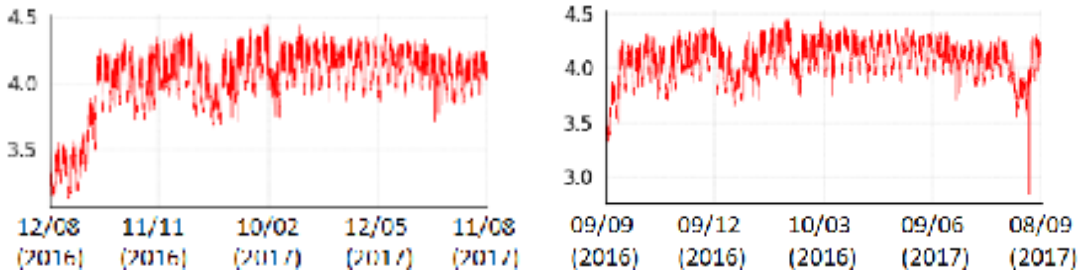


FIGURE 11.6 – Some consumption curves of customers from cluster C17.

Experiment 2 : In this experiment, we are interested in the capacity of our method to detect if a customer changes his way of consuming. For this purpose, we apply a Sliding Window technique with slide duration of four weeks, thus we obtain 13 different one-year-window datasets. Each one-year window datasets is processed by Algorithm 11.1.

We now analyze a customer whose consumption behavior changes over time. Consider the case of customer 10010. As we can see in Figure 11.7, this customer has a sharp drop in consumption in August 2015 while there is a much smaller decline in August 2016. This customer has therefore clearly changed his mode of consumption. In Figure 11.8, we show the clusters to which customer 10010 belongs during the 12 monthly runs of our segmentation algorithm. We see that up to the 12/08/2016, customer 10010 belongs to cluster C2. As it was detected by our algorithm, this customer changes his mode of consumption in August 2016. By 09/09/2016, customer 10010 is assigned to a new cluster (cluster C7).

In Figure 11.9 and Figure 11.10, we show some customers in cluster C2 and cluster C7. We observe the similarities between the load curve of customer 10010 from August 12th, 2015 to August 11th, 2016 (Figure 11.7a) and those of customers in cluster C2 (Figure 11.9). This same remark is valid between customer 10010's consumption curve from September 9th, 2015 to September 8th, 2016 (Figure 11.7b) and other customers of cluster C7 (Figure 11.10).



(a) August 12th, 2016 (b) September 09th, 2016

FIGURE 11.7 – The consumption of customer 10010 from 12/08/2015 to 11/08/2016 (left figure) and from 09/09/2015 to 08/09/2016 (right figure).

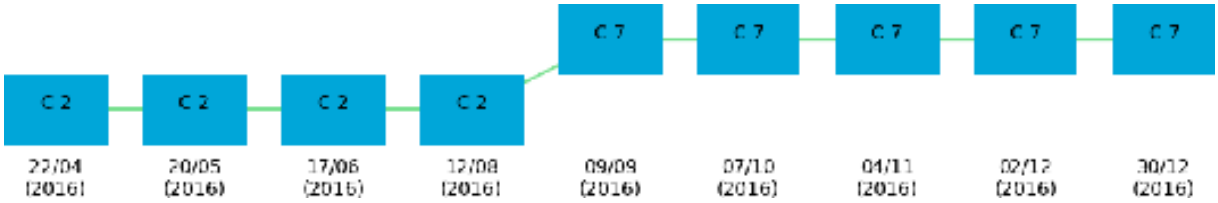


FIGURE 11.8 – The cluster of customer 10010 during nine runs. The first four runs' results are also C2, thus it is cropped out for visibility reason. The date (i.e. 22/04/2016) represents the staring date of the one-year window.

FIGURE 11.9 – Some customers from cluster C2 on 12/08/2016.

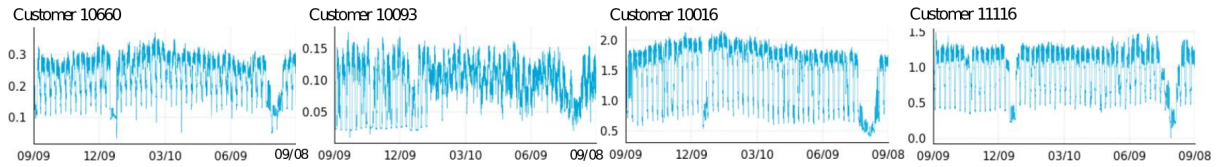


FIGURE 11.10 – Some customers from cluster C7 on 09/09/2016.

11.4 Conclusion

The proposed algorithm in this work is the result of in-depth studies using advanced theoretical and algorithmic tools for large-scale time-series data clustering. We have efficiently tackled the three challenges for our time-series clustering task : the similarity distance measure, the clustering algorithm, and the Big data. The innovative character intervenes in all stages of the proposed approach : the data transformation via t-SNE with the DTW measure in the original data space and the Euclidean distance in the transformed space is original. Indeed, on one hand, the DTW measure is appropriate to time-series clustering for pattern discovery. In another hand, the use of Euclidean distance in the transformed space allows us to investigate efficient clustering algorithms based on this distance. This transformation is very efficient to confront the high-dimensional data. For the first time the DTW distance is considered in the t-SNE transformation model and the resulting DCA scheme is particularly effective. Last but not least, the combination of two powerful clustering algorithms - DCA-Modularity and DCA-MSSC - is interesting, which gives rise to a performant method of clustering.

From a business point of view, this clustering has led to results that are already interpretable and useful for RTE. The possibility of monitoring potential customer developments is of particular interest to customer relationships managers, who can thus offer customized services.

Chapitre 12

Research Perspectives

This chapter presents the research directions that I would like to explore in the future. These perspectives are in accordance with my research developed thus far. I will continue to investigate new advanced techniques in DC programming and DCA, with a focus on stochastic optimization. From my best knowledge, the development of stochastic methods for non-convex stochastic optimization is still limited. Based on our previous works on Stochastic DCA, I aim to design novel efficient stochastic versions of DCA for other classes of stochastic optimization problems.

12.1 Introduction

Stochastic optimization refers to a wide class of programming methodologies and optimization algorithms where the randomness/uncertainty is present. It is arguably that stochastic optimization plays a significant role in almost all fields of applied sciences, in which it is the main tool to model, design complex systems as well as to tackle the resulting problems. In the era of unprecedented growth of data, stochastic optimization is a mean to resolve many associated challenges including data processing, storage bottleneck, noisy measurements, high dimensionality, etc.

A typical stochastic optimization problem takes the following form :

$$\begin{aligned} &\min f(x) := \mathbb{E}_\xi(F(x,\xi)) && \text{(SP)}\\ &\text{subject to} \quad x \in \mathcal{X}. \end{aligned}$$

There is a vast literature that studies the problem (SP) in the convex setting : $F(\cdot,\xi)$ are convex for all ξ, $\mathcal{X}$ is deterministic and convex. However, when either $F(\cdot,\xi)$ or $\mathcal{X}$ becomes non-convex, the problem becomes more difficult. More than that, the problem (SP) will be much more challenging if the constraint is given in a stochastic form, i.e., $\mathbb{E}_\xi(Q(x,\xi)) \leq 0$.

In our previous works, we have addressed the Stochastic DC program where the objective is DC under deterministic convex constraint. However, there are still many open questions. We aim firstly to extend our previous works on Stochastic DC program, e.g. study the non-asymptotic convergence properties of these algorithms. Secondly, we will study a larger class of problems, namely the Stochastic DC programs under Stochastic DC constraints. This class of problems is very large and is so far under studied.

12.2 Stochastic DC programs

Let $(\Omega, \Sigma_\Omega, \mathbb{P})$ be a probability space. Consider the Stochastic DC program :

$$\min\{f(x) := \Phi(x) + r(x) : \quad x \in \mathbb{R}^n\}, \qquad \text{(SDC)}$$

where $r : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a lower semicontinuous DC function given by

$$r(x) := r_1(x) - r_2(x), \quad x \in \mathbb{R}^n,$$

with $r_1, r_2 : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ being semicontinuous convex functions, and the expected loss function

$$\Phi(x) := G(x) - H(x) = \mathbb{E}_\xi(g(x,\xi)) - \mathbb{E}_\xi(h(x,\xi))$$

where $g(\cdot,\xi), h(\cdot,\xi), \xi \in \Omega$ are convex functions defined on $\mathbb{R}^n$.

In practice, the distribution of ξ is unknown by nature, so is Φ. We can, however, approximate the function Φ by Monte-Carlo's method. Given a large number of i.i.d. samples $\{\xi_1, \xi_2, \ldots, \xi_N\}$ obtained from the distribution of ξ, the function Φ can be approximated by the empirical loss

$$\Phi(x) \approx \Phi_{\text{emp}}(x) := \frac{1}{N}\sum_{i=1}^{N} g(x,\xi_i) - \frac{1}{N}\sum_{i=1}^{N} h(x,\xi_i),$$

which leads to the following approximation optimization problem

$$\min\{\bar{f}(x) := \Phi_{\text{emp}}(x) + r(x) : \quad x \in \mathbb{R}^n\}. \qquad \text{(ERM)}$$

The Stochastic DC program (SDC) and its empirical problem (ERM) are very challenging : they are non-convex, nonsmooth programs. The problem (SDC) is stochastic by nature while the problem (ERM) is a large-sum problem (N must be large for a good approximation). Consequently, stochastic approaches are more suitable than deterministic approaches to tackle these problems.

Literature Review. The current body of research on the problem (SDC) as well as (ERM) remains limited.

— Le Thi et al. (2017) [131] considered a large sum of L-smooth function with $\ell_{2,0}$ regularization term. The $\ell_{2,0}$ is then approximated by DC functions, which leads to a special case of (ERM). A stochastic DCA was developed and the authors established an asymptotic convergence to DC critical points (with probability 1).

— Le Thi et al. (2020) [133] considered the problem (ERM) directly, where no smoothness condition is required. A stochastic DCA was developed and the asymptotic convergence with probability 1 to DC critical points was proved.

— Le Thi et al. (2021) [136] studied the problem (SDC) over a compact convex set in the context of streaming data. An online stochastic DCA scheme was developed based on Sample Average Approximation (SAA). The asymptotic convergence with probability 1 to critical DC points was established.

— Le Thi et al. (2020) [127] developed several Stochastic DCA schemes for solving the general problem (SDC) in an incremental fashion. The convergence with probability 1 to DC critical points and the iteration convergence rate were proved.

— Liu et al. (2020) [158] studied the two-stage stochastic programming with linearly bi-parameterized quadratic recourse. The problem was then formulated in the form of (SDC) and the authors developed a stochastic scheme based on DCA.

— Nitanda and Suzuki (2017) [177] studied the problem (SDC) where $r = 0$, G and H are differentiable. The authors developed a proximal Stochastic DCA scheme and obtained the first non-asymptotic convergence result.

— Xu et al. (2019) [244] developed proximal stochastic DCA for solving the problem (SDC) as well as (ERM) where two cases of the regularizer r have been considered : r is convex or r is non-convex whose proximal operator is simple. Their meta-algorithm can be considered as the standard DCA, where the convex subproblems are solved by SPG [257] or Adagrad [69] (in case of problem (SDC)) or by SVRG [242] (in case of problem (ERM)).

Perspectives. In previous works, we have established the asymptotic convergence of Stochastic DCA to DC critical points. In order to evaluate theoretically the robustness of our algorithms, in future works, we aim to :

— Establish non-asymptotic convergence rate of these Stochastic DCA schemes. To be specific, we want to evaluate iteration convergence rate (note that, in [127] we have provided such a rate). Furthermore, the computation complexity of each iteration is taken into account (it consists of the complexity of computing the stochastic gradient and the complexity of the underlying convex solver for solving sub-problem). Overall, the total computation complexity will be established. Moreover, in machine learning, sample complexity is also a very important concept. The sample complexity measures how many i.i.d. samples need to use to obtain certain level of accuracy. We will study this kind of complexity.

— We will design new stochastic DCA schemes with competitive complexity comparing with existing results ([177, 244]).

12.3 Stochastic General DCA

We consider the General Stochastic DC problem, i.e. Stochastic DC programs with Stochastic DC constraints, as follows

$$\begin{aligned} &\min f_0(x) := \mathbb{E}_{\xi_0}(g_0(x,\xi_0)) - \mathbb{E}_{\xi_0}(h_0(x,\xi_0)) + r_0(x) - s_0(x), \\ &\text{subject to} \\ &f_i(x) := \mathbb{E}_{\xi_i}(g_i(x,\xi_i)) - \mathbb{E}_{\xi_i}(h_i(x,\xi_i)) + r_i(x) - s_i(x) \leq 0, \quad i = 1,2,\ldots,m. \end{aligned} \tag{GSDC}$$

where $g_i(\cdot,\xi_i), h_i(\cdot,\xi_i)$ are convex defined on $\mathbb{R}^n$, r_i, s_i are lower semicontinuous convex functions : $\mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$.

The problem (GSDC) is really challenging as both the objective and constraints are given in form of Stochastic DC. To our best knowledge, there is no existing work in the literature that addresses this problem. Liu et al. (2019) [157] and Ye and Cui (2019) [249] are the only works that study a stochastic non-convex program with non-convex constraints. More precisely, they consider the following stochastic (non-convex) L-smooth programs with stochastic L-smooth constraints :

$$\begin{array}{l} \min f_0(x) := \mathbb{E}_\xi(p_0(x,\xi)), \\ \text{subject to} \\ f_i(x) := \mathbb{E}_\xi(p_i(x,\xi)) \leq 0, \quad i = 1,2,\ldots,m, \end{array} \qquad \text{(SLC)}$$

where $p_i(\cdot,\xi)$ are L-smooth. Clearly, the class of (SLC) programs is strictly contained in the class of (GSDC) programs.

Even with the (SLC) problem, there are many difficulties that have been encountered. For example, as the constraints are non-convex and stochastic, how can we design an algorithm that ensures the feasibility of such constraints. Moreover, how to prove the convergence with probability one to critical/stationary points ?

Challenges of (GSDC)

— The DC structures of both objective and constraints.

— The stochastic nature of both objective and constraints : we can only access the objective and constraints through noisy random observations.

— How to design an iterative algorithm whose subproblems are feasible at each iteration and the generated sequence converges to a feasible point of (GSDC) ?

Perspectives. In the deterministic context where both objective and constraints are (deterministic) DC functions, General DCA ([125]) can be developped. Based on this spirit, we aim to extend the General DCA for the stochastic setting to handle the problem (GSDC). Then, we will study both asysmptotic and non-asymptotic convergence results where (computation/sample) complexities of the proposed algorithms will be analyzed. Finally, we will apply the newly proposed algorithms to solve some important stochastic optimization problems such as chance constrained programs, interference networks in Telecommunications, Neyman-Pearson classification in machine learning, etc.

Two approaches can be considered for solving (GSDC) : penalty technique and feasible point pursuit.

Approach 1 : Penalty technique. Let $p(x) := \max\{f_1(x), f_2(x), \ldots, f_m(x)\}$ and $p^+(x) = \max\{p(x), 0\}$. Then we can reformulate the problem (GSDC) by using the penalty technique as follows

$$\min\{\varphi(x) := f_0(x) + \beta p^+(x)\} \qquad \text{(Pen-GSDC)}$$

The problem (Pen-GSDC) is an unconstrained Stochastic DC program for which we can employ directly, as well as further develop, our existing Stochastic DCA schemes to tackle.

Some important open questions :

— When does the exact penalty hold, i.e., the problem (Pen-GSDC) is equivalent to (GSDC) in the sense that they have the same optimal value and the same solution set ?

— Even when (Pen-GSDC) is equivalent to (GSDC) in terms of globality, the connections of these two problems in terms of locality/stationarity/criticality should be established since iterative algorithms can only - in theory - find the latter kind of points.

— How to develop good algorithms to solve (Pen-GSDC) ?

Approach 2 : Feasible Point Pursuit. In this approach, we aim to convexify each DC constraint by linearizing their concave part. By this way, at each iteration we will obtain a Stochastic DC programs over a (stochastic) convex set. The stochastic convex constraints can be further approximated by the sample convex constraints. However, these convex constraints can be infeasible. To avoid the infeasbility, we use the feasible point pursuit technique that consists in introducing slack variables. We consider the following problem

$$\begin{aligned}
&\min_{x,z} f_0(x) := \mathbb{E}_{\xi_0}(g_0(x,\xi_0)) - \mathbb{E}_{\xi_0}(h_0(x,\xi_0)) + r_0(x) - s_0(x) + \beta \sum_{i=1}^{m} z_i, \\
&\text{subject to} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(FPP_GSDC)} \\
&f_i(x) := \mathbb{E}_{\xi_i}(g_i(x,\xi_i)) - \mathbb{E}_{\xi_i}(h_i(x,\xi_i)) + r_i(x) - s_i(x) \leq z_i, \quad i = 1,2,\ldots,m, \\
&z_1, z_2, \ldots, z_m \geq 0,
\end{aligned}$$

where $z_1, z_2, \ldots, z_m$ are slack variables and $\beta > 0$ is the penalty parameter. At each iteration, we can linearize $\mathbb{E}_{\xi_i}(h_i(x,\xi_i))$ (and further approximate the affine minorant by the sample affine minorant) and $s_i(x)$. The resulting convex constraints are always feasible thanks to $z_1, z_2, \ldots, z_m$.

In the deterministic context, this technique has been successfully employed in the General DCA [125]. In the stochastic context, it has been successfully employed in the Stochastic SCA developed for solving L-smooth stochastic objective with L-smooth stochastic constraints [249].

12.3.1 Some Applications of Stochastic General DCA

12.3.1.1 Chance constrained optimization

In many systems, it is desirable to impose such constraints $u_i(x,\xi) \leq 0$ for all uncertain data ξ. In many situations, this requirement is too strictly or even impossible due to the uncertainty nature of ξ [169]. Chance (or probabilistic) constraint is an important concept that can address this issue. Instead of requiring "$u_i(x,\xi) \leq 0$ for all $\xi \in \Omega$" (appear in the context of robust optimization, worst-case analysis), we can ask for a weaker constraint of the form "$u_i(x,\xi) \leq 0$ with at least probability of η". Formally, a chance constraint is described as

$$\mathbf{Prob}(u_i(x,\xi) \leq 0) \geq \eta, \tag{12.1}$$

which is equivalent to

$$\mathbf{Prob}(u_i(x,\xi) > 0) \leq 1 - \eta$$

where η is usually a huge value in $[0,1]$, e.g. $0.9, 0.95, 0.99$.

The chance constraint is the same as $\mathbf{VaR}(u_i(x,\xi);\eta) \leq 0$ where $\mathbf{VaR}(z;\eta)$ is the value-at-risk of random variable z at level η.

Example. In portfolio selection [4], let $R = (R_1, R_2, \ldots, R_n)$ be rates of return on n assets, $x = (x_1, x_2, \ldots, x_n)$ be the amount money allocated to each asset, t be a specified lower bound of porfolio return, and α be stipulated probability. Then, the following chance constraint is imposed

$$\mathbf{Prob}(\langle R, x\rangle < t) \leq \alpha.$$

Clearly, chance constraints can be written in form of expectation since

$$\mathbf{Prob}(u_i(x,\xi) > 0) = \mathbb{E}\left(1_{(0,+\infty)}(u_i(x,\xi))\right).$$

The function $1_{(0,+\infty)}$ is discountinuous. We can, however, use (convex) continuous functions to approximate this discountinuity [169]. Let $\phi : \mathbb{R} \to \mathbb{R}$ be nonnegative convex nondecreasing function with $\phi(0) = 1$. We see that, for each $\alpha_i > 0, \phi(z/\alpha_i) \geq 1_{(0,+\infty)}(z), \quad \forall z \in \mathbb{R}$. Therefore

$$\mathbf{Prob}(u_i(x,\xi) > 0) \leq \mathbb{E}\phi(u_i(x,\xi)/\alpha_i). \tag{12.2}$$

Consequently, the constraint $\mathbb{E}\phi(u_i(x,\xi)/\alpha_i) \leq 1 - \eta$ implies the chance constraint (12.1). When $u_i(\cdot,\xi)$ are convex for all ξ, $\phi(u_i(\cdot,\xi))$ are convex. Hence we obtain a stochastic convex constraint.

Some common choices of ϕ :

— Markov bound : $\phi(u) = (u+1)_+$.

— Chebyshev bound : $\phi(u) = (u+1)_+^2$.

— Chernoff bound : $\phi(u) = \exp(u)$.

If $u_i(\cdot,\xi)$ is non-convex or ϕ is chosen to be non-convex (to get a better bound in (12.2) for example), then $\phi(u_i(\cdot,\xi))$ is generally non-convex . When for each ξ, $u_i(\cdot,\xi)$ is DC, if we further assume that ϕ is DC where both DC components are Lipschitz continuous, then the composite $\phi(u_i(\cdot,\xi))$ is DC [15].

12.3.1.2 Interference networks in Telecommunications

Let K be a number of pairs frequency-selective interference channel. Let H_{kj} be the random coefficient of the channel between the k-th transmitter and the j-th receiver. Assume that H_{kj} are i.i.d. whose law is $\mathcal{CN}(0,\delta_{kj})$. Let p_k be the transmit power for the k-th transmitter and P_k be the power limit for the k-th transmitter.

Ergodic Sum-Rate Maximization with Coupled Constraints [249]

$$\begin{aligned}
&\max_p r_0(p) = \sum_{k=1}^{K} r_k(p),\\
&\text{subject to}\\
&0 \leq p_k \leq P_k, \quad k = 1,2,\ldots,K,\\
&r_k(p) \geq R_k, \quad k = 1,2,\ldots,K,
\end{aligned}$$

where $r_1, r_2, \ldots, r_K$ are ergodic rates defined as follows

$$r_k(p) = \mathbb{E}\left[\log\left(1 + \frac{|H_{kk}|^2 p_k}{\sum_{j\neq k}|H_{kj}|^2 p_j + \sigma_k^2}\right)\right].$$

Many related problems in telecommunications such as Ergodic Sum-rate maximization with decoupled constraints, MIMO transmit signal design with imperfect CSI, Robust beaming design, Massive MIMO Hybrid Beaming Design, etc. can be found in [249, 157].

12.3.1.3 Neyman-Pearson Classification in Machine Learning

Let (X,Y) be a pair of random variables where X is a random feature vector and $Y \in \{-1,1\}$ is the corresponding label. In the classification task, we learn the classifier ϕ that maps X to its label Y.

The overall classification error that the classifier ϕ incurs is

$$R(\phi) = \mathbf{Prob}(\phi(X) \neq Y) = \mathbb{E}(1_{\phi(X)\neq Y}).$$

Traditional approach is to find a classifier ϕ that minimizes $R(\phi)$.

By law of total probability, $R(\phi)$ can be written as

$$\begin{aligned} R(\phi) &= \mathbf{Prob}(Y=-1) \times \mathbf{Prob}(\phi(X) \neq Y | Y=-1) + \mathbf{Prob}(Y=1) \times \mathbf{Prob}(\phi(X) \neq Y | Y=1) \\ &:= \mathbf{Prob}(Y=-1) R_0(\phi) + \mathbf{Prob}(Y=1) R_1(\phi). \end{aligned}$$

The traditional approach indeed seeks to minimize R_0 and R_1 simultaneously. In contrast, the Neyman-Pearson paradigm looks for a classifier ϕ that minimizes R_1 while guaranteeing the term R_0 smaller than some level α. The (α-level) Neyman-Pearson oracle classifier is the minimizer of the following problem [225]

$$\begin{aligned} &\min R_1(\phi) = \mathbf{Prob}(\phi(X) \neq Y | Y=1) \\ &\text{subject to} \\ &\mathbf{Prob}(\phi(X) \neq Y | Y=-1) \leq \alpha, \end{aligned}$$

which literary reads "minimize the misclassification error of class 1 while keeping the misclassification error of class -1 under α".

Usually, the classifier ϕ takes the following form

$$\phi(x) = \begin{cases} +1 & \text{if } h(w,x) > 0, \\ -1 & \text{if } h(w,x) < 0 \end{cases},$$

where $h(w,x)$ is a regression function parameterized by w. Then,

$$R(\phi) = \mathbf{Prob}(\phi(X) \neq Y) = \mathbf{Prob}(Yh(w,X) < 0) = \mathbb{E}\left(1_{(-\infty,0)}(Yh(w,X))\right).$$

In practice, the above "0-1" loss function $1_{(-\infty,0)}$ is not tractable in general. Therefore, we replace it by a suitable surrogate loss φ function (a common practice is to use a convex surrogate for facility) [19] :

$$\bar{R}(\phi) = \mathbb{E}(\varphi(Yh(w,X))).$$

The "surrogate" (α-level) Neyman-Pearson is as follows :

$$\begin{aligned} &\min_w \bar{R}_1(\phi) = \mathbb{E}_{X|(Y=1)}\left(\varphi(h(w,X))\right), \\ &\text{subject to} \\ &\bar{R}_0(\phi) := \mathbb{E}_{X|(Y=-1)}(\varphi(-h(w,X))) \leq \alpha. \end{aligned}$$

This is a stochastic program with both the objective and the constraint given in the expectation forms. Classical approach uses linear function h ($h(w,X) := w^\top X$) and convex φ, which leads to a convex optimization problem. With other choices of h and φ, the resulting problems can be non-convex, nonsmooth.

12.3.1.4 Bayesian optimization

Bayesian statistical inference is a comprehensive framework to incorporate newly arriving information into the current probabilistic system based on the Bayes' rule. In the Bayesian philosophy, probability has subjective nuances : (unknown) deterministic objects are modeled as random variables (the uncertainty actually comes from the lack of knowledge about the interested objects). Suppose we want to perform some computations on the deterministic unknown parameter ξ. Since ξ is unknown, we treat it as a random variable, where we have some initial beliefs (prior) $\pi(\xi)$ about its distribution (if there is no knowledge about ξ, uniform priors can be used). When new data X_n that reveals some information about ξ arrives, our belief about ξ is updated via Bayes' theorem :

$$\pi(\xi|X_n) = \frac{p(X_n|\xi)\pi(\xi)}{\int p(X_n|\xi)\pi(\xi)d\xi},$$

where $\pi(\xi|X_n)$ is the posterior distribution of ξ, $p(X_n|\xi)$ is the likelihood of observing X_n.

The above Bayesian framework allows one to draw inference under uncertainty, making the best use of new relevant information.

In Bayesian optimization, suppose we are dealing with the following program parameterized by ξ [113],

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & f(x,\xi) \\ \text{subject to } & g_i(x,\xi) \leq 0, \forall i \in \{1,2,\ldots,m\}. \end{aligned}$$

In many situations, the true value of ξ is unknown beyond lying in some set Θ. Hence, ξ is modeled as a random variable where we assume some prior knowledge π about its distribution over Θ. When a new relevant observation X_n is revealed, our belief is updated, giving rise to the posterior distribution $\pi(\xi|X_n)$ which encodes the most up-to-date information about ξ. We then solve the following stochastic optimization problem

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & \mathbb{E}_{\pi(\xi|X_n)}[f(x,\xi)] \\ \text{subject to } & \Pi(g_i(x,\xi) \leq 0, \quad i = 1,2,\ldots,m|X_n) \geq \beta, \end{aligned}$$

where, for any $A \subset \Theta$, $\Pi(A|X_n) = \int_A \pi(\xi|X_n)d\xi$.

The above problem is in general non-convex stochastic optimization program when $f(\cdot,\xi), g_i(\cdot,\xi)$ are non-convex.

12.3.1.5 Distributionally Robust Chance Constrained Programming based on Generative adversarial networks (GANs)

Generative adversarial networks. Generative adversarial networks (GANs) [84] are a special family of generative models (to learn how to generate data from an interested distribution). Unlike traditional generative models that learn by maximizing the likelihood , GANs consist of two components : a Generator (G) and a Discriminator (D). To learn how to generate data from the target distribution over data x, the generator learns a mapping from a prior noise model $p_z(z)$ to the space of samples x. On the other hand, the Discriminator learns how to distinguish samples obtained from the true distribution and samples generated by the Generator by performing binary classification. Formally, D and G play the following minimax game with value

$V(G, D)$:

$$\min_G \max_D V(G, D) := \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] .$$

Usually, the Generator G and the Discriminator D are parameterized as two neural networks $G(z; \theta_g)$ and $D(z; \theta_d)$. Then, the parameters θ_g and θ_d are optimization variables.

This problem is - in general - a non-convex stochastic program. The conventional approach to solve it is to use Stochastic Gradient Descent (and Stochastic Gradient Ascent) alternatively between the Generator's parameter space θ_g and the Discriminator's parameter space θ_d.

GAN-based Data-Driven Distributionally Robust Chance Constrained Programming. The distributionally robust chance constrained programming takes the following form

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & \inf_{P \in D} \mathbb{P}_{\varsigma \sim P} \left(u(x, \varsigma) \leq 0 \right) \geq 1 - \alpha, \\ & x \in X, \end{aligned} \qquad (P)$$

where D is an *ambiguity set* and X is a deterministic constraint.

If the ambiguity set is constructed based on ϕ-divergence, the distributionally robust chance constrained program can be formulated as the conventional chance constrained program (introduced in [258]) with a modified risk level α',

$$\inf_{P \in D} \mathbb{P}_{\varsigma \sim P} \left(u(x, \varsigma) \leq 0 \right) \geq 1 - \alpha \Leftrightarrow \mathbb{P}_{\varsigma \sim P_0} \left(u(x, \varsigma) \leq 0 \right) \geq 1 - \max\{0, \alpha'\},$$

where the ambiguous probability distribution P is replaced by the empirical probability distribution P_0.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & \mathbb{P}_{\varsigma \sim P_0} \left(u(x, \varsigma) \leq 0 \right) \geq 1 - \max\{0, \alpha'\}, \\ & x \in X. \end{aligned} \qquad (Q)$$

Recently, GANs have been used to learn a good estimation for the empirical probability distribution P_0 [258]. Then, the Generator of GANs will generate data which are further used to construct SAA problem (Sample Average Approximation) for the problem (Q). If the constraints and the objective of (P) are linear, the SAA problem is a mixed-integer linear programming which can be solved by existing packages such as CPLEX, GUROBI, etc. When the objective and constraints are non-convex, the SAA problems are much more difficult. We can employ the continuous approximation technique in Section 12.3.1.1 to transform the (continuous) Stochastic non-convex program.

Bibliographie

[1] J. Aach and G. M. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6) :495–508, 2001.

[2] S. Adams and P. Beling. A survey of feature selection methods for gaussian mixture models and hidden markov models. *Artificial Intelligence Review*, 52 :1739–1779, 2019.

[3] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah. Time-series clustering – A decade review. *Information Systems*, 53 :16–38, 2015.

[4] N. Agnew, R. Agnew, J. Rasmussen, and K. Smith. An application of chance constrained programming to portfolio selection in a casualty insurance firm. *Management Science*, 15(10) :B–512, 1969.

[5] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *n Proc. 2nd Int. Symp. Information Theory, Tsahkadsor*, page 267–281, 1973.

[6] Z. Allen-Zhu. Natasha 2 : Faster non-convex optimization than SGD. *Advances in Neural Information Processing Systems*, 31 :2675–2686, 2018.

[7] Z. Allen-Zhu and Y. Li. Neon2 : finding local minima via first-order oracles. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3720–3730, 2018.

[8] Z. Allen-Zhu and Y. Yuan. Improved SVRG for non-strongly-convex or sum-of-non-convex objectives. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1080–1089, 2016.

[9] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. Np-hardness of euclidean sum-of-squares clustering. *Cahiers du GERAD G-2008-33*, 2008.

[10] F. J. Aragon Artacho, R. Fleming, and T. V. Phan. Accelerating the DC algorithm for smooth functions. *Math. Program. B*, 169, 2018.

[11] F. J. Aragon Artacho and T. V. Phan. The boosted difference of convex functions algorithm for nonsmooth functions. *SIAM Journal on Optimization*, 30(1), 2020.

[12] H. Attouch and J. Bolte. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116(1) :5–16, 2009.

[13] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal Alternating Minimization and Projection Methods for Nonconvex Problems : An Approach Based on the Kurdyka-Łojasiewicz Inequality. *Mathematics of Operations Research*, 35(2) :438–457, Apr. 2010.

[14] H. Attouch, J. Bolte, and B. F. Svaiter. Convergence of descent methods for semi-algebraic and tame problems : proximal algorithms, forward-backward splitting, and regularized gauss-seidel methods. *Mathematical Programming*, 137(1) :91–129, 2013.

[15] M. Bacak and J. M. Borwein. On difference convexity of locally lipschitz functions. *Optimization*, 60(8-9) :961–978, 2011.

[16] S. C. Bagley, H. White, and B. A. Golomb. Logistic regression in the medical literature : Standards for use and reporting, with particular attention to one medical domain. *Journal of Clinical Epidemiology*, 54(10) :979–985, Oct. 2001.

[17] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off : A review and experimental evaluation of recent algorithmic advances. *Data Min Knowl Disc*, 31(3) :606–660, 2017.

[18] W. Bajwa, J. Haupt, S. A., and R. Nowak. Compressive wireless sensing. *Proceedings of Fifth Int. Conf. on Information Processing in Sensor Networks*, pages 134–142, 2006.

[19] M. Barlaud, P. L. Combettes, and L. Fillatre. Constrained convex neyman-pearson classification using an outer approximation splitting method. hal-01160831, 2015.

[20] A. Beck and M. Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11) :2419–2434, 2009.

[21] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2 :183–202, 2009.

[22] P. K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *In Proceedings of the conference on Advances in neural information processing systems II, Cambridge, MA, USA*, pages 368–374, 1999.

[23] D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex analysis and optimization*. Athena Scientific, 2003.

[24] D. P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization : A survey. Technical report, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, 2010.

[25] D. P. Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical Programming*, 129(2) :163–195, 2011.

[26] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. J. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerson. Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. *Proc Natl Acad Sci USA*, 98(24) :13790–13795, 2001.

[27] S. Bhattacharya and P. D. McNicholas. A lasso-penalized bic for mixture model selection. *Advances in Data Analysis and Classification*, 8 :45 – 61, 2014.

[28] T. D. Bie and N. Cristianini. Convex methods for transduction. *Proceedings of the 16th International Conference on Neural Information Processing Systems*, 2004.

[29] J. Bolte, A. Daniilidis, A. Lewis, and M. Shiota. Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2) :556–572, 2007.

[30] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1) :459–494, Aug 2014.

[31] L. Bottou. On-line learning and stochastic approximations. In D. Saad, editor, *On-line Learning in Neural Networks*, pages 9–42. Cambridge University Press, New York, NY, USA, 1998.

[32] C. Bouveyron and C. Brunet. Model-based clustering of high-dimensional data : A review. *Computational Statistics and Data Analysis*, 71 :52 – 78, 2013.

[33] C. R. Boyd, M. A. Tolson, and W. S. Copes. Evaluating trauma care : The TRISS method. Trauma Score and the Injury Severity Score. *The Journal of Trauma*, 27(4) :370–378, 1987.

[34] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

[35] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Machine Learning Proceedings of the Fifteenth International Conference (ICML '98)*, pages 82–90. Morgan Kaufmann, 1998.

[36] P. S. Bradley, O. L. Mangasarian, and J. B. Rosen. Parsimonious least norm approximation. *Comput. Optim. Appl.*, 11(1) :5–21, 1998.

[37] M. J. Brusco. A repetitive branch-and-bound procedure for minimum within-cluster sum of squares partitioning. *Psychometrika*, 71 :347–363, 2006.

[38] E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles : Exact signal reconstruction from highly incomplete fourier information. *IEEE Trans. Inform. Theory*, 52, 2006.

[39] E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8) :1207–1223, 2006.

[40] E. Candès and T. Tao. Near optimal signal recovery from random projections : Universal encoding strategies ? *IEEE Trans. Inform. Theory*, 52(12) :5406–5425, 2006.

[41] E. Candès and T. Tao. The dantzig selector : Statistical estimation when p is much larger than n. *Ann. Stat.*, 35(6) :2313–2351, 2007.

[42] E. J. Candes and P. Randall. Highly robust error correction by convex programming. *IEEE Trans. Inform. Theory*, 54 :2829–2840, 2006.

[43] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Inform. Theory*, 51(12) :4203–4215, 2005.

[44] E. J. Canes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted-l_1 minimization. *J. Four. Anal. and Appli.*, 14 :877–905, 2008.

[45] G. Celeux and G. Govaert. A classification em algorithm for clustering and two stochastic versions. *Comput. Statist. Data Anal*, 14(3) :315–332, 1992.

[46] G. Celeux and G. Govaert. Comparison of the mixture and the classification maximum likelihood in cluster analysis. *J. Stat. Comput. Simul.*, 47 :127–146, 1993.

[47] E. Y. Chan, W. K. Ching, K. N. Michael, and Z. J. Huang. An optimizationalgorithm for clustering using weighted dissimilarity measures. *Pattern Recognition*, 37(5) :943–952, 2004.

[48] O. Chapelle and Z. A. Semi-supervised classification by low density separation. In *In Proc. 10th Internat. Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005.

[49] O. Chapelle, V. Sindhwani, and S. Keerthi. Branch and bound for semi-supervised support vector machines. *Advances in Neural Information Processing Systems, MIT Press, Cambridge*, 2006.

[50] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9, 2008.

[51] R. Chartrand and W. Yin. Iteratively reweighted algorithms for compressive sensing. *IEEE International Conference on Acoustics, Speech and Signal Processing 2008*, 2008.

[52] J. Chen and A. Khalili. Order selection in finite mixture models with a nonsmooth penalty. *Journal of the American Statistical Association*, 103(484) :1674–1683, 2008.

[53] X. Chen, F. M. Xu, and Y. Ye. Lower bound theory of nonzero entries in solutions of l2-lp minimization. *SIAM J. Sci. Comp.*, 32(5) :2832–2852, 2010.

[54] S. Chu, E. Keogh, D. Hart, and M. Pazzani. Iterative deepening dynamic time warping for time series. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, pages 195–212. SIAM, 2002.

[55] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. *Machine Learning*, 7, 2006.

[56] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings of the 23th International Conference on Machine Learning (ICML 2006)*, pages 2832–2852, 2006.

[57] P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4) :1168–1200, 2005.

[58] D. Cox. The regression analysis of binary sequences (with discussion). *J Roy Stat Soc B*, 20 :215–242, 1958.

[59] M. Davenport, M. Duarte, Y. Eldar, and G. Kutyniok. *Introduction to Optimization.* Cambridge University Press, 2012.

[60] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga : A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proceedings of Advances in Neural Information Processing Systems*, 2014.

[61] A. Defazio, T. Caetano, and J. Domke. Finito : A faster, permutable incremental gradient method for big data problems. In *Proceedings of the* 31^{st} *International Conference on Machine Learning*, 2014.

[62] S. Ding, Z. Zhu, and X. Zhang. An overview on semi-supervised support vector machine. *Neural Computing and Applications*, 28, 2017.

[63] H. Do, A. Kalousis, and M. Hilario. Feature weighting using margin and radius based error bound optimization in svms. *Machine Learning and Knowledge Discovery in Databases*, 26 :315–329, 2009.

[64] D. L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52, 2006.

[65] O. Du Merle, P. Hansen, B. Jaumard, and N. Mladenovi'c. An interior point algorithm for minimum sum of squares clustering. *SIAM J. Sci. Comput.*, 21(4) :1485–1505, 2006.

[66] D. Dua and K. Taniskidou. Uci machine learning repository. Irvine, CA : University of California, School of Information and Computer Science, 2017.

[67] M. F. Duarte and Y. C. Eldar. Compressive sensing. *IEEE Sig. Process. Mag.*, 24, 2007.

[68] M. F. Duarte and Y. C. Eldar. Structured compressed sensing : From theory to applications. *IEEE Transactions on Signal Processing*, 59, 2011.

[69] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(7), 2011.

[70] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96 :1348–1360, 2001.

[71] C. Fang, C. J. Li, Z. Lin, and T. Zhang. Spider : Near-optimal non-convex optimization via stochastic path integrated differential estimator. *arXiv preprint arXiv :1807.01695*, 2018.

[72] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction : Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4) :586–597, 2007.

[73] C. Fraley and A. E. Raftery. How many clusters ? which clustering method ? answers via model-based cluster analysis. *The Computer Journal*, 41(8) :578–588, 1998.

[74] P. Frankel, G. Garrigos, and P. Peypouquet. Splitting methods with variable metric for kurdyka–łojasiewicz functions and general convergence rates. *Journal of Optimization Theory and Applications*, 165(3) :874–900, 2015.

[75] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3) :432–441, 2007.

[76] W. J. Fu. Penalized regression : the bridge versus the lasso. *J. Comp. Graph. Stat.*, 7 :397–416, 2008.

[77] M. Fukushima and H. Mine. A generalized proximal point algorithm for certain non-convex minimization problems. *Int. J. Syst. Sci.*, 12(8), 1981.

[78] C. Gao, Y. Zhu, X. Shen, and W. Pan. Estimation of multiple networks in gaussian mixture models. *Electronic Journal of Statistics*, 10(1) :1133–1154, 2016.

[79] G. Gasso, A. Rakotomamonjy, and S. Canu. Recovering sparse signals with a certain family of nonconvex penalties and DC programming. *IEEE Trans. Sign. Proc.*, 57 :4686–4698, 2009.

[80] A. Genkin, D. D. Lewis, and D. Madigan. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3) :291–304, 2007.

[81] F. Gieseke, A. Airola, and T. Pahikkala. Fast and simple gradient-based optimization for semi-supervised support vector machines. *Neurocomputing*, 123, 2014.

[82] F. Gieseke, A. Airola, T. Pahikkala, and O. Kramer. Sparse quasi-newton optimization for semi-supervised support vector machines. In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 45–54, 2012.

[83] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data : Constraint specification and implementation. In U. Montanari and F. Rossi, editors, *Principles and Practice of Constraint Programming — CP '95*, Lecture Notes in Computer Science, pages 137–153. Springer Berlin Heidelberg, 1995.

[84] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[85] I. F. Gorodnitsky and B. D. Rao. Sparse signal reconstructions from limited data using focuss : A re-weighted minimum norm algorithm. *IEEE Trans. Signal Processing*, 45 :600–616, 1997.

[86] G. Govaert and M. Nadif. Clustering with block mixture models. *Pattern Recognition*, 36 :463–473, 2003.

[87] G. Govaert and M. Nadif. Block clustering with bernoulli mixture models : Comparison of different approaches. *Computational Statistics & Data Analysis*, 52 :3233–3245, 2008.

[88] R. Gribonval and M. Nielsen. Sparse representation in union of bases. *IEEE Trans. on Information Theory*, 49 :3320–3325, 2003.

[89] L. Grippo and M. Sciandrone. Nonmonotone globalization techniques for the barzilai-borwein gradient method. *Computational Optimization and Applications*, 23(2) :143–169, 2002.

[90] B. Gu, Z. Huo, and H. Huang. Inexact proximal gradient methods for non-convexand non-smooth optimization. In *32th AAAI Conference on Artificial Intelligence AAAI-18*, 2018.

[91] W. Guan and A. Gray. Sparse high-dimensional fractional-norm support vector machine via DC programming. *Computational Statistics and Data Analysis*, 67 :136–148, 2013.

[92] J. Guo, E. Levina, G. Michailidis, and J. Zhu. Pairwise variable selection for high-dimensional model-based clustering. *Biometrics*, 66 :793 – 804, 2009.

[93] E. Hale, W. Yin, and Y. Zhang. Fixed-point continuation for ℓ_1-minimization : Methodology and convergence. *SIAM Journal on Optimization*, 19(3) :1107–1130, 2008.

[94] B. Hao, W. Sun, Y. Liu, and G. Cheng. Simultaneous clustering and estimation of heterogeneous graphical models. *Electronic Journal of Statistics*, 18(217) :1–58, 2018.

[95] P. Hartman. On functions representable as a difference of convex functions. *Pacific J. Math*, 9 :707–713, 1959.

[96] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning 2nd edition.* Springer,Heidelberg, 2009.

[97] K. Hayashi, M. Nagahara, and T. Tanaka. A user's guide to compressed sensing for communications systems. *IEICE Trans. on Comm.*, 96(3) :685–712, 2013.

[98] K. Healy and L. W. Schruben. Retrospective simulation response optimization. In *1991 Winter Simulation Conference Proceedings.*, pages 901–906, 1991.

[99] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines : Global versus component-based approach. In *Proc. 8th International conference on computer vision*, pages 688–694, 2001.

[100] Herbrich. *Learning kernel classifiers.* MIT Press, 2002.

[101] F. Hichem and O. Nasraoui. Unsupervised learning of prototypes and attribute weights. *Journal Pattern Recognition*, 37(3) :567–581, 2004.

[102] S. M. Hill. *Sparse graphical models for cancer signalling.* PhD thesis, University of Warwick, 2012.

[103] J. B. Hiriart-Urruty. Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. *Lecture Note in Economics and Math. Systems*, 256 :37–70, 1985.

[104] E. Hjelmas. Face detection : A survey. *Computer Vision and Image Understanding*, 83, 2001.

[105] V. T. Ho. *Advanced machine learning techniques based on DC programming and DCA*. PhD thesis, University of Lorraine, France, 2017.

[106] V. T. Ho, H. A. Le Thi, and D. C. Bui. Online DC optimization for online binary linear classification. In *Intelligent Information and Database Systems : ACIIDS 2016, Proceedings, Part II*, pages 661–670, 2016.

[107] C. Hsieh, M. Sustik, I. Dhillon, and P. Ravikumar. Quic : Quadratic approximation for sparse inverse covariance estimation. *Journal of Machine Learning Research*, 15 :2911–2947, 2014.

[108] T. Huang, H. Peng, and K. Zhang. Model selection for gaussian mixture models. *Statistica Sinica*, 27(1) :147–169, 2017.

[109] W. Huang, K. N. Michael, R. Hongqiang, and Z. Li. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5) :657–668, 2005.

[110] L. Hubert and P. Arabie. Comparing partitions. *Journal of the Classification*, 2 :193–218, 1985.

[111] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11) :1413–1457, 2004.

[112] S. J Reddi, S. Sra, B. Poczos, and A. J. Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. *Advances in neural information processing systems*, 29 :1145–1153, 2016.

[113] P. Jaiswal, H. Honnappa, and V. A. Rao. Bayesian chance constrained optimization : Approximations and statistical consistency. *arXiv preprint arXiv :2106.12199*, 2021.

[114] L. F. James, C. E. Priebe, and D. J. Marchette. Consistent estimation of mixture complexity. *The Annals of Statistics*, 29(5) :1281–1296, 2001.

[115] T. Joachims. Transductive inference for text classification using support vector machines. In *16th Inter. Conf. on Machine Learning ,SF, USA*, pages 200–209, 1999.

[116] J. J. Joaquim, M. Raydan, S. S. Rosa, and S. A. Santos. On the solution of the symmetric eigenvalue complementarity problem by the spectral projected gradient algorithm. *Numerical Algorithms*, 47(4) :391–407, 2008.

[117] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[118] J. Judice, M. Raydan, and S. Rosa. On the solution of the symmetric eigenvalue complementarity problem by the spectral projected gradient algorithm. *Numerical Algorithms*, 47 :391 – 407, 2008.

[119] J. Kim, Y. Kim, and Y. Kim. A Gradient-Based Optimization Algorithm for LASSO. *Journal of Computational and Graphical Statistics*, 17(4) :994–1009, 2008.

[120] G. King and L. Zeng. Logistic Regression in Rare Events Data. *Political Analysis*, 9 :137–163, 2001.

[121] H. M. Le, H. Le Thi, and M. C. Nguyen. DCA based algorithms for feature selection in semi-supervised support vector machines. *Lecture Notes in Computer Science (LNCS)*, 7988, 2013.

[122] H. A. Le Thi. A new approximation for the ℓ_0-norm. Technical report, LITA EA 3097, University of Lorraine, France, 2012.

[123] H. A. Le Thi, T. Belghiti, and T. Pham Dinh. A new efficient algorithm based on DC programming and DCA for clustering. *Journal of Global Optimization*, 37 :593–608, 2007.

[124] H. A. Le Thi and V. T. Ho. Online learning based on online DCA and application to online classification. *Neural Computation*, 32(4) :759–793, 2020.

[125] H. A. Le Thi, V. N. Huynh, and T. Pham Dinh. DC programming and DCA for general DC programs. In *Advanced Computational Methods for Knowledge Engineering*, pages 15–35, Cham, 2014. Springer International Publishing.

[126] H. A. Le Thi, V. N. Huynh, and T. Pham Dinh. Error bounds via exact penalization with applications to concave and quadratic systems. *Journal of Optimization Theory and Applications*, 171(1), 2016.

[127] H. A. Le Thi, V. N. Huynh, T. Pham Dinh, and H. P. H. Luu. Stochastic difference-of-convex algorithms for solving nonconvex optimization problems. *arXiv preprint arXiv :1911.04334v2*, 2020.

[128] H. A. Le Thi, H. M. Le, V. V. Nguyen, and T. Pham Dinh. A DC programming approach for feature selection in support vector machines learning. *Advances in Data Analysis and Classification*, 2(3) :259–278, Dec. 2008.

[129] H. A. Le Thi, H. M. Le, and T. Pham Dinh. New and efficient DCA based algorithms for minimum sum-of-squares clustering. *Pattern Recognition*, 47(1) :388 – 401, 2014.

[130] H. A. Le Thi, H. M. Le, and T. Pham Dinh. Feature selection in machine learning : an exact penalty approach using a different of convex function algorithm. *Machine Learning*, 101(1) :163–186, 2015.

[131] H. A. Le Thi, H. M. Le, D. N. Phan, and B. Tran. Stochastic DCA for the Large-sum of Non-convex Functions Problem and its Application to Group Variable Selection in Classification. In *Proceedings of the 34th International Conference on Machine Learning* , volume 70, pages 3394–3403, 2017.

[132] H. A. Le Thi, H. M. Le, D. N. Phan, and B. Tran. Stochastic DCA for sparse multiclass logistic regression. In *Advanced Computational Methods for Knowledge Engineering*, pages 1–12, 2018.

[133] H. A. Le Thi, H. M. Le, D. N. Phan, and B. Tran. Stochastic DCA for minimizing a large sum of DC functions with application to multi-class logistic regression. *Neural Networks*, 132 :220–231, 2020.

[134] H. A. Le Thi, H. M. Le, D. N. Phan, and B. Tran. Novel DCA based algorithms for a special class of nonconvex problems with application in machine learning. *Applied Mathematics and Computation*, 409, 125904, November 2021.

[135] H. A. Le Thi, H. P. H. Luu, H. M. Le, and T. Pham Dinh. Stochastic DCA with variance reduction and applications in machine learning. submitted to Journal of Machine Learning Research.

[136] H. A. Le Thi, H. P. H. Luu, and T. Pham Dinh. Online stochastic DCA with applications to principal component analysis. *arXiv preprint arXiv :2108.02300*, 2021.

[137] H. A. Le Thi and M. C. Nguyen. Efficient algorithms for feature selection in multi-class support vector machine. In *Advanced Computational Methods for Knowledge Engineering, Studies in Computational Intelligence 479*, 2013.

[138] H. A. Le Thi, M. C. Nguyen, and T. Pham Dinh. A DC Programming Approach for Finding Communities in Networks. *Neural Computation*, 26(12) :2827–2854, 2014.

[139] H. A. Le Thi, T. B. T. Nguyen, and H. M. Le. Sparse signal recovery by difference of convex functions algorithms. *Lecture Notes in Computer Science*, 7803 :387–397, 2013.

[140] H. A. Le Thi, V. V. Nguyen, and S. Ouchani. Gene selection for cancer classification using DCA. *Journal of Fonctiers of Computer Science and Technology*, 3(6) :62–72, 2009.

[141] H. A. Le Thi and T. Pham Dinh. A continuous approach for globally solving linearly constrained quadratic zero-one programming problems. *Optimization*, 50 :93–120, 2001.

[142] H. A. Le Thi and T. Pham Dinh. The DC (Difference of Convex Functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems. *Annals of Operations Research*, 133(1) :23–46, Jan. 2005.

[143] H. A. Le Thi and T. Pham Dinh. Minimum sum-of-squares clustering by DC programming and DCA. In *ICIC'09 Proceedings of the Intelligent computing 5th internationalconference on Emerging intelligent computing technology and applications*, pages 327–340, 2009.

[144] H. A. Le Thi and T. Pham Dinh. DC programming and DCA : thirty years of developments. *Mathematical Programming*, pages 1–64, 2018.

[145] H. A. Le Thi, T. Pham Dinh, and V. N. Huynh. Exact penalty and error bounds in DC programming. *Journal of Global Optimization, Special Issue in Memory of Reiner Horst, Founder of the Journal*, 52(3), 2012.

[146] H. A. Le Thi, T. Pham Dinh, and D. M. Le. Exact penalty in DC programming. *Vietnam J. Math.*, 27(2), 1999.

[147] H. A. Le Thi, T. Pham Dinh, H. M. Le, and X. T. Vo. DC approximation approaches for sparse optimization. *European Journal of Operational Research*, 244(1) :26–46, 2015.

[148] H. A. Le Thi, T. Pham Dinh, H. P. H. Luu, and H. M. Le. Deterministic and stochastic DCA for DC programming. In H. Pham, editor, *Handbook of Engineering Statistics 2nd edition*. Springer, 2022.

[149] H. A. Le Thi and D. N. Phan. DC Programming and DCA for Sparse Optimal Scoring Problem. *Neurocomput.*, 186(C) :170–181, 2016.

[150] H. A. Le Thi, D. N. Phan, and H. M. Le. DCA-like and its boosted scheme for a class of nonconvex optimization problems. submitted.

[151] H. A. Le Thi, D. N. Phan, and D. T. Pham. DCA based approaches for bi-level variable selection and application for estimating multiple sparse covariance matrices. *Neurocomputing*, 466, 2021.

[152] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller. Efficient backprop. In G. B. Orr and K.-R. Müller, editors, *TNeural Networks : Tricks of the Trade*, pages 9–50. Springer Berlin Heidelberg, 1998.

[153] L. Lei, C. Ju, J. Chen, and M. I. Jordan. Non-convex finite-sum optimization via scsg methods. *arXiv preprint arXiv :1706.09156*, 2017.

[154] H. Li and Z. Lin. Accelerated proximal gradient methods for nonconvex programming. In *Advances in Neural Information Processing Systems*, pages 377–387, 2015.

[155] J. G. Liao and K.-V. Chin. Logistic regression for disease classification using microarray data : Model selection in a large p and small n case. *Bioinformatics*, 23(15) :1945–1951, 2007.

[156] J. Liping, N. K. Michael, and Z. 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.

[157] A. Liu, V. K. Lau, and B. Kananian. Stochastic successive convex approximation for non-convex constrained stochastic optimization. *IEEE Transactions on Signal Processing*, 67(16) :4189–4203, 2019.

[158] J. Liu, Y. Cui, J. S. Pang, and S. Sen. Two-stage stochastic programming with linearly bi-parameterized quadratic recourse. *SIAM J. Optim.*, 30(3) :2530–2558, 2020.

[159] J. Ma and X. Zhang. A full smooth semi-support vector machine based on the cubic spline function. In *6th International Conference on Biomedical Engineering and Informatics (BMEI)*, pages 650–655, 2013.

[160] L. v. d. Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1) :3221–3245, 2014.

[161] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov) :2579–2605, 2008.

[162] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Trans. Signal Processing*, 41(12) :3397–3415, 1993.

[163] O. L. Mangasarian. Machine learning via polyhedral concave minimization. In *Applied Mathematics and Parallel Computing*, pages 175–188, 2017.

[164] G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley and Sons, 2000.

[165] V. Mechelen, H. Bock, and P. De Boeck. Two-mode clustering methods : A structured overview. *Statistical Methods in Medical Research*, 13 :363–394, 13.

[166] M. Meinard. Dynamic Time Warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[167] N. Mladenović and P. Hansen. Variable neighborhood search. *Comput. Oper. Res.*, 24 :1097–1100, 1997.

[168] H. Mohimani, M. Babaie-Zadeh, and C. Jutten. A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm. *IEEE Transactions on Signal Processing*, 57(1) :289–301, 2009.

[169] A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4) :969–996, 2007.

[170] Y. Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. *Soviet Mathematics Doklady*, 27 :372–376, 1983.

[171] J. Neumann, G. Schnörr, and G. Steidl. Combined svm-based feature selection and classification. *hine Learning*, 61(1-3) :129–150, 2005.

[172] J. Neveu. *Discrete-Parameter Martingales*, volume 10 of *North-Holland Mathematical Library*. Elsevier, 1975.

[173] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. Sarah : A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621. PMLR, 2017.

[174] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. Stochastic recursive gradient algorithm for nonconvex optimization. *arXiv preprint arXiv :1705.07261*, 2017.

[175] L. M. Nguyen, M. van Dijk, D. T. Phan, P. H. Nguyen, T.-W. Weng, and J. R. Kalagnanam. Finite-sum smooth optimization with sarah. *arXiv preprint arXiv :1901.07648*, 2019.

[176] V. A. Nguyen, H. A. Le Thi, and H. M. Le. A DCA based algorithm for feature selection in model-based clustering. In *Lecture Notes in Computer Science 12033*, pages 404–415, 2020.

[177] A. Nitanda and T. Suzuki. Stochastic difference of convex algorithm and its application to training deep boltzmann machines. In *Artificial intelligence and statistics*, pages 470–478, 2017.

[178] J. Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151) :773–782, 1980.

[179] C. L. Nutt, D. R. Mani, R. A. Betensky, P. Tamayo, J. G. Cairncross, C. Ladd, U. Pohl, C. Hartmann, M. E. McLaughlin, T. T. Batchelor, P. M. Black, A. V. Deimling, S. L. Pomeroy, T. R. Golub, and D. M. Louis. Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Res*, 63(7) :1602–1607, 2003.

[180] C. S. Ong and H. A. Le Thi. Learning sparse classifiers with difference of convex functions algorithms. *Optimization Methods and Software*, 28(4) :830–854, 2013.

[181] E. Osuna, R. Freund, and F. Girosi. Training support vector machines : an application to face detection. In *Proc. of Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[182] W. Pan and X. Shen. Penalized model-based clustering with application to variable selection. *J. Mach. Learn. Res*, 8 :1145 – 1164, 2007.

[183] J. Paparrizos and L. Gravano. K-Shape : Efficient and Accurate Clustering of Time Series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1855–1870. ACM Press, 2015.

[184] N. Parikh and S. Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3) :127–239, Jan. 2014.

[185] Y. C. Pati, R. Rezaifar, and P. S. Krishnaprasa. Orthogonal matching pursuit : recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Asilomar Conf. on Signals, Systems and Comput.*, 1993.

[186] D. Peleg and R. Meir. A bilinear formulation for vector sparsity optimization. *Signal Processing*, 8(2) :375–389, 2008.

[187] J. Peng and Y. Xiay. A cutting algorithm for the minimum sum-of-squared error clustering. *In Proceedings of the SIAM International Data Mining Conference*, 2005.

[188] N. H. Pham, L. M. Nguyen, D. T. Phan, and Q. Tran-Dinh. Proxsarah : An efficient algorithmic framework for stochastic composite nonconvex optimization. *J. Mach. Learn. Res.*, 21 :110–1, 2020.

[189] T. Pham Dinh and H. A. Le Thi. Convex analysis approach to DC programming : Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1) :289–355, 1997.

[190] T. Pham Dinh and H. A. Le Thi. Recent advances in DC programming and DCA. *Transactions on Computational Collective Intelligence*, 8342, 2014.

[191] D. N. Phan. *DCA based algorithms for learning with sparsity in high dimensional setting and stochastical*. PhD thesis, University of Lorraine, France, 2016.

[192] D. N. Phan, H. M. Le, and H. A. Le Thi. Accelerated difference of convex functions algorithm and its application to sparse binary logistic regression. In *Proceedings of 27th International Joint Conference on Artificial Intelligence and 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 2018)*, pages 1369–1375, 2018.

[193] D. N. Phan and H. A. Le Thi. Group variable selection via $\ell_{p,0}$ regularization and application to optimal scoring. *Neural Networks*, 2019.

[194] D. N. Phan, H. A. Le Thi, and T. Pham Dinh. Sparse covariance matrix estimation by DCA-Based Algorithms. *Neural Computation*, 29(11) :3040–3077, 2017.

[195] V. Piccialli, A. M. Sudoso, and A. Wiegele. Sos-sdp : an exact solver for minimum sum-of-squares clustering. *arXiv preprint arXiv :2104.11542*, 2021.

[196] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. H. Kim, L. C. Goumnerova, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, M. O. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. B. Mesirov, E. S. Lander, and T. R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415 :436–442, 2002.

[197] S. Qaisar, R. Muhammad Bilal, W. I. Muqaddas Naureen, and S. Lee. Compressive sensing : From theory to applications, a survey. *J. of Comm. and Net.*, 15(5) :443–456, 2013.

[198] A. Rakotomamonjy, R. Flamary, and G. Gasso. DC Proximal Newton for Nonconvex Optimization Problems. *IEEE Transactions on Neural Networks and Learning Systems*, 27(3) :636–647, Mar. 2016.

[199] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '12*, page 262, Beijing, China, 2012. ACM Press.

[200] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, M. J. P., T. Poggio, W. Gerald, M. Loda, E. S. Lander, and T. R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proc Natl Acad Sci USA*, 98 :15149–15154, 2001.

[201] M. Rani, S. B. Dhok, and R. B. Deshmukh. A systematic review of compressive sensing : Concepts, implementations and applications. *IEEE Access*, 6 :4875–4894, 2018.

[202] B. D. Rao, K. Engan, S. F. Cotter, P. J., and K. KreutzDelgado. Subset selection in noise based on diversity measure minimization. *IEEE Trans. Signal Processing*, 51(3) :760–770, 2003.

[203] B. D. Rao and K. Kreutz-Delgado. An affine scaling methodology for best basis selection. *IEEE Trans. Signal Processing*, 47 :87–200, 1999.

[204] F. Rinaldi, F. Schoen, and M. Sciandrone. Concave programming for minimizing the zero-norm over polyhedral sets. *Comput. Optim. Appl.*, 46(3) :467–486, 2010.

[205] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3) :400–407, 1951.

[206] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[207] J. V. Rosmalen, P. J. F. Groenen, and W. Trejos, J.and Castillo. Global optimization strategies for two-mode clustering. Econometric Institute Report, EI-2005-33, 2005.

[208] M. Saquib Sarfraz, O. Hellwich, and Z. Riaz. Feature extraction and representation for face recognition. In *Face Recognition*, pages 688–694, 2001.

[209] P. Schäfer. The BOSS is concerned with time series classification in the presence of noise. *Data Min Knowl Disc*, 29(6) :1505–1530, 2015.

[210] M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1) :83–112, Mar 2017.

[211] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2) :461–464, 1978.

[212] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. Mclust 5 : Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R journal*, 8 :29, 2016.

[213] S. Shalev-Schwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14 :567–599, 2013.

[214] S. Shalev-Shwartz. SDCA without duality, regularization, and individual convexity. In *International Conference on Machine Learning*, pages 747–754. PMLR, 2016.

[215] H. D. Sherali and J. Desai. A global optimization rlt-based approach for solving the hard clustering problem. *Journal of Global Optimization*, 32 :281–306, 2005.

[216] V. Sindhwani and S. Keerthi. Large scale semi-supervised linear svms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 477–484, 2006.

[217] V. Sindhwani, S. Keerthi, and O. Chapelle. Deterministic annealing for semi-supervised kernel machines. In *Proceedings of ICML'06, NY, USA*, pages 841–848, 2006.

[218] A. Subasi and E. Erçelebi. Classification of EEG signals using neural network and logistic regression. *Comput. Methods Programs Biomed.*, 78(2) :87–99, 2005.

[219] D. Takhar, J. N. Laska, M. B. Wakin, M. Duarte, D. Baron, S. Sarvotham, K. F. Kelly, and R. G. Baraniuk. A new compressive imaging camera architecture using optical-domain compression. In *Proceedings of SPIE - The International Society for Optical Engineering*, 2006.

[220] M. Thiao. *Approches de la programmation DC et DCA en Data mining.* PhD thesis, INSA-Rouen, France, 2011.

[221] M. Thiao, T. Pham Dinh, and H. A. Le Thi. DC programming approach for solving a class of nonconvex programs dealing with zero-norm. *Modelling, Computation and Optimization in Information Systems ans Management Science*, 14 :348–357, 2008.

[222] M. Thiao, T. Pham Dinh, and H. A. Le Thi. A DC programming approach for sparse eigenvalue problem. In *Proceedings of the 27th International Conference on Machine Learning, ICML 2010*, pages 1063–1070, 2010.

[223] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc.*, 46 :431–439, 1996.

[224] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 63(2) :411–423, 2001.

[225] X. Tong, L. Xia, J. Wang, and Y. Feng. Neyman-pearson classification : parametrics and sample size requirement. *J. Mach. Learn. Res.*, 21 :12–1, 2020.

[226] V. Vapnik and A. Sterin. On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10, 1977.

[227] M. Vincent and N. R. Hansen. Sparse group lasso and high dimensional multinomial classification. *Comput. Stat. Data Anal.*, 71 :771–786, 2014.

[228] H. D. Vinod. Integer programming and the theory of grouping. *J. Amer. Stat. Assoc.*, 64 :506–519, 1969.

[229] M. Vladymyrov and M. Carreira-Perpinan. Partial-Hessian strategies for fast learning of nonlinear embeddings. *arXiv preprint arXiv :1206.4646*, 2012.

[230] X. T. Vo. *Learning with sparsity and uncertainty by difference of convex functions optimization.* PhD thesis, University of Lorraine, France, 2015.

[231] L. Wang, J. Zhu, and H. Zou. The doubly regularized support vector machine. *Statistica Sinica*, 16 :589–615, 2006.

[232] S. Wang and J. Zhu. Model-based high-dimensional clustering and its application to microarray data. *Biometrics*, 64 :440 – 448, 2008.

[233] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh. Spiderboost and momentum : Faster stochastic variance reduction algorithms. *arXiv : Optimization and Control*, 2018.

[234] T. Warren Liao. Clustering of time series data—a survey. *Pattern Recognition*, 38(11) :1857–1874, 2005.

[235] J. Weston, A. Elisseeff, B. Scholkopf, and M. Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3 :1439–1461, 2003.

[236] D. M. Witten and R. Tibshirani. Penalized classification using Fisher's linear discriminant. *Journal of the Royal Statistical Society : Series B*, 73(5) :753–772, 2011.

[237] M. Woo and T. N. Sriram. Robust estimation of mixture complexity. *Journal of the American Statistical Association*, 101(476) :1475–1486, 2006.

[238] D. Wozabal, R. Hochreiter, and G. C. Pflug. A difference of convex formulation of value-at-risk constrained optimization. *Optimization*, 59(3) :377–400, 2010.

[239] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7) :2479–2493, 2009.

[240] M. Y. Wu, D. Q. Dai, X. F. Zhang, and Y. Zhu. Cancer subtype discovery and biomarker identification via a new robust network clustering algorithm. *PLOS One*, 2013.

[241] A. E. Xavier and V. L. Xavier. Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions. *SIAM J. Sci. Comput.*, 44 :70–77, 2011.

[242] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM J. Optim.*, 24(4) :2057–2075, 2014.

[243] B. Xie, W. Pan, and X. Shen. Penalized model-based clustering with cluster-specific diagonal covariance matrices and grouped variables. *Electronic Journal of Statistics*, 2 :168–212, 2008.

[244] Y. Xu, Q. Qi, Q. Lin, R. Jin, and T. Yang. Stochastic optimization for DC functions and non-smooth non-convex regularizers with non-asymptotic convergence. In *International Conference on Machine Learning*, pages 6942–6951. PMLR, 2019.

[245] L. Yang and L. Wang. Simultaneous feature selection and classification via semi-supervised models. In *Proceeding ICNC '07, Third International Conference on Natural Computation-Cover*, pages 646–650, 2007.

[246] Z. Yang, I. King, Z. Xu, and E. Oja. Heavy-tailed symmetric stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, pages 2169–2177, 2009.

[247] Z. Yang, J. Peltonen, and S. Kaski. Majorization-minimization for manifold embedding. In *Artificial Intelligence and Statistics*, pages 1088–1097, 2015.

[248] Q. Yao, J. Kwok, F. Gao, W. Chen, and T. Liu. Efficient inexact proximal gradient algorithm for nonconvex problems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3308–3314, 2017.

[249] C. Ye and Y. Cui. Stochastic successive convex approximation for general stochastic optimization problems. *IEEE Wireless Communications Letters*, 9(6) :755–759, 2019.

[250] J. K. Y.F. Li and Z. Zhou. Semi-supervised learning using label mean. In *Proceedings of the 26th International Conference on Machine Learning*, pages 1–8, 2009.

[251] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *VLDB*, volume 385, page 99, 2000.

[252] P. Yin, E. Esser, and J. Xin. Ratio and difference of l1 and l2 norms and sparse representation with coherent dictionaries. *Commun Inf Syst.*, 14 :87–109, 2015.

[253] P. Yin, Y. Lou, Q. He, and J. Xin. Minimization of l1–l2 for compressed sensing. *SIAM J Sci Comput.*, 37 :36–63, 2015.

[254] H. Zhang, J. Ahn, X. Lin, and C. Park. Gene selection using support vector machines with non-convex penalty. *Bioinformatics*, 2(1) :88–95, 2006.

[255] R. Zhang, T. B. Liu, and M. W. Zheng. Semi-supervised learning for classification with uncertainty. *Advanced Materials Research*, 440, 2012.

[256] T. Zhang. Some sharp performance bounds for least squares regression with regularization. *Ann. Statist.*, 37 :2109–2144, 2009.

[257] P. Zhao and T. Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *International conference on machine learning*, pages 1–9. PMLR, 2015.

[258] S. Zhao and F. You. Distributionally robust chance constrained programming with generative adversarial networks (gans). *AIChE Journal*, 66(6) :e16963, 2020.

[259] D. Zhou, P. Xu, and Q. Gu. Stochastic nested variance reduction for nonconvex optimization. *Journal of machine learning research*, 2020.

[260] H. Zhou, W. Pan, and X. Shen. Penalized model-based clustering with un-constrained covariance matrices. *Electronic Journal of Statistics*, 3 :1473 – 1496, 2009.

[261] Z. Zhou and J. Yu. A new nonconvex sparse recovery method for compressive sensing. *Frontiers in Applied Mathematics and Statistics*, 5 :14, 2019.

[262] X. Zhu and A. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan and Claypool, 2009.

[263] H. Zou. The adaptive lasso and its oracle properties. *J. Amer. Stat. Ass.*, 101 :1418–1429, 2006.

[264] H. Zou and R. Li. One-step sparse estimates in nonconcave penalized likelihood models. *Ann. Statist.*, 36(4) :1509–1533, 2008.

Résumé

Ce mémoire d'HDR est consacré au développement des méthodes avancées en programmtion DC (Difference of Convex functions) et DCA (DC Algorithm) pour différentes classes des problèmes d'optimisation non convexes et leurs applications à nombreuses thématiques de l'apprentissage automatique. Le mémoire est divisé en trois parties contenant de douze chapitres. Dans la première partie, nous présentons trois approches avancées de DCA pour plusieurs classes de problèmes d'optimisation non convexe. La première approche, nommée Accelerated DCA (ADCA), est pour but d'améliorer la vitesse de convergence de DCA. La deuxième approche, appelée DCA-Like, permet de résoudre deux classes des problèmes d'optimisation non convexes sans mettre en évidence une décomposition DC. La troisième approche, DCA Stochastique, est développée pour faire face aux problèmes avec big data. La deuxième partie du mémoire concerne une thématique difficile et extrêmement importante - l'optimisation parcimonieuse. Nous présentons deux approches majeures pour l'optimisation parcimonieuse : l'approximation DC et la reformulation via la pénalité exacte. Nous avons prouvé plusieurs résultats théoriques importants et dévelopé quatre méthodes DCA pour résoudre les problèmes résultants. Nous déployons ensuite nos deux approches proposées à la résolution de trois problèmes d'apprentissage automatique. La troisième partie concerne le développement des méthodes DCA pour un sujet fondamental de l'apprentissage et fouille de données : le clustering. Cinq problèmes en clustering sont considérés : le MSSC (Minimum of Sum of Squares Clustering), le MSSC avec pondération de variables, le clustering par blocs, le modèle de mélange gaussien, le clustering des séries temporelles.

Mots-clés: Non-convexe optimisation, Programmation DC, DCA, Apprentissage automatique

Abstract

This habilitation thesis is devoted to the development of advanced methods in DC (Difference of Convex functions) and DCA (DC Algorithm) programming for different classes of non-convex optimization problems and their applications in machine learning. The dissertation is divided into three parts containing twelve chapters. In the first part, we present three advanced DCA approaches for several classes of non-convex optimization problems. The first approach, Accelerated DCA (ADCA), aims to improve the convergence speed of DCA. The second approach, DCA-Like, allows to solve two classes of non-convex optimization problems without highlighting a DC decomposition of the objective function. The third approach, Stochastic DCA, is developed to deal with problems with big data. The second part of the thesis concerns a difficult and extremely important topic - sparse optimization. We present two major approaches for sparse optimization : DC approximation and nonconvex exact reformulation. Several important theoretical results and efficient DCA based method are presented. We then deploy our two proposed approaches to solve three machine learning applications. The third part concerns the development of DCA methods for a fundamental subject of learning and data mining : clustering. Five clustering problems are considered : MSSC (Minimum of Sum of Squares Clustering), MSSC with weighted variables, block clustering, Gaussian mixture model, clustering of large-scale time-series.

Keywords: Nonconex optimization, DC Programming, Advanced DCA, Machine Learning