Titre: Title:	Exact algorithms for minimum sum-of-squares clustering
Auteur: Author:	Daniel Aloise
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Aloise, D. (2009). Exact algorithms for minimum sum-of-squares clustering [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8451/

UNIVERSITÉ DE MONTRÉAL

EXACT ALGORITHMS FOR MINIMUM SUM-OF-SQUARES CLUSTERING

DANIEL ALOISE
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph.D.)
(MATHÉMATIQUES DE L'INGÉNIEUR)
JUIN 2009

Your file Votre référence
ISBN: 978-0-494-53792-3
Our file Notre référence
ISBN: 978-0-494-53792-3

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

EXACTS ALGORITHMS FOR MINIMUM SUM-OF-SQUARES CLUSTERING

présentée par: ALOISE Daniel

en vue de l'obtention du diplôme de: Philosophiæ Doctor

a été dûment acceptée par le jury d'examen constitué de:

M. AUDET Charles, Ph.D., président

M. ROUSSEAU Louis-Martin, Ph.D., membre et directeur de recherche

M. HANSEN Pierre, D. Agr., membre et codirecteur de recherche

M. CAPOROSSI Gilles, Ph.D., membre

M. BRUSCO Michael J., Ph.D., membre externe

à Caroline et à nos rêves

ACKNOWLEDGEMENTS

My first words are dedicated to Professor Pierre Hansen from whom I learned that beauty will always be out there, either in the progress of science or in a Chagall's painting. More than just a pleasure, I feel honored to have worked with him these last four years. I will never know how to thank him properly. I was also privileged to receive scientific advice and to benefit from the contribution of Professor Louis-Martin Rousseau. Additionally, I would like to thank Professors Charles Audet and Michael J. Brusco whose comments have helped improving this thesis.

I thank the Brazilian people for conceiving my Ph.D. scholarship, as well as the CAPES agency for providing the necessary support that allowed me to focus only on my studies. I am also grateful to the staff of the *Groupe d'Études et de Recherche en Analyse des Décisions* (GERAD) and to that of the *École Polytechnique de Montréal* for providing an excellent work environment during these four years.

I thank my Quebecer friends Simon Blanchard, Simon Boivin, Julie Paquette and Marie-Ève Rancourt, who presented to me a country full of colors and full of life. I wish to thank as well as "los amigos" Gerardo Berbeglia and Claudio Contardo.

I also use this opportunity to thank Alysson Costa, Patricia Hammes, Gisela Lima and Rafael Silveira for making my life happier here in Montreal. If I could, I would take all of them back to Brazil in my luggage.

I also thank my dear parents, grandparents as well as my brother for their counsels and for being there overseas when I called home.

Finally, I must thank the most important person in my life: Caroline, who was a fundamental element for this accomplishment. I will always be grateful of her support, friendship and love.

RÉSUMÉ

Étant donné un ensemble de n points dans l'espace Euclidien, la classification automatique selon le critère des moindres carrés consiste à partitionner cet ensemble en k classes de sorte que la somme des carrés des distances de chaque point au centroïde de sa classe soit minimum. Ceci est un problème fondamental dans le domaine de la classification automatique ayant de nombreuses applications dans diverses disciplines. Plusieurs heuristiques pour ce problème ont été et continuent à être proposées. Les méthodes exactes de résolution sont rares mais une variété d'approches ont été explorées.

Le premier chapitre de la thèse traite de la complexité du problème, un sujet qui mérite d'être clarifié. On remarque un manque de rigueur de la part de certains articles qui font des affirmations incorrectes ou non justifiées sur la difficulté du problème, souvent en l'associant avec d'autres problèmes de la classification automatique. Récenment, une preuve de NP-complétude pour le problème a été proposée par Drineas, Frieze, Kanan, Vempala et Vinay dans *Machine Learning*, 2004. Cependant, on montre que cette preuve n'est pas correcte. Une courte preuve alternative, dûe à Amit Deshpande et Preyas Popat, est donc fournie.

Les trois chapitres suivants de la thèse étudient trois approaches parmi les plus importantes pour la résolution exacte du problème. Au Chapitre 2, nous étudions un article récent de Sherali et Desai dans le *Journal of Global Optimization*, 2005. Dans cet article les auteurs proposent un algorithme de séparation et évaluation basé sur une reformulation-linéarisation du problème, déclarant avoir résolu des problèmes ayant jusqu'à 1000 points. Nous étudions leur algorithme en détail, en reproduisant une partie de leurs expériences de calcul. Toutefois, notre implantation a donné des temps de calcul qui se sont révélés être beaucoup plus élevés. En effet, pour deux ensembles de données de la littérature, seuls des exemples ayant jusqu'à 20 points ont pu être résolus en moins de 10 heures de temps de calcul. Les raisons possibles de cette grande différence sont discutées. On explore également l'effet d'une règle pour rompre la symétrie dûe à Plastria (*European Journal of*

Operational Research, 2002) et de l'introduction des inégalités valides appartenant à la fermeture convexe en deux dimensions des points qui peuvent appartenir à chaque classe.

Au Chapitre 3, on étudie l'article de Peng et Xia dans *Studies in Fuziness and Soft Computing*, 2005 sur l'équivalence entre la programmation 0-1 semi-définie positive et la classification automatique selon le critère de la moindre somme des carrés. En vue de la croissance rapide de l'ensemble de contraintes dans leur modèle, les auteurs n'ont fourni qu'une esquisse d'un algorithme pour résoudre le problème de façon exacte. On a donc développé un algorihme de branchement et coupes en suivant leurs lignes directrices mais en n'ajoutant que l'ensemble de contraintes violées. L'algorithme obtient des solutions exactes avec des temps de calculs comparables à ceux des meilleures méthodes exactes précédemment trouvées dans la littérature.

Finalement, le Chapitre 4 est dédié à l'approche par génération de colonnes dûe à du Merle, Hansen, Jaumard et Mladenović (*SIAM Journal on Scientific Computing*, 2000) et à ses améliorations. L'étape cruciale est la résolution du problème auxiliaire qui consiste à trouver une colonne avec un coût réduit négatif. Nous proposons une nouvelle manière de résoudre ce problème auxiliaire, basée sur des arguments géométriques. Ceci améliore grandement l'efficacité de l'algorithme entier et permet la résolution exacte d'exemples dans le plan ayant jusqu'à $n = 2392$ points et $k \geq 2$ classes, c'est-à-dire, 10 fois plus que précédemment. De plus, des exemples allant jusqu'à 19 dimensions et ayant jusqu'à $n = 2310$ points sont résolus de façon exacte dans le cas où beaucoup de classes sont utilisées.

ABSTRACT

Minimum sum-of-squares clustering (MSSC) consists in, given a set of n entities associated with points in s-dimensional Euclidean space, partitioning this set into k clusters in such a way that the sum of squared distances from each entity to the centroid of its cluster is minimum. This much studied problem is a basic one in cluster analysis and has application in numerous and diverse fields. Many heuristic algorithms for MSSC have been and continue to be regularly proposed. Exact solution methods are rare but a variety of approaches have been explored.

The first chapter of the thesis concerns complexity analysis of MSSC, a topic in which there seems to have been much confusion. We note indeed that several dozen papers have made incorrect or unjustified statements about NP-hardness of MSSC, usually confusing it with some other clustering problem. Recently, a proof was proposed by Drineas, Frieze, Kanan, Vempala and Vinay in *Machine Learning*, 2004. Unfortunately, as shown in this chapter, this proof is not correct. An alternate short proof, due to Amit Deshpande and Preyas Popat, is then provided.

The next three chapters of the thesis consider three of the main approaches to exact solution of MSSC. In chapter 2 we study a recent paper of Sherali and Desai in *Journal of Global Optimization*, 2005. In this paper the authors proposed a reformulation-linearization based branch-and-bound algorithm for this problem, claiming to solve instances with up to 1000 points. We investigated their method in further detail, reproducing some of their computational experiments. However, our computational times turn out to be drastically larger. Indeed, for two data sets from the literature only instances with up to 20 points could be solved in less than 10 hours of computer time. Possible reasons for this discrepancy are discussed. The effect of a symmetry breaking rule due to Plastria (*European Journal of Operational Research*, 2002) and of the introduction of valid inequalities of the convex hull of points in two dimensions which may belong to each cluster is also explored.

In chapter 3, we study the work of Peng and Xia (*Studies in Fuziness and Soft Computing*, 2005) on a 0-1 semidefinite programming (0-1 SDP) reformulation of MSSC. In view of the rapid increase in size of the set of constraints in their model, the authors only sketched an algorithm to exactly solve the problem. We then developed a branch-and-cut algorithm following those lines but adding only sets of violated constraints. The algorithm obtains exact solutions with computing times comparable with those of the best exact method previously found in the literature.

Finally, Chapter 4 is devoted to the column generation approach of du Merle, Hansen, Jaumard and Mladenović (*SIAM Journal on Scientific Computing*, 2000) and its improvements. The bottleneck of that algorithm is the resolution of the auxiliary problem of finding a column with negative reduced cost. We propose a new way to solve this auxiliary problem based on geometric arguments. This greatly improves the efficiency of the whole algorithm and leads to exact solution of instances in the plane with up to $n = 2392$ entities and $k \geq 2$ clusters, i.e., more than 10 times as much as previously done. Moreover, instances in up to 19 dimensions and with up to $n = 2310$ entities could be solved exactly when there are many clusters.

CONDENSÉ EN FRANÇAIS

La classification automatique est un outil puissant pour l'analyse de données. Étant donné un ensemble d'entités, elle consiste à trouver des sous-ensembles, appelés classes, qui sont homogènes et/ou bien séparés.

Un des plus importants types de classification automatique est la *partition*, où étant donné un ensemble $O = \{o_1, o_2, \ldots, o_n\}$ avec n entités, on cherche à trouver la partition $P_k = \{C_1, C_2, \ldots, C_k\}$ de O en k classes telle que

- $C_j \neq \emptyset \quad j = 1, \ldots, k$;
- $C_{j_1} \bigcap C_{j_2} = \emptyset \quad j_1, j_2 = 1, \ldots, k$ et $j_1 \neq j_2$; et
- $\bigcup_{j=1}^{k} C_j = O$.

qui optimise un critère donné.

Plusieurs critères ont déjà été utilisés dans la littérature pour exprimer l'homogénéité et/ou la séparation des classes qui doivent êtres trouvées (voir e.g. [53]). Un critère clé est celui de la moindre somme des carrés des distances Euclidiennes de chaque point au centre de sa classe. Le problème de trouver la partition optimale des entités selon ce critère est dénoté par MSSC (a partir de l'anglais *Minimum Sum-of-Squares Clustering*). L'heuristique classique k-means [79] résout approximativement MSSC. Cet algorithme a été considéré par *IEEE Computer Society* comme le deuxième plus influent dans la comunnauté d'exploitation de données [125].

Le partitionnement selon le critère de la moindre somme des carrés des distances Euclidiennes a plusieurs propriétés. Voici certaines d'entre elles:

(i) Il exprime l'homogénéité et la séparation comme expliqué dans [111], pp. 60–61.

(ii) Étant donné les affectations, les centres des classes sont situés à leurs centroïdes, dû aux conditions d'optimalité du premier ordre. Ceux-ci sont déterminés par une expression simple.

(iii) Étant donné les centroïdes, chaque entité est affectée à son centroïde le plus près en raison de l'optimalité locale. Ceci n'exige que quelques comparaisons.

(iv) Les classes obtenues sont sphéroïdales du fait de la minimisation des carrés des distances. Cette propriété peut être souhaitable ou non, selon le problème étudié.

Une formulation mathématique pour MSSC est donnée par:

$$\min_{x,y} \quad \sum_{i=1}^{n}\sum_{j=1}^{k} x_{ij}\|p_i - y_j\|^2$$

sujet à

$$\sum_{j=1}^{k} x_{ij} = 1 \qquad \forall i = 1,\ldots,n$$

$$x_{ij} \in \{0,1\} \qquad \forall i = 1,\ldots,n; \forall j = 1,\ldots,k.$$

Les n entités $\{o_1, o_2, \ldots, o_n\}$ à être classifiées sont situées aux points $p_i = (p_i^r, r = 1,\ldots,s)$ de $\mathbb{R}^s$ pour $i = 1,\ldots,n$; k centres de classes doivent être situés à des points inconnus $y_j \in \mathbb{R}^s$ pour $j = 1,\ldots,k$; la norme $\|\cdot\|$ dénote la distance Euclidienne entre les deux points de l'argument dans l'espace à s dimensions considéré. Les variables de décision x_{ij} expriment l'affectation de l'entité o_i à la classe j. On assume que le nombre d'entités n est plus grand que k, autrement le problème est résolu trivialement en situant un centre de classe à la position de chaque entité.

Les propriétés mathématiques du problème sont abordées dans les livres de Späth [111], Mirkin [88] et Kogan [69]. Plusieurs centaines d'articles ont été écrits sur des heuristiques visant à résoudre MSSC et plusieurs milliers sur des applications dans divers domaines (voir e.g. la synthèse d'un demi-siècle faite par Steinley [113]). Les principales heuristiques pour MSSC comprennent la méthode j-means de Hansen et Mladenović [55], la méthode global k-means de Likas, Vlassis and Verbeek [77], qui a été analysée dans [58] et puis modifiée

par Bagirov [6], l'algorithme d'optimisation non-lisse de Bagirov and Yearwood [7], les algorithmes d'optimisation lisse attribués à Teboulle et Kogan [116] et Xavier et al. [126], les approaches métaheuristiques développées dans [85, 93, 92, 115, 72, 73], la méthode de partionnement par génération de colonnes restrictive de Christou [15], et l'heuristique D.C. de An, Belgueti et Tao [3]. Une comparison systématique de douze heuristiques pour MSSC a été effectuée par Brusco et Steinley dans [13].

Les méthodes exactes sont beaucoup moins nombreuses que les heuristiques. Au meilleur de notre connaissance, il y a moins d'une douzaine d'articles sur le sujet. En 1973, Diehr a déclaré dans [23] (p.17) que "*Les chercheurs doivent garder à l'esprit que dans la plupart des cas les buts de la classification automatique ne justifient pas les temps de calcul nécessaires pour trouver ou vérifier la solution optimale*" (traduction libre de l'anglais). Cette déclaration, cependant, ne prend pas en compte trois faits:

- Les méthodes exactes sont largement utilisées maintenant pour ajuster ou pour découvrir des écueils dans les méthodes heuristiques ou bien pour suggérer des nouvelles approches;
- La performance des ordinateurs s'est beaucoup améliorée au cours des dernières décennies;
- La programmation mathématique a beaucoup évolué au cours des trente dernières années.

Du point de vue de la programmation mathématique, selon un rapporteur d'un de nos articles, "*La classification automatique selon le critère de la moindre somme des carrés des distances est un problème stimulant d'optimisation globale .*" (traduction libre de l'anglais)

Pour $k \geq 2$ en une dimension, MSSC peut être résolu en temps $O(n^3)$ [111]. Si k et la dimension s sont fixés, le problème peut être résolu en temps $O(n^{sk+1})$ [61], ce qui peut être très coûteux même pour des exemples dans le plan.

Plusieurs affirmations incorrectes ont été énoncées quant au caractère NP-complet du problème pour une dimension Euclidienne s quelconque. Une source fréquente de confusion

est la lecture trop rapide d'un article de Brücker [11] où l'auteur prouve la NP-complétude de plusieurs problèmes de la classification automatique, quoique rien ne soit dit à propos de MSSC. De plus, une preuve de NP-complétude de Garey, Johnson et Witsenhausen [44] est applicable seulement au problème de quantification. Ce dernier est en fait un problème particulier de k-médiane où chaque centre de classe est choisi a partir d'un ensemble fini de positions.

Récemment, une preuve de NP-complétude pour MSSC avec $k = 2$ en s dimensions a été donnée par Drineas et al. dans *Machine Learning* 56, 9–33, 2004. On montre que cette preuve est, toutefois, invalide. Une courte preuve alternative, dûe à Deshpande et Popat [20], est fournie via une réduction du problème de la coupe la plus dense. Plus récemment, Mahajan, Nimbhorkar et Varadarajan [80] ont prouvé que MSSC est NP-complet pour des valeurs k quelconques même dans le plan.

L'objectif de cette thèse est double: d'un côté estimer l'état de l'art concernant les méthodes exactes pour MSSC et d'autres parts d'améliorer autant que possible ces méthodes.

Récemment, Sherali et Desai [108] ont proposé un algorithme de séparation et évaluation basé sur une reformulation linéaire du problème. Ce modèle est obtenu après avoir généré de nouvelles contraintes via l'emploi de multiplications de contraintes existantes et en redéfinissant quelques variables.

Sherali et Desai [108] ont rapporté des résultats de calculs pour des grands exemples ayant jusqu'à 1000 points en 8 dimensions. Toutefois, quelques détails ont mérité d'être investigués. En particulier, les valeurs d'écart rapportées entre les bornes inférieures et supérieures semblent être trop grandes. De ce fait, le nombre de noeuds évalué par la méthode de séparation et d'évaluation devrait être élevé, mais demeure moderé. De plus, les auteurs résolvent un petit exemple pour lequel l'algorithme k-means donne un résultat avec une valeur deux fois plus grande que celle obtenue par l'algorithme de séparation et d'évaluation, ce qui semble a priori peu probable.

On a essayé de reproduire de tels résultats sans succès. À cet effet, on a implanté l'algorithme de Sherali and Desai en suivant autant que possible la description donnée

dans leur article. On a considéré de petits bases de données obtenues en sélectionant des sous-ensembles de l'ensemble de données de Fisher avec 150 entités [36]. On a ainsi observé que les temps de calculs obtenus par notre implantation pour la résolution d'un petit exemple avec 20 entités était déjà assez grands (c'est-à-dire, plus de 6 heures de calculs sur un Pentium IV 2 GHz). On a discuté avec Sherali et Desai à propos des raisons possibles d'une telle différence entre leurs résultats et les nôtres. Aucune explication n'a pu être donnée puisque "*Malheureusement, il* [Jitamitra Desai] *semble avoir supprimé ses codes et ses données*" [106] (traduction libre de l'anglais). L'explication la plus probable semble être que les exemples de tests utilisés par Sherali et Desai étaient trop faciles à résoudre (c'est-à-dire, les classes étaient très bien séparées).

Quoique les résultats de Sherali et Desai [108] n'ont pas pu être reproduits, on a évalué l'intérêt de différentes règles pour éliminer la symmétrie dans leur modèle. En particulier, Plastria a proposé dans [100] d'éliminer la symmétrie en n'acceptant que des solutions lexicographiques minimales, c'est-à-dire, tel que chaque classe j contient le point d'index le plus bas n'appartenant à aucune classe d'index $1, \ldots, j-1$. Selon cette propriété, il n'y a qu'une seule façon d'indexer les classes. Cette règle d'élimination de la symmétrie semble être meilleure que celles proposés dans [108], à la fois en termes de réduction du nombre de noeuds ainsi qu'en termes de temps de calculs.

De plus, on a étudié l'impact de l'ajout des contraintes valides obtenues à partir de l'enveloppe convexe de points qui peuvent être affectés à une classe. Dans [108], les auteurs utilisent plutôt un hyperrectangle $\overline{H}(I_j)$ qui inclut l'enveloppe convexe de points qui peuvent encore être affectés à une classe donnée j, dénoté I_j, pour chaque $j = 1, \ldots, k$:

$$\overline{H}(I_j) = \{y_j : \alpha_j^r \leq y_j^r \leq \beta_j^r, r = 1, \ldots, s\},$$

où, $\alpha_j^r = \min\{p_i^r : i \in I_j\}$ et $\beta_j^r = \max\{p_i^r : i \in I_j\}$, $\forall r = 1, \ldots, s$.

Puisque chaque paire de points extrêmes de l'enveloppe convexe peut définir un demi-espace dans le plan euclidien, les coordonnées des centroïdes sont confinées à être dans le polyhèdre défini par l'intersection de ces demi-espaces. Malheureusement, le nombre de contraintes à la sortie est sensible au nombre de points extrêmes donnés par l'algorithme de

Graham [46]. Un nombre $O(kn)$ de contraintes sont nécessaires dans le modèle quand les hyperrectangles sont utilisés, tandis que ce nombre augmente à $O(kn^2)$ avec les inégalités de l'enveloppe convexe, puisque toutes les entités peuvent être des points extrêmes de l'enveloppe convexe. Les expériences de calculs réalisées pour le cas à deux dimensions ont montré que le nombre de noeuds de l'arbre de résolution est réduit. Cependant, une telle réduction n'amène pas nécessairement une réduction du temp de calcul. En effet, cela est dû à l'augmentation du nombre de contraintes qui implique que la résolution du modèle est plus coûteuse.

La tâche la plus difficile au moment de développer des méthodes exactes pour MSSC est celle de calculer de bonnes bornes inférieures dans un temps de calculs raisonable. Récemment, Peng and Xia [98] ont utilisé des opérations matricielles pour modéliser le problème comme un programme 0-1 sémi-défini positif (0-1 SDP) de la façon suivante:

$$\begin{aligned} &\min_{Z} \ Tr(W_p W_p^T (I - Z)) \\ &\text{sujet à} \\ &Ze = e, Tr(Z) = k, \\ &Z \geq 0, Z = Z^T, Z^2 = Z. \end{aligned}$$

où $W_p \in \mathbb{R}^{n \times s}$ est la matrice dont la i-ème ligne est le vecteur p_i. Ceci peut ensuite être relaxé et donner un problème SDP convexe ou un programme linéaire.

En utilisant les résultats de Peng and Xia [98], on propose un algorithme de branchement et coupes afin d'exploiter de façon efficiente les bornes inférieures obtenues à partir de la relaxation linéaire du modèle 0-1 SDP. Cette relaxation consiste à remplacer les contraintes $Z = Z^2$. Peng and Xia [98] prouvent que les inégalités suivantes sont satisfaites par toutes les solutions de leur formulation.

$$\begin{aligned} Z_{ij} &\leq Z_{ii} && \forall i, j \\ Z_{ij} + Z_{i\ell} &\leq Z_{ii} + Z_{j\ell} && \forall i, j, \ell \end{aligned}$$

En vue de la croissance rapide de l'ensemble de contraintes dans leur modèle, les auteurs n'ont fourni qu'une esquisse d'un algorithme pour résoudre le problème de façon exacte. On a donc développé un algorihme de branchement et coupes en suivant leurs lignes directrices, mais en n'ajoutant que l'ensemble de contraintes violées. L'algorithme obtient des solutions exactes avec des temps de calculs comparables à ceux des meilleures méthodes exactes précédemment trouvées dans la littérature, c'est-à-dire, l'algorithme de génération de colonnes proposé par du Merle et al. [28] et l'algorithme de séparation et d'évaluation répétitive de Brusco [12]. Plus précisément, l'algorithme de branchement et coupes basé sur la rélaxation linéaire du modèle 0-1 SDP obtient des solutions exactes pour des exemples avec $n = 202$ entités et $k = 9$ classes dans le plan en moins de 12 heures.

Une méthode de génération de colonnes pour MSSC a été proposée par du Merle et al. dans [28]. En effet, les problèmes de partitionnement dans le domaine de la classification automatique peuvent aussi être formulés mathématiquement en considérant toutes les classes possibles. Soit une classe C_t pour laquelle

$$a_{it} = \begin{cases} 1 & \text{si l'entité } o_i \text{ appartient à la classe } C_t \\ 0 & \text{sinon,} \end{cases}$$

et soit y_t le centroïde des points p_i tels que $a_{it} = 1$. Ainsi, le coût c_t de la classe C_t peut être écrit par:

$$c_t = \sum_{i=1}^{n} \|p_i - y_t\|^2 a_{it}.$$

Une formulation alternative pour MSSC est donc donnée par

$$\min_z \sum_{t \in T} c_t z_t$$

sujet à

$$\sum_{t \in T} a_{it} z_t = 1 \qquad \forall i = 1, \ldots, n$$

$$\sum_{t \in T} z_t = k$$

$$z_t \in \{0, 1\} \qquad \forall t \in T,$$

où $T = \{1, \ldots, 2^n - 1\}$. Les variables z_t sont égales à 1 si la classe C_t est dans la partition optimale et égales 0 sinon. Le premier ensemble de contraintes permet d'assurer que chaque entité appartient à une classe, et la contrainte suivante impose que la partition optimale contienne exactement k classes.

Cette formulation correspond à un problème de partitionnement d'ensembles de grande taille avec une contrainte additionelle dont le nombre des variables est exponentiel en termes du nombre n d'entités. La méthode de génération de colonnes proposée dans [28] travaille avec un petit sous-ensemble de colonnes du modèle généré itérativement. Elle a résolu pour la première fois des exemples de taille moyenne (c'est-à-dire, des exemples avec 100-200 entités), incluant l'ensemble de données de Fisher avec 150 entités [36]. Le problème maître est résolu par la méthode de points intérieurs (ACCPM, *Analytical Center Cutting Plane Method*) de Goffin, Haurie et Vial [45]. Le problème auxiliaire dont l'objectif est de trouver une colonne avec un coût réduit négatif est exprimé comme un programme hyperbolique en variables 0-1. Ce problème est résolu par un algorithme inspiré de celui de Dinkelbach [24] qui utilise lui même un algorithme de séparation et d'évaluation pour résoudre des problèmes d'optimisation de fonctions quadratiques en variables 0-1 sans contraintes. Un autre algorithme de séparation et d'évaluation appliqué au problème maître conduit, si nécessaire, à une solution entière. Finalement, des heuristiques de recherche à voisinages variables (VNS) sont utilisées à la fois au début pour trouver une bonne solution initiale avec des bornes sur les variables duales, ainsi que dans la résolution du problème auxiliaire afin de l'accélérer. La partie la plus coûteuse de l'algorithme réside dans la résolution de son problème auxiliaire qui est exprimé par:

$$\pi^* = \sigma + \min_{y_v \in \mathbb{R}^s, v \in \mathbb{B}^n} \sum_{i=1}^{n} (\|p_i - y_v\|^2 - \lambda_i) v_i,$$

où y_v qui dénote le centroïde de points p_i pour lequels $v_i = 1$. Si $\pi^* < 0$, alors la solution optimale v^* pour le problème ci-dessus est ajoutée sous la forme d'une colonne au problème de partitionnement d'ensembles avec sa variable associée. Autrement, le problème maître relaxé a déjà été résolu.

On propose une nouvelle façon de résoudre le problème auxiliaire basée sur des arguments géométriques. Ce problème peut être vu par analogie comme une minimisation de la somme de fonctions égales aux distances au carré de chaque centre de classe y_v à chacune des entités, mais avec une limite pour chacune de ces distances, après laquelle la fonction correspondante n'augmente plus. En effet, étant donné une localisation y_v, v_i est égal à 1 si $\|p_i - y_v\|^2 \leq \lambda_i$, et à 0 sinon. Géométriquement, dans le plan, ceci est équivalent à la condition où $v_i = 1$ si y_v appartient au disque de rayon $\sqrt{\lambda_i}$ ayant p_i comme centre, sinon $v_i = 0$.

L'adaptation et la complémentation d'un algorithme énumératif de Drezner, Mehrez et Wesolowsky [26] permettent la résolution du problème auxiliaire avant d'effectuer un branchement en temps $O(n^3)$. Si un branchement est nécessaire, la règle de branchement classique de Ryan et Foster [104] est appliquée.

À plusieurs dimensions, l'énumération serait trop longue quoique la propriété de base peut encore être exploitée. Une condition suffisante pour que deux entités ne soient pas dans la même classe est utilisée afin de remplacer des coefficients dans le problème non-contraint quadratique en variables 0-1 par des valeurs arbitrairement grandes. Ensuite, un algorithme de séparation et d'évaluation est appliqué dans un schéma de suppresion de noeuds. À cet effet, un graphe est construit ayant des noeuds associés aux entités et des arêtes associées à des paires d'entités qui ne sont pas trop éloignées l'une de l'autre, c'est-à-dire pour lesquelles les hypersphères de rayon $\sqrt{\lambda_i}$ et $\sqrt{\lambda_j}$ s'intersectent. Récursivement, un noeud de degré minimal dans ce graphe est sélectionné et le sous-graphe induit par ses noeuds adjacents est examiné. Le problème quadratique en variables 0-1 sans contraintes associé à ce sous-graphe est résolu et la solution optimale sauvegardée dans le cas où elle est meilleure que la solution courante.

L'application de ces nouvelles règles a conduit à un progrès substantiel. En effet, des exemples dans le plan ayant jusqu'à $n = 2392$ entités et $k \geq 2$ ont pu être résolus en de temps de calculs (longs, mais encore) raisonables. De plus, des exemples allant jusqu'à 19 dimensions et ayant jusqu'à $n = 2310$ points ont pu être résolus de façon exacte dans le cas de l'utilisation de plusieurs classes.

Pour conclure, les approches exactes de résolution de MSSC peuvent être séparées en trois familles:

1. celles qui résolvent de petits exemples ($n \approx 25$), c'est à dire la programmation dynamique non-sérielle [119], la programmation concave [127] et la technique de reformulation-linéarisation [108].

2. celles qui résolvent des exemples de taille moyenne ($n \approx 100 - 200$), c'est-à-dire la méthode de séparation et d'évaluation répétitive [12], la méthode de branchement et coupes basée sur le modèle de programmation sémi-définie positive 0-1 [2, 98] et la méthode de génération de colonnes sans les améliorations géométriques [28].

3. celle qui peut résoudre des problèmes de grande taille ($n \approx 2000$), c'est-à-dire la méthode de génération de colonnes améliorée presentée dans cette thèse.

D'une façon générale, on peut considérer nos résultats comme une preuve de la réalisabilité de l'approche par génération de colonnes pour résoudre des problèmes appartenant au domaine de la classification automatique. Il y a plusieurs critères proposés dans la littérature pour exprimer l'homogénéité et/ou la séparation des classes. Un projet de construction d'un progiciel de génération de colonnes pour la classification automatique, impliquant plusieurs professeurs du GERAD ainsi que plusieurs étudiants, est actuellement en cours d'éxpérimentation. Certainement, le succès du progiciel sur un ou plusieurs critères dépendra grandement de deux facteurs: la facilité de résolution du problème auxiliaire et la présence d'un petit ou grand saut de dualité. Plus d'effort algorithmique et d'implémentation seraient nécessaires. Les critères qui seront étudiés sont des critères récemment proposés dans les communautés de l'exploitation de données et de la physique: e.g. la coupe par ratio [50], la coupe normalisée [109] et la modularité [17].

De plus, quoique on ait mis l'accent sur une approche basée sur une méthode de points intérieurs, (c'est-à-dire ACCPM et génération de colonnes) la programmation linéaire stabilisée [29] pourrait encore être un concurrent en particulier si elle est combinée avec les récents travaux de [31, 32] sur le traitement efficace de la dégénérescence.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Clustering is a basic chapter in data analysis. It addresses the following problem: given a set of entities find subsets, called clusters, which are homogeneous and/or well separated (e.g. Hartigan [59]; Jain, Murty and Flynn [62]; Mirkin [87]). Homogeneity means that entities in the same cluster must be similar and separation that entities in different clusters must differ one from another.

One of the most used types of clustering is *partitioning*, where given a set $O = \{o_1, o_2, \ldots, o_n\}$ of n entities, we look for a partition $P_k = \{C_1, C_2, \ldots, C_k\}$ of O into k clusters such that

- $C_j \neq \emptyset \quad j = 1, \ldots, k$;
- $C_{j_1} \bigcap C_{j_2} = \emptyset \quad j_1, j_2 = 1, \ldots, k$ and $j_1 \neq j_2$; and
- $\bigcup_{j=1}^{k} C_j = O$.

Many different criteria are used in the literature to express homogeneity and/or separation of the clusters to be found (see [53] for a survey). For instance, one may desire to maximize the *split* of a partition, i.e., the minimum dissimilarity between two entities assigned to two different clusters [19, 37], or to minimize the *diameter*, i.e., the largest dissimilarity between a pair of entities in the same cluster [52]. Among these criteria, a frequently used one is the minimum sum of squared Euclidean distances from each entity to the centroid of the cluster to which it belongs. Partitioning n entities into k clusters with this criterion is known as minimum sum-of-squares clustering (MSSC).

For $k \geq 2$ and one dimensional data, MSSC can be solved in $O(n^3)$ time [111]. The problem is NP-hard in the plane for general values of k [80]. In general dimension, MSSC is NP-hard even for $k = 2$ [1]. If both k and dimension s are fixed, the problem can be solved in $O(n^{sk+1})$ time [61], which may be very time-consuming even for instances in the plane.

MSSC has several properties:

(i) It expresses both homogeneity and separation as explained in Späth's book [111], pages 60–61;

(ii) Given the assignments, the cluster centers are located in their centroids, due to first order optimality conditions. These are determined by a simple closed-form expression;

(iii) Given the centroids, each entity is assigned to its closest centroid, due to local optimality. This just requires a few comparisons;

(iv) Clusters obtained are spheroidal due to minimization of squared Euclidean distances. This may be desirable or not, depending on the problem considered.

A mathematical programming formulation of MSSC is as follows:

$$\min_{x,y} \quad \sum_{i=1}^{n}\sum_{j=1}^{k} x_{ij}\|p_i - y_j\|^2$$

subject to

$$\sum_{j=1}^{k} x_{ij} = 1 \qquad \forall i = 1,\ldots,n \tag{1}$$

$$x_{ij} \in \{0,1\} \qquad \forall i = 1,\ldots,n; \forall j = 1,\ldots,k.$$

The n entities $\{o_1, o_2, \ldots, o_n\}$ to be clustered are at given points $p_i = (p_i^r, r = 1,\ldots,s)$ of $\mathbb{R}^s$ for $i = 1,\ldots,n$; k cluster centers must be located at unknown points $y_j \in \mathbb{R}^s$ for $j = 1,\ldots,k$; the norm $\|\cdot\|$ denotes the Euclidean distance between the two points in its argument in the s-dimensional space under consideration. The decision variables x_{ij} express the assignment of the entity o_i to the cluster j. We assume that the number of entities n is greater than k, otherwise the problem is trivially solved by locating one cluster center at the position of each entity.

If y is fixed, the condition $x_{ij} \in \{0,1\}$ can be replaced by $x_{ij} \in [0,1]$, since in an optimal solution for the resulting problem each entity belongs to the cluster with the nearest center.

Besides, for a fixed x, first order conditions on the gradient of the objective function require that at an optimal solution

$$\sum_{i=1}^{n} x_{ij}(y_j^r - p_i^r) = 0, \ \forall j, r, \quad \text{i.e.,} \quad y_j^r = \frac{\sum_{i=1}^{n} x_{ij} p_i^r}{\sum_{i=1}^{n} x_{ij}}, \ \forall j, r. \tag{2}$$

Hence, the optimal cluster centers are always at the centroids of the clusters.

Other mathematical properties of MSSC are discussed in the books of Späth [111], Mirkin [88] and Kogan [69]. Several hundred papers have been written on heuristics for MSSC and several thousand on their applications in many domains (see, for instance, Steinley's half century synthesis [113]). The best known heuristic for MSSC is k-means [38, 79] (the continuous version of k-means for space partitioning was previously described by Steinhaus in [112]) which was identified by the IEEE Computer Society as the 2nd most influential algorithm in the data mining community [125]. Indeed MSSC is sometimes called the k-means problem. This heuristic alternately applies properties (ii) and (iii) above until a local optimum is reached. It has been shown by Hansen and Mladenović [55] that while k-means usually gives good results for small number of clusters its performance deteriorates, sometimes drastically, when this number increases. Modifying k-means by adding a *jump* move of a centroid to an entity location gives a much better heuristic called j-means. Finally, combining j-means with a Variable Neighborhood Search (VNS) heuristic [56, 57, 89] gives a heuristic which often provides optimal solutions or best known ones.

Other recent heuristics for MSSC include the global k-means method of Likas, Vlassis and Verbeek [77], analyzed in [58] and modified by Bagirov [6], Bagirov and Yearwood's nonsmooth optimization algorithm [7], smoothing optimization algorithms due to Teboulle and Kogan [116] and Xavier et al. [126], Merz's iterated local search [85], Pacheco's scatter search [92], Pacheco and Valencia's hybrids [93], Taillard's decomposition methods [115], Laszlo and Mukherjee's genetic algorithms [72, 73], Christou's restricted column generation and partitioning method [15], and the D.C. heuristic of An, Belghiti and Tao [3]. A

systematic comparison of twelve heuristics for MSSC was made by Brusco and Steinley in [13].

Exact algorithms for MSSC are much less numerous than heuristics. To the best of our knowledge, there are less than a dozen papers published on that topic. In 1973, Diehr stated in [23] (p. 17) that "*Researchers must keep in mind that in most of cases the goals of clustering do not justify the computational time to locate or verify the optimal solution*". This statement, however, does not take into account three facts:

- Exact methods are extensively used nowadays to tune or discover pitfalls on existing approximate methods as well as to derive new approaches.
- Computer performance has greatly improved in the last decades.
- Mathematical programming has evolved a lot in 30 years.

From the mathematical programming point of view, as pointed out by a referee of one of our papers, "*Minimum sum-of-squares clustering is a challenging global optimization problem*". Indeed this thesis will cover quite diverse approaches that can be used to exactly solve the problem.

Early branch-and-bound algorithms are due to Koontz, Narendra and Fukunaga [70] and Diehr [22]. Bounds depend on distances between entities assigned to the same cluster and a limited look-ahead component.

A column generation method for MSSC was proposed by du Merle et al. in [28]. It solved for the first time medium size benchmark instances (i.e., instances with 100-200 entities), including Fisher's Iris [36]. The master problem is solved by the ACCPM interior point method of Goffin, Haurie, and Vial [45]. The auxiliary problem of finding a column with negative reduced cost is expressed as a hyperbolic program in 0-1 variables. It is solved by a Dinkelbach-like algorithm [24] which relies on a branch-and-bound algorithm for unconstrained quadratic 0-1 optimization. Another branch-and-bound on the master problem leads, if needed, to an integer solution. Finally, VNS heuristics are used both at the outset to find a good initial solution together with tentative bounds on the dual

variables, as well as in the auxiliary problem to accelerate its solution. The bottleneck of the algorithm lies in the resolution of its auxiliary problem, and more precisely, in the unconstrained quadratic 0-1 optimization problem arising there.

More recently, Xia and Peng [127] proved that the objective function of MSSC is concave in the relaxed feasible domain. In their paper, they propose an adaptation of Tuy's [118] cutting plane method to solve it. Approximate results are reported for a version where this algorithm is halted before global convergence. Some experiments of ours showed that small instances with about 25 entities can be solved exactly with that approach.

MSSC can also be solved by non-serial dynamic programming as shown by Jensen [63]. An improved implementation due to van Os and Meulman [119] allows solutions of instances with about 28 entities.

Brusco [12] proposed a repetitive branch-and-bound procedure which, after ordering the entities, solves by branch-and-bound the problem defined by the $k+1$ last ones, then the problem with $k+2$ last ones, and so on, until the problem with all given entities is solved. The bound used at any iteration of one of those iterated branch-and-bound procedures comprises two components, i.e., an usual one corresponding to distances between already assigned entities and a sophisticated look-ahead one which corresponds to distances in an optimal solution for the set of unassigned entities. These much improved bounds led to efficient solution of some well-known benchmark instances, including Fisher's 150 Iris [36], particularly when the number of cluster is small. Artificially generated examples with well-separated clusters and up to $n = 220$ entities could be solved also.

The hardest task when devising exact algorithms for MSSC is to compute good lower bounds in a reasonable amount of time. Sherali and Desai [108] proposed to obtain such bounds by linearizing the model via the reformulation-linearization technique (RLT) [107]. They claim to solve instances with up to 1,000 entities by means of a branch-and-bound algorithm. Recently, Peng and Xia [98] proved the equivalence of MSSC and a model called 0-1 semidefinite programming (0-1 SDP), in which eigenvalues are binary. The authors report in [98] values of lower bounds obtained from LP and SDP relaxations of this 0-1 SDP MSSC formulation.

This thesis consists of four main chapters which are largely independent. Chapter 1 is dedicated to the computational complexity of MSSC, a topic in which there seems to have been much confusion. We show that a recent proof provided by Drineas et al. in [27] regarding the complexity of MSSC in general Euclidean dimension is invalid. An alternate short proof due to Amit Deshpande and Preyas Popat (our co-authors to a forthcoming paper) is then given. Chapter 2 concerns an extensive empirical evaluation of the RLT-based branch-and-bound algorithm of [108], trying to reproduce the same results obtained in that paper without success. In chapter 3, we study the 0-1 SDP MSSC formulation of Peng and Xia [98]. On the basis of their work, we propose a branch-and-cut algorithm based on cutting with violated triangle inequalities, i.e., if the pairs of entities (o_i,o_j) and (o_i,o_ℓ) belong to the same cluster, then entities o_i and o_ℓ also belong to the same cluster. The resulting algorithm obtains exact solutions for some benchmark data sets with computing times comparable with those of the best exact methods previously found in the literature [12, 28]. In Chapter 4, the column generation approach of du Merle et al. [28] is revisited and an alternate geometric-based approach for the solution of its auxiliary problem is proposed. This greatly improves the efficiency of the whole algorithm and leads to exact solution of instances with over 2300 entities.

CHAPTER 1 : NP-HARDNESS OF EUCLIDEAN SUM-OF-SQUARES CLUSTERING

1.1 Computational complexity

First of all, it is important to remark that the computational complexity of a clustering problem depends on the criterion used. For instance, split maximization is polynomially solvable [19] while diameter minimization is NP-hard [11, 52].

To the best of our knowledge, the computational complexity of minimum sum-of-squares clustering in general Euclidean space for $k \geq 2$ was unknown before the present work. However, several incorrect statements have been made about this problem being known to be NP-hard, many of them without providing a reference [35, 40, 81, 83, 97, 98, 101, 110, 124, 128]

Some confusion is also made in [14, 21, 41, 42, 49, 75, 90] by referring to a paper of Garey, Johnson and Witsenhausen [44], which provides a NP-hardness proof for the *quantization* problem by a reduction from the exact covering problem by triples, which is known to be a NP-complete problem [43]. The quantization problem is defined in [44] as follows.

> "A source produces one sample of a random variable X with equiprobable values in $\{1, 2, \ldots, n\}$.
> The encoder (quantizer) maps X into a variable Y with values in $\{1, 2, \ldots, k\}$. The decoder maps Y into a decision variables Z with values in $\{1, 2, \ldots, m\}$. If $X = i$ and $Z = j$ the resulting distortion is d_{ij}. All entries in the $n \times m$ matrix $[d_{ij}]$ are zeros or ones. The goal is to find an encoder function, $f : X \to Y$, and a decoder function, $g : Y \to Z$, such that the average distortion
>
> $$\frac{1}{n} \sum_{i=1}^{n} d_{ig(f(i))}$$
>
> is as small as possible."

However, this is in fact a particular k-median problem (see e.g. [71] for a survey) where each cluster center is taken from a given finite set of fixed potential locations. This problem was already known to be NP-hard for $k \geq 2$ [66].

Other results due to Brücker [11] led to further confusion. This author proved that the partitioning problem is NP-hard for many different clustering criteria. In the classical book *Computers and Intractability* of Garey and Johnson [43], this paper is referenced in the following way:

> "[MS9] CLUSTERING
> INSTANCE: Finite set X, a distance $d(x, y) \in Z_0^+$ for each pair $x, y \in X$, and two positive integers K and B.
> QUESTION: Is there a partition of X into disjoint sets $X_1, X_2, \ldots, X_k$ such that, for $1 \leq i \leq k$ and all pairs $x, y \in X_i$, $d(x, y) \leq B$?
> Reference: **[Brücker, 1978]** Transformation from GRAPH 3-COLORABILITY.
> Comment: Remains NP-complete even for fixed $K = 3$ and all distances in $\{0, 1\}$. Solvable in polynomial time for $K = 2$. Variants in which we ask that the sum, over all X_i, of $\max\{d(x, y) : x, y \in X_i\}$ or of $\sum_{x,y \in X_i} d(x, y)$ be at most B are similarly NP-complete (with the last one NP-complete even for $K = 2$)."

The problem described here is minimum diameter partitioning.

Despite the fact that nothing is mentioned about squared Euclidean distances in [11], many papers cited it to state that the MSSC is NP-hard [25, 55, 84, 85, 86, 92, 93, 96, 117, 131].

The papers [34, 64, 99, 116, 122] also cite Garey and Johnson's book without mentioning Brücker as a reference for MSSC to be NP-hard. This error may be due to the paragraph cited above or possibly to another one which refers to minimum sum-of-squares,

> "[SP19] MINIMUM SUM OF SQUARES
> INSTANCE: Finite set A, a size $s(a) \in Z^+$ for each $a \in A$, positive integers $K \leq |A|$ and J.

QUESTION: Can A be partitioned into K disjoint sets $A_1, A_2, \ldots, A_K$ such that

$$\sum_{i=1}^{K} \left(\sum_{a \in A_i} s(a) \right)^2 \leq J.$$"

Clearly, this last problem is different from MSSC.

Recently, a proof of NP-hardness of MSSC for $k = 2$ in general dimension s was given by Drineas et al. in *Machine Learning* 56, 9–33, 2004. As shown in the next section the proof is, however, invalid. An alternate short proof, due to A. Deshpande and P. Popat [20], is given in Section 1.3. Note that another longer proof was obtained independently, and almost at the same time, by Dasgupta [18]. Moreover, a proof which is essentially the same as that of [20] was obtained independently and more recently by Kanade, Nimbhorkar and Varadarajan [65].

1.2 An incorrect reduction from the k-section problem

Drineas et al. [27] propose a NP-hardness proof for the MSSC with $k = 2$ and general dimension by a reduction from the minimum bisection problem, whose objective is to partition a graph into two equal-sized parts so as to minimize the number of edges going between the two parts. The authors state that a proof for $k > 2$ is similar via a reduction to the minimum k-section problem. The paper is cited in [4, 8, 16, 91] as giving a proof that MSSC is NP-hard.

The polynomial transformation for performing the reduction from the bisection problem is described as follows:

> "Let $G = (V, E)$ be the given graph with n vertices 1,...,n, with n even. Let $d(i)$ be the degree of the i'th vertex. We will map each vertex of the graph to a point with $|E| + |V|$ coordinates. There will be one coordinate for each edge and one coordinate for each vertex. The vector X^i for a vertex i is defined as $X^i(e) = 1$ if e is adjacent to i and 0 if e is not adjacent to i; in addition $X^i(i) = M$ and $X^i(j) = 0$ for all $j \neq i$."

Figure 1.1 illustrates an example of such a transformation for a given graph. It can be checked in the example that all partitions with non-empty clusters have the same cost value regarding the last $|V|$ coordinates. Correcting an error in the proof presented in [27], we will show that this is always true for any MSSC instance constructed by the proposed transformation.

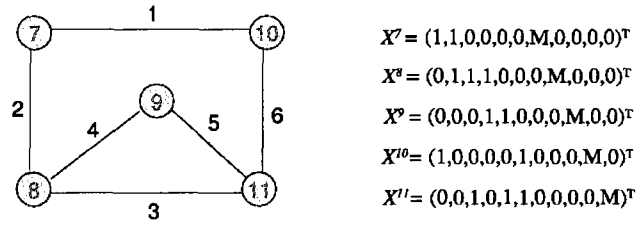

Figure 1.1: Transformation of a graph into an MSSC instance as defined in [27]

Let us consider a bipartition of the entities into two clusters P and Q whose cardinalities are denoted by p and q, respectively. Regarding the last $|V|$ coordinates of the centroids $z^P, z^Q \in \mathbb{R}^{|E|+|V|}$, we have for $i = 1, \ldots, |V|$

$$z^P_{|E|+i} = \begin{cases} \frac{M}{p} & : \text{ if } i \in P \\ 0 & : \text{ otherwise} \end{cases} \qquad z^Q_{|E|+i} = \begin{cases} \frac{M}{q} & : \text{ if } i \in Q \\ 0 & : \text{ otherwise} \end{cases}$$

Therefore, the sum of squared distances of each entity to its centroid, limited to the last $|V|$ coordinates, is equal to

$$\begin{aligned} & p\left(M - \tfrac{M}{p}\right)^2 + q\left(M - \tfrac{M}{q}\right)^2 + \mathbf{p(p-1)}\left(\mathbf{0} - \tfrac{\mathbf{M}}{\mathbf{p}}\right)^2 + \mathbf{q(q-1)}\left(\mathbf{0} - \tfrac{\mathbf{M}}{\mathbf{q}}\right)^2 \\ & = nM^2 - 4M^2 + M^2\left(\tfrac{1}{p} + \tfrac{1}{q}\right) + 2M^2 - M^2\left(\tfrac{1}{p} + \tfrac{1}{q}\right) \\ & = (n-2)M^2. \end{aligned}$$

In Drineas et al. [27], the authors forget to add the squared distances of the null components to the centroids, which are indicated in boldface in the expression. If they are not taken into consideration, then the sum-of-squares limited to the last $|V|$ coordinates is equal to

$$nM^2 + M^2\left(\frac{1}{p} + \frac{1}{q}\right) - 4M^2,$$

which is minimized whenever $p = q = n/2$. Thus, if M is made sufficiently large, balanced bipartitions have costs strictly smaller than unbalanced ones, since the contribution for the cost limited to the first $|E|$ coordinates is upper bounded. In fact, for $p = q$, this last value is minimized when the solution of MSSC is the balanced bipartition that corresponds to the minimum bisection in the original graph (see Drineas et al. [27], page 16). Unfortunately, after correcting the expression of the cost regarding the last $|V|$ coordinates, there is no dependence on the cardinalities of the clusters. This implies that the proposed reduction from minimum bisection is invalid.

1.3 A new proof by reduction from the densest cut problem

Nevertheless, there is a similar (valid) reduction that shows that the problem is in fact NP-hard.

Theorem 1.1. *MSSC in general dimension is NP-hard for $k = 2$*

Proof. The reduction is from the densest cut problem, whose objective is to maximize for a given graph $G = (V, E)$ the ratio $|E(P, Q)|/|P| \cdot |Q|$ over all bipartitions (P, Q) of the vertices in G, where $E(P, Q)$ denotes the edge set of the cut. The problem is equivalent to the sparsest cut problem on the complement graph, which was shown to be NP-hard in [82].

Given a graph G with no parallel edges, let us define a $|V|$ by $|E|$ matrix M as follows. An entry (v, e) in M is equal to 0, if edge $e \in E$ is not incident to vertex $v \in V$. Otherwise, it is $+1$ for one endpoint of e and -1 for the other. It does not matter which endpoint corresponds to $+1$ and which to -1. Thus, each column of M has exactly one entry equal to $+1$ and exactly one entry equal to -1.

Now, let us suppose that the rows of M are points in $\mathbb{R}^{|E|}$ and compute the value of the MSSC criterion for a bipartition into two clusters P and Q, with $|P| = p$, $|Q| = q$ and $p+q = n$. The centroid of cluster P has in its e-th coordinate a value equal to either $+1/p$ or $-1/p$ if $e \in E(P, Q)$, or 0 otherwise. The same holds for the coordinates of the centroid

of cluster Q. Then, by computing the total cost of the bipartition, we have that

$$\begin{aligned}
&\sum_{e\in E} \text{cost of } P \text{ due to the } e\text{-th coordinate} + \text{cost of } Q \text{ due to the } e\text{-th coordinate} \\
&= \sum_{e\in E(P,Q)} (p-1)\frac{1}{p^2} + \left(1-\frac{1}{p}\right)^2 + (q-1)\frac{1}{q^2} + \left(1-\frac{1}{q}\right)^2 + \sum_{e\notin E(P,Q)} 2 \\
&= \left(2-\frac{1}{p}-\frac{1}{q}\right)|E(P,Q)| + 2|E(P,P)| + 2|E(Q,Q)| \\
&= 2|E| - \frac{n}{p\cdot q}|E(P,Q)|,
\end{aligned}$$

by using $p+q=n$. The MSSC for $k=2$ minimizes the above, which means that it maximizes $|E(P,Q)|/p\cdot q$ and hence finds the densest cut in the given graph G. □

CHAPTER 2 : EVALUATING A BRANCH-AND-BOUND RLT-BASED ALGORITHM FOR MINIMUM SUM-OF-SQUARES CLUSTERING

Recently, Sherali and Desai [108] proposed an exact branch-and-bound RLT-based algorithm for minimum sum-of-squares clustering based on a model obtained with the reformulation-linearization technique (RLT) [107]. These authors reported results for large instances with up to 1000 points and dimension 8. However, some details in that paper deserve further investigation. In particular, reported values of the ratio gaps between the lower and upper bounds (obtained by branch-and-bound with the RLT model and by a heuristic, respectively) appear to be large, while the number of branch-and-bound nodes in the branch-and-bound tree are strikingly moderate. Moreover, the authors solve a small example for which the well established k-means algorithm gives a result with a value twice larger than that one obtained by the branch-and-bound algorithm.

2.1 Reformulation-Linearization Technique for the MSSC

The RLT method can be used to transform a zero-one mixed-integer quadratic program into an equivalent zero-one mixed-integer linear programming problem. In this approach, a tight linear programming relaxation, with an outer-approximation to the convex envelope of the objective function over the constrained region, is constructed for the problem by generating new constraints through the process of employing suitable products of constraints and using variable redefinitions. The RLT technique applied to the MSSC together with other valid inequalities, as in [108], will next be recalled.

First of all, we remark that the objective function of (1) can be manipulated. From the KKT conditions expressed in (2), the objective function can be rewritten as

$$\begin{aligned}
& \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{r=1}^{s} x_{ij}(y_j^r - p_i^r)^2 \\
= & \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{r=1}^{s} x_{ij}(y_j^r - p_i^r)y_j^r - \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{r=1}^{s} x_{ij}(y_j^r - p_i^r)p_i^r \\
= & \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{r=1}^{s} x_{ij}(p_i^r)^2 - \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{r=1}^{s} p_i^r x_{ij} y_j^r.
\end{aligned}$$

Since $\sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{r=1}^{s} x_{ij}(p_i^r)^2 = \sum_{i=1}^{n}\sum_{r=1}^{s}(p_i^r)^2$ is a constant, the MSSC is equivalent to:

$$\max_{x,y} \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{r=1}^{s} p_i^r x_{ij} y_j^r$$

subject to

$$\begin{aligned}
& \sum_{i=1}^{n} x_{ij}(y_j^r - p_i^r) = 0 && \forall j = 1,\ldots,k, \forall r = 1,\ldots,s \qquad (2.1)\\
& \sum_{j=1}^{k} x_{ij} = 1, && \forall i = 1,\ldots,n \\
& x_{ij} \in \{0,1\} && \forall i = 1,\ldots,n, \forall j = 1,\ldots,k.
\end{aligned}$$

From equation (2), we notice that for a given solution x, y_j is a convex combination (with equal weights) of all the points p_i, such that $x_{ij} = 1$. Therefore, constraints can be added stating that $y_j, \forall j = 1,\ldots,k$ must be in the convex hull of all the points p_i that can be associated to cluster j, denoted I_j.

For a two-dimensional space, the convex hull can be polynomially calculated in $O(n \log n)$ by Graham's algorithm [46]. However, for a higher dimension, it is generally an expensive task. In [108], the authors use instead a hyperrectangle $\overline{H}(I_j)$ that includes the convex

hull of the allowed points I_j, for each $j = 1, \ldots, k$.

$$\overline{H}(I_j) = \{y_j : \alpha_j^r \leq y_j^r \leq \beta_j^r, r = 1, \ldots, s\},$$

where, $\alpha_j^r = \min\{p_i^r : i \in I_j\}$ and $\beta_j^r = \max\{p_i^r : i \in I_j\}$, $\forall r = 1, \ldots, s$.

Instead of simply imposing these constraints, the product of each of them with both x_{ij} and $(1 - x_{ij})$ $\forall i \in I_j$ is considered for each $j = 1, \ldots, k$, following the RLT method. The resulting constraints are:

$$\begin{aligned}
&\alpha_j^r x_{ij} \leq y_j^r x_{ij} \leq \beta_j^r x_{ij} \quad \forall i \in I_j \quad \forall j, r, \\
&\alpha_j^r (1 - x_{ij}) \leq y_j^r (1 - x_{ij}) \leq \beta_j^r (1 - x_{ij}) \quad \forall i \in I_j \quad \forall j, r.
\end{aligned}$$

Additional constraints can still be included in order to tighten the mathematical formulation of the model. Since $n > k$, we have that

$$1 \leq \sum_{i=1}^{n} x_{ij} \leq (n - k + 1), \quad \forall j = 1, \ldots, k.$$

They assert that at least one point must be assigned to each cluster, and therefore, each cluster contains at most $n - k + 1$ points assigned to it. The reformulation of these constraints leads to

$$y_j^r \leq \sum_{i=1}^{n} x_{ij} y_j^r \leq (n - k + 1) y_j^r \quad \forall j, r.$$

Although not mentioned in [108], these last constraints are valid only if $y_j^r \geq 0$, $\forall j, r$. Thus, the data sets used in the computational experiments must eventually be translated in order to satisfy these conditions.

2.1.1 Dealing with symmetry

Symmetry in the problem structure can make difficult the resolution via a branch-and-bound approach. For any given solution to the MSSC, alternative equivalent solutions could be obtained by simply re-indexing clusters. This matter was previously studied

by Klein and Aronson in [68] where they propose the use of some valid inequalities in order to reduce the effects of symmetry. In [108], Sherali and Desai propose two different strategies to that purpose, though recognizing that symmetry in the problem structure is not completely eliminated by them.

The first strategy imposes the following constraints:

$$x_{11} = 1, x_{1j} = 0, \quad \forall j = 2, \ldots, k$$
$$\sum_{i=1}^{n} x_{ij} \geq \sum_{i=1}^{n} x_{i,j+1}, \quad \forall j = 2, \ldots, k-1.$$

It means that point p_1 is assigned to the first cluster. Regarding the other clusters, indexing is required to be performed in nonincreasing order of their size. However, symmetry still occurs if there is a solution having different clusters with the same size.

In the second strategy, a dispersed set of points $P = \{p_{i_1}, p_{i_2}, \ldots, p_{i_{k-1}}\}$ is built from the complete set of points to be clustered. First, a point p_{i_1} is arbitrarily chosen and inserted in P. Then, among the points outside of P, we select a point p_{i_2} whose distance to p_{i_1} is maximum. After that, each new point to be included in P is selected if its minimum distance to a point in P is maximum among those belonging to the complement of P.

Then, each point p_{i_h}, $h = 1, \ldots, k-1$ in P is restricted to belong to one of the first h ($< k$) clusters. Since the points in P are dispersed there is a good chance that they actually belong to different clusters. For the instances where this happens, the strategy actually eliminates symmetry. However, Figure 2.1 shows a simple case for which the symmetry effect still remains. Indeed should both points 1 and 2 be assigned to cluster 1, these constraints allow point 3 to belong to cluster 2 or to cluster 3. As reported in [108], the second strategy is preferred over the first in the sense that, if it is used, the algorithm has better initial gaps and finds optimal solutions more quickly.

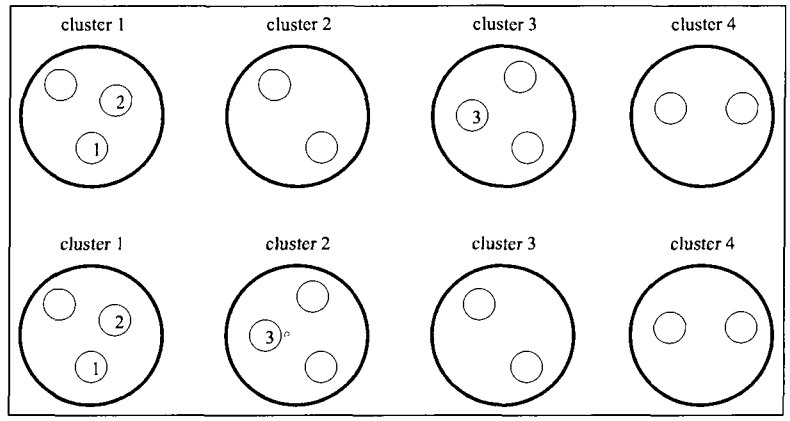

Figure 2.1: Two symmetric solutions allowed by the second strategy

The final MIP model is obtained after linearizing the products $x_{ij}y_j^r$ by z_{ij}^r, $\forall i, j, r$, following the RLT approach. The model, as given in [108], is:

$$\max_{x,y,z} \sum_{i \in I_j} \sum_{j \in J_i} \sum_{r=1}^{s} p_i^r z_{ij}^r$$

subject to

$$\sum_{i \in I_j} z_{ij}^r - \sum_{i \in I_j} p_i^r x_{ij} = 0 \qquad \forall j, r$$

$$\alpha_j^r x_{ij} \leq z_{ij}^r \leq \beta_j^r x_{ij} \qquad \forall i \in I_j \;\; \forall j, r$$

$$\alpha_j^r (1 - x_{ij}) \leq y_j^r - z_{ij}^r \leq \beta_j^r (1 - x_{ij}) \qquad \forall i \in I_j \;\; \forall j, r$$

$$\sum_{j \in J_i} x_{ij} = 1 \qquad \forall i$$

$$\sum_{i \in I_j} x_{ij} \geq 1 \qquad \forall j$$

$$y_j^r \leq \sum_{i \in I_j} z_{ij}^r \leq (n - k + 1) y_j^r \qquad \forall j, k$$

symmetry breaking strategy 1 or 2

$$x \in X$$

where,

$$X = \{x \text{ binary} : x_{ij} = 0 \text{ for all } (i,j) \in I^-, x_{ij} = 1 \text{ for all } (i,j) \in I^+\}$$

with $I^+ = \{(i,j) : x_{ij}$ has been fixed at 1$\}$, $I^- = \{(i,j) : x_{ij}$ has been fixed at 0$\}$, and $I^f = \{(i,j) : x_{ij}$ is free$\}$. Hence, the sets I_j for each $j = 1, \ldots, k$ are given by

$$I_j = \{i \in \{1, \ldots, n\} : (i,j) \in I^+ \cup I^f\}.$$

Moreover, for each $i \in \{1, \ldots, n\}$, we define:

$$J_i \equiv \{j \in \{1, \ldots, k\} : (i,j) \in I^+ \cup I^f\}.$$

The RLT model is valid since for any feasible solution of the RLT model, we have that $z_{ij}^r = x_{ij} y_j^r$ holds true. Hence, the RLT model is an equivalent linear 0-1 mixed integer programming (MIP) representation of the original formulation of the MSSC [108].

2.2 Branch-and-bound for the MSSC

When developing a standard branch-and-bound method [123], three elements are essential: upper bounds obtained by means of a (usually) linear relaxation, lower bound solutions, and a branching rule.

Accordingly to the RLT theory, the resultant model is supposed to obtain tight upper bounds. For the MSSC, upper bounds are computed by using the LP relaxation of the MIP model, taking into account the current definitions of the sets I_j and J_i, for each $i = 1, \ldots, n$ and $j = 1, \ldots, k$ at each branch-and-bound node of the tree. Lower bounds $\underline{x}$ on variables x can be obtained by a rounding heuristic applied to this LP solution. This heuristic rounds the relaxed solution $\overline{x}$ to the nearest binary one subject to $\sum_{j \in J_i} x_{ij} = 1, \forall i = 1, \ldots, n$. For example, suppose that we have a problem with $n = 3$ and $k = 2$ for which $\overline{x}_{11} = 0.4$, $\overline{x}_{12} = 0.6$, $\overline{x}_{21} = 0.2$, $\overline{x}_{22} = 0.8$, $\overline{x}_{31} = 1$, and $\overline{x}_{32} = 0$. Then, the heuristic will provide as lower bound $\underline{x}_{11} = 0$, $\underline{x}_{12} = 1$, $\underline{x}_{21} = 0$, $\underline{x}_{22} = 1$, $\underline{x}_{31} = 1$, and $\underline{x}_{32} = 0$. If the solution $\overline{x}$ is already binary, the LP solution is optimal for the subproblem.

The LP solution of a branch-and-bound node can also be used to generate a valid inequality involving the incumbent best lower bound. This is done by using the dual values of the LP solution and surrogating all constraints in x-variables, except for constraints $0 \leq x_{ij} \leq 1$, $\forall(i,j)$. Then, standard 0-1 logical tests are performed on this inequality to possibly fix additional x-variables.

Exploiting the structure of the inherent generalized upper bounding (GUB) constraints, the authors of [108] explore an alternative specially ordered set (SOS) branching strategy. For this purpose, θ_{ij} is defined as the total absolute discrepancy in the linearized objective terms (z^r_{ij}) relative to the original nonlinear product terms ($\overline{x}_{ij} \cdot \overline{y}^r_j$). This is expressed as

$$\theta_{ij} = \sum_{r=1}^{s} |p^r_i (z^r_{ij} - \overline{x}_{ij}\overline{y}^r_j).|$$

Then, values $\theta_i \equiv \sum_{j \in J_i} \theta_{ij}$ are defined for each $i = 1, \ldots, n$, computing $u \in \text{argmax}\{\theta_i\}$. The branching rule is to partition the set J_u into two children nonempty sets, J_{u_1} and J_{u_2}, as follows. Two subproblem nodes are constructed for the branch-and-bound tree corresponding to the respective imposed branching restrictions $\sum_{j \in J_{u_1}} x_{ij} = 1$ and $\sum_{j \in J_{u_2}} x_{ij} = 1$. In order to obtain the partitions J_{u_1} and J_{u_2}, the values θ_{uj}, with $j \in J_u$, are sorted in nonincreasing order $\{\theta_{uj_1}, \theta_{uj_2}, \ldots, \theta_{uj_l}\}$, where $l = |J_u| \geq 2$. Then, a value $\gamma \geq 1$ is defined to be the smallest integer such that $\sum_{c=1}^{\gamma} \theta_{uj_c} \geq \theta_u/2$. Finally, $J_{u_1} = \{j_1, \ldots, j_\gamma\}$ and $J_{u_2} = \{j_{\gamma+1}, \ldots, j_l\}$. Finally, for each of the children nodes, the sets I^+, I^- and I^f are updated, and new convex hull constraints for each cluster are used to strengthen the formulation.

The branch-and-bound algorithm proposed in [108] adopted a depth-first strategy to develop the enumeration tree. The algorithm was implemented in C++, and the commercial software CPLEX 8.1.0 was invoked for the purpose of solving the LP relaxations at each node. Besides, the optimal basis of the parent nodes were used as an advanced-start basis for the children nodes.

2.3 An attempt at reproducing computational results

In order to verify and validate the computational experiments reported in [108] via the branch-and-bound described in the previous section, we followed the same algorithmic development steps described in that paper. Except for the platform used, a Pentium IV 2 GHz with 512 MB RAM under Linux in our experiments, the implementations are supposed to be equivalent. Our codes were compiled with g++ (option -O3) version 3.4.4, and are available at `http://www.gerad.ca/~aloise/publications.html`. For the computational experiments reported in this section, the second symmetry breaking strategy presented in section 2.1.1 is used in the resolution of the RLT-based MIP model.

We start by the study of a small illustrative example with three clusters used in the referred paper. Table 2.1 provides this data set (which is a subset of 22 points corresponding to coordinates of German towns given in [111]) where the points $p_1, \ldots, p_{10}$ are given by their coordinates in Euclidean space.

Table 2.1: Coordinates of 10 points in the Euclidean space

Coordinates	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
x	-57	54	46	8	-36	-22	34	74	-6	21
y	28	-65	79	111	52	-76	129	6	-41	45

The authors report that 27 branch-and-bound nodes were created before the algorithm could reach the optimal solution and prove its optimality. The initial gap ratio was $UB_0/LB_0 = 1.9236$. However, the results obtained by our implementation are different. To solve this problem, our branch-and-bound algorithm required 848 nodes beginning with an initial gap ratio of 1.23. Recall that the authors must have changed the data set in order to validate their model, though such transformation was not described in the paper. We performed translations in order to obtain non-negative data. Another transformation, suggested by Sherali [105], which could accelerate the algorithm would be to rotate the axes in order to find the smallest volume hyperrectangle.

Comparing with k-means, the authors report that the ratio between the best solution obtained by five executions of the heuristic from randomly generated initial solutions

(=34404.85) and the optimal solution obtained by the branch-and-bound (=15805.25) is 2.17. Moreover, calculating the same ratio with the first feasible solution LB_0 (=22434.68) found at the root node instead of the optimal solution, a value of 1.86 is obtained.

Computational experiments were carried out with a standard k-means heuristic implementation without any enhanced feature. Five different executions were performed 100 times. The worst of the best solutions obtained by each group of five executions, denoted here $Z_{k-means}$, was found to be 20743.133, providing a ratio of 1.312 between the $Z_{k-means}$ and the optimal solution. This ratio is $0.925 < 1$ with respect to LB_0, while regarding the first lower bound value LB'_0 obtained by our implementation, this same ratio is equal to 0.598. In fact, the overall worst solution obtained for the 500 executions of k-means was 32611.6, the optimal solution was found 67 times ($\approx 13\%$), and the average value was 19083.444. Table 2.2 presents the ratios for two benchmark data sets extracted from [111] which contains cartesian coordinates for 22 and 59 German towns, and for the classical Fisher's Iris [36] with 150 points in four dimensions. We can notice that k-means always obtained better solutions than the first lower bound value obtained by our RLT-based branch-and-bound implementation.

Table 2.2: Relative results of our branch-and-bound implementation versus the k-means algorithm in three benchmark data sets with three and five cluster centers. Results reported in [108] regarding the best solution among five executions of k-means are shown in parenthesis

Data set	Points	k	$Z_{k-means}$/optimal	$Z_{k-means}/LB'_0$
German22	22	3	1.05 (3.41)	0.33
	22	5	1.32 (1.80)	0.26
German59	59	3	1.16 (7.66)	0.46
	59	5	1.28 (4.57)	0.25
Fisher150	150	3	1.00	0.50
	150	5	1.23	0.14

A further investigation also revealed that the solution provided in the paper, i.e. that one obtained by k-means for the small example with 10 points (page 294 of [108]), is not a local optimum. Figure 2.2 presents with dashed lines the composition of the clusters indicated by the authors as the solution provided by the k-means heuristic, while bold lines illustrate the actual local optimum if the algorithm proceeds.

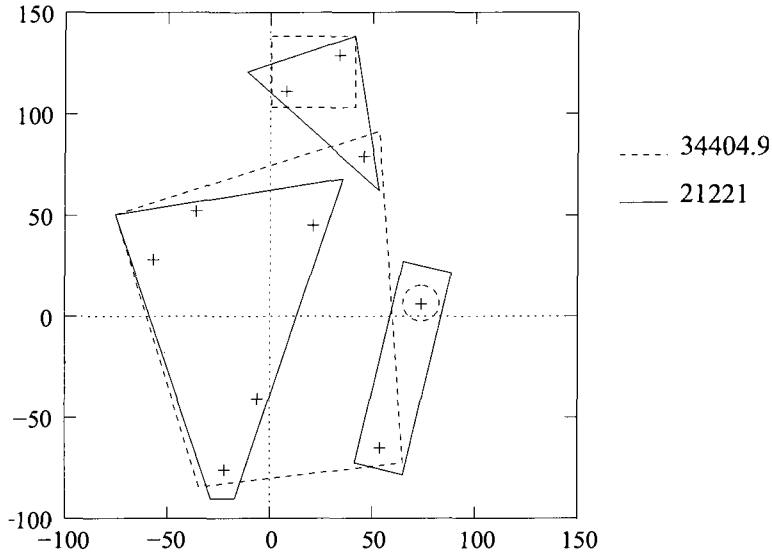

Figure 2.2: k-means solution as provided by Sherali and Desai (2005) and k-means actual local optimum. The legends indicate the cost of the two solutions

Our branch-and-bound implementation was also tested for other data sets. The initial data set with 10 points was progressively increased with additional points from the 22 German towns data set of [111], until it was no more possible to solve the generated instance within 10 hours. The same procedure was done to generate data sets from Fisher's [36] 150 Iris. Table 2.3 shows for each data set the number of nodes as well as the gap between the initial upper bound value (UB'_0) and lower bound value (LB'_0) at the root node of the enumeration, which is calculated as $\frac{UB'_0 - LB'_0}{UB'_0}$. Comparing with some results presented in [108] for larger instances with 250 and 500 nodes, it is remarkable that a much higher amount of branch-and-bound nodes are generated by our algorithm for much smaller data sets.

Finally, computational tests were performed to test the efficacy of solving the RLT model directly by the commercial software CPLEX. While in [108] the authors claim that their implementation solves the data sets faster than CPLEX 8.1 with default settings, Table 2.4 presents opposing results obtained by our implementation, when CPU times in seconds are compared with those spent by CPLEX 8.1 and CPLEX 10.1.

Table 2.3: Results on data sets generated from 22 German towns coordinates presented in [111] and from the Fisher's 150 Iris presented in [36]

Benchmark	Points	k	# solved nodes	root gap(%)
German22	10	3	848	18.9
	12	3	2630	14.8
	14	3	5758	25.3
	16	3	15922	24.2
	18	3	30946	24.5
	20	3	310308	25.0
	10	5	1202	13.0
	12	5	26430	11.3
	14	5	63740	21.3
Fisher150	10	3	196	0.2
	12	3	1580	0.3
	14	3	9684	0.5
	16	3	19082	0.6
	18	3	67618	0.6
	20	3	340702	0.7
	10	5	1208	0.1
	12	5	13716	0.2
	14	5	146552	0.3

Table 2.4: CPU times in seconds obtained by our implementation and by the commercial softwares CPLEX 8.1 and CPLEX 10.1 with default settings

Benchmark	Points	k	Branch-and-bound	CPLEX 8.1	CPLEX 10.1
German22	10	3	6.11	0.47	0.44
	12	3	24.26	0.81	0.84
	14	3	65.12	3.28	2.91
	16	3	225.87	9.28	7.34
	18	3	562.51	23.69	15.12
	20	3	14097.68	98.08	78.88
	10	5	12.62	2.26	2.61
	12	5	355.80	13.33	13.03
	14	5	1223.08	81.19	69.71
Fisher150	10	3	3.06	0.41	0.52
	12	3	29.40	1.66	1.67
	14	3	219.68	6.72	6.64
	16	3	527.04	17.56	14.08
	18	3	2435.81	52.78	34.04
	20	3	21909.95	150.14	85.79
	10	5	34.86	1.25	1.49
	12	5	499.23	25.36	31.64
	14	5	8267.44	440.58	322.94

We also noticed that while in [108] CPLEX 8.1 was able to solve directly a model with $n = 250$, $k = 3$ and dimension 4 in 263.44 seconds, and another with the same number of points and clusters but with dimension 6 in 360.83 seconds, for a small data set with 20 points, the solver already takes 150.14 seconds according to our experiments. It is important to remark that the hardness of a MSSC instance is not directly measured by the values of n, k, and s. It also depends on the distribution of points. To illustrate, consider an example of MSSC with n entities divided into k clusters all points of which are each time within a unit diameter ball in $\mathbb{R}^s$. Assume these balls are pairwise at least $n+1$ units apart. Then the simplest branch-and-bound algorithm will quickly find this partition and confirm its optimality without branching as any misclassification more than doubles the objective function value. Indeed, the contribution to the bound of the cluster containing both misclassified points is then at least $\frac{n^2}{n-k} > n$, while the optimal solution has a value less than n. Note that n, k and s can be arbitrarily large.

2.4 Breaking symmetry and convex hull inequalities

In this section, the effects of a symmetry breaking rule due to [100] are investigated. In that paper, the permutation symmetry is broken by accepting only lexicographically minimal solutions, i.e., each cluster j contains the lowest indexed point which does not belong to any of the previous clusters $1, \ldots, j-1$. Following this property, there is only one way to index the clusters, and therefore, symmetry is broken. To illustrate, among all the equivalent numbering of a clustering of the set $P = \{a, b, c, d\}$ into the three sets $\{a\}, \{b, d\}$ and $\{c\}$, i.e.,

Cluster 1	Cluster 2	Cluster 3
{b,d}	{a}	{c}
{b,d}	{c}	{a}
{a}	{b,d}	{c}
{a}	{c}	{b,d}
{c}	{b,d}	{a}
{c}	{a}	{b,d},

only the third one is accepted.

Below, the steps leading to the full formulation of the desired property as a set of linear inequality constraints are presented.

The first point p_1 must be assigned to the first cluster, which means that

$$x_{11} = 1.$$

For the second one, if point p_2 is not in cluster 1, then it must be in cluster 2.

$$(1 - x_{22}) \leq x_{21}.$$

Generalizing, if points $p_2, \ldots, p_{c-1}$ are in cluster 1 and the point p_c is not ($c \geq 3$) then p_c must be assigned to cluster 2.

$$(1 - x_{c2}) \leq \sum_{i=2}^{c-1} (1 - x_{i1}) + x_{c1}.$$

For cluster $j = 3, \ldots, k-1$, the smallest indexed point which is in none of the clusters $1, \ldots, j-1$ is forced to belong to cluster j. In other words, if for all $i = 2, \ldots, c-1$, point p_i belongs to some cluster $\ell < j$ and point p_c does not, then p_c must be in cluster j. For each cluster j, this is expressed by the constraints

$$(1 - x_{cj}) \leq \sum_{i=2}^{c-1} \left(1 - \sum_{\ell=1}^{j-1} x_{i\ell} \right) + \sum_{\ell=1}^{j-1} x_{c\ell} \quad \forall c = j, \ldots, n.$$

Note that no variable x_{ij} is needed for $i < j$, since the lexicographic order guarantees that no point is ever assigned to a cluster with a higher number than the point itself. Moreover, the last cluster k does not need to be considered because it will automatically contain all remaining points that do not belong to any of the lower indexed clusters. Therefore, the symmetry breaking rule proposed by [100] requires $O(kn)$ additional constraints to any MSSC instance.

As we can notice, the constraints are sensitive to the initial indexing of the points. In order to analyze the effect of this indexing, we tested the symmetry breaking rule with three different types of ordering for the points.

In the first ordering, denoted Symm3_F, the points are selected in the same way as described in section 2.1.1 for the second strategy devised by [108]. The second ordering, called Symm3_R, uses the ordering that comes in the data set, which can be considered as a random one. The third one, is similar to the first, but instead of selecting points whose minimum distance to a point already selected is maximum, a point is selected if its *maximum* distance to a point already selected is *minimum* among those yet to be selected. This last ordering is denoted Symm3_C.

The strategies conceived to deal with symmetry were computationally tested for some of the data sets described above, the results being reported in Table 2.5. The labels Symm1 and Symm2 refer, respectively, to the first and second strategies devised by [108].

These computational experiments confirm that the second strategy Symm2 has a better performance than Symm1. Regarding the symmetry breaking rule based on [100], the best results were obtained by the versions that ordered the points by dispersion (Symm3_F) and randomly (Symm3_R), except for the data set with 14 points generated from the Fisher's 150 Iris. The reasons for such difference in performance caused by the different orderings must be further investigated. Also, Symm3_F always presented a better performance than Symm2 for the tested data sets, though the initial upper bounds have the same value. In fact, if the points are ordered by dispersion as explained above, the effect at the root node with the two strategies tends to be similar, since the affected points, i.e. $p_{i_h}, h = 1, \ldots, k-1$, are really supposed to belong to different clusters. If this happens, both strategies imply the same cluster's indexing. Otherwise, algorithm Symm3_F is preferable since its associated breaking rule strategy is more restrictive than that used by algorithm Symm2. Table 2.6 shows initial upper bounds obtained by these algorithms for different values of k. As the value of k increases, points $p_{i_h}, h = 1, \ldots, k-1$ are less likely to belong each one to a different cluster.

Table 2.5: Results obtained for the different symmetry breaking strategies for data sets generated from the 22 German towns coordinates and from the Fisher's 150 Iris. The CPU times are given in seconds. The symbol "-" is used to represent that the algorithm was not able to solve the instance within 10 hours of computing time

Benchmark	Points	k	Symm1			Symm2		
			CPU time	# of nodes	UB'_0	CPU time	# of nodes	UB'_0
German22	10	3	7.86	1038	258956	6.11	848	257795
	12	3	36.67	3754	345830	24.26	2630	343502
	14	3	93.83	7648	379776	65.12	5758	378659
	16	3	293.40	18952	449054	225.87	15922	445628
	18	3	655.74	36616	487644	562.51	30946	484159
	20	3	37519.50	503362	538440	14097.68	310308	535152
Fisher150	10	5	140.25	4118	369.69	34.85	1208	369.25
	12	5	4568.11	98780	452.15	499.23	13716	451.78
	14	5	-	-	515.19	8267.44	146552	514.86

Benchmark	Points	k	Symm3_F			Symm3_R			Symm3_C		
			CPU time	# of nodes	UB'_0	CPU time	# of nodes	UB'_0	CPU time	# of nodes	UB'_0
German22	10	3	5.57	736	257795	4.53	626	257795	5.49	756	258377
	12	3	21.00	2194	343502	19.09	2080	344838	36.53	4060	345547
	14	3	43.53	3866	378659	50.64	4544	378649	136.77	12090	379475
	16	3	210.80	15022	445628	188.66	13664	447320	540.73	35066	448889
	18	3	498.27	29490	484159	608.02	34752	486002	3441.03	133534	487643
	20	3	12730.51	289570	535152	16085.13	331230	536886	30504.15	463090	538440
Fisher150	10	5	21.06	674	369.25	37.25	1128	369.68	40.78	1268	369.42
	12	5	365.14	9638	451.78	639.07	15856	452.14	434.67	10028	451.88
	14	5	5375.03	98818	514.86	8525.15	138502	515.18	4032.01	68982	514.86

Table 2.6: Initial upper bounds UB'_0 obtained by algorithms Symm2 and Symm3_F for different number of clusters

Benchmark	Points	k	Symm2	Symm3_F
German22	20	5	535057	535055
		7	534538	534536
		9	533511	533493
		11	532010	531949
		13	530139	530025
Fisher150	14	7	814.85	814.84
		9	514.75	514.74
		11	514.62	514.60
		13	514.42	514.38

We also decided to investigate the effect of the actual convex hull constraints in the MIP model of section 2.1.1 for data sets in $\mathbb{R}^2$. Recall that constraints for the coordinates of each centroid y_j^r, for each $j = 1, \ldots, k$ and $r = 1, \ldots, s$, were introduced by means of hyperrectangles that covered all the entities in the sets I_j.

Since each pair of consecutive extreme points in $conv(P)$ can define a halfspace in the Euclidean space, the coordinates of the centroids are confined to be in the polyhedron defined by the intersection of these halfspaces. Unfortunately, the number of constraints in the output is sensitive to the number of extreme points provided by Graham's algorithm [46]. While $O(kn)$ constraints are necessary in the model when using the hyperrectangles, this number raises to $O(kn^2)$ with the convex hull inequalities, since all the entities can be extreme points of the convex hull. Tables 2.7 and 2.8 present comparative results in the same data sets generated from the 22 German towns coordinates of [111] for a new implementation which includes the convex hull inequalities just described. The resulting algorithm (Symm3_F+CH) uses the symmetry breaking rule of [100] with the dispersion ordering, and is compared with the implementation of the previous section (Symm3_F).

There is a clear reduction provided by the convex hull inequalities in the number of branch-and-bound nodes solved by the RLT-based branch-and-bound, with average reduction of approximately 79%. This is partially justified by the initial upper bounds obtained at the root of the enumeration tree which are also smaller relatively to the previous implementation. Moreover, since the convex hull constraints are updated at each branch-and-

Table 2.7: Comparison in terms of the number of branch-and-bound nodes solved by two implementations with (Symm3_F+CH) and without (Symm3_F) the convex hull inequalities

Benchmark	Points	k	Symm_3F	Symm_3F+CH	Reduction(%)
German22	10	3	736	161	78.1
	12	3	2194	593	72.9
	14	3	3866	995	74.2
	16	3	15022	4473	70.2
	18	3	29490	4251	85.5
	10	5	792	149	81.1
	12	5	14834	1765	88.1
	14	5	32672	4507	86.2

Table 2.8: Comparison in terms of initial upper bounds obtained by two implementations with (Symm3_F+CH) and without (Symm3_F) the convex hull inequalities

Benchmark	Points	k	Symm_3F	Symm_3F+CH	Reduction(%)
German22	10	3	257795	248129	3.7
	12	3	343502	333573	2.9
	14	3	378659	368908	2.6
	16	3	445628	428689	3.8
	18	3	484159	460801	4.8
	10	5	257965	248698	3.5
	12	5	343711	333890	2.8
	14	5	378826	369307	2.5

bound node according to the current definition of the sets I_j, the upper bounds at each node are also supposed to be better, therefore reducing the number of branch-and-bound nodes evaluated.

However, a large reduction in the number of nodes does not imply the same effect for computing times as can be verified in Table 2.9. Indeed, this is due to the augmentation on the number of constraints which makes the resolution of the relaxed model more time consuming. Note that for the data set with 16 points CPU times are even worse with the implementation that uses the convex hull inequalities.

Table 2.9: Comparison in terms of CPU times in seconds used by two implementations with (Symm3_F+CH) and without (Symm3_F) the convex hull inequalities

Benchmark	Points	k	Symm_3F	Symm_3F+CH	Reduction(%)
German22	10	3	5.57	3.08	44.7
	12	3	21.00	16.53	21.2
	14	3	43.53	36.10	17.0
	16	3	210.80	332.26	negative
	18	3	498.27	428.92	13.9
	10	5	10.09	5.33	47.1
	12	5	267.85	121.22	54.7
	14	5	747.78	467.61	37.4

2.5 Concluding remarks

The first aim of this chapter was to reproduce results of Sherali and Desai [108] in solving the minimum sum-of-squares clustering problem with a branch-and-bound RLT-based algorithm. Following the guidelines presented by the authors in that paper, we tried our best to devise an equivalent branch-and-bound implementation. However, results obtained were drastically different even for small data sets, and inconsistencies were found with the results reported. Indeed, computing times and number of nodes for these small instances were larger than those reported in [108] for much larger problems. There can be several possible explanations for this:

(a) Our implementation is not correct;

(b) Our implementation is not efficient;

(c) The implementation of [108] is not correct;

(d) The data sets generated in the examples of [108] are extremely easy to solve;

(e) The platforms used are very different.

Regarding (a), we doubt this as the same results were obtained by our branch-and-bound implementation and by CPLEX for all instances. Regarding (b), while we followed [108] as closely as possible, this cannot be excluded a priori. Indeed computing times of our branch-and-bound implementation grow much quicker than those of CPLEX. However, the fact

remains that computing times of CPLEX are large even for small problems. Regarding (c), we note that in view of the fact that there are errors in the solution of the small example of [108] this again cannot be excluded a priori. Sherali and Desai [108] report similar computing times for CPLEX and the RLT-based branch-and-bound algorithm in contrast to our results. Regardless of correctness of the RLT-based branch-and-bound implementation of Sherali and Desai [108], the discrepancy between computing times with CPLEX remains to be explained. Regarding (d), we note that, as discussed above, it is very easy to generate arbitrarily large data sets in any number of dimensions for which the MSSC problem can be solved with a very small amount of branching. Regarding (e), we have been careful to use in our comparative experiments the same version of CPLEX as Sherali and Desai [108]. These authors do not mention the computer and compiler used. However, we believe that differences in computing platforms cannot explain the vast discrepancies observed.

To find out which is (or are) the true reasons among those listed we asked the authors of [108] to provide either a copy of their code or the data sets or a precise description of how they are generated (details are not given in their paper). This request could not be answered because, as mentioned by Sherali in recent email [106], "*Unfortunately, he* [Jitamitra Desai] *appears to have deleted his codes and data sets*".

The second aim of this chapter was to assess the interest of symmetry breaking rules, in particular that one of [100]. This last one completely breaks symmetry and appears to be better than the two ones of [108], both in terms of reduction in the number of nodes and of computing time.

The third aim of this chapter was to study the impact of adding valid inequalities obtained from the convex hull of the points which belong or can be added to each cluster. Tests were made in the case of two dimensions. It appears that the number of nodes of the branch-and-bound tree is reduced. The overall computing time may be either reduced or increased.

CHAPTER 3 : A BRANCH-AND-CUT SDP-BASED ALGORITHM FOR MINIMUM SUM-OF-SQUARES CLUSTERING

The hardest task while devising exact algorithms for MSSC is to compute good lower bounds in a reasonable amount of time. Recently, Peng and Xia [98] used matrix arguments to model MSSC as a so-called 0-1 semidefinite programming (0-1 SDP) which can be further relaxed to convex SDP or to linear programming. On the basis of their work, we propose in this chapter a branch-and-cut algorithm to efficiently exploit the tight lower bounds obtained from the linear relaxation of the underlying 0-1 SDP model.

3.1 Equivalence of MSSC to 0-1 SDP

In general, SDP refers to the problem of minimizing a linear function over the intersection of a polyhedron and the cone of symmetric and positive semidefinite matrices [120]. The canonical SDP has the following form:

$$(SDP)\begin{cases} \min\limits_{Z} & Tr(WZ) \\ s.t. & Tr(B_iZ) = b_i \quad \text{for } i = 1,\dots,m \\ & Z \succeq 0 \end{cases}$$

where W and B_i for $i = 1,\dots,m$ are matrices of coefficients, $Tr(\cdot)$ denotes the trace of a matrix, and $Z \succeq 0$ means that Z is positive semidefinite. If the latter is replaced by the constraint $Z^2 = Z$, then the following problem is obtained

$$(\text{0-1 } SDP)\begin{cases} \min\limits_{Z} & Tr(WZ) \\ s.t. & Tr(B_iZ) = b_i \qquad \text{for } i = 1,\dots,m \\ & Z^2 = Z, Z = Z^T \end{cases}$$

It is called 0-1 SDP due to the similarity of the constraint $Z^2 = Z$ to the obvious constraints on binary integer programming variables (see e.g. [10, 39]). Moreover, the eigenvalues of matrix Z are equal to 0 or 1.

From Huygens' theorem (see e.g. [30]), the MSSC objective function in (1) can be rewritten as

$$\sum_{i=1}^{n}\sum_{j=1}^{k} x_{ij}\|p_i - y_j\|^2 = \sum_{j=1}^{k} \frac{\sum_{i=1}^{n-1}\sum_{\ell=i+1}^{n} x_{ij}x_{\ell j}\|p_i - p_\ell\|^2}{|C_j|},$$

Then, by rearranging it, the MSSC cost function can be expressed by

$$\begin{aligned}\sum_{j=1}^{k} \frac{\sum_{i=1}^{n-1}\sum_{\ell=i+1}^{n}\|p_i - p_\ell\|^2 x_{ij}x_{\ell j}}{\sum_{i=1}^{n} x_{ij}} &= \sum_{i=1}^{n}\|p_i\|^2 - \sum_{j=1}^{k}\frac{\|\sum_{i=1}^{n} x_{ij}p_i\|^2}{\sum_{i=1}^{n} x_{ij}} \\ &= Tr(W_pW_p^T) - \sum_{j=1}^{k}\frac{\|\sum_{i=1}^{n} x_{ij}p_i\|^2}{\sum_{i=1}^{n} x_{ij}},\end{aligned}$$

where $W_p \in \mathbb{R}^{n\times s}$ is the matrix whose i-th row is the vector p_i. Note that the same matrix arguments were used by Zha et al. [130] and Steinley [114] in order to look for orthonormal matrices which optimize the second term of the expression.

In [98], maximization of the second term is shown to be equivalent to maximizing a 0-1 SDP problem. Their development starts by considering a feasible assignment matrix X, and then, defining a matrix $Z = X(X^TX)^{-1}X^T$. Note that Z is a matrix that satisfies $Z^2 = Z$ and $Z = Z^T$ with nonnegative elements.

Thus, the objective function can be rewritten as $Tr(W_pW_p^T(I - Z)) = Tr(W_pW_p^T) - Tr(W_pW_p^TZ)$. The constraint $\sum_{j=1}^{k} x_{ij} = 1$ can be written as $Xe^k = e^n$, which implies that

$$Ze^n = ZXe^k = Xe^k = e^n.$$

Moreover, the trace of Z is equal to k, the number of clusters, i.e.,

$$Tr(Z) = k.$$

Thus, the following 0-1 SDP model for the MSSC is obtained

$$\begin{aligned} &\min_{Z} \; Tr(W_p W_p^T (I - Z)) \\ &\text{subject to} \\ &Ze = e, Tr(Z) = k, \\ &Z \geq 0, Z = Z^T, Z^2 = Z. \end{aligned} \tag{3.1}$$

Peng and Xia [98] proved that any feasible solution Z for this 0-1 SDP model is necessarily associated to a feasible MSSC assignment matrix X. Therefore, an equivalence relation among the MSSC formulations (1) and (3.1) is established. Regarding complexity, the 0-1 SDP model is linear except for the constraint $Z^2 = Z$.

3.1.1 Valid inequalities for the 0-1 SDP formulation

Peng and Xia [98] also derived valid inequalities for (3.1) from a property of semidefinite positive matrices. Suppose Z a feasible solution for (3.1). Since Z is semidefinite positive, it follows that there exists an index $i_1 \in 1, \ldots, n$ such that

$$Z_{i_1 i_1} = \max_{i,j} Z_{ij} > 0.$$

Since $Z^2 = Z$, then $\sum_{j \in \mathcal{I}_1} (Z_{i_1 j})^2 = Z_{i_1 i_1}$, where $\mathcal{I}_1 = \{j : Z_{i_1 j} > 0\}$. This implies that

$$\sum_{j \in \mathcal{I}_1} \frac{Z_{i_1 j}}{Z_{i_1 i_1}} Z_{i_1 j} = 1.$$

From the choice of i_1 and the constraint $\sum_{j=1}^{n} Z_{i_1 j} = \sum_{j \in \mathcal{I}_1} Z_{i_1 j} = 1$, Peng and Xia [98] concluded that

$$Z_{i_1 j} = Z_{i_1 i_1}, \quad \forall j \in \mathcal{I}_1.$$

If the respective columns and lines associated to the index set $\mathcal{I}_1$ are eliminated, the remaining matrix is still semidefinite positive with the same aforementioned properties.

Therefore, if the process is repeated, the following valid inequalities are obtained

$$Z_{i_\beta j} = Z_{i_\beta i_\beta}, \quad \forall j \in \mathcal{I}_\beta, \beta = 1, \ldots, k.$$

3.2 A branch-and-cut algorithm for the 0-1 SDP formulation

Peng and Xia [98] have proposed an LP relaxation for the MSSC 0-1 SDP formulation by removing the constraint that $Z^2 = Z$. Then, valid inequalities are used to strengthen the model based on the fact that if the pairs of entities (o_i,o_j) and (o_i,o_ℓ) belong to the same cluster, then o_i and o_ℓ also belong to the same cluster. From the definition of Z, these relationships imply that

$$Z_{ij} = Z_{j\ell} = Z_{i\ell} = Z_{ii} = Z_{jj} = Z_{\ell\ell}.$$

In their paper, such inequalities are partially characterized by the following ones

$$\begin{array}{lll} Z_{ij} \leq Z_{ii} & \forall i, j & \text{(pair inequalities)} \\ Z_{ij} + Z_{i\ell} \leq Z_{ii} + Z_{j\ell} & \forall i, j, \ell & \text{(triangular inequalities).} \end{array}$$

This partial polyhedron characterization was inspired by the work of Lisser and Rendl [78] for graph partitioning. Thus, the resulting LP relaxed model is expressed by

$$\begin{array}{ll} \min\limits_{Z} \; Tr(W_p W_p^T (I - Z)) & \\ \text{subject to} & \\ Ze = e, Tr(Z) = k, & \\ Z \geq 0 & \\ Z_{ij} \leq Z_{ii} & \forall i, j \\ Z_{ij} + Z_{i\ell} \leq Z_{ii} + Z_{j\ell} & \forall i, j, \ell \end{array} \tag{3.2}$$

The authors report some results on benchmark instances for which the lower bounds provided by this LP relaxation are very close to the optimal values. However, they claim that its resolution is unpractical for large-sized data due to the huge amount $O(n^3)$ of triangular inequalities. We propose here to tackle this limitation via a cutting plane procedure which adds triangular inequalities only if they are violated.

Although the focus of Peng and Xia [98] is not on exact methods, the authors suggest a simple branching scheme. Suppose that for the optimal solution of the LP relaxation Z^R there are indices i, j such that $Z^R_{ij}(Z^R_{ii} - Z^R_{ij}) \neq 0$, then one can produce a branch with $Z_{ii} = Z_{ij}$ and another one with $Z_{ij} = 0$. With this branching scheme, the number of different branches is limited to at most 2^{n^2}.

Regarding variable selection, we propose to choose indices i, j as the $\text{argmax}_{i,j} \min\{Z^R_{ij}, Z^R_{ii} - Z^R_{ij}\}$. The reason behind this selection is to choose indices i and j with the least tendency to assign o_i and o_j to the same cluster, or to different ones. Consequently, it is expected to have, in both branches, a considerable impact on the LP relaxation.

Algorithm 1 summarizes the whole branch-and-cut method. In Line 1, the list L of unsolved problems is initialized with the 0-1 SDP model (3.1). List L is implemented with a stack data structure so that a depth-first search is performed while exploring the enumeration tree. In Line 3, the best current solution s^* is initialized by variable neighborhood search (VNS) [89, 55] which is allowed to execute for one minute of CPU time.

Lines 4–23 consist of the main loop of the branch-and-cut method which is repeated until the tree is completely explored. In Lines 5–6, a problem P is removed from L, and its relaxation P^R as in (3.2) is considered for being solved without its $O(n^3)$ triangular constraints. In the loop of Lines 7–10, the relaxed problem P^R is solved via cutting planes until there are no longer triangular inequalities which are violated. Limited computational experiments showed that adding the 3000 most violated cuts is a good choice for the number of cutting planes added in Line 9. Thus, the LP relaxation is kept fairly small as compared to the full set of constraints.

If P^R is feasible in Line 11, then due to equivalence between (1) and (3.1), a feasible solution s is obtained to (1) from Z^R in Line 13. If $cost(s)$ is better than $cost(s^*)$ then

the latter is updated, where function $cost(\cdot)$ returns the cost of a solution to either formulation (1) or (3.1). Branching is performed whenever the lower bound Z^R is smaller than the current upper bound $cost(s^*)$ in Line 18. Consequently, problem P is split into two subproblems in Line 20 according to variables selected by the rule of Line 19. These subproblems are added to L in Line 20. Finally, the optimal solution s^* is returned in Line 24 when L is empty.

```
1  Algorithm: BC-SDP-MSSC
2  Let L be a list of unsolved problems. Initialize L with (3.1);
3  Solution s* is initialized by VNS ;
4  repeat
5  |  Select a problem P from L and remove it from L;
6  |  Consider the linear relaxation P^R of P as in (3.2) without the triangular
   |  inequalities;
7  |  repeat
8  |  |  Solve P^R. Let Z^R be an optimal solution if one exists;
9  |  |  Look for violated triangular inequalities and add them to P^R;
10 |  until there are no violated triangular inequalities ;
11 |  if P^R is feasible then
12 |  |  if Z^R is feasible for P then
13 |  |  |  Obtain a feasible solution s to (1) from Z^R;
14 |  |  |  if cost(s) < cost(s*) then
15 |  |  |  |  s* ← s;
16 |  |  |  end
17 |  |  end
18 |  |  if cost(Z^R) < cost(s*) then
19 |  |  |  Calculate (i, j) ∈ argmax_{i,j} min{Z^R_ij, Z^R_ii − Z^R_ij};
20 |  |  |  Branch P into two subproblems by means of cuts Z_ij = 0 and
   |  |  |  Z_ii − Z_ij = 0 and add them to L;
21 |  |  end
22 |  end
23 until L = ∅ ;
24 return s*;
```

Algorithm 1: Branch-and-cut SDP-based algorithm for MSSC

3.3 Computational experiments

In this section we report on the computational experiences with our SDP-based branch-and-cut algorithm for MSSC. Results were obtained using an AMD 2 GHz architecture

with g++ (Option -O3) C compiler. Package CPLEX 10.0 is called to solve with dual simplex the LP relaxations of the problems generated.

In order to better evaluate the cutting plane procedure (Lines 7–10) of the proposed BC-SDP-MSSC algorithm, three distinct versions of the program were devised:

1. BC-tri adds all pair inequalities a priori and exploits the triangular inequalities cuts as cutting planes.
2. BC-all exploits both pair inequalities and triangular inequalities as cutting planes.
3. BC-hpair adds a half of the pair inequalities a priori and exploits the remaining ones as well as the triangular inequalities as cutting planes.

Comparisons were made using some standard problems from the cluster analysis literature (i. Ruspini's 75 points in the Euclidean plane [103], ii. Späth's 89 Bavarian postal codes in three dimensions [111], iii. the synthetic HATCO data set published in [51] consisting of 100 objects in seven dimensions, iv. Fisher's 150 iris problem in four dimensions [36], and v. Grötschel and Holland's 202 European cities coordinates [47]). To the best of our knowledge, problems (iii) and (v) were never reported to be solved exactly in the literature.

In all tables presented here, the first column gives values of k and the second column gives the optimal objective function values. More information is provided when available. Peng and Xia [98] report CPU times used for solving LP relaxation (3.2) with all its $O(n^3)$ triangular inequalities for data sets (i), with $k = 2, \ldots, 5$, and (ii), with $k = 2, \ldots, 9$. Thus, the third column of Tables 3.1 and 3.2 present those computing times, which were obtained with an IBM RS-6000 workstation and CPLEX 7.1 with AMPL interface. Remaining columns are associated to CPU times of exact methods, i.e, the column generation algorithm (CGA) of du Merle et al. [28] obtained with a SUN ULTRA 200 MHz station, the repetitive branch-and-bound algorithm (RBBA) of Brusco [12], and the three versions of BC-SDP-MSSC. Moreover, a last column is included in the tables to present gap values between upper and lower bounds obtained by the solution of (3.2) at the root node, denoted UB^0 and LB^0 respectively, which are calculated as $(UB^0 - LB^0)/LB^0$. The letter

'i' indicates that no initial gap exists, i.e., the problem is already solved by our approach at the root node, without branching. Otherwise, the number of nodes of the branch-and-cut tree is given in parenthesis.

Table 3.1: Results for Ruspini's data set

k	Opt. sol.	CPU times (seconds)						% gap
		LP relax.	CGA	RBBA	BC-tri	BC-all	BC-hpair	
2	89337.8	27.81	12.33	0.05	3.54	14.33	3.56	i
3	51063.4	66.58	16.50	2.10	8.79	15.10	8.34	i
4	12881.0	7.22	7.30	136.29	0.69	2.83	0.48	i
5	10126.7	9.47	14.43	1699.75	0.78	2.91	0.60	i
6	8575.4		25.45	> 12h	1.97	3.46	1.03	i
7	7126.2		30.19	> 12h	1.20	2.22	0.98	i
8	6149.6		43.11	> 12h	8.24	12.67	7.27	0.5 (7)
9	5181.6		31.26	> 12h	2.90	4.54	2.87	0.3 (3)
10	4446.3		27.62	> 12h	2.08	3.88	2.39	0.2 (3)
20	1721.2			> 12h	0.31	0.41	0.28	i
30	741.8			> 12h	0.17	0.16	0.14	i

Table 3.2: Results for Späth's data set

k	Opt. sol.	CPU times (seconds)						% gap
		LP relax.	CGA	RBBA	BC-tri	BC-all	BC-hpair	
2	$6.02546\ 10^{11}$	283.26	19.92	603.33	9.96	150.36	10.78	i
3	$2.94506\ 10^{11}$	418.07	1479.75	> 12h	27.33	136.50	27.58	i
4	$1.04474\ 10^{11}$	99.54	70.49	> 12h	31.81	96.61	26.31	i
5	$5.97615\ 10^{10}$	60.67	39.59	> 12h	18.23	53.28	10.07	i
6	$3.59085\ 10^{10}$	52.55	87.61	> 12h	17.88	47.00	13.13	i
7	$2.19832\ 10^{10}$	61.78	106.55	> 12h	38.85	52.35	25.06	i
8	$1.33854\ 10^{10}$	26.91	76.86	> 12h	10.11	18.41	9.04	i
9	$8.42375\ 10^{9}$	18.04	75.58	> 12h	8.84	12.36	5.51	i
10	$6.44647\ 10^{9}$		84.33	> 12h	8.02	10.46	4.26	i
20	$7.48215\ 10^{8}$			> 12h	0.98	0.99	0.74	i
30	$1.71392\ 10^{8}$			> 12h	0.31	0.34	0.26	i

Table 3.3: Results for Fisher's data set

k	Opt. sol.	CPU times (seconds)					% gap
		CGA	RBBA	BC-tri	BC-all	BC-hpair	
2	152.3479	18697.59	0.05	166.82	549.28	169.44	i
3	78.8514	497.55	2.10	512.85	454.70	283.24	i
4	57.2284	505.49	136.29	301.12	299.40	240.19	i
5	46.4461	350.25	1699.75	152.18	237.36	145.54	i
6	39.0399	584.04	> 12h	141.88	163.32	147.51	i
7	34.2982	427.04	> 12h	1061.16	847.93	742.83	0.0 (7)
8	29.9889	695.37	> 12h	89.92	123.18	108.73	i
9	27.7860	855.23	> 12h	92.19	88.78	70.04	i
10	25.8340	628.92	> 12h	76.47	73.07	59.66	i
20	14.2208		> 12h	155.75	104.91	87.06	0.0 (5)
30	9.5552		> 12h	175.87	145.24	155.52	0.1 (23)

Table 3.4: Results for HATCO's data set

k	Opt. sol.	CPU times (seconds)				% gap
		RBBA	BC-tri	BC-all	BC-hpair	
2	600.108	0.80	108.79	101.88	69.68	i
3	506.962	> 12h	158.15	146.19	141.52	i
4	426.602	> 12h	85.39	88.54	68.00	i
5	383.831	> 12h	61.46	80.33	52.56	i
6	344.534	> 12h	188.49	151.39	145.03	0.0 (3)
7	313.582	> 12h	246.91	191.20	181.13	0.1 (5)
8	288.601	> 12h	377.88	378.52	286.67	0.5 (11)
9	264.599	> 12h	421.22	381.70	314.30	0.6 (13)
10	241.128	> 12h	315.99	284.10	217.74	0.4 (15)
20	114.032	> 12h	5.00	6.07	4.15	0.0 (3)
30	62.992	> 12h	2.51	2.72	2.35	0.0 (5)

Tables 1–4 suggest the following conclusions:

- For data sets (i) and (ii), branch-and-cut algorithms are able to prove optimality of model (3.1) in less computing time than solving only its LP relaxation given by (3.2) with all $O(n^3)$ triangular constraints. This shows the efficiency of the cutting-plane approach.

- Algorithm BC-all is in most of cases outperformed by either BC-tri or BC-hpair. Mainly for small k, a large amount of pair inequalities are active at the LP optimal solution, and therefore, exploiting all of them as cutting planes is not a worthwhile strategy.

- Algorithm RBBA is particularly efficient for small values of k, while its performance quickly deteriorates as k increases. This is due to the fact that the number of branches in RBBA is $O(k^n)$.

- Except for one case in data sets (i–iii), algorithms BC-tri and BC-hpair always proved optimality in smaller CPU times than those obtained by the column generation algorithm of [28]. This comparison is not completely fair since their results were obtained in an older computing architecture. If a straightforward scale factor of 10 (2 GHz $= 10 \cdot 200$ MHz) is used for the column generation algorithm of [28], then it performs better than our branch-and-cut algorithms in 14 out of 27 instances.

- Relaxation (3.2) provides very good bounds for MSSC since initial gap values are never larger than 0.6%. Moreover, more than 65% of the tested data sets are exactly solved after considering only the root node of the enumeration. This may be due to the inclusion of the triangle inequalities in the formulation of the problem. Grötschel and Wakabayashi [48] used triangular inequalities within a branch-and-cut algorithm for partitioning with the sum-of-cliques criterion. Such constraints appear to suffice in almost all of their computational tests too.

- Computing times of the branch-and-cut algorithms does not increase as the number of clusters k increases. In fact, there is no evident relationship between the complexity of solving (3.1) and the value of k. However, performance seems to improve for large values of k, as shown by the results for data sets (i), (ii) and (iv).

The tests also assessed the quality of the solutions obtained by VNS for MSSC since all initial upper bounds proved to be optimal.

Table 3.5 present results for Grötschel and Holland's 202 European cities coordinates [47] whose value of n is the largest among data sets (i–v). Results show that BC-hpair is able to determine proved minimum sum-of-squares partitions when k is large, while their performance deteriorates as the value of k decreases. In our tests, the algorithms were not able to solve instances with $k \leq 8$ in less than 12 hours.

Table 3.5: Results for Grötschel and Holland's data set

k	Opt. sol.	CPU times (seconds)	% gap
		BC-hpair	
9	4376.1937	48885.38	0.2 (9)
10	3794.4880	23680.84	0.0 (7)
20	1523.5086	3839.77	0.1 (13)
30	799.3109	1060.77	0.0 (13)

Finally, note that our branch-and-cut algorithm based on solving LP relaxations of the 0-1 SDP formulation proposed in [98] can be extended to other related clustering problems (e.g. normalized k-cut minimization, balanced clustering; see [97] for details).

CHAPTER 4 : AN IMPROVED COLUMN GENERATION ALGORITHM FOR MINIMUM SUM-OF-SQUARES CLUSTERING

A column generation algorithm for MSSC was given in du Merle et al. [28]. The bottleneck of that algorithm is the resolution of the auxiliary problem of finding a column with negative reduced cost. We propose in this chapter a new way to solve this auxiliary problem based on geometric arguments.

4.1 Column generation algorithm revisited

Partitioning problems in cluster analysis can be mathematically formulated by considering all possible clusters. Let us consider any cluster C_t for which

$$a_{it} = \begin{cases} 1 & \text{if entity } o_i \text{ belongs to cluster } C_t \\ 0 & \text{otherwise,} \end{cases}$$

and let us denote by y_t the centroid of points p_i such that $a_{it} = 1$. Thus, the cost c_t of cluster C_t can be written as

$$c_t = \sum_{i=1}^{n} \|p_i - y_t\|^2 a_{it}.$$

An alternative formulation for MSSC is then given by

$$\begin{aligned}
&\min_{z} \sum_{t \in T} c_t z_t \\
&\text{subject to} \\
&\sum_{t \in T} a_{it} z_t = 1 && \forall i = 1, \ldots, n \qquad (4.1) \\
&\sum_{t \in T} z_t = k \\
&z_t \in \{0, 1\} && \forall t \in T,
\end{aligned}$$

where $T = \{1, \ldots, 2^n - 1\}$. The z_t variables are equal to 1 if cluster C_t is in the optimal partition and to 0 otherwise. The first set of constraints state that each entity belongs to one cluster, and the following constraint expresses that the optimal partition contains exactly k clusters. Without loss of generality, they can be replaced by

$$\sum_{t \in T} a_{it} z_t \geq 1, \quad \forall i = 1, \ldots, n, \quad \text{and} \quad \sum_{t \in T} z_t \leq k,$$

because (i) a covering of O which is not a partition cannot be optimal, and (ii) any partition with less than k clusters has objective value greater or equal to the optimal partition with k clusters.

This is a large set partitioning problem with a side constraint, for which the number of variables is exponential in the number n of entities. Therefore, it cannot be explicitly written and solved in a straightforward way unless n is small. The column generation method proposed in [28] works with a reasonably small subset $T' \subseteq T$ of the columns in (4.1), i.e., with a *restricted master problem*. The method is combined with branch-and-bound in order to solve exactly (4.1) for medium size (about 100-200 entities) to fairly large instances (1000 entities or more).

Problem (4.1) is solved iteratively, augmenting the number of columns in the restricted master problem until optimality is proved with the columns at hand. Entering columns are found by solving an auxiliary problem, i.e., finding the list of entities of a cluster whose

associated variable in (4.1) has negative reduced cost. Since a standard column generation method for solving the linear relaxation of the formulation (4.1) suffers from very slow convergence due to high degeneracy, two strategies for stabilizing column generation [29] were used and compared in [28]. That one for which the linear relaxation is solved by an interior-point algorithm, i.e., the weighted version of the analytic center cutting plane method (ACCPM) of Goffin, Haurie, and Vial [45], was found to be the best.

Once the linear relaxation of the problem is solved, the integrality of the obtained solution is checked (and often found to hold for small to medium size problems with few clusters). Then, if the solution is not integer, branching is needed. The branching rule used in [28] is the standard one, due to Ryan and Foster [104], i.e., branching by imposing in one hand that two entities belong to the same cluster and on the other hand that at most one of these entities belongs to any given cluster.

4.1.1 Auxiliary problem

The biggest obstacle for an efficient exact resolution of the MSSC via column generation is the difficulty of the auxiliary problem. The dual of the formulation (4.1) is expressed by

$$
\begin{aligned}
&\max_{\lambda,\sigma} \; k\sigma + \sum_{i=1}^{n} \lambda_i \\
&\text{subject to} \\
&-\sigma + \sum_{i=1}^{n} a_{it}\lambda_i \leq c_t && \forall t \in T \qquad (4.2)\\
&\lambda_i \geq 0 && \forall i = 1, \ldots, n \\
&\sigma \geq 0,
\end{aligned}
$$

where the λ_i for $i = 1, \ldots, n$ and σ are dual variables associated with the covering constraints and with the side constraint.

Problem (4.2) is solved using a cutting plane method, starting with a relaxation and adding constraints as necessary. In the classical cutting plane method by Kelley [67], cuts

are generated at an extreme point of the relaxed dual formulation. However, Kelley's method is known to slow down considerably in the presence of degeneracy [29]. ACCPM tackles this shortcoming by generating cuts at an analytic center of the current dual feasible region (cf. [33]). In both cases, given dual values λ, σ, a violated cut is searched to be added to the relaxed dual problem. The violation π_t of a constraint is given by

$$\pi_t = c_t + \sigma - \sum_{i=1}^{n} \lambda_i a_{it}.$$

Since we are interested in finding violated constraints, $\pi_t < 0$. The auxiliary problem is then given by $\pi^* = \min_t \pi_t$. Although the enumeration of π_t for all $t \in T$ is too expensive, the value of π^* can be found by solving

$$\pi^* = \sigma + \min_{y_v \in \mathbb{R}^s, v \in \mathbb{B}^n} \sum_{i=1}^{n} (\|p_i - y_v\|^2 - \lambda_i) v_i. \tag{4.3}$$

with y_v denoting the centroid of points p_i for which $v_i = 1$. If $\pi^* < 0$, then the optimal solution v^* to (4.3) is added as a cut to the relaxed dual problem (in the primal, this is equivalent to adding a column to the restricted master problem together with its associated primal variable). Otherwise, problem (4.2) (or equivalently, problem (4.1)) is solved optimally.

From Huygens' theorem (e.g., Edwards and Cavalli-Sforza [30]), which states that the sum of squared distances from all entities of a given cluster to its centroid is equal to the sum of squared distances between pairs of entities of this cluster divided by its cardinality, problem (4.3) can be expressed by

$$\begin{aligned} \pi^* &= \sigma + \min_{v \in \mathbb{B}^n} \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \|p_i - p_j\|^2 v_i v_j}{\sum_{i=1}^{n} v_i} - \sum_{i=1}^{n} \lambda_i v_i \\ &= \sigma + \min_{v \in \mathbb{B}^n} \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (\|p_i - p_j\|^2 - \lambda_i - \lambda_j) v_i v_j - \sum_{i=1}^{n} \lambda_i v_i}{\sum_{i=1}^{n} v_i}. \end{aligned} \tag{4.4}$$

It is a hyperbolic (or fractional) program in 0-1 variables with quadratic numerator and linear denominator. This problem is solved in [28] by an adaptation to binary variables of Dinkelbach's algorithm [24]. This algorithm begins with a tentative value for (4.4) then reduces the problem to unconstrained quadratic 0-1 optimization by multiplying both sizes by the denominator and regrouping terms. If a positive value is obtained for the optimal solution of this last problem its corresponding value in (4.4) is computed and the procedure iterated. Its most expensive step is the resolution of a sequence of unconstrained quadratic 0-1 programs, which are solved in [28] by a VNS heuristic until optimality must be checked by a branch-and-bound algorithm.

4.2 A geometric approach

The auxiliary problem (4.3) can be viewed as minimizing the sum of functions equal to squared distances from the cluster center y_v to each of the entities, but with a limit on each of the distances, after which the corresponding function does not increase anymore. Clearly, for a given location y_v, v_i is equal to 1 if $\|p_i - y_v\|^2 \leq \lambda_i$, and to 0 otherwise. Geometrically, in the plane, this is equivalent to the condition that $v_i = 1$ if y_v belongs to a disc with radius $\sqrt{\lambda_i}$ centered at p_i, and 0 otherwise.

A branch-and-bound algorithm based on the vector v would consider implicitly all 2^n subproblems generated by branching on binary variables v_i for $i = 1, \ldots, n$, while adding constraints $\|p_i - y_v\|^2 \leq \lambda_i$ and $\|p_i - y_v\|^2 \geq \lambda_i$ to the resulting subproblems. However, the resulting problems pertain to D.C. programming and are difficult to solve. Another possibility is to focus on components v_i of v which are equal to 1. We then consider subproblems of the following type:

$$\begin{aligned} &\min_{y} \quad \sum_{i \in S} \|p_i - y\|^2 \\ &\text{subject to} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (4.5) \\ &\|p_i - y\|^2 \leq \lambda_i \qquad\qquad\qquad\qquad \forall i \in S, \end{aligned}$$

where $S \subseteq \{1, 2, \ldots, n\}$ is a non-empty set. Subproblems of type (4.5) are convex programming problems. Proposition 1 shows that an optimal solution for (4.3) is guaranteed to be an optimal solution to a subproblem of type (4.5).

Proposition 4.1. *Let (y_v^*, v^*) be the optimal solution to (4.3). Then, y_v^* is the optimal solution to a subproblem of type (4.5) with a set S for which $\|p_i - y_v^*\|^2 > \lambda_i$ for all $i \notin S$.*

Proof. Define S^* as the index set of all points p_i such that $\|p_i - y_v^*\|^2 \leq \lambda_i$. Thus, for $i \notin S^*$, $\|p_i - y_v^*\|^2 > \lambda_i$. Now let y' be the optimal solution for (4.5) with S^* and suppose that y_v^* is not the optimal solution for it. Since, $\|p_i - y_v^*\|^2 > \min\{\|p_i - y'\|^2, \lambda_i\}$ for all $i \notin S^*$, the cost of (y', v^*) is smaller than that of (y_v^*, v^*) in (4.3), which is a contradiction. □

The auxiliary problem (4.3) still has another very important property which states that at optimal solution (v_v^*, v^*), y_v^* is at the centroid of points p_i for which $v_i^* = 1$. Given a subproblem of type (4.5) with index set S, this implies that if the centroid of the points p_i such that $i \in S$ is not a feasible solution, then we conclude that the subproblem does not contain the optimal solution to (4.3). In the plane, it amounts to say that the centroid must belong to the intersection of all discs with index $i \in S$ (which includes the particular case where S is a singleton).

Let us define A as the set of discs whose boundaries intersect at least one other boundary of a disc in two points, and B as the set of discs that do not belong to A. They include isolated discs and nested discs (i.e., discs that contain other discs in their interior and discs that are entirely contained into other ones). An useful result is shown by the following proposition:

Proposition 4.2. *The number T of distinct regions which are intersection of discs $\|p_i - y\|^2 \leq \lambda_i$ is bounded by $2n(n-1)$.*

Proof. The total number of points of intersection among discs in A is at most $|A|(|A|-1)$. Since each one of them can be associated with at most 4 different regions, and as each

of these regions contains at least two of these points, the number of regions r_A which are delimited by discs in A is bounded by $2|A|(|A|-1)$.

Each one of the discs in B can delimit at most one region. Consequently, the number of regions r_B delimited by discs in B is equal to $|B|$.

Thus,

$$\begin{aligned} T = r_A + r_B &\leq 2|A|(|A|-1) + |B| \\ &\leq 2(|A|+|B|)(|A|+|B|-1) \\ &\leq 2n(n-1) \end{aligned}$$

□

Proposition 2 implies that the number of subproblems of type (4.5) that need to be solved in order to obtain an optimal solution to (4.3) is polynomially bounded.

An algorithm was proposed in [26] for a similar problem in location theory, i.e., the 1-center Weber problem with limited distances. The only difference between this problem and (4.3) lies in the fact that Euclidean distances are used instead of squared ones. The algorithm proceeds by considering all intersection points between discs in the plane, and then solves, for each one of these points, the subproblems of type (4.5) corresponding to the four possible regions which are adjacent to the point. For instance, suppose that p is an intersection point between discs centered at points p_i and p_j, then the four possible non-empty index sets corresponding to regions for which p can be a vertex are formed by: $S_a = \{\ell : \|p_\ell - p\|^2 \leq \lambda_\ell, \ell \neq i,j\}$; $S_b = \{i\} \cup S_a$; $S_c = \{j\} \cup S_a$; and $S_d = \{i,j\} \cup S_a$.

It appears that the algorithm of [26] implicitly assumes that regions delimited by discs in B either do not exist or can be discarded for evaluation. However, this is not true either for the 1-center Weber problem with limited distances or for (4.3), which makes the algorithm proposed in [26] incomplete.

Figure 4.1 exhibits an auxiliary problem configuration which appears after 11 iterations of our column generation algorithm while clustering the 10 points described at the top of

Figure 2.2 into 3 clusters. The shaded region (2) in the figure corresponds to the optimal solution of the auxiliary problem while region (1) is the solution provided if the algorithm of [26] is used instead.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
	690	190	823	73	782	338	287	410	769	962
	166	887	695	125	979	894	340	263	768	831
$\sqrt{\lambda}$	382.78	360.47	203.34	379.22	208.24	168.79	211.61	198.70	138.14	332.88

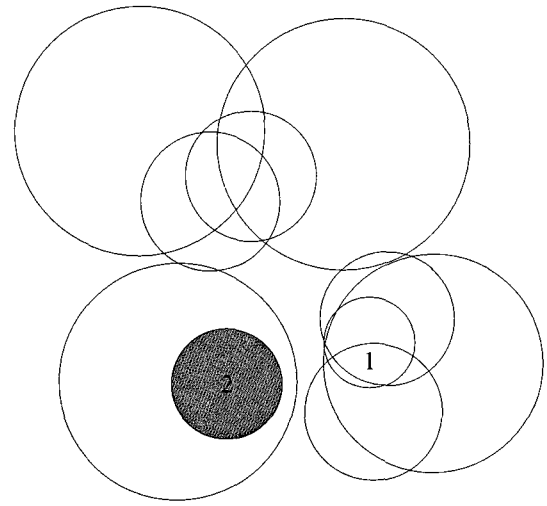

Figure 4.1: Configuration of convex regions experimentally obtained

Algorithm 1 below is the new algorithm obtained after adapting and completing the algorithm of [26] in order to consider sets S corresponding to regions delimited by discs of B. This algorithm requires $O(n^3)$ time since there are $O(n^2)$ possible intersection points and step 6 takes $O(n)$ time per subproblem. Additional operations due to steps 9–13 are performed in $O(n^2)$ time.

The following simple condition holds if two discs associated to points p_i and p_j intersect

$$\|p_i - p_j\| \leq \sqrt{\lambda_i} + \sqrt{\lambda_j},$$

Algorithm 1
Enumerate all intersection points of pairs of discs in the plane as well as all the points whose associated disc does not intersect any other one. Let L_1 and L_2 be the corresponding lists;
for *each point* $p \in L_1$ *defined by the intersection of discs centered at points* p_i *and* p_j **do**
    Find the set S of all k such that $k \neq i, j$ and $\|p_k - p\|^2 \leq \lambda_k$;
    Consider four sets: $S, S \cup \{i\}, S \cup \{j\}$, and $S \cup \{i, j\}$;
    Solve subproblems of type (4.5) defined by each of these sets;
    Update the best solution if an improving one is found;
end
for *each point* $q \in L_2$ **do**
    Find the set S' composed by q and the indices of all points associated to discs containing that associated to q;
    Solve subproblems of type (4.5) defined by each S';
    Update the best solution if an improving one is found;
end
return *best solution found*

one disc being contained in the other if

$$\|p_i - p_j\| \leq |\sqrt{\lambda_i} - \sqrt{\lambda_j}|.$$

Based on these conditions, an acceleration procedure for Algorithm 1 is to build for each point p_i, $i = 1, \ldots, n$ a list of non-decreasing distances to any other point. In step 2 of Algorithm 1, each point p_i is tested in turn with all other points p_j for $j = 1, \ldots, n$, such that $j > i$, in order to know if their respective discs intersect. Indeed these points can be considered in the order given by the sorted list of p_i and the search for intersections halted as soon as

$$\|p_i - p_j\| > \sqrt{\lambda_i} + \sqrt{\lambda_{max}},$$

where $\lambda_{max} = \max\{\lambda_i\}$ for $i = i + 1, \ldots, n$. Note that exactly the same test can be used in order to speed up step 4 of the algorithm.

4.2.1 Branching

The classical branching rule is applied whenever branching is needed to solve (4.1). It consists on finding two rows i_1, i_2 such that there are two columns t_1 and t_2 with fractional values at the optimum and such that $a_{i_1 t_1} = a_{i_2 t_1} = 1$ and $a_{i_1 t_2} = 1, a_{i_2 t_2} = 0$. Then, constraints are introduced in the auxiliary problem of both subproblems in the form (i) $v_{i_1} = v_{i_2}$ for one branch, and (ii) $v_{i_1} + v_{i_2} \leq 1$ for the other one. Problem (4.3) in the presence of branching constraints can be expressed as

$$\begin{aligned} &\min_{y_v \in \mathbb{R}^s, v \in \mathbb{B}^n} \quad \sum_{i=1}^{n} (\|p_i - y_v\|^2 - \lambda_i) v_i \\ &\text{subject to} \qquad\qquad (4.6) \\ &v_i = v_j \qquad\qquad \forall (i,j) \in I_1 \\ &v_i + v_j \leq 1 \qquad\qquad \forall (i,j) \in I_2 \end{aligned}$$

where I_1, I_2 are the index sets of pairs of entities involved in constraints of form (i) and (ii), respectively.

Algorithm 1 is not able to solve problem (4.6), since optimal solutions may now be associated to index sets which do not correspond directly to a region in the plane. In fact, Proposition 1 is no longer valid in the presence of branching constraints. A very simple example consists of two points p_i, p_j whose discs of radius $\sqrt{\lambda_i}$ and $\sqrt{\lambda_j}$ do not intersect while a constraint states that points p_i and p_j must be together. In this case, none of the index sets S scanned by Algorithm 1 is able to provide a feasible solution to the problem.

Fortunately, Proposition 3 below shows that Algorithm 1 can be slightly modified in order to solve problem (4.6) exactly. Let us first define three index sets associated with any vector y_v

- $S_1(y_v)$ is the index set of points p_i for which $\|p_i - y_v\|^2 \leq \lambda_i$, and for which $(i,j) \in I_1$ or $(i,j) \in I_2$ with $j \in S_1(y_v) \cup S_2(y_v)$;
- $S_2(y_v)$ is the index set of points p_i for which $\|p_i - y_v\|^2 > \lambda_i$, and for which $(i,j) \in I_1$ with $j \in S_1(y_v)$;

- $S_3(y_v)$ is the index set of points p_i for which $\|p_i - y_v\|^2 \leq \lambda_i$, and such that $i \notin S_1(y_v)$.

Proposition 4.3. *Let (y_v^*, v^*) be the optimal solution of (4.6) and let $\bar{v}^* = (v_i^* \mid i \in S_1(y_v^*) \cup S_2(y_v^*))$. Then, $(y_v^*, \bar{v}^*)$ is the optimal solution of a subproblem given by*

$$
\begin{aligned}
&\min \sum_{i \in S_1 \cup S_2} \|p_i - y\|^2 v_i + \sum_{i \in S_3} \|p_i - y\|^2 \\
&\text{subject to} \\
&\|p_i - y\|^2 v_i \leq \lambda_i && \forall i \in S_1 \\
&\|p_i - y\|^2 \leq \lambda_i && \forall i \in S_3 \qquad (4.7) \\
&v_i \in \{0,1\} && \forall \in S_1 \cup S_2 \\
&v \in X \\
&y \in \mathbb{R}^s
\end{aligned}
$$

with sets $S_1, S_2, S_3 \subseteq \{1, \ldots, n\}$ and where X is the polyhedron of branching constraints.

Proof. From the definition of $S_1(y_v^*)$, $S_2(y_v^*)$ and $S_3(y_v^*)$, $\|p_i - y_v^*\| > \lambda_i$ for all $i \notin S_1(y_v^*) \cup S_2(y_v^*) \cup S_3(y_v^*)$.

Now let $(y_v', \bar{v}')$ be the optimal solution to (4.7) regarding $S_1 = S_1(y_v^*)$, $S_2 = S_2(y_v^*)$ and $S_3 = S_3(y_v^*)$, and suppose that the optimal solution of (4.6) $(y_v^*, \bar{v}^*)$ is not optimal for (4.7). Then, we can construct v' as:

- $v_i' = \bar{v}_i', \forall i \in S_1 \cup S_2$;
- $v_i' = 1, \forall i \in S_3$;
- $v_i' = 0$, otherwise;

such that the cost of (y_v', v') is smaller than that of (y_v^*, v^*) in (4.6), which is a contradiction. □

The importance of Proposition 3 lies in the fact that, given the optimal y_v^*, the optimal subproblem of type (4.7) with sets $S_1 = S_1(y_v^*), S_2 = S_2(y_v^*)$ and $S_3 = S_3(y_v^*)$ is by

definition associated to the region in the plane originated from the intersection of discs $\|p_i - y_v^*\|^2 \leq \lambda_i$. This fact implies that the number of subproblems of type (4.7) which need to be considered in order to solve (4.6) is polynomially bounded. However, (4.7) is a problem with binary variables for which an enumeration method of resolution is needed.

Algorithm 1 can be modified to solve subproblems of type (4.7). For each region in the plane, sets S_1, S_2 and S_3 are determined to form a subproblem of type (4.7) (remark that any location y in a given region of the plane defines the same sets $S_1(y)$, $S_2(y)$ and $S_3(y)$). Then, the subproblem is solved by a branch-and-bound procedure. Note that whenever $S_1, S_2 = \emptyset$, subproblem (4.7) turns out to be equivalent to subproblem (4.5), and therefore, enumeration is not needed.

Decisions in the branch-and-bound algorithm are made by presence-absence dichotomy on variables v_i, for $\forall i \in S_1 \cup S_2$. Lower bounds are calculated in each node as the difference of two values:

1. the cost of the node solution, which is calculated with respect to the centroid of points p_i for which decision $v_i = 1$ is fixed;

2. the sum of the prices λ_i of the free variables v_i.

When (4.6) contains a few branching constraints, sets S_1 and S_2 have small cardinality by definition. So, the given branch-and-bound method to solve (4.7) performs very well in practice.

REMARK: In the presence of a larger number of branching rules, solving (4.7) becomes a more difficult task. To this purpose, we note that (4.7) can be reformulated exactly (in the sense of [76]) by introducing parameters:

$$M_i \geq \max_j \|p_i - p_j\|^2 \quad \forall i \in S_1 \cup S_2,$$

decision variables:

$$\omega_i \in [0, M_i] \quad \forall i \in S_1 \cup S_2,$$

and constraints:

$$\|p_i - y\|^2 \leq \omega_i + (1 - v_i)M_i \quad \forall i \in S_1 \cup S_2$$

to (4.7). We then replace constraints $\|p_i - y\|^2 v_i \leq \lambda_i \; \forall i \in S_1$ by

$$\|p_i - y\|^2 \leq \lambda_i + (1 - v_i)M_i \quad \forall i \in S_1,$$

and the terms $\|p_i - y\|^2 v_i$ for $i \in S_1 \cup S_2$ in the objective function by ω_i. We thus obtain the reformulated problem:

$$\begin{aligned}
&\min \sum_{i \in S_1 \cup S_2} \omega_i + \sum_{i \in S_3} \|p_i - y\|^2 \\
&\text{subject to} \\
&\|p_i - y\|^2 \leq \lambda_i + (1 - v_i)M_i && \forall i \in S_1 \\
&\|p_i - y\|^2 \leq \omega_i + (1 - v_i)M_i && \forall i \in S_1 \cup S_2 \\
&\|p_i - y\|^2 \leq \lambda_i && \forall i \in S_3 \qquad (4.8) \\
&v_i \in \{0, 1\} && \forall i \in S_1 \cup S_2 \\
&v \in X \\
&y \in \mathbb{R}^s \\
&\omega_i \in [0, M_i] && \forall i \in S_1 \cup S_2
\end{aligned}$$

which is a convex MINLP, for which there exist practically efficient algorithms (e.g. [9, 74]). We also remark that its continuous relaxation is a continuous NLP which can be solved in polynomial time [121].

Finally, note that Algorithm 1 can be used without modifications to provide approximate solutions to (4.6). This can be done up to the moment that the exact resolution of (4.6) is required to prove that (4.1) was in fact optimally solved.

4.3 Generalization to the Euclidean space

Let us consider a graph $G = (N, E)$ for which there is a node $n_i \in N$ corresponding to each point p_i, for $i = 1, \ldots, n$. Besides, an edge e_{ij} exists in G if and only if

$$\|p_i - p_j\| \leq \sqrt{\lambda_i} + \sqrt{\lambda_j},$$

i.e., $e_{ij} \in E$ if and only if the hyperspheres centered at p_i and p_j with radius $\sqrt{\lambda_i}$ and $\sqrt{\lambda_j}$ intersect.

The following result allows us to generalize the geometric approach in the plane by considering the intersection graph of hyperspheres centered at the points p_i, for $i = 1, \ldots, n$.

Proposition 4.4. *If a solution (y_v^*, v^*) is optimal to (4.3) then the elements of the set $N^* = \{n_i | v_i^* = 1\}$ form a clique in G.*

Proof. Let us suppose that (y_v^*, v^*) is the optimal solution of (4.3) and that the elements of N^* do not form a clique in G. Hence, there are two nodes n_i, n_j in N^* for which $e_{ij} \notin E$, i.e., the hyperspheres centered at p_i and p_j with radius $\sqrt{\lambda_i}$ and $\sqrt{\lambda_j}$ do not intersect. In such a case, y_v^* is certainly located outside at least one of these hyperspheres. Suppose $\|p_i - y_v^*\| > \sqrt{\lambda_i}$, then a reduction in the cost of the solution is obtained by setting $v_i^* = 0$, which contradicts the optimality of (y_v^*, v^*). □

The number of distinct regions resulting from the intersection of hyperspheres is not polynomially bounded in n only. However, Proposition 4 allows to better exploit (4.4) above. Indeed it can be written as

$$\sigma + \min_{v_i \in \{0,1\}} \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (d_{ij}^2 - \lambda_i - \lambda_j) v_i v_j - \sum_{i=1}^{n} \lambda_i v_i}{\sum_{i=1}^{n} v_i},$$

where d_{ij} represents the Euclidean distance between the entities associated to variables v_i and v_j. Coefficients $d_{ij}^2 - \lambda_i - \lambda_j$ of the product $v_i v_j$ can be made arbitrarily large in (4.4) if $d_{ij} > \sqrt{\lambda_i} + \sqrt{\lambda_j}$ due to Preposition 4, since $v_i = v_j = 1$ does not occur in the optimal solution.

4.3.1 Branching

As proposed in [28], branching constraints of type $v_i = v_j$ can be added to the auxiliary problem (4.4) by reducing by one the number of its variables and updating coefficients accordingly. In the case of branching constraints of type $v_i + v_j \leq 1$, it suffices to set coefficient $d_{ij}^2 - \lambda_i - \lambda_j$ to an arbitrary large value. Thus, the auxiliary problem is expressed by

$$\sigma + \min_{v_{i'} \in \{0,1\}} \frac{\sum_{i'=1}^{n'-1} \sum_{j'=i'+1}^{n'} (d_{i'j'}^2 - w_{j'}\lambda_{i'} - w_{i'}\lambda_{j'}) v_{i'} v_{j'} - \sum_{i'=1}^{n'} (w_{i'}\lambda_{i'} - d_{i'i'}^2) v_{i'}}{\sum_{i'=1}^{n'} w_{i'} v_{i'}}, \tag{4.9}$$

where $w_{i'}$ is the number of variables merged in variable $v_{i'}$. Note that the form of the auxiliary problem is not changed. It is still a fractional program in 0-1 variables with quadratic numerator and linear denominator.

An observation must be made when setting coefficients based on the intersection graph of hyperspheres in the presence of branching constraints of type $v_i = v_j$. Suppose entities o_i and o_j for which there is a constraint stating that $v_i = v_j$. Consequently, variables v_i and v_j are merged together in a single variable $v_{i'}$ of (4.9). Let us consider now $v_{k'}$ the variable associated to entity o_k, then coefficient $d_{i'k'}^2 - \lambda_{i'} - 2\lambda_{k'}$ is set to an arbitrary large value in (4.9) only if

$$d_{ik} > \sqrt{\lambda_i} + \sqrt{\lambda_k} \quad \text{and} \quad d_{jk} > \sqrt{\lambda_j} + \sqrt{\lambda_k},$$

i.e., only if

$$d_{i'k} > \sqrt{\lambda_i} + \sqrt{\lambda_j} + 2\sqrt{\lambda_k}.$$

This can be generalized to any pair of variables $v_{i'}, v_{j'}$. Let us consider $\mu_{i'}$ and $\mu_{j'}$ the index set of variables merged at variables $v_{i'}$ and $v_{j'}$, respectively. Thus, if

$$d_{i'j'} > w_{j'} \sum_{i \in \mu_{i'}} \sqrt{\lambda_i} + w_{i'} \sum_{j \in \mu_{j'}} \sqrt{\lambda_j}$$

then $d^2_{i'j'} - w_{j'}\lambda_{i'} - w_{i'}\lambda_{j'}$ can be set to an arbitrary large value in (4.9).

4.3.2 Solving by cliques

Moreover, Proposition 4 permits to exactly solve the auxiliary problem by directly searching for cliques in G. Algorithm 2 presents the steps to compute the optimal solution to (4.9) from the intersection graph of hyperspheres $G = (N, E)$.

```
1 Algorithm 2
2 while G is not empty do
3 |   Find a vertex n_i with smallest degree in G;
4 |   Consider G^i = (N^i, E^i) the subgraph composed by n_i and its adjacent
  |   vertices;
5 |   Solve (4.9) for variables v_ℓ such that n_ℓ ∈ G^i;
6 |   Save the clique obtained if it is the best found so far;
7 |   Remove n_i and its adjacent edges from G;
8 end
9 return best clique found
```

Clearly, Algorithm 2 is more efficient for sparse graphs G than for dense ones as subproblems (4.9) solved in (c) tend to have less variables Indeed, the sparsity of G depends on the dual values λ, which tends to decrease with the number of clusters. This is due to the fact that when k is large, entities are likely to be close to their second-closest centroids in the optimal solution. Consequently, a second copy of an entity has little impact on the objective function value which means that the values λ of the dual variables are small.

4.4 Computational results

Computational experiments were performed on a AMD 64 bits platform with a 2 GHz clock and 10 Gigabytes of RAM memory. The algorithms were implemented in C++ and

compiled by gcc 3.4. Unconstrained 0-1 quadratic programs are solved by a specialized algorithm proposed in [54] which was observed to perform better than CPLEX 10.1 for that purpose. Eleven real-world data sets were used in our numerical experiments. They are briefly listed in Table 4.1 together with references to where more information about them can be found.

Table 4.1: List of data sets

Data sets	n	s
Ruspini's data [103]	75	2
Grötschel and Holland's 202 cities coordinates [47]	202	2
Grötschel and Holland's 666 cities coordinates [47]	666	2
Reinelt's hole-drilling data [102]	1060	2
Padberg and Rinaldi's hole-drilling data [94]	2392	2
Fisher's Iris [5]	150	4
Glass identification [5]	214	9
Body measurements[1] [60]	507	5
Telugu Indian vowel sounds [95]	871	3
Concrete compressive strength [5, 129]	1030	8
Image segmentation [5]	2310	19

[1] the attributes used are: weight, height, chest girth, waist girth and hip girth

For all experiments reported here, initial upper bound solutions are obtained by j-means [55]. They are used to add initial cuts to (4.2), as well as to estimate initial dual bounds which may be adjusted throughout execution if necessary. Lower and upper bounds for dual variables λ can be estimated from any given upper bound solution UB. For each dual variable λ_i, for $i = 1, \ldots, n$, a lower bound value lb_i is estimated by calculating the cost variation in UB caused by omitting entity o_i from its associated cluster in UB. The estimation of an upper bound value ub_i is done by calculating the cost variation in UB caused by assigning entity o_i to its second-closest centroid. These estimations are exact whenever UB is the optimal solution and no integrality gap exists (cf. [28]).

4.4.1 Results in the plane

In this subsection we compare the column generation of [28], denoted accpm-vns-qp, with two improved ones, i.e., (i) accpm-a1 which uses Algorithm 1 to exactly solve all auxiliary problems, and (ii) accpm-vns-a1 which uses heuristic VNS to provide approximate solutions

to auxiliary problems until optimality must be proved by Algorithm 1. The VNS heuristic used by algorithms accpm-vns-qp and accpm-vns-a1 is set to run for one iteration, i.e., it reaches the largest neighborhood only once. Note that it is not worthwhile to use VNS for many iterations since Algorithm 1 is polynomially bounded in $O(n^3)$.

The results are also compared to those of two other methods proposed in the literature, i.e., the repetitive branch-and-bound algorithm (rbba) of Brusco [12] and the best branch-and-cut SDP-based algorithm (bb-sdp) of [2].

Tables 2–7 show results for data sets in the plane. They present in the first column the number k of clusters, and optimal solution values f_{opt} are reported in the second column. The values associated to each algorithm refer to their respective CPU times (in seconds) spent on solving exactly the instance. Finally, a last column is included to present gap values between upper and lower bounds obtained at the root node, denoted UB^0 and LB^0 respectively, which are calculated as $(UB^0 - LB^0)/LB^0$. The letter 'i' indicates that no initial gap exists, i.e., the problem is already solved by the accpm algorithms at the root node, without branching. Otherwise, the number of branch-and-bound nodes is given in parenthesis.

Table 4.2 shows that all methods perform well or very well for Ruspini's data set with $n = 75$ entities. Algorithm rbba is particularly efficient for small values of k, while its performance quickly deteriorates as k increases. This is due to the fact that the number of branches in RBBA is $O(k^n)$. For $k \geq 5$, algorithms accpm-a1 and accpm-vns-a1 are always faster than the other methods.

Table 4.2: Results for Ruspini data set with 75 entities

k	f_{opt}	rbba	bb-sdp	accpm-vns-qp	accpm-vns-a1	accpm-a1	$gap(\%)$
2	0.893378e+05	0.01	3.56	0.55	0.24	0.39	i
3	0.510634e+05	0.28	8.34	0.57	0.20	0.42	i
4	0.128810e+05	0.01	0.48	0.53	0.14	0.07	i
5	0.101267e+05	0.17	0.57	0.59	0.16	0.10	i
6	0.857541e+04	21.97	1.03	0.91	0.27	0.18	i
7	0.712620e+04	181.90	0.98	1.12	0.28	0.18	i
8	0.614964e+04	2921.93	7.27	1.04	0.44	0.23	0.01(3)
9	0.518165e+04	> 1h	2.87	1.20	0.30	0.17	i
10	0.444628e+04	> 1h	2.39	1.17	0.26	0.12	i

Table 4.3 presents results obtained in less than 12 hours of CPU time for the Grötschel and Holand's data set with $n = 202$. Algorithm rbba is not able to solve even the problem with $k = 2$ clusters in less than 12 hours. So, we do not refer to its results in the subsequent tables. As empirically observed in [2], the performance of algorithm bb-sdp improves as k increases, in contrast with algorithm rbba. It is unable to solve problems for $k \leq 8$ in less than 12 hours. It also appears that it is better to approximately solve the auxiliary problems by VNS up to $k = 15$. For $k \geq 20$, the sparsity of the discs in the plane, which is implied by small dual values, makes Algorithm 1 more efficient than VNS to solve the auxiliary problems. So, algorithm accpm-a1 performs better than accpm-vns-a1 for these values of k. The sparsity effect also appears to be advantageous to the unconstrained 0-1 quadratic programming solver since the algorithm is faster for instances with larger number of clusters.

Table 4.3: Results for Grötschel and Holland's data set with 202 entities

k	f_{opt}	bb-sdp	accpm-vns-qp	accpm-vns-a1	accpm-a1	$gap(\%)$
2	0.234374e+05	> 12h	> 12h	19.85	61.54	i
3	0.153274e+05	> 12h	> 12h	19.64	79.65	i
4	0.114556e+05	> 12h	> 12h	21.87	82.89	i
5	0.889490e+04	> 12h	> 12h	15.62	63.95	i
6	0.676488e+04	> 12h	> 12h	26.33	69.97	i
7	0.581757e+04	> 12h	> 12h	33.79	85.56	i
8	0.500610e+04	> 12h	1526.63	48.80	65.56	i
9	0.437619e+04	48885.38	1334.06	33.79	47.87	i
10	0.379249e+04	23680.84	496.85	16.42	35.84	i
15	0.232008e+04	39756.23	41.49	18.43	30.71	i
20	0.152351e+04	3839.77	59.90	18.87	17.75	i
25	0.108556e+04	1915.05	33.95	18.24	11.05	i
30	0.799311e+03	1060.77	27.03	17.78	5.96	i

Regarding the results for the Grötschel and Holland's data set with $n = 666$ entities presented in Table 4.4, a CPU time limit of 1 day was established, which proved not to be enough for algorithms bb-sdp and accpm-vns-qp. Therefore, the results of these algorithms will not be reported from now on since they demand too much time to exactly solve instances of the largest data sets. Table 4.4 shows that algorithm accpm-a1 is faster than accpm-vns-a1 from $k \geq 4$.

Table 4.4: Results for Grötschel and Holland's data set with 666 entities

k	f_{opt}	accpm-vns-a1	accpm-a1	$gap(\%)$
2	$1.754012e+06$	1179.68	2723.48	i
3	$0.772707e+06$	1525.10	1758.92	i
4	$0.613995e+06$	3585.39	3290.45	i
5	$0.485088e+06$	3277.55	2410.83	i
6	$0.382676e+06$	3162.39	1909.23	i
7	$0.323283e+06$	3082.65	1909.49	i
8	$0.285925e+06$	4314.00	2469.90	i
9	$0.250989e+06$	4134.31	2162.06	i
10	$0.224183e+06$	3131.41	2108.38	i
20	$0.106276e+06$	10504.30	4819.84	0.00(3)
50	$0.351795e+05$	6161.84	447.48	i

The results in Table 4.5 show that accpm-a1 is faster than accpm-vns-a1 from $k \geq 7$. The algorithms appear to be scalable for larger values of k due to increasing sparsity of discs in the auxiliary problems. It is worthwhile to mention that some of the state-of-art heuristics proposed in [15, 55, 72, 73, 92, 115] did not report the optimal solutions found here for the Reinelt's drilling data set with $n = 1060$ entities and $k = 120, 150$. To the best of our knowledge, this is the first time that such solutions are reported in the literature.

Table 4.5: Results for Reinelt's drilling data set with 1060 entities

k	f_{opt}	accpm-vns-a1	accpm-a1	$gap(\%)$
2	$0.983195e+10$	7417.92	13657.78	i
3	$0.670578e+10$	17897.19	30016.73	i
4	$0.475197e+10$	13429.61	26921.27	i
5	$0.379100e+10$	15966.45	26049.23	i
6	$0.317701e+10$	15128.71	19970.91	i
7	$0.270386e+10$	39966.71	22289.93	i
8	$0.226315e+10$	24863.21	19942.57	i
9	$0.198104e+10$	21810.90	16438.40	i
10	$0.175484e+10$	349793.97	56625.07	0.01(3)
100	$0.963178e+08$	17017.10	496.85	i
110	$0.848396e+08$	14930.74	373.54	i
120	$0.755366e+08$	8165.25	393.21	i
130	$0.675542e+08$	8296.29	301.77	i
140	$0.611196e+08$	13886.32	299.75	i
150	$0.559082e+08$	4998.90	292.37	i
200	$0.361572e+08$	4234.54	229.74	i

Finally, algorithms accpm-vns-a1 and accpm-a1 were tested for Padberg and Rinaldi's data set with $n = 2392$ entities. From the geometric interpretation of the auxiliary problem corroborated by the results presented in the previous tables, we concluded that algorithm accpm-vns-a1 is the most efficient one for instances with a small number of clusters. Therefore, Table 4.6 presents only the results of accpm-vns-a1 for $2 \leq k \leq 10$. Note that these instances require a lot of computing time to be exactly solved (e.g. more than one week was necessary to solve the instance with $k = 9$).

Table 4.6: Results for Padberg and Rinaldi's data set with 2392 entities for $2 \leq k \leq 10$

k	f_{opt}	accpm-vns-a1	$gap(\%)$
2	$0.296723e+11$	180581.30	i
3	$0.212012e+11$	393564.16	i
4	$0.141184e+11$	298724.00	i
5	$0.115842e+11$	416314.64	i
6	$0.948900e+10$	218403.68	i
7	$0.818180e+10$	565361.77	i
8	$0.701338e+10$	482525.96	i
9	$0.614600e+10$	663595.15	i
10	$0.532491e+10$	478613.29	i

Table 4.7 presents the results obtained by algorithm accpm-a1 for the Padberg and Rinaldi's data set with $n = 2392$ entities using large values of k. For these instances, approximately 3-5% of the total computing time is spent solving the auxiliary problems, revealing that at this point ($\approx$ 2000 entities) the resolution of the restricted master problem by ACCPM is the most expensive step of the algorithm. Note that the largest CPU time reported in Table 4.7 is of approximately 29 hours for $k = 150$.

Table 4.7: Results for Padberg and Rinaldi's data set with 2392 entities for large values of k

k	f_{opt}	accpm-a1	$gap(\%)$
100	$0.404498e+09$	21528.56	i
150	$0.245685e+09$	105852.43	0.01(7)
200	$0.175431e+09$	18918.16	i
250	$0.132352e+09$	16460.46	i
300	$0.101568e+09$	35939.04	0.00(3)
350	$0.804783e+08$	8131.32	i
400	$0.657989e+08$	9336.05	i

4.4.2 Results in general Euclidean space

Two other algorithms were implemented in order to check the computational effect of the geometric arguments in general Euclidean space. They are: (i) accpm-vns-qp+, which is similar to accpm-vns-qp proposed in [28] except that some coefficients are modified to arbitrarily large values in the auxiliary problem following the geometrical arguments presented in Section 4.3, and (ii) accpm-vns-a2, which uses one iteration of VNS to obtain approximate solutions to auxiliary problems until optimality is certified by Algorithm 2.

Table 4.8 shows CPU times spent by the different algorithms in order to solve exactly instances of the Fisher's Iris data set with $n = 150$ entities in $s = 4$ dimensions. The results shows that again rbba is very efficient for small number of clusters, though its performance deteriorates very fast as k increases. Moreover, except for $k = 2$, algorithm accpm-vns-qp+ performs better than accpm-vns-qp. Finally, since the auxiliary problems are small for this data set ($n = 150$), Algorithm 2 is not very advantageous for solving them. In fact, for the instance with $k = 2$, algorithm accpm-vns-a2 is much less efficient than the others.

Table 4.8: Results for Fisher's Iris with 150 entities in 4 dimensions

k	f_{opt}	rbba	bb-sdp	accpm-vns-qp	accpm-vns-qp+	accpm-vns-a2	$gap(\%)$
2	0.152348e+03	0.05	169.44	251.04	486.62	1958.06	i
3	0.788514e+02	2.10	283.24	83.09	19.88	19.55	i
4	0.572285e+02	136.29	240.19	138.85	32.71	17.22	i
5	0.464462e+02	1699.75	145.54	42.00	6.52	8.80	i
6	0.390400e+02	> 12h	147.51	15.50	11.70	10.47	i
7	0.342982e+02	> 12h	742.83	10.50	7.83	6.65	i
8	0.299889e+02	> 12h	108.73	7.82	6.41	6.74	i
9	0.277861e+02	> 12h	70.04	6.44	6.11	7.48	i
10	0.25834e+02	> 12h	59.66	8.51	8.38	9.03	i

The results in Table 4.9 give CPU times spent on solving exactly instances of the Glass identification data set with $n = 214$ in $s = 9$ dimensions. We notice that instances with $k \leq 10$ cannot be solved in less than 1 day of computation. In particular, algorithm rbba takes more than 1 day to solve even its most favorable case with $k = 2$. Therefore, the next tables will not refer to its results. Likewise, results of algorithm bb-sdp will not be reported in the following tables since it is clearly outperformed by ACCPM algorithms.

Table 4.9: Results for the Glass identification data set with 214 entities in 9 dimensions

k	f_{opt}	bb-sdp	accpm-vns-qp	accpm-vns-qp+	accpm-vns-a2	$gap(\%)$
15	0.155766e+03	> 1 day	> 1 day	37714.82	7983.52	i
20	0.114646e+03	> 1 day	> 1 day	30065.43	13365.79	0.02(3)
25	0.842515e+02	> 1 day	> 1 day	24568.26	19011.65	0.00(3)
30	0.632478e+02	49831.18	269.36	52.80	39.50	i
35	0.492386e+02	25629.86	22.60	16.33	18.87	i
40	0.394983e+02	6272.84	27.87	16.85	18.32	i
45	0.320395e+02	17437.27	43.27	29.37	32.21	0.00(3)
50	0.267675e+02	10032.09	21.69	20.51	21.46	i

From the results on Table 4.9, algorithm accpm-vns-qp+ outperforms accpm-vns-qp in all tested instances. Since this is also true for the computational experiments on the other data sets, we will not report the results of accpm-vns-qp from now on. This fact confirms the benefits derived from the geometric interpretation of the auxiliary problem. Moreover, algorithm accpm-vns-a2 was more efficient than accpm-vns-qp+ for the instances with the most difficult auxiliary problems (i.e., $15 \leq k \leq 30$), showing that solving (4.9) by isolating cliques is a good strategy in these cases.

Taking into account the increasing computing times spent by VNS as the value of n increases, one may ask if it would not be better to solve exactly the auxiliary problems at each iteration of ACCPM. In order to answer this question, two other algorithms are considered for comparison in Tables 4.10, 4.11, 4.12. They differ only in the way that auxiliary problems are dealt with. While accpm-qp+ always uses Dinkelbach's algorithm to solve the auxiliary problems, accpm-a2 uses Algorithm 2 instead, i.e., using Dinkelbach's algorithm on each clique.

Table 4.10: Results for the Body measurements data set with 507 entities in 5 dimensions

k	f_{opt}	accpm-vns-qp+	accpm-qp+	accpm-vns-a2	accpm-a2	$gap(\%)$
30	0.195299e+05	79819.81	> 2 days	12433.74	> 2 days	0.00(3)
40	0.162318e+05	3981.92	25196.16	3954.62	13396.05	0.00(3)
50	0.139547e+05	26991.10	> 2 days	22945.66	67178.35	0.04(11)
60	0.121826e+05	2847.94	3284.43	2242.53	1860.72	0.00(3)
70	0.107869e+05	2606.16	2421.93	2534.06	1329.71	0.00(3)
80	0.964873e+04	5565.30	5026.03	6191.68	2705.14	0.01(5)

From Table 4.10, we notice that the algorithms that solve auxiliary problems by cliques (i.e., accpm-vns-a2 and accpm-a2) perform usually better than their counterparts that solve the auxiliary problems by considering the whole intersection graph of hyperspheres (accpm-vns-qp+ and accpm-qp+, respectively). In particular accpm-a2 is the best algorithm from $k \geq 60$. The same conclusions can be extended to Tables 4.11 and 4.12, except that for these larger data sets accpm-a2 is very often the best algorithm for the instances that can be exactly solved within a CPU time limit of 2 days.

Table 4.11: Results for the Telugu Indian vowel sounds data set with 871 entities in 3 dimensions

k	f_{opt}	accpm-vns-qp+	accpm-qp+	accpm-vns-a2	accpm-a2	$gap(\%)$
40	0.636653e+07	26059.64	83537.53	10232.80	8209.48	i
50	0.524020e+07	5070.60	14304.07	4314.11	2450.54	i
60	0.442262e+07	> 2 days	> 2 days	> 2 days	107905.07	0.10(21)
70	0.375286e+07	7439.66	8853.31	6524.48	1726.57	0.00(3)
80	0.324801e+07	2538.37	2320.29	2389.09	323.95	i
90	0.285069e+07	2227.94	1929.68	1980.14	282.73	i
100	0.251058e+07	5717.78	1606.62	5054.39	195.53	0.00(3)

Table 4.12: Results for the Concrete compressive strength data set with 1030 entities in 9 dimensions

k	f_{opt}	accpm-vns-qp+	accpm-qp+	accpm-vns-a2	accpm-a2	$gap(\%)$
60	0.288107e+07	> 2 days	> 2 days	93018.98	114291.96	i
70	0.247893e+07	32524.80	33373.40	8671.61	2825.70	i
80	0.215791e+07	5622.55	7538.82	5717.15	1405.62	i
90	0.189364e+07	> 2 days	> 2 days	64518.66	88849.78	0.01(7)
100	0.168778e+07	3330.97	3530.60	3773.60	380.75	i
110	0.151334e+07	2950.36	2465.40	2714.67	301.46	i
120	0.136737e+07	3883.50	2754.42	3835.67	310.24	i

We have still obtained results for a larger data set consisting of 2310 entities in 19 dimensions taken from [5] by means of algorithm accpm-a2. The results presented in Table 4.13 shows that instances with a ratio of $n/k \approx 10$ can be exactly solved in a reasonable amount of time by the column generation algorithm, which is a new record for benchmark data sets of this magnitude ($n = 2310$) and this dimension ($s = 19$).

Table 4.13: Results for the Image segmentation data set with 2310 entities in 19 dimensions

k	f_{opt}	accpm-a2	$gap(\%)$
230	0.463938e+06	16717.39	i
250	0.421018e+06	10864.30	i
300	0.338072e+06	25693.02	0.00(3)
350	0.276957e+06	7036.09	i
400	0.230310e+06	99554.55	0.00(11)
450	0.195101e+06	66655.32	0.00(7)
500	0.157153e+06	36772.86	0.01(5)

4.4.3 Comparison of approaches in the plane and in general Euclidean space

Finally, we compare our approach in the plane with that tailored for problems in general Euclidean space. Since the superiority of the approach in the plane for a small number of clusters is obvious, we decided to focus this comparison on instances with large values of k. The best algorithm regarding each one of the approaches is then selected for comparison, i.e., accpm-a1 from the class of algorithms which tackles exclusively instances in the plane and accpm-a2 from the class of algorithm dealing with instances in general Euclidean space.

In the graph of Figure 4.2, we plot the percentage of CPU time spent by algorithm accpm-a2 in excess of the CPU time spent by algorithm accpm-a1 when solving different instances of the Reinelt's planar data set with 1060 entities.

From the graph, we notice that accpm-a1 tends to be increasingly better than accpm-a2 as k augments, though the computing times are smaller for instances with a large number of clusters.

4.5 Conclusions

MSSC is a central problem in cluster analysis. Numerous heuristics as well as a variety of exact algorithms have been proposed for its solution. These last ones include the column generation algorithm of du Merle et al. [28] which is the point of departure of this chapter. The bottleneck step of that algorithm appeared within the auxiliary problem and was

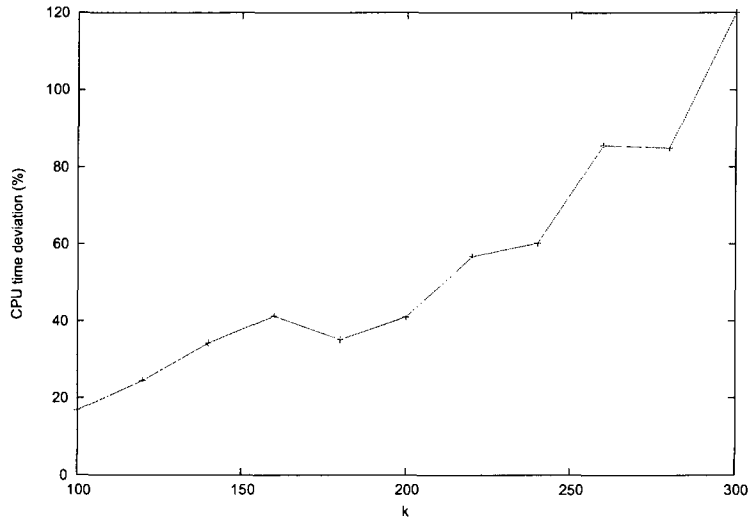

Figure 4.2: Percentage of CPU time spent by algorithm accpm-a2 in excess of the CPU time spent by algorithm accpm-a1 for instances of the Reinelt's planar data set with 1060 entities

the solution of unconstrained 0-1 quadratic programs. Based on geometric reasoning, a different and more efficient way of solving this auxiliary problem is proposed in this chapter. It exploits systematically the property that far apart points will not belong to the same cluster. This property is made precise by proving that it is the case when their mutual distance exceeds the sum of square roots of the corresponding dual variables at the current iteration. Geometrically, solutions in the plane correspond to a quadratic number of regions which are determined by a $O(n^2)$ algorithm. This leads to solution of the auxiliary problem in $O(n^3)$, at least when there is little branching in the master problem which appears to be most often the case. Finding all similar regions in a higher dimensional space would be time consuming. However, the way to solve the auxiliary problem can still be improved by replacing by a large value coefficients in the unconstrained 0-1 quadratic programs corresponding to far apart entities. This has led to substantially increase the size of instances solved exactly. In the plane, instances with n up to 2392 entities and $k \geq 2$ have been solved exactly, most of them for the first time. The increase in the size of the instances exactly solved has thus been multiplied by more than 10. In general Euclidean space problems with up to $n = 2310$ and $k = 230$ clusters in 19 dimensions have been

solved. However, it appears that the number of entities per cluster should be small, i.e. n/k roughly equal to 10, in order to solve such instances in reasonable time.

CONCLUSION

MSSC consists in, given a set of n entities associated with points in s-dimensional Euclidean space, partitioning this set into clusters in such a way that the sum of squared distances from each entity to the centroid of its cluster is minimum. This much studied problem is a basic one in cluster analysis and has applications in numerous and diverse fields.

Many heuristics algorithms for MSSC have been and continue to be regularly proposed. Exact solution methods are rare but a variety of approaches have been explored.

The aim of this thesis is twofold: on the one hand to assess the state of the art concerning exact solution methods for MSSC and on the other hand to improve as much as possible these methods.

A first chapter concerns complexity analysis of MSSC, a topic in which there seems to have been much confusion. We note indeed that several dozen paper have made incorrect or unjustified statements about NP-hardness of MSSC, usually confusing it with some other clustering problem. Recently, a proof was proposed by Drineas et al. in *Machine Learning*, 2004 [27]. Unfortunately, as shown in that chapter, this proof is not correct. We next provide, with A. Deshpande and P. Popat, a new proof of NP-hardness of MSSC in general Euclidean space, exploiting a reduction from the densest cut problem.

The remaining three chapters consider several of the main approaches to exact solution of MSSC.

In chapter 2 we study a recent paper of Sherali and Desai in *Journal of Global Optimization*, 2005 [108]. In this paper, they apply the reformulation-linearization technique (RLT) of Sherali and Adams [107] in order to get precise bounds for a branch-and-bound algorithm. These authors claim to have solved large instances, i.e., problems with up $n = 1,000$ entities in $s = 8$ dimensions. We attempted to reproduce these results without success. To that effect we wrote an implementation of the Sherali and Desai algorithm following as closely as possible the description given in their paper. Moreover, we used CPLEX to solve their basic model directly. We then considered small instances of MSSC

obtained by selecting a subset of the 150 Fisher's Iris data [36]. We observed that computing times spent for solution with our implementation were already large (i.e., more than 6 hours on a Pentium IV 2 GHz) for a data set with only 20 entities. We discussed by email with both Sherali and Desai possible causes for this vast discrepancy between their results and ours. *"Unfortunately, he* [Jitamitra Desai] *appears to have deleted his codes and data sets"* [106]. The most likely explanation seems to be that the test problems used by Sherali and Desai were very easy to solve (i.e., that the clusters were very well separated).

In chapter 3, we studied the work of Peng and Xia [98] on a 0-1 semidefinite programming (0-1 SDP) reformulation of MSSC. Instead of directly applying SDP, these authors consider a formulation in which the constraint $Z \succeq 0$ is replaced by idempotency $Z = Z^2$ and symmetry $Z = Z^T$. In this model, all eigenvalues of Z are equal to either 0 or 1. On this basis, Xia and Peng provided a reformulation of MSSC. They then proved important properties of the relaxation obtained by relaxing the constraints $Z = Z^2$. Namely, they proved that the following inequalities would be satisfied in any solution of their formulation:

$$\begin{array}{lll} Z_{ij} \leq Z_{ii} & \forall i, j & \text{(pair inequalities)} \\ Z_{ij} + Z_{i\ell} \leq Z_{ii} + Z_{j\ell} & \forall i, j, \ell & \text{(triangular inequalities).} \end{array}$$

However, in view of the rapid increase in size of this set of constraints, Peng and Xia [98] only sketched an algorithm. We developed a branch-and-cut algorithm following those lines but adding only sets of violated constraints. Computational experiments showed that this algorithm was competitive with the previously best ones, i.e., the column generation algorithm of du Merle et al. [28] and repetitive branch-and-bound of Brusco [12]. Specifically, the 0-1 SDP branch-and-cut algorithm can solve instances with $n = 202$ and $k = 9$ in the plane in less than 12 hours.

Chapter 4 is devoted to the column generation approach of du Merle et al. [28] and its improvements. This algorithm exploits the ACCPM (Analytical Center Cutting Plane Method) of Goffin, Haurie and Vial [45] to solve the master problem. The auxiliary problem turns out to be a hyperbolic program in 0-1 variables which can be reduced to a sequence

of unconstrained 0-1 quadratic problems. These last ones are the bottleneck of the whole procedure. Despite some progress, it is still difficult to solve such problems with 100% dense matrices and more than 200 entities. Therefore, we propose a different approach to the solution of the auxiliary problem. Essentially, it makes precise and exploits systematically the property that two far apart entities cannot belong both to the same cluster. It is based on geometric arguments and an analogy with the 1-center Weber problem with maximum distances introduced by Drezner, Mehrez and Wesolowsky [26]. In the plane, the auxiliary problem consists in minimizing the sum for all entities of functions centered at each entity position p_i and equal to the squared distance from each p_i with, however, a limit of $\sqrt{\lambda_i}$ on the distance (where λ_i is the corresponding dual variable in the current solution of the master problem) after which the functions remain constant. Adapting and completing an enumerative algorithm of Drezner et al. [26] solves the auxiliary problem before branching in $O(n^3)$ time. If branching is needed, which appears to be very rarely the case, the classical branching rule of Ryan and Foster [104] can be used.

In higher dimensions, the enumeration would become too cumbersome but the basic property can still be exploited: a sufficient condition for two entities not to belong to the same cluster is used to replace coefficients in the unconstrained quadratic 0-1 problem by arbitrarily large values. Then, a branch-and-bound algorithm is applied within a vertex removal scheme. To this effect, a graph is constructed with nodes associated to the entities and edges associated to pairs of entities which do not have an arbitrarily large coefficient, i.e., for which the hyperspheres of radius $\sqrt{\lambda_i}$ and $\sqrt{\lambda_j}$ do intersect. Recursively, a vertex of minimum degree in this graph is selected and the subgraph induced by its closed neighborhood considered. The unconstrained 0-1 quadratic problem associated to this subgraph is solved and the optimal solution kept if it is better than the incumbent one.

Application of these new rules led to very substantial progress: indeed instances in the plane with up to $n = 2392$ entities and $k \geq 2$ clusters could be solved in (large but still) reasonable time. Moreover, instances in up to 19 dimensions and with up to $n = 2310$ entities could be solved exactly when there are many clusters.

To conclude, exact approaches to resolution of MSSC can be divided into three families:

1. those which solve small instances ($n \approx 25$), i.e., non-serial dynamic programming [119], concave programming [127] and RLT [108].

2. those which solve medium size instances ($n \approx 100 - 200$), i.e., repetitive branch-and-bound [12], 0-1 SDP-based branch-and-cut [2, 98], and column generation without geometric enhancements [28].

3. those which can solve fairly large instances ($n \approx 2000$), i.e., the improved column generation approach presented in this thesis.

Taking a larger view we can consider these results as a feasibility proof for exact solution of clustering problems by column generation. There are many criteria proposed in the literature to express homogeneity and/or separation of the clusters. A project for building a column generation package for clustering, involving several professors at GERAD as well as many students, is currently in the experimental stage. Clearly, success of the package on one or another criterion will depend largely on two factors: easy of resolution of the auxiliary problem and presence of a small or large gap. More algorithmic and computational work is needed here. Several likely candidates are recently proposed criteria in the data mining and physics communities: e.g. ratio cut [50], normalized cut [109], and modularity [17].

Also, while we have focused on an interior point based approach, i.e., ACCPM and column generation, stabilized linear programming [29] might still be a competitor particularly if combined with the recent work of [31, 32] on efficient treatment of degeneracy.

BIBLIOGRAPHY

[1] ALOISE, D., DESHPANDE, A., HANSEN, P., and POPAT, P. (2009). NP-hardness of Euclidean sum-of-squares clustering, *Machine Learning*, vol. 75, pp. 245–249.

[2] ALOISE, D. and HANSEN, P. (2008). A branch-and-cut SDP-based algorithm algorithm for minimum sum-of-squares clustering, *submitted to Pesquisa Operacional.*

[3] AN, L., BELGHITI, M., and TAO, P. (2007). A new efficient algorithm based on DC programming and DCA for clustering, *Journal of Global Optimization*, vol. 37, pp. 593–608.

[4] ARTHUR, D. and VASSILVITSKII, S. (2007). K-means++: the advantages of careful seeding, in *2007 ACM-SIAM Symposium on Discrete Algorithms (SODA'07).*

[5] ASUNCION, A. and NEWMAN, D. (2007). UCI machine learning repository, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[6] BAGIROV, A. (2008). Modified global k-means algorithm for minimum sum-of-squares clustering problems, *Pattern Recognition*, vol. 41, pp. 3192–3199.

[7] BAGIROV, A. and YEARWOORD, J. (2006). Hierarchical grouping to optimize an objective function, *European Journal of Operational Research*, vol. 170, pp. 578–596.

[8] BERINGER, J. and HÜLLERMEIER, E. (2006). Online clustering of parallel data streams, *Data & Knowledge Engineering*, vol. 58, pp. 180–204.

[9] BONAMI, P. and LEE, J. (June 2007). `BONMIN` user's manual, tech. rep., IBM Corporation.

[10] BOOLE, G. (1854). *An Investigation of the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities.* London: Walton and Maberley.

[11] BRÜCKER, P. (1978). On the complexity of clustering problems, *Lecture Notes in Economic and Mathematical Systems*, vol. 157, pp. 45–54.

[12] BRUSCO, M. (2006). A repetitive branch-and-bound procedure for minimum within-cluster sum of squares partitioning, *Psychometrika*, vol. 71, pp. 347–363.

[13] BRUSCO, M. and STEINLEY, D. (2007). A comparison of heuristics procedures for minimum within-cluster sums of squares partitioning, *Psychometrika*, vol. 72, pp. 583–600.

[14] CHEN, H.-L., CHUANG, K.-T., and CHEN, M.-S. (2005). Labeling unclustered categorical data into clusters based on the important attribute values, in *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05).*

[15] CHRISTOU, I. (2009). Exact method-based coordination of cluster ensembles, Athens Information Technology Technical Report 2055; Provisionally accepted for publication in IEEE Transactions in Pattern Analysis and Machine Intelligence.

[16] CILIBRASI, R., VAN IERSEL, L., KELK, S., and TROMP, J. (2005). On the complexity of several haplotyping problems, *Lecture Notes in Computer Science*, vol. 3692, pp. 128–139.

[17] CLAUSET, A., NEWMAN, M., and MOORE, C. (2004). Finding community structure in very large networks, *Physics Review E*, vol. 70.

[18] DASGUPTA, S. (17 January 2008). The hardness of k-means clustering, tech. rep. CS2008-0916, University of California.

[19] DELATTRE, M. and HANSEN, P. (1980). Bicriterion cluster analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 4, pp. 277–291.

[20] DESHPANDE, A. and POPAT, P. (22 January 2008). Email sent to Ravi Kannan et al. and transmitted by Nina Mishra to the first and third authors.

[21] DHILLON, I. and MODHA, D. (2002). A data-clustering algorithm on distributed memory multiprocessors, *Lecture Notes in Artificial Intelligence*, vol. 1759, pp. 245–260.

[22] DIEHR, G. (1985). Evaluation of a branch and bound algorithm for clustering, *SIAM Journal Scientific and Statistical Computing*, vol. 6, pp. 268–284.

[23] DIEHR, G. (April 1973). Minimum variance partitions and mathematical programming. Paper presented at the National Meetings of the Classification Society, Atlanta, Georgia.

[24] DINKELBACH, W. (1967). On nonlinear fractional programming, *Management Science*, vol. 13, pp. 492–498.

[25] DOMINGO-FERRER, J. and MATEO-SANZ, J. M. (2002). Practical data-oriented microaggregation for statistical disclosure control, *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, pp. 189–201.

[26] DREZNER, Z., MEHREZ, A., and WESOLOWSKY, G. (1991). The facility location problem with limited distances, *Transportation Science*, vol. 25, pp. 183–187.

[27] DRINEAS, P., FRIEZE, A., KANNAN, R., VEMPALA, S., and VINAY, V. (2004). Clustering large graphs via the singular value decomposition, *Machine Learning*, vol. 56, pp. 9–33.

[28] DU MERLE, O., HANSEN, P., JAUMARD, B., and MLADENOVIĆ, N. (2000). An interior point algorithm for minimum sum-of-squares clustering, *SIAM Journal Scientific Computing*, vol. 21, pp. 1485–1505.

[29] DU MERLE, O., VILLENEUVE, D., DESROSIERS, J., and HANSEN, P. (1999). Stabilized column generation, *Discrete Mathematics*, vol. 194, pp. 229–237.

[30] EDWARDS, A. and CAVALLI-SFORZA, L. (1965). A method for cluster analysis, *Biometrics*, vol. 21, pp. 362–375.

[31] ELHALLAOUI, I., METRANE, A., SOUMIS, F., and DESAULNIERS, G. (2008). Multiphase dynamic constraint aggregation for set partitioning type problems, *to appear in Mathematical Programming.*

[32] ELHALLAOUI, I., VILLENEUVE, D., SOUMIS, F., and DESAULNIERS, G. (2005). Dynamic aggregation of set-partitioning constraints in column generation, *Operations Research*, vol. 53, pp. 632–645.

[33] ELHEDHLI, S. and GOFFIN, J.-L. (2004). The integration of an interior-point cutting plane method within a branch-and-price algorithm, *Mathematical Programming*, vol. 100, pp. 267–294.

[34] FASULO, D. (1999). An analysis of recent work on clustering algorithms, tech. rep. UW-CSE-01-03-02, University of Washington.

[35] FISCHER, B., ROTH, V., and BUHMANN, J. (2004). Clustering with the connectivity kernel, *Advances in Neural Information Processing Systems*, vol. 16.

[36] FISHER, R. (1936). The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, vol. VII, pp. 179–188.

[37] FLOREK, K., LUKASZEWICZ, J., PERKAL, H., STEINHAUS, H., and ZUBRZYCKI, S. (1951). Sur la liaison et la division des points d'un emsemble fini, *Colloquium Mathematicum*, vol. 2, pp. 282–285.

[38] FORGY, E.W. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications, *Biometrics*, vol. 21, pp. 768.

[39] FORTET, R. (1959). L'algèbre de boole et ses applications en recherche opérationnelle, *Cahiers du Centre d'Études de Recherche Opérationnelle*, vol. 1, pp. 5–36.

[40] FRADKIN, D., MUCHNIK, I., and STRELTSOV, S. (2003). Image compression in real-time multiprocessor systems using divisive k-means clustering, in *International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMA'03)*, pp. 506–511.

[41] FRÄNTI, P. and KIVIJÄRVI, J. (2000). Randomised local search algorithm for the clustering problem, *Pattern Analysis & Applications*, vol. 3, pp. 358–369.

[42] FRÄNTI, P., VIRMAJOKI, O., and KAUKORANTA, T. (2002). Branch-and-bound technique for solving optimal clustering, in *International Conference on Pattern Recognition (ICPR'02)*, pp. 232–235.

[43] GAREY, M. and JOHNSON, D. (1979). *Computers and Intractability.* New York: W.H. Freeman and Company.

[44] GAREY, M., JOHNSON, D., and WITSENHAUSEN, H. (1982). The complexity of the generalized lloyd-max problem, *IEEE Transactions on Information Theory*, vol. IT-28, pp. 255–256.

[45] GOFFIN, J.-L., HAURIE, A., and VIAL, J.-P. (1992). Decomposition and nondifferentiable optimization with the projective algorithm, *Management Science*, vol. 38, pp. 284–302.

[46] GRAHAM, R. (1972). An efficient algorithm for determining the convex hull of a finite point set, *Information Processing Letters*, vol. 1, pp. 132–133.

[47] GRÖTSCHEL, M. and HOLLAND, O. (1991). Solution of large-scale symmetric traveling salesman problems, *Mathematical Programming*, vol. 51, pp. 141–202. Data sets available at [http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp].

[48] GRÖTSCHEL, M. and WAKABAYASHI, Y. (1989). A cutting plane algorithm for a clustering problem, *Mathematical Programming*, vol. 45, pp. 59–96.

[49] GÜNGÖR, Z. and ÜNLER, A. (2007). k-harmonic means data clustering with simulated annealing heuristic, *Applied Mathematics and Computation*, vol. 184, pp. 199–209.

[50] HAGEN, L. and KAHNG, A. (1992). New spectral methods for ratio cut partitioning and clustering, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, pp. 1074–1085.

[51] HAIR, J., ANDERSON, R., TATHAM, R., and BLACK, W. (1998). *Multivariate Data Analysis*. New York: Prentice-Hall.

[52] HANSEN, P. and DELATTRE, M. (1978). Complete-link cluster analysis by graph coloring, *Journal of the American Statistical Association*, vol. 73, pp. 397–403.

[53] HANSEN, P. and JAUMARD, B. (1997). Cluster analysis and mathematical programming, *Mathematical Programming*, vol. 79, pp. 191–215.

[54] HANSEN, P., JAUMARD, B., and MEYER, C. (2000). A simple enumerative algorithm for unconstrained $0-1$ quadratic programming, Cahier du GERAD G-2000-59.

[55] HANSEN, P. and MLADENOVIĆ, N. (2001). J-means: a new local search heuristic for minimum sum of squares clustering, *Pattern Recognition*, vol. 34, pp. 405–413.

[56] HANSEN, P. and MLADENOVIĆ, N. (2001). Variable neighborhood search: principles and applications, *European Journal of Operational Research*, vol. 130, pp. 449–467.

[57] HANSEN, P., MLADENOVIĆ, N., and PÉREZ, J. (2008). Variable neighborhood search: methods and applications, *to appear in 4OR*.

[58] HANSEN, P., NEGAI, E., CHEUNG, B., and MLADENOVIĆ, N. (2005). Analysis of global k-means, an incremental heuristic for minimum sum-of-squares clustering, *Journal of Classification*, vol. 22, pp. 287–310.

[59] HARTIGAN, J. (1975). *Clustering Algorithms.* New York, Wiley.

[60] HEINZ, G., PETERSON, L., JOHNSON, R., and KERK, C. (2003). Exploring relationships in body dimensions, *Journal of Statistics Education*, vol. 11. Data set available at [www.amstat.org/publications/jse/v11n2/datasets.heinz.html].

[61] INABA, M., KATOH, N., and IMAI, H. (1994). Applications of weighted voronoi diagrams and randomization to variance-based k-clustering, in *Proceedings of the 10th ACM Symposium on Computational Geometry*, pp. 332–339.

[62] JAIN, A., MURTY, M., and FLYNN, P. (1999). Data clustering: A review, *ACM Computing Surveys*, vol. 31, pp. 264–323.

[63] JENSEN, R. (1969). A dynamic programming algorithm for cluster analysis, *Operations Research*, vol. 17, pp. 1034–1057.

[64] JUNG, Y., PARK, H., DU, D.-Z., and DRAKE, B. (2003), A decision criterion for the optimal number of clusters in hierarchical clustering, *Journal of Global Optimization*, vol. 25, pp. 91–111.

[65] KANADE, G., NIMBHORKAR, P., and VARADARAJAN, K. (manuscript of 14 February 2008). On the NP-hardness of the 2-means problem.

[66] KARIV, O. and HAKIMI, S. (1969). An algorithmic approach to network location problems; part 2. the p-medians, *SIAM Journal on Applied Mathematics*, vol. 37, pp. 539–560.

[67] KELLEY, J. (1960). The cutting plane method for solving convex programs, *J. SIAM*, vol. 8, pp. 703–712.

[68] KLEIN, G., ARONSON, J.E. (1991). Optmal clustering: A model and method, *Naval Research Logistics*, vol. 38, pp. 447–461.

[69] KOGAN, J. (2006). *Introduction to Clustering Large and High-Dimensional Data.* New York: Cambridge University Press.

[70] KOONTZ, W., NARENDRA, P., and FUKUNAGA, K. (1975). A branch and bound clustering algorithm, *IEEE Trans. Comput.*, vol. C-24, pp. 908–915.

[71] LABBÉ, M., PETTERS, D., and THISSE, J. (1995). Location on networks, in *Network Routing* (BALL, M., MAGNANTI, T., MONMA, C., and NEMHAUSER, G., eds.), pp. 551–624, North-Holland.

[72] LASZLO, M. and MUKHERJEE, S. (2006). A genetic algorithm using hyper-quadtrees for low-dimensional k-means clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 533–543.

[73] LASZLO, M. and MUKHERJEE, S. (2007). A genetic algorithm that exchanges neighboring centers for k-means clustering, *Pattern Recognition Letters*, vol. 36, pp. 451–461.

[74] LEYFFER, S. (1999). User manual for MINLP_BB, tech. rep., University of Dundee, UK.

[75] LI, Y. and CHUNG, S. (2007). Parallel bisecting k-means with prediction clustering algorithm, *The Journal of Supercomputing*, vol. 39, pp. 19–37.

[76] LIBERTI, L. (2009). Reformulations in mathematical programming: Definitions and systematics, *RAIRO-RO*, vol. 43, no. 1, pp. 55–86.

[77] LIKAS, A., VLASSIS, N., and VERBEEK, J. (2003). The global k-means clustering algorithm, *Pattern Recognition*, vol. 36, pp. 451–461.

[78] LISSER, A. and RENDL, F. (2003). Graph partitioning using linear and semidefinite programming, *Mathematical Programming*, vol. Series B 95, pp. 91–101.

[79] MACQUEEN, J. (1967). Some methods for classification and analysis of multivariate observations, in *Proceedings of* 5^{th} *Berkeley Symposium on Mathematical Statistics and Probability*, vol. 2, (Berkeley, CA), pp. 281–297.

[80] MAHAJAN, M., NIMBHORKAR, P., and VARADARAJAN, K. (2009). The planar k-means problem is NP-hard, *Lecture Notes in Computer Science*, vol. 5431, pp. 274–285.

[81] MARZOUK, Y. and GHONIEM, A. (2005). k-means clustering for optimal partitioning and dynamic load balancing of parallel hierarchical n-body simulations, *Journal of Computational Physics*, vol. 207, pp. 493–528.

[82] MATULA, D. and SHAHROKHI, F. (1990). Sparsest cuts and bottlenecks in graphs, *Discrete Applied Mathematics*, vol. 27, pp. 113–123.

[83] MEILĂ, M. (2006). The uniqueness of a good optimum for k-means, *ACM International Conference Proceeding Series*, vol. 148, pp. 625–632.

[84] MERZ, P. (2003). Analysis of gene expression profiles: an application of memetic algorithms to the minimum sum-of-squares clustering problem, *Biosystems*, vol. 72, pp. 99–109.

[85] MERZ, P. (2003). An iterated local search for minimum sum-of-squares clustering, *Lecture Notes in Computer Science*, vol. 2810, pp. 286–296.

[86] MERZ, P. and ZELL, A. (2002). Clustering gene expression profiles with memetic algorithms, *Lecture Notes in Computer Science*, vol. 2439, pp. 811–820.

[87] MIRKIN, B. (1996). *Mathematical Classification and Clustering.* Dordrecht, The Netherlands: Kluwer.

[88] MIRKIN, B. (2005). *Clustering for Data Mining: A Data Recovery Approach.* Boca Raton: Chapman and Hall/CRC.

[89] MLADENOVIĆ, N. and HANSEN, P. (1997). Variable neighborhood search, *Computers and Operations Research*, vol. 24, pp. 1097–1100.

[90] NIU, K., ZHANG, S., and CHEN, J. (2006). An initializing cluster centers algorithm based on pointer ring, in *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06).*

[91] OSTROVSKY, R., RABANI, Y., SCHULMAN, L., and SWAMY, C. (2006). The effectiveness of Lloyd-type methods for the k-means problem, in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06).*

[92] PACHECO, J. (2005). A scatter search approach for the minimum sum-of-squares clustering problem, *Computers & Operations Research*, vol. 32, pp. 1325–1335.

[93] PACHECO, J. and VALENCIA, O. (2003). Design of hybrids for the minimum sum-of-squares clustering problem, *Computational Statistics & Data Analysis*, vol. 43, pp. 235–248.

[94] PADBERG, M. and RINALDI, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, *SIAM Review*, vol. 33, pp. 60–100. Data set available at [http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp].

[95] PAL, S. and MAJUMDER, D. (1977). Fuzzy sets and decision making approaches in vowel and speaker recognition, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, pp. 625–629. Data set available at [http://www.isical.ac.in/ sushmita/patterns/vowel.dat].

[96] PATERLINI, S. and KRINK, T. (2006). Differential evolution and particle swarm optimisation in partitional clustering, *Computational Statistics and Data Analysis*, vol. 50, pp. 1220–1247.

[97] PENG, J. and WEI, Y. (2007). Approximating k-means-type clustering via semidefinite programming, *SIAM Journal on Optimization*, vol. 18, pp. 186–205.

[98] PENG, J. and XIA, Y. (2005). A new theoretical framework for k-means-type clustering, *Studies in Fuzziness and Soft Computing*, vol. 180, pp. 79–96.

[99] PETROVIC, S. and ÁLVAREZ, G. (2003). A method for clustering web attacks using edit distance, *CoRR*, vol. cs.IR/0304007.

[100] PLASTRIA, F. (2002). Formulating logical implications in combinatorial optimisation, *European Journal of Operational Research*, vol. 140, pp. 338–353.

[101] RAMOS, V. and MUGE, F. (2000). Map segmentation for colour cube genetic k-mean clustering, *Lecture Notes in Computer Science*, vol. 1923, pp. 319–323.

[102] REINELT, G. (1991). TSPLIB – a traveling salesman library, *ORSA Journal on Computing*, vol. 3, pp. 319–350. [www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95].

[103] RUSPINI, E. (1970). Numerical method for fuzzy clustering, *Information Sciences*, vol. 2, pp. 319–350.

[104] RYAN, D. and FOSTER, B. (1981). An integer programming approach to scheduling, in *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling* (WREN, A., ed.), pp. 269–280, North-Holland.

[105] SHERALI, H. (11 March 2008). Personal correspondance sent to Pierre Hansen and Daniel aloise.

[106] SHERALI, H. (19 November 2007). Personal correspondance sent to Pierre Hansen and Daniel Aloise.

[107] SHERALI, H. and ADAMS, W. (1999). Reformulation-linearization techniques for discrete optimization problems, in *Handbook of combinatorial optimization 1* (DU, D. and PARDALOS, P., eds.), pp. 479–532, Kluwer.

[108] SHERALI, H. and DESAI, J. (2005). A global optimization RLT-based approach for solving the hard clustering problem, *Journal of Global Optimization*, vol. 32, pp. 281–306.

[109] SHI, J. and MALIK, J. (2000). Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905.

[110] SONG, M. and RAJASEKARAN, S. (2005). Fast k-means algorithms with constant approximation, *Lecture Notes in Computer Science*, vol. 3827, pp. 1029–1038.

[111] SPÄTH, H. (1980). *Cluster analysis algorithm for data reduction and classification of objects.* New York: John Wiley & sons.

[112] STEINHAUS, H. (1956). Sur la division des corps matériels en parties, *Bulletin de l'Académie Polonaise des Sciences*, vol. IV, no. 12, pp. 801-804.

[113] STEINLEY, D. (2006). K-means clustering: A half-century synthesis, *British Journal of Mathematical and Statistical Psychology*, vol. 59, pp. 1–34.

[114] STEINLEY, D. (2007), Validating clusters with the lower bound for sum-of-squares error, *Psychometrika*, vol. 72, pp. 93–106.

[115] TAILLARD, É. (2003). Heuristic methods for large centroid clustering problems, *Journal of Heuristics*, vol. 9, pp. 51–73.

[116] TEBOULLE, M. (2007). A unified continuous optimization framework for center-based clustering methods, *Journal of Machine Learning Research*, vol. 8, pp. 65–102.

[117] TEN EIKELDER, H. and VAN ERK, A. (2004). Unification of some least squares clustering methods, *Journal of Mathematical Modelling and Algorithms*, vol. 3, pp. 105–122.

[118] TUY, H. (1964). Concave programming under linear constraints, *Soviet Mathematics*, vol. 5, pp. 1437–1440.

[119] VAN OS, B. and MEULMAN, J. (2004). Improving dynamic programming strategies for partitioning, *Journal of Classification*, vol. 21, pp. 207–230.

[120] VANDENBERGHE, L. and BOYD, S. (1996). Semidefinite programming, *SIAM Review*, vol. 38, pp. 49–95.

[121] VAVASIS, S. (1991). *Nonlinear Optimization: Complexity Issues.* Oxford: Oxford University Press.

[122] WANG, J. (1999). A linear assignment clustering algorithm based on the least similar cluster representatives, *IEEE Transactions on systems, man, and cybernetics - part A: systems and humans*, vol. 29, pp. 100–104.

[123] WOLSEY, L. (1998). *Integer Programming.* New York: John Wiley & sons.

[124] WU, F.-X., ZHANG, W., and KUSALIK, A. (2003). A genetic k-means clustering algorithm applied to gene expression data, *Lecture Notes in Artificial Intelligence*, vol. 2671, pp. 520–526.

[125] WU, X., KUMAR, V., QUINLAN, J. R., GHOSH, J., YANG, Q., MOTODA, H., MCLACHLAN, G. J., NG, A., LIU, B., YU, P. S., ZHOU, Z.-H., STEINBACH, M., HAND, D. J., and STEINBERG, D. (2008). Top 10 algorithm in data mining, *Knowledge and Information Systems*, vol. 14, pp. 1–37.

[126] XAVIER, A., NEGREIROS, M., MACULAN, N., and MICHELON, P. (2005). The use of the hyperbolic smoothing clustering method for planning the tasks of sanitary agents in combating dengue, in *Proceedings of IFORS 2005.*

[127] XIA, Y. and PENG, J. (2005). A cutting algorithm for the minimum sum-of-squared error clustering, in *Proceedings of the fifth SIAM International Data Mining Conference*, pp. 150–160.

[128] XU, M. and FRÄNTI, P. (2004). Delta-MSE dissimilarity in suboptimal k-means clustering, in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*.

[129] YEH, I.-C. (1998). Modeling of strength of high performance concrete using artificial neural networks, *Cement and Concrete Research*, vol. 28, pp. 1797–1808. Data set available at [http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength].

[130] ZHA, H., DING, C., GU, M., HE, X., and SIMON, H. (2002). Spectral relaxation for k-means clustering, in *Advances in Neural Information Processing Systems 14* (DIETTERICH, T., BECKER, S., and GHAHRAMANI, Z., eds.), pp. 1057–1064, MIT Press.

[131] ZHOU, W., ZHOU, C., HUANG, Y., and WANG, Y. (2005). Analysis of gene expression data: application of quantum-inspired evolutionary algorithm to minimum sum-of-squares clustering, *Lecture Notes in Artificial Intelligence*, vol. 3642, pp. 383–391.