versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

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

Daniel Aloise*
GERAD and École Polytechnique de Montréal
Montréal, QC, H3C 3A7, Canada
daniel.aloise@gerad.ca

Pierre Hansen
GERAD and HEC Montréal
Montréal, QC, H3T 2A7, Canada
pierre.hansen@gerad.ca

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 10/2008; aceito em 07/2009 após 1 revisão
Received October 2008; accepted July 2009 after one revision

Abstract

Minimum sum-of-squares clustering (MSSC) consists in partitioning a given set of *n* points into *k* clusters in order to minimize the sum of squared distances from the points to the centroid of their cluster. Recently, Peng & Xia (2005) established the equivalence between 0-1 semidefinite programming (SDP) and MSSC. In this paper, we propose a branch-and-cut algorithm for the underlying 0-1 SDP model. The algorithm obtains exact solutions for fairly large data sets with computing times comparable with those of the best exact method found in the literature.

Keywords: clustering; sum-of-squares; semidefinite programming.

Resumo

Clusterização por soma mínima de distâncias quadráticas consiste em particionar um dado conjunto de *n* pontos em *k* clusters a fim de minimizar a soma das distâncias quadráticas entre os pontos e o centróide de seus respectivos clusters. Recentemente, Peng & Xia (2005) estabeleceram a equivalência entre o problema e programação semidefinida 0-1. Neste artigo, um algoritmo *branch-and-cut* é proposto para o modelo baseado em programação semidefinida 0-1. O algoritmo obtém soluções exatas para instâncias reais de grande porte em tempos computacionais comparáveis àqueles do melhor método exato proposto na literatura.

Palavras-chave: clusterização; soma de distâncias quadráticas; programação semidefinida.

1. Introduction

Clustering is a powerful tool for automated analysis of data. It addresses the following general problem: given a set of entities, find subsets, or clusters, which are homogeneous and/or well separated (cf. Hartigan, 1975; Kaufman & Rousseeuw, 1990; Mirkin, 1996). 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 $O = \{o_1,o_2,...,o_n\}$ of n entities, we look for a partition $P_k = \{C_1, C_2,...,C_k\}$ of O into k clusters such that

(i) $C_j \neq \emptyset$ for $j = 1,\ldots, k$;

(ii) $C_{j_1} \cap C_{j_2} = \emptyset$ for $j_1, j_2 = 1,\ldots k$ and $j_1 \neq j_2$;

(iii) $\bigcup_{j=1}^{k} C_j = O$.

Many different criteria are used in the literature to express homogeneity and/or separation of the clusters to be found (see Hansen & Jaumard (1997) 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 (Delattre & Hansen, 1980; Florek *et al.*, 1951), or to minimize the *diameter*, i.e., the largest dissimilarity between a pair of entities in the same cluster (Hansen & Delattre, 1978). Among these criteria, the minimum sum of squared distances from each entity to the centroid of the cluster to which it belongs is one of the most used. It expresses both homogeneity and separation (see Späth (1980), pages 60-61). The resulting problem is called minimum sum-of-squares clustering (MSSC) for short. A mathematical programming formulation of MSSC is as follows:

$$\begin{aligned}
&\min_{x,y} \sum_{i=1}^{n}\sum_{j=1}^{k} x_{ij} \left\| p_i - y_j \right\|^2 \\
&\textit{subject to} \\
&\sum_{j=1}^{k} x_{ij} = 1, \quad \forall i = 1,\ldots,n \\
&x_{ij} \in \{0,1\}, \quad \forall i = 1,\ldots n; \ \forall j = 1,\ldots k.
\end{aligned} \tag{1}$$

The n entities $\{o_1,o_2,\ldots o_n\}$ to be clustered are at given points $p_i = (p_i^t, t = 1,\ldots,s)$ of $\Re^s$ for $i = 1,\ldots,n$; k cluster centers must be located at unknown points $y_j \in \Re^s$ 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.

The problem is also sometimes referred to in the literature as the *discrete clustering problem* or the *hard clustering problem*. Besides, it is well-known as the problem tackled by the classical k-means clustering heuristic (Forgy, 1965; MacQueen, 1967). From an initial set of k points viewed as initial centroids (or from an initial partition), k-means proceeds by (after computing initial centroids, if needed) reassigning the entities to their closest centroids and updating their positions until stability is reached. Although it does not provide the global optimum solution, *k*-means is popular due to its simplicity and fast convergence to a local optimum observed in practice. Moreover, it takes advantage of some mathematical properties of the MSSC formulation.

If the centroids y_j for $j = 1,\ldots, k$ are defined a priori, 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. This is exactly what *k*-means does after recomputing the centroids. Besides, for a fixed *x*, first order conditions on the gradient of the objective function require that at an optimal solution

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

In other words, it states that the optimal cluster centers are always at the centroids of the clusters. The *k*-means heuristic recomputes the centroids of the clusters whenever reassignments are performed, thereby improving the cost of the MSSC solution according to the optimality condition just mentioned. Because reassignments are performed only if profitable and the number of partitions is finite, we can conclude that *k*-means always converges to a local minimum.

Regarding computational complexity, MSSC is NP-hard in the plane for general values of *k* (Mahajan *et al.*, 2009). In general dimension, MSSC is NP-hard even for *k* = 2 (Aloise *et al.*, 2009). If both *k* and *s* are fixed, the problem can be solved in $O(n^{sk+1})$ time (Inaba *et al.*, 1994), which may be very time-consuming even for instances in the plane.

Not much effort was devoted to the exact resolution of the problem. To the best of our knowledge, there are less than a dozen papers published on the topic. Diehr (1973, p. 17) stated 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.

Early branch-and-bound algorithms are due to Koontz *et al.* (1975) and Diehr (1985). However, these methods are confined to small data sets. 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 ball in $\Re^s$. Assume these balls are pairwise at least *n* units apart. Then any reasonable 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. Note that *n*, *k* and *s* can be arbitrarily large. Recently, Brusco (2006) proposed a repetitive branch-and-bound procedure which gives bounds containing two components, i.e., an usual one corresponding to distances between already assigned entities and a 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, particularly when the number of clusters is small.

We can still select from the literature a column generation method proposed by du Merle *et al.* (2000) which transfers the complexity of the MSSC to the resolution of the pricing problem: an unconstrained hyperbolic problem in 0-1 variables with a quadratic numerator and linear denominator. It is solved approximately by Variable Neighborhood Search (VNS) (Mladenović & Hansen, 1997; Hansen & Mladenović, 2001) up to the moment that optimality must be checked by means of a Dinkelbach's-like algorithm (Dinkelbach, 1967). This algorithm clustered exactly, for the first time, several data sets, including the famous Fisher's 150 iris (Fisher, 1936). Xia & Peng (2005) casted the MSSC as a concave minimization problem and adapted Tuy's cutting plane method (Tuy, 1964) to solve it. In their paper, good approximate results are reported for a version where the cutting plane algorithm is halted before global convergence. Unfortunately, some further computational experiments of ours showed that this approach can only solve exactly fairly small instances.

The hardest task while devising exact algorithms for MSSC is to compute good lower bounds in a reasonable amount of time. Sherali & Desai (2005) proposed to obtain such bounds by linearizing the model via the reformulation-linearization technique (Sherali & Adams, 1999), claiming to solve instances with up to 1,000 points by means of a branch-and-bound algorithm. However, their algorithm was investigated in further detail by Aloise & Hansen (2008) and shown to require a very large computing time already for small real data sets.

Recently, Peng & Xia (2005) used matrix arguments to model MSSC as a so-called 0-1 semidefinite programming (SDP) which can be further relaxed to convex SDP or to linear programming. The branch-and-cut algorithm proposed here exploits the linear relaxation of the underlying 0-1 SDP model in order to obtain tight lower bounds for MSSC.

The paper is organized as follows. Section 2 shows the results of Peng & Xia (2005) on the equivalence between MSSC and 0-1 SDP as well as on proposing valid inequalities for the problem. In Section 3, our branch-and-cut algorithm is presented. Finally, computational results and conclusions are given in Section 4.

2. 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 (Vandenberghe & Boyd, 1996). The canonical SDP has the following form:

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

where W and B_i for $i = 1,\dots, m$ are matrices of coefficients, $Tr(\cdot)$ denotes the trace of the 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

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

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. Boole (1854); Fortet (1959)). Moreover, the eigenvalues of matrix *Z* are equal to 0 or 1.

From Huygens' theorem (see e.g. Edwards & Cavalli-Sforza (1965)), the MSSC objective function aforementioned can be rewritten as

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

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

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

$$= Tr(W_pW_p^T) - \sum_{j=1}^{k}\frac{\left\|\sum_{i=1}^{n} x_{ij}\,p_i\right\|^2}{\sum_{i=1}^{n} x_{ij}},$$

where $W_p \in \Re^{nxs}$ is the matrix whose *i*-th row is the vector p_i. These matrix arguments were used by Zha *et al.* (2002) and Steinley (2007) in order to look for orthonormal matrices which optimize the second term of the expression.

In Peng & Xia (2005), maximization of the second term is shown to be equivalent to maximizing a 0-1 SDP problem. Their development is based on the definition of a matrix $Z=X\,(X^TX)^{-1}\,X^T$ from a feasible assignment matrix *X*. Thus, the following 0-1 SDP model for MSSC is obtained

$$\begin{aligned}&\min Tr(W_pW_p^T(I-Z))\\ &s.t.\ \ Ze=e,\ Tr(Z)=k,\\ &\qquad Z\ge 0,\ Z=Z^T,\ Z^2=Z.\end{aligned} \tag{2}$$

Peng & Xia (2005) then 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 (2) is established. Regarding complexity, the 0-1 SDP model is all linear except for the constraint $Z^2 = Z$.

2.1 Valid inequalities for the 0-1 SDP formulation

Peng & Xia (2005) also derived valid inequalities for (2) from a property of semidefinite positive matrices. Suppose Z a feasible solution for (2). 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 $Z^2 = Z$, $\sum_{j \in \mu_1} (Z_{i_1 j})^2 = Z_{i_1 i_1}$, where $\mu_1 = \{ j : Z_{i_1 j} > 0 \}$. This implies that

$$\sum_{j \in \mu_1} \frac{Z_{i_1 j}}{Z_{i_1 i_1}} Z_{i_1 j} = 1.$$

From the choice of i_1 and the constraint $\sum_{j=1}^{n} Z_{i_1 j} = \sum_{j \in \mu_1} Z_{i_1 j} = 1$, Peng & Xia (2005) concluded that

$$Z_{i_1 j} = Z_{i_1 i_1}, \quad \forall j \in \mu_1.$$

If the respective columns and lines associated to the index set μ_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

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

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

Peng & Xia (2005) have proposed a 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

$$Z_{ij} \le Z_{ii} \qquad \forall i, j \qquad (\textit{pair inequalities})$$

$$Z_{ij} + Z_{i\ell} \le Z_{ii} + Z_{j\ell} \qquad \forall i, j, \ell \qquad (\textit{triangular inequalities})$$

This partial polyhedron characterization was inspired by the work of Lisser & Rendl (2003) for graph partitioning. Thus, the resulting LP relaxed model is expressed by

$$
\begin{aligned}
\min\ & Tr(W_p W_p^T (I - Z)) \\
s.t.\ \ & Ze = e, Tr(Z) = k, \\
& Z \geq 0, \\
& Z_{ij} \leq Z_{ii} \quad \forall i, j, \\
& Z_{ij} + Z_{i\ell} \leq Z_{ii} + Z_{j\ell} \quad \forall i, j, \ell.
\end{aligned}
\tag{3}
$$

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 & Xia (2005) is not on exact methods, the authors suggest a simple branching scheme. Suppose that for the optimal solution Z^r of the LP relaxation there are indices i and j such that $Z_{ij}^r (Z_{ii}^r - Z_{ij}^r) \neq 0$, then one can produce a branch with $Z_{ii}^r = Z_{ij}^r$ and another one with $Z_{ij}^r = 0$. With this branching scheme, the number of different branches is limited to $O(2^{n^2})$.

Regarding variable selection, we propose to choose indices i and j as the $\arg\max_{i,j} \min\{Z_{ij}^r, Z_{ii}^r - Z_{ij}^r\}$. 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 2, the list L of unsolved problems is initialized with the 0-1 SDP model (2). 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 VNS 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) 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 3,000 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.

If P^r is feasible for P in line 11, then due to equivalence between (1) and (2), 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 (2). 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 (2);
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) 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                 $s^* \leftarrow s$;
16             end
17         end
18         if $cost(Z^r) < cost(s^*)$ then
19             Calculate $\operatorname{argmax}_{i,j} \min\{Z^r_{ij}, Z^r_{ii} - Z^r_{ij}\}$;
20             Branch $P$ into two subproblems by means of cuts $Z^r_{ij} = 0$ and
               $Z^r_{ii} - Z^r_{ij} = 0$ and add them to $L$;
21         end
22     end
23 until $L = \emptyset$ ;
24 return $s^*$;
```

Algorithm 1 – Branch-and-cut SDP-based algorithm for MSSC.

4. Computational Results

In this section we report on the computational experiences with our SDP-based branch-and-cut algorithm for MSSC. Results were obtained using a Dual Core AMD Opteron^{TM} 2 GHz architecture and 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-halfp 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 (Ruspini, 1970), ii. Späth's 89 Bavarian postal codes in three dimensions (Späth, 1980), iii. the synthetic HATCO data set published by (Hair *et al.*, 1998) consisting of 100 objects in seven dimensions, iv. Fisher's 150 iris problem in four dimensions (Fisher, 1936), v. Grötschel and Holland's 202 European cities coordinates (Grötschel & Holland, 1991)). 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 optimal objective function values. The third column presents the CPU times spent on solving the LP relaxation proposed by Peng & Xia (2005) to their 0-1 SDP formulation (recall that their model includes all inequalities a priori). Remaining columns are associated to CPU times of exact methods, i.e, the column generation algorithm (CGA) of du Merle *et al.* (2000) and the three versions of BC-SDP-MSSC. Twenty iterations were used as stopping condition to the VNS heuristic that obtains approximate solutions to the pricing problems arising in column generation. Moreover, a last column is included in the tables 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 'c' 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 1 – Ruspini's data set.

k	Opt. Sol.	CPU times (seconds)					% gap
		LP relax.	CGA	BC-tri	BC-all	BC-halfp	
2	89337.8	10.48	**1.69**	3.54	14.33	3.56	c
3	51063.4	13.75	**2.62**	8.79	15.10	8.34	c
4	12881.0	3.62	1.16	0.69	2.83	**0.48**	c
5	10126.7	4.80	1.92	0.78	2.91	**0.60**	c
6	8575.4	10.02	3.80	1.97	3.46	**1.03**	c
7	7126.2	10.92	4.26	1.20	2.22	**0.98**	c
8	6149.6	15.97	14.29	8.24	12.67	**7.27**	0.5 (7)
9	5181.6	15.29	4.87	2.90	4.54	**2.87**	0.3 (3)
10	4446.3	17.69	4.02	**2.08**	3.88	2.39	0.3 (3)
20	1721.2	21.15	4.52	0.31	0.41	**0.28**	c
30	741.8	27.16	3.25	0.17	0.16	**0.14**	c

Table 2 – Späth's data set.

k	Opt. Sol.	CPU times (seconds)					% gap
		LP relax.	CGA	BC-tri	BC-all	BC-halfp	
2	$6.02546\ 10^{11}$	129.65	**1.44**	9.96	150.36	10.78	c
3	$2.94506\ 10^{11}$	169.38	**6.33**	27.33	136.50	27.58	c
4	$1.04474\ 10^{11}$	97.81	**8.36**	31.81	96.61	26.31	c
5	$5.97615\ 10^{10}$	420.91	15.00	18.23	53.28	**10.07**	c
6	$3.59085\ 10^{10}$	567.97	**5.26**	17.88	47.00	13.13	c
7	$2.19832\ 10^{10}$	889.99	**7.44**	38.85	52.35	25.06	c
8	$1.33854\ 10^{10}$	927.22	12.52	10.11	18.41	**9.04**	c
9	$8.42375\ 10^{9}$	723.69	8.62	8.84	12.36	**5.51**	c
10	$6.44647\ 10^{9}$	996.35	8.07	8.02	10.46	**4.26**	c
20	$7.48215\ 10^{8}$	578.69	6.80	0.98	0.99	**0.74**	c
30	$1.71392\ 10^{8}$	433.77	4.99	0.31	0.34	**0.26**	c

Tables 1–4 suggest the following conclusions:

- The cutting plane algorithms are able to prove optimality of model (2) in less computing time than solving its LP relaxation with all inequalities added a priori in 87.09% of the instances solved at the root node (indicated by letter 'c'). When branch occurs, our branch-and-cut algorithms find optimal integer solutions in less computing time than solving only the LP relaxation with all inequalities in 8 out of 13 instances.
- Algorithm BC-all is in most of cases outperformed by one of its counterparts. 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.

Table 3 – Fisher's data set.

k	Opt. Sol.	CPU times (seconds)					% gap
		LP relax.	CGA	BC-tri	BC-all	BC-halfp	
2	152.3479	9473.85	293.34	**166.82**	549.28	169.44	c
3	78.8514	2098.29	**66.57**	512.85	454.70	283.24	c
4	57.2284	1927.25	**186.00**	301.12	299.40	240.19	c
5	46.4461	2303.65	**104.65**	152.18	237.36	145.54	c
6	39.0399	2331.69	**102.43**	141.88	163.32	147.51	c
7	34.2982	2155.89	**102.97**	1061.16	847.93	742.83	0.0 (7)
8	29.9889	2557.59	106.10	**89.92**	123.18	108.73	c
9	27.7860	2568.36	122.16	92.19	88.78	**70.04**	c
10	25.8340	2294.35	110.00	76.47	73.07	**59.66**	c
20	14.2208	1791.42	137.10	155.75	104.91	**87.06**	0.0 (5)
30	9.5552	1641.43	**91.65**	175.87	145.24	155.52	0.1 (23)

Table 4 – HATCO's data set.

k	Opt. Sol.	CPU times (seconds)					% gap
		LP relax.	CGA	BC-tri	BC-all	BC-halfp	
2	600.108	73.57	**13.95**	108.79	101.88	69.68	c
3	506.962	149.82	**18.50**	158.15	146.19	141.52	c
4	426.602	105.75	**34.46**	85.39	88.54	68.00	c
5	383.831	155.93	**16.21**	61.46	80.33	52.56	c
6	344.534	151.85	**14.29**	188.49	151.39	145.03	0.0 (3)
7	313.582	88.83	**18.01**	246.91	191.20	181.13	0.1 (5)
8	288.601	105.54	**19.04**	377.88	378.52	286.67	0.5 (11)
9	264.599	72.32	**161.16**	421.22	381.70	314.30	0.6 (13)
10	241.128	67.61	**132.07**	315.99	284.10	217.74	0.4 (15)
20	114.032	21.91	6.73	5.00	6.07	**4.15**	0.0 (3)
30	62.992	22.78	9.09	2.51	2.72	**2.35**	0.0 (5)

- In 50% of the instances, the best CPU time obtained among our branch-and-cut algorithms is smaller than that obtained by the column generation algorithm of du Merle *et al.* (2000).
- Relaxation (3) provides very good bounds for MSSC since initial gap values are never larger than 0.6%. Moreover, more than 65% of the tested instances 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 & Wakabayashi (1989) 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.
- 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 (2) 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 5 presents results for Grötschel and Holland's 202 European cities coordinates (Grötschel & Holland, 1991) whose value of *n* is the largest among data sets (i–v). Results show that BC-halfp 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 5 – Grötschel and Holland's data set.

k	Opt. Sol	CPU times (seconds)	% gap
		BC-halfp	
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 approach based on solving LP relaxations of the 0-1 SDP formulation provided by Peng & Xia (2005) can be extended to other related clustering problems (e.g. normalized *k*-cut minimization, balanced clustering; see Peng & Wei (2007) for details).

Acknowledgments

Research of the first author was supported by CAPES/Brazil grant number 2479-04-4. Research of the second author was supported by NSERC grant number 105574-07, FQRNT grant 2007-PR-112176 and the Data Mining Chair of HEC Montréal.

References

(1) Aloise, D.; Deshpande, A.; Hansen, P. & Popat, P. (2008). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, **75**, 245-249.

(2) Aloise, D. & Hansen, P. (2008). Evaluating a branch-and-bound RLT-based algorithm for minimum sum-of-squares clustering. *Les Cahiers du GERAD*, G-2008-26.

(3) Boole, G. (1854). *An Investigation of the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities*. Walton and Maberley, London.

(4) Brusco, M. (2006). A repetitive branch-and-bound procedure for minimum within-cluster sum of squares partitioning. *Psychometrika*, **71**, 347-363.

(5) Delattre, M. & Hansen, P. (1980). Bicriterion cluster analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(**4**), 277-291.

(6) Diehr, G. (April 1973). Minimum variance partitions and mathematical programming. Paper presented at the *National Meetings of the Classification Society*, Atlanta, Georgia.

(7) Diehr, G. (1985). Evaluation of a branch and bound algorithm for clustering. *SIAM Journal on Scientific and Statistical Computing*, **6**, 268-284.

(8) Dinkelbach, W (1967). On nonlinear fractional programming. *Management Science*, **13**, 492-498.

(9) du Merle, O.; Hansen, P.; Jaumard, B. & Mladenović, N. (2000). An interior point algorithm for minimum sum-of-squares clustering. *SIAM Journal on Scientific Computing*, **21**, 1485-1505.

(10) Edwards, A. & Cavalli-Sforza, L. (1965). A method for cluster analysis. *Biometrics*, **21**, 362-375.

(11) Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **VII**, 179-188.

(12) Florek, K.; Lukaszewicz, J.; Perkal, H.; Steinhaus, H. & Zubrzycki, S. (1951). Sur la liaison et la division des points d'un emsemble fini. *Colloquium Mathematicum*, **2**, 282-285.

(13) Forgy, E.W. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, **21**, 768.

(14) Fortet, R. (1959). L'algèbre de boole et ses applications en recherche opérationnelle. *Cahiers du Centre d'Études de Recherche Opérationnelle*, **1**, 5-36.

(15) Grötschel, M. & Holland, O. (1991). Solution of large-scale symmetric traveling salesman problems. *Mathematical Programming*, **51**, 141-202.

(16) Grötschel, M. & Wakabayashi, Y. (1989). A cutting plane algorithm for a clustering problem. *Mathematical Programming*, **45**, 59-96.

(17) Hair, J.; Anderson, R.; Tatham, R. & Black, W. (1998). *Multivariate Data Analysis*. Prentice-Hall, New York.

(18) Hansen, P. & Delattre, M. (1978). Complete-link cluster analysis by graph coloring. *Journal of the American Statistical Association*, **73**, 397-403.

(19) Hansen, P. & Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical Programming*, **79**, 191-215.

(20) Hansen, P. & Mladenović, N. (2001). J-means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, **34**, 405-413.

(21) Hartigan, J. (1975). *Clustering Algorithms*. Wiley, New York.

(22) Inaba, M.; Katoh, N. & 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*, 332-339.

(23) Kaufman, L. & Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York.

(24) Koontz, W.; Narendra, P. & Fukunaga, K. (1975). A branch and bound clustering algorithm. *IEEE Transactions on Computers*, **C-24**, 908-915.

(25) Lisser, A. & Rendl, F. (2003). Graph partitioning using linear and semidefinite programming. *Mathematical Programming*, Series B **95**, 91-101.

(26) MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. **In**: *Prooceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, **2**, 281-297, Berkeley, California.

(27) Mahajan, M.; Nimbhorkar, P. & Varadarajan, K. (2009). The planar *k*-means problem is NP-hard. *Lecture Notes in Computer Science*, **5431**, 274-285.

(28) Mirkin, B. (1996). *Mathematical Classification and Clustering*. Kluwer, Dordrecht, The Netherlands.

(29) Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, **24**, 1097-1100.

(30) Peng, J. & Wei, Y. (2007). Approximating *k*-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*, **18**, 186-205.

(31) Peng, J. & Xia, Y. (2005). A new theoretical framework for *k*-means-type clustering. *Studies in Fuzziness and Soft Computing*, **180**, 79-96.

(32) Ruspini, E. (1970). Numerical method for fuzzy clustering. *Information Sciences*, **2**, 319-350.

(33) Sherali, H.D. & Adams, W. (1999). Reformulation-linearization techniques for discrete optimization problems. **In**: *Handbook of combinatorial optimization* [edited by D. Du and P. Pardalos], Kluwer, 479-532.

(34) Sherali, H. & Desai, J. (2005). A global optimization RLT-based approach for solving the hard clustering problem. *Journal of Global Optimization*, **32**, 281-306.

(35) Späth, H. (1980). *Cluster analysis algorithm for data reduction and classification of objects*. John Wiley & sons, New York.

(36) Steinley, D. (2007). Validating clusters with the lower bound for sum-of-squares error. *Psychometrika*, **72**, 93-106.

(37) Tuy, H. (1964). Concave programming under linear constraints. *Soviet Mathematics*, **5**, 1437-1440.

(38) Vandenberghe, L. & Boyd, S. (1996). Semidefinite programming. *SIAM Review*, **38**, 49-95.

(39) Xia, Y. & Peng, J. (2005). A cutting algorithm for the minimum sum-of-squared error clustering. **In**: *Proceedings of the SIAM International Data Mining Conference*.

(40) Zha, H.; Ding, C.; Gu, M.; He, X. & Simon, H. (2002). Spectral relaxation for k-means clustering. **In**: *Advances in Neural Information Processing Systems 14* [edited by T. Dietterich, S. Becker and Z. Ghahramani], MIT Press, 1057-1064.