Block Clustering Based on Difference of Convex Functions (DC) Programming and DC Algorithms

Hoai Minh Le
minh.le@univ-lorraine.fr
Hoai An Le Thi
hoai-an.le-thi@univ-lorraine.fr
Laboratory of Theoretical and Applied Computer Science, University of Lorraine, 57045 Metz, France

Tao Pham Dinh
pham@insa-rouen.fr
Laboratoire of Mathematics, National Institute for Applied Sciences—Rouen, 76800 Saint-Étienne-du-Rouvray cedex, France

Van Ngai Huynh
ngaivn@yahoo.com
University of Qui Nhon, Quy Nhon City, Vietnam

We investigate difference of convex functions (DC) programming and the DC algorithm (DCA) to solve the block clustering problem in the continuous framework, which traditionally requires solving a hard combinatorial optimization problem. DC reformulation techniques and exact penalty in DC programming are developed to build an appropriate equivalent DC program of the block clustering problem. They lead to an elegant and explicit DCA scheme for the resulting DC program. Computational experiments show the robustness and efficiency of the proposed algorithm and its superiority over standard algorithms such as two-mode K-means, two-mode fuzzy clustering, and block classification EM.

1 Introduction

Clustering, which aims at dividing a data set into groups or clusters containing similar data, is a fundamental problem in unsupervised learning and has many applications in various domains. In this letter, we are interested in block clustering, or biclustering or two-mode clustering, which consists of simultaneous clustering on the set of samples (objects) and on the set of their features in order to find homogeneous blocks. Biclustering has great significance in different fields, particularly for biomedical applications. When it is performed with high reliability, we are able not only to diagnose conditions represented by sample classes but also to identify...
features (e.g., genes or proteins) responsible for them or serving as their markers.

We have a data set of \( n \) objects (samples), each of them composed of \( m \) features (variables). This data set is presented by a \( n \times m \) real-valued rectangular matrix \( X = (x_{ij})_{n \times m} \) where \( i \in I := \{1, \ldots, n\} \), the set of \( n \) rows, and \( j \in J := \{1, \ldots, m\} \), the set of \( m \) columns, and the value \( x_{ij} \) is the expression of the \( j \)th feature in the \( i \)th object. More precisely, each row of \( X \) refers to an object, while each column of \( X \) corresponds to a feature.

A biclustering of a data set is a collection of pairs of object and feature subsets \( B = \{(O_1, F_1), \ldots, (O_k, F_l), \ldots\} \) such that the collection \( \{O_1, \ldots, O_k\} \) forms a partition of the set of objects on \( K \) clusters (row clusters) and the collection \( \{F_1, \ldots, F_l\} \) forms a partition of the set of features on \( L \) clusters (column clusters). A pair \( (O_k, F_l) \) will be called a bicluster. In other words, biclustering (two-mode clustering) assigns each element of \( X \) to a row cluster and a column cluster. If \( L = m \), two-mode clustering reduces to one-mode clustering (partitioning the set of rows), and the same is true if \( K = n \) (partitioning the set of columns).

Depending on the definition of the distance measure, biclustering covers a large variety of problems. In this letter, we focus on biclustering using the squared Euclidean metric as the distance measure. The problem can then be stated as follows. Let

- \( P_{n \times K} = (p_{ik})_{I \times K} \) be the binary matrix containing row cluster membership values (called a row-classification matrix), that is, \( p_{ik} = 1 \) if row \( i \) belongs to row cluster \( k \), and \( p_{ik} = 0 \) otherwise
- \( Q_{m \times L} = (q_{jl})_{J \times L} \) be the binary matrix containing column cluster membership values (called a column-classification matrix), that is, \( q_{jl} = 1 \) if column \( j \) belongs to column cluster \( l \), and \( q_{jl} = 0 \) otherwise
- \( V_{K \times L} = (v_{kl})_{K \times L} \) be the matrix representing prototype values for the entries in a block, that is, \( x_{ij} \) is in the block with prototype \( v_{kl} \) when \( p_{ik} q_{jl} = 1 \)

We search for the block cluster prototype matrix \( V \) and the two cluster membership values matrices \( P, Q \), which minimize the sum of squared Euclidean distances from the entries to their respective prototype values.

Define the inner product \( \langle X, Y \rangle \) in \( M_{n,m}(\mathbb{R}) \) (the vector space of \( n \times m \) real-valued matrices) as the trace of the matrix \( X^T Y \), which is denoted by \( \text{Tr}(X^T Y) \), say,

\[
\langle X, Y \rangle_{M_{n,m}(\mathbb{R})} = \sum_{i=1}^{n} X_i Y_i^T = \sum_{j=1}^{m} (X^j)^T Y^j = \text{Tr}(X^T Y),
\]

where \( X_i \) and \( X^k \) are, respectively, the \( i \)th row and \( k \)th column of the matrix \( X \). Denote by \( \| \cdot \| \) the corresponding Euclidean norm on \( M_{n,m}(\mathbb{R}) \). Then the
function to be minimized in biclustering is given by
\[
f(P, Q, V) := \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ik}q_{jl}(x_{ij} - v_{kl})^2 = \|X - PVQ^T\|^2. \tag{1.1}
\]

As usual, the classification matrices must satisfy the following constraints:

- The cluster membership values of each row and column object must sum to one:
  \[\sum_{k=1}^{K} p_{ik} = 1 \quad \forall i = 1, \ldots, n, \quad \sum_{l=1}^{L} q_{jl} = 1 \quad \forall j = 1, \ldots, m.\]

- None of the row or column clusters is empty:
  \[\sum_{i=1}^{n} p_{ik} \geq 1 \quad \forall k = 1, \ldots, K, \quad \sum_{j=1}^{m} q_{jl} \geq 1 \quad \forall l = 1, \ldots, L.\]

- All cluster membership values must be either zero or one:
  \[p_{ik} \in \{0, 1\} \quad \forall i = 1, \ldots, n, k = 1, \ldots, K,
  \quad q_{jl} \in \{0, 1\} \quad \forall j = 1, \ldots, m, l = 1, \ldots, L.\]

Finally, the mathematical block clustering problem can be written as
\[
\begin{aligned}
\text{min}_{(P, Q, V)} & \quad f(P, Q, V) = \|X - PVQ^T\|^2 \\
\text{s.t.} & \quad \sum_{k=1}^{K} p_{ik} = 1; \quad \sum_{l=1}^{L} q_{jl} = 1; \quad \forall i = 1, \ldots, n; \quad j = 1, \ldots, m \\
& \quad \sum_{i=1}^{n} p_{ik} \geq 1; \quad \sum_{j=1}^{m} q_{jl} \geq 1; \quad \forall k = 1, \ldots, K; \quad l = 1, \ldots, L \\
& \quad P \in \{0, 1\}^{n \times K}, \quad Q \in \{0, 1\}^{m \times L}, \quad V \in \mathbb{R}^{K \times L}.
\end{aligned} \tag{1.2}
\]

Problem 1.2 is a nonconvex nonlinear mixed 0-1 program, known to be NP-hard and for which no efficient global algorithm is available. Even for small \(n\) and \(m\), the number of possible partitions can become extremely large. There are several optimization methods for two-mode clustering. All of these algorithms are heuristics (often based on meta-heuristic, multistart procedures) that are not guaranteed to find the global optimum and often get stuck in local minima. An extensive overview of two-mode clustering methods can be found in Mechelen, Bock, and De Boeck (2004).
Recently, Rosmalen, Groenen, Trejos, and Castillo (2005) gave an overview of existing optimization methods based on meta-heuristics and introduced new fuzzy algorithms (two-mode fuzzy) for solving problem 1.2. A simulation study has been performed to compare these methods and determine how effective they are at finding the optimal clustering. It turns out that the application of the multistart heuristic, in combination with the K-means algorithm, usually has the best average performance.

We investigate in this work an efficient nonconvex programming approach for solving the biclustering problem 1.2. Our methods are based on DC (difference of convex functions) programming and DCA (DC algorithms) that were introduced by Pham Dinh Tao in preliminary form in 1985. They have been extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao (see Le Thi, 1997; Le Thi & Pham Dinh, 1997, 2005; Pham Dinh & Le Thi, 1998) and are now classic and increasingly popular (see Collobert, Sinz, Weston, & Bottou, 2006; Liu & Shen, 2006; Liu, Shen, & Doss, 2005; Neumann, Schnörr, & Steidl, 2004; Shen, Tseng, Zhang, & Wong, 2003; and the list of references in Le Thi, n.d.). Our work is motivated by the fact that DCA has been successfully applied to many smooth and nonsmooth large-scale nonconvex programs in various domains of applied sciences, in particular, in machine learning (Collobert et al., 2006; Krause & Singer, 2004; Le Thi, Belghiti, & Pham Dinh, 2006; Le Thi, Le, & Pham Dinh, 2006, 2007; Le Thi, Le, Nguyen, & Pham Dinh, 2008; Le Thi, Nguyen, & Ouchani, 2008; Le Thi, Pham Dinh, & Huynh, 2012; Liu et al., 2005; Liu & Shen, 2006; Neumann et al., 2004; Ong & Le Thi, 2011; Shen et al., 2003; Symons, 1981; Yuille & Rangarajan, 2002; Weber, Schüle, & Schnörr, 2005) for which they provided quite often a global solution and proved to be more robust and efficient than standard methods.

DC programming and DCA have been already developed in Le Thi et al. (2006) for (one-mode) clustering via the bilevel formulation of the minimum sum-of-squares Euclidean distance problem. Numerical experiments on real-world databases show the efficiency and superiority of the proposed DCA with respect to the standard K-means algorithm. DCA has also been extensively investigated in Le Thi et al. (2007) with regard to the fuzzy C-Mean model, which was introduced first in Bezdek (1973). Numerical results in Le Thi et al. (2007) also proved the efficiency and the scalability of DCA as well as its superiority over the FCM standard method.

A DC program is of the form

\[
\min \{ f(x) := g(x) - h(x) : x \in C \},
\]  

where \( g \) and \( h \) are convex functions defined on \( \mathbb{R}^p \) and \( C \) is a convex set in \( \mathbb{R}^p \). The construction of DCA involves the convex DC components \( g \) and \( h \) but not the DC function \( f \) itself. A DC function \( f \) has infinitely many DC decompositions \( g - h \). Chosing an appropriate decomposition has a crucial
impact on the qualities (speed of convergence, robustness, efficiency, globality of computed solutions) of DCA. How to develop an efficient algorithm based on the generic DCA scheme for a practical problem is a question to be studied, and the answer depends on the structure of the problem being considered.

The purpose of this letter is to develop an efficient and scalable DCA-based algorithm for block clustering problem 1.2. The investigation of DC programming and DCA to the hard optimization block clustering problem, 1.2, requires rigorous study for reformulating it in terms of a DC program. Using an interesting exact penalty result developed recently in Le Thi et al. (2012), we first transform equation 1.2 into an equivalent continuous optimization problem. Then we carefully study DC reformulation techniques for the continuous optimization problem to get finally a good DC program. A simple and elegant DCA scheme for solving the resulting DC program is then developed. The proposed DCA scheme is original and very inexpensive because it amounts to computing, at each iteration, the projection of points onto a simplex or onto a box that are all determined in the explicit form. Numerical experiments on several data sets have shown the efficiency of the proposed method and its superiority over two-mode K-means, two-mode Fuzzy (the two best algorithms presented in Rosmalen et al., 2005), and block classification expectation maximization (BCEM; Govaert & Nadif, 2003; see section 4).

The rest of the letter is organized as follows. In the next section, we introduce a continuous formulation of problem 1.2 by using an exact penalty result in DC programming. Section 3 is devoted to DC programming and DCA for solving the block clustering problem in the continuous optimization form. We briefly introduce DC programming and DCA and then reformulate the problem as a DC program for which a customized DCA is developed. Comparative experimental results with two-mode K-means, two-mode fuzzy (Rosmalen et al., 2005), and BCEM (Govaert & Nadif, 2003) are reported in section 4. Some conclusions end the letter. In the appendix, we describe the three comparative algorithms: two-mode K-means, two-mode fuzzy and BCEM.

2 A Continuous Formulation of the Block Clustering Problem 1.2

Let us show that the variable $V$ in problem 1.2 can be bounded in a box. Let $(P^*, Q^*, V^*)$ be a solution to the problem. For every $k, l$, there exist $i, j$ such that $p_{ik} = q_{jl} = 1$. Thus,

$$(x_{ij} - v_{kl})^2 \leq f(P^*, Q^*, V^*) \leq \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}^2.$$
Hence,

\[ x_{ij} - a \leq v_{kl} \leq x_{ij} + a, \quad \text{where} \quad a^2 := \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}^2. \]

Denote by \( \alpha = \min_{i=1, \ldots, n; j=1, \ldots, m} (x_{ij} - a), \quad \beta = \max_{i=1, \ldots, n; j=1, \ldots, m} (x_{ij} + a); \) then \( v_{kl} \in [\alpha, \beta], \quad \forall \ k, l. \) By this observation, it suffices to consider problem 1.2 with \( V \in \mathcal{V} = [\alpha, \beta]^{K \times L}. \)

\[ P = \left\{ P = (p_{ik})_{I \times K} : \sum_{k=1}^{K} p_{ik} = 1; \ p_{ik} \in \{0, 1\}, \ i = 1, \ldots, n, \right\}, \]

\[ Q = \left\{ Q = (q_{jl})_{J \times L} : \sum_{l=1}^{L} q_{jl} = 1; \ q_{jl} \in \{0, 1\}, \ j = 1, \ldots, m, \right\}. \]

Problem 1.2 is equivalent to

\[ \min \{ f(P, Q, V) : \ (P, Q, V) \in \mathcal{P} \times \mathcal{Q} \times \mathcal{V} \}. \quad (2.1) \]

**Lemma 1.** The function \( f(\cdot, \cdot, V) \) is uniformly Lipschitz on \( \mathcal{P} \times \mathcal{Q} \) for \( V \in \mathcal{V} \) with uniform Lipschitz constant \( \mathcal{L} \) given by

\[ \mathcal{L} = \sqrt{2} \max\{\alpha^2, \beta^2\} \max\{nK, mL\}. \quad (2.2) \]

**Proof.** For \( x, y, z; \ x', y' \in R, \) by the Cauchy-Schwarz inequality, we have the following bound:

\[
xyz^2 - x'y'z^2 = z^2y(x - x') + z^2x'(y - y') \\
\leq z^2 \sqrt{y^2 + x'^2} \sqrt{(x - x')^2 + (y - y')^2}.
\]

By taking \( z = x_{ij} - v_{kl}, \ x = p_{ik}, \ x' = p_{ik}', \ y = q_{jl}, \ y' = q_{jl}' \) into account, it follows directly that \( f(\cdot, \cdot, V) \) is uniformly Lipschitz on \( \mathcal{P} \times \mathcal{Q} \) for \( V \in \mathcal{V} \) with uniformly Lipschitzian constant

\[ \mathcal{L} = \sqrt{2} \max\{\alpha^2, \beta^2\} \max\{nK, mL\}. \quad (2.3) \]
For reformulating equation 1.2 in a continuous optimization form, first recall the following exact penalty result. For a point \( x \in \mathbb{R}^n \) and a set \( S \subset \mathbb{R}^n \), denote by \( d(x, S) \) the distance from \( x \) to \( S \):

\[
d(x, S) := \inf_{z \in S} \| x - z \|.
\]

We have:

**Lemma 2.** (see, e.g., Le Thi et al., 2012) Suppose that \( f \) is Lipschitz on \( \mathcal{X} \subset \mathbb{R}^n \) with constant \( \mathcal{L}_0 \). Let \( \varphi : \mathcal{X} \to \mathbb{R}^n \cup \{+\infty\} \) be a nonnegative function defined on \( \mathcal{X} \) and

\[
S := \{ x \in \mathcal{X} : \varphi(x) = 0 \}.
\]

If \( d(x, S) \leq \varphi(x) \) for all \( x \in \mathcal{X} \), then for all \( \mathcal{L} > \mathcal{L}_0 \), the two problems

\[
\inf \{ f(x) : x \in S \} \quad \text{and} \quad \inf \{ f(x) + \mathcal{L}\varphi(x) : x \in \mathcal{X} \}
\]

are equivalent insofar as they have the same optimal value and the same solution set.

Denote by

\[
S_p = \left\{ P = (p_{ik})_{1 \times K} : \sum_{k=1}^{K} p_{ik} = 1; \ p_{ik} \in [0, 1], \ i = 1, ..., n; \ k = 1, ..., K \right\}
\]

\[
S_q = \left\{ Q = (q_{jl})_{J \times L} : \sum_{l=1}^{L} q_{jl} = 1; \ q_{jl} \in [0, 1], \ j = 1, ..., m; \ l = 1, ..., L \right\}
\]

\[
S^1_p = \{ P \in S_p : p_{ik} \in \{0, 1\}, \ i = 1, ..., n; \ k = 1, ..., K \}
\]

\[
S^1_q = \{ Q \in S_q : q_{jl} \in \{0, 1\}, \ j = 1, ..., m; \ l = 1, ..., L \}.
\]

For using lemma 2, we find an upper bound for \( d((P, Q), P \times Q) \) with \( (P, Q) \in S_p \times S_q \).

**Step 1. Estimation Bounds for** \( d((P, Q), S^1_p \times S^1_q) \) with \( (P, Q) \in S_p \times S_q \).

The following lemma will be needed:

**Lemma 3.** Let

\[
K := \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1, \ x_i \in [0, 1], \ i = 1, ..., n \right\}
\]
and

\[
C := \left\{ x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1, \ x_i \in \{0, 1\}, i = 1, \ldots, n \right\}.
\]

Then for all \( x = (x_1, \ldots, x_n) \in K \), one has

\[
d(x, C) \leq 2(1 - \max_{1 \leq i \leq n} \{x_i\}).
\]

**Proof.** For \( x = (x_1, \ldots, x_n) \in K \), one has

\[
d^2(x, C) = \min \left\{ \sum_{i=1}^n (x_i - y_i)^2 : y = (y_1, \ldots, y_n) \in C \right\}
\]

\[
= \min \left\{ \sum_{i=1}^n x_i^2 - 2 \sum_{i=1}^n x_iy_i + 1 : y = (y_1, \ldots, y_n) \in C \right\}
\]

\[
= \sum_{i=1}^n x_i^2 + 1 - 2 \max_{1 \leq i \leq n} \{x_i\}.
\]

Let \( r \in \{1, \ldots, n\} \) be such that \( \max_{1 \leq i \leq n} \{x_i\} = x_r \). Then

\[
d^2(x, C) = \sum_{i \neq r} x_i^2 + (1 - x_r)^2 \leq \left( \sum_{i \neq r} x_i + 1 - x_r \right)^2
\]

\[
= 4(1 - x_r)^2 = 4(1 - \max_{1 \leq i \leq n} \{x_i\})^2.
\]

The proof is complete.

**Remark 1.** Let \( x = (x_1, \ldots, x_n) \in K \) and \( \varphi(x) = 2(1 - \max_{1 \leq i \leq n} \{x_i\}) \). Then

\[
d(x, C) \leq \varphi(x) \ \forall x \in K \ \text{and} \ \ x \in C \ \text{iff} \ \varphi(x) = 0.
\]

From this lemma, one obtains the following bound,

\[
d((P, Q), S_p^1 \times S_q^1) \leq 2 \left( \sum_{i=1}^n \left(1 - \max_{1 \leq k \leq K} \{p_{ik}\} \right) + \sum_{j=1}^m \left(1 - \max_{1 \leq l \leq L} \{p_{jl}\} \right) \right)
\]

(2.4)

for all \((P, Q) \in S_p \times S_q\).
Step 2. Bound for $d((P, Q), \mathcal{P} \times \mathcal{Q})$ with $(P, Q) \in S^1_p \times S^1_q$. Let $(P, Q) \in (S^1_p \times S^1_q) \setminus (\mathcal{P} \times \mathcal{Q})$. It is easy to see that

$$d(P, P) \leq \sqrt{2n}; \quad d(Q, Q) \leq \sqrt{2m}.$$ 

On the other hand, since $(P, Q) \notin \mathcal{P} \times \mathcal{Q}$, there exist indices $k_0, l_0$ such that

$$\sum_{i=1}^n p_{ik_0} = 0 \quad \text{and} \quad \sum_{j=1}^m p_{jl_0} = 0.$$ 

Therefore,

$$d((P, Q), \mathcal{P} \times \mathcal{Q}) \leq \sqrt{2n} \sum_{k=1}^K \max \left\{0, 1 - \sum_{i=1}^n p_{ik}\right\} + \sqrt{2m} \sum_{l=1}^L \max \left\{0, 1 - \sum_{j=1}^m q_{jl}\right\}.$$ 

By using the relation $\max\{0, x\} = |x| - \max\{0, -x\}$ and observing that $1 - \sum_{i=1}^n p_{ik}, 1 - \sum_{j=1}^m q_{jl}$ are integers for $(P, Q) \in \mathcal{P} \times \mathcal{Q}$, one has the following inequalities:

$$\max \left\{0, 1 - \sum_{i=1}^n p_{ik}\right\} = \left|1 - \sum_{i=1}^n p_{ik}\right| - \max \left\{0, \sum_{i=1}^n p_{ik} - 1\right\} \leq \left(\sum_{i=1}^n p_{ik} - 1\right)^2 - \left|1 - \sum_{i=1}^n p_{ik}\right| \max \left\{0, \sum_{i=1}^n p_{ik} - 1\right\} \leq \left(\sum_{i=1}^n p_{ik} - 1\right)^2 - \max \left\{0, \sum_{i=1}^n p_{ik} - 1\right\}^2.$$ 

Similarly, one has

$$\max \left\{0, 1 - \sum_{j=1}^m q_{jl}\right\} \leq \left(\sum_{j=1}^m q_{jl} - 1\right)^2 - \max \left\{0, \sum_{j=1}^m q_{jl} - 1\right\}^2.$$
Combining the preceding inequalities, one obtains
\[
d((P, Q), \mathcal{P} \times \mathcal{Q}) \leq \sqrt{2n} \left( \sum_{k=1}^{K} \left( \sum_{i=1}^{n} p_{ik} - 1 \right)^2 \right)
- \sum_{k=1}^{K} \max \left\{ 0, \sum_{i=1}^{n} p_{ik} - 1 \right\}^2
+ \sqrt{2m} \left( \sum_{l=1}^{L} \left( \sum_{j=1}^{m} q_{jl} - 1 \right)^2 \right)
- \sum_{l=1}^{L} \max \left\{ 0, \sum_{j=1}^{m} q_{jl} - 1 \right\}^2.
\] (2.8)

Denote by
\[
\varphi(P, Q) := 2 \left( \sum_{i=1}^{n} (1 - \max_{1 \leq k \leq K} \{ p_{ik} \}) + \sum_{j=1}^{m} (1 - \max_{1 \leq l \leq L} \{ p_{jl} \}) \right)
\] (2.9)
and
\[
\psi(P, Q) := \sqrt{2n} \left( \sum_{k=1}^{K} \left( \sum_{i=1}^{n} p_{ik} - 1 \right)^2 \right)
- \sum_{k=1}^{K} \max \left\{ 0, \sum_{i=1}^{n} p_{ik} - 1 \right\}^2
+ \sqrt{2m} \left( \sum_{l=1}^{L} \left( \sum_{j=1}^{m} q_{jl} - 1 \right)^2 \right)
- \sum_{l=1}^{L} \max \left\{ 0, \sum_{j=1}^{m} q_{jl} - 1 \right\}^2.
\] (2.10)

Obviously \( \varphi, \psi \) are nonnegative on \( S_p \times S_q \) and
\[
S_p^1 \times S_q^1 = \{ x \in S_p \times S_q : \varphi(P, Q) = 0 \}
\]
\[
\mathcal{P} \times \mathcal{Q} = \{ x \in S_p^1 \times S_q^1 : \psi(P, Q) = 0 \}.
\]

The following lemma gives an upper bound for \( d((P, Q), \mathcal{P} \times \mathcal{Q}) \) in terms of the function \( \varphi \) and \( \psi \):

**Lemma 4.** For all \( (P, Q) \in S_p \times S_q \), one has
\[
d((P, Q), \mathcal{P} \times \mathcal{Q}) \leq \tau \varphi(P, Q) + \psi(P, Q),
\] (2.11)
\[
\text{where } \tau = t + 1 = \max \{ 4n \sqrt{2n}, 4m \sqrt{2m} \} + 1.
\]
Proof. Observe that \( \psi \) is Lipschitz on \( S_p \times S_q \) with constant \( t \). Let \((P, Q) \in S_p \times S_q \). Let \((P_1, Q_1) \in S^1_p \times S^1_q \) be the projection of \((P, Q) \) into \( S^1_p \times S^1_q \). One has

\[
\begin{align*}
&d((P, Q), \mathcal{P} \times \mathcal{Q}) \leq d((P, Q), S^1_p \times S^1_q) + d((P_1, Q_1), \mathcal{P} \times \mathcal{Q}) \\
&\leq (t+1)d((P, Q), S^1_p \times S^1_q) + \psi(P_1, Q_1) \\
&\leq (t+1)d((P, Q), S^1_p \times S^1_q) + \psi(P, Q) \\
&\leq (t+1)\tau \psi(P, Q) + \psi(P, Q).
\end{align*}
\]

The lemma is then proved.

From lemmas 2 and 4, we derive that problem 1.2 is equivalent to the following continuous optimization program:

\[
\begin{align*}
\min \{ F(P, Q, V) := f(P, Q, V) + \gamma \tau \psi(P, Q) + \gamma \psi(P) : (P, Q, V) \in S_p \times S_q \times V \}, \\
(2.12)
\end{align*}
\]

for all \( \gamma > \mathcal{L} \), where \( \mathcal{L} \) is defined by equation 2.3. In the next section, we consider the block clustering problem in the form of equation 2.12 with a sufficiently large value \( \gamma \), say, \( \gamma > \sqrt{2} \max\{\alpha^2, \beta^2\} \max\{nK, mL\} \).

3 DC Programming and DCA for Solving the Equivalent Block Clustering Problem 2.12

3.1 A Brief Presentation of DC Programming and DCA. DC programming and DCA constitute the backbone of smooth and nonsmooth nonconvex programming and global optimization. They address general DC programs of the form

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

where \( g, h \in \Gamma_0(\mathbb{R}^p) \), the convex cone of all lower semicontinuous proper convex functions defined on \( \mathbb{R}^p \) and taking values in \( \mathbb{R} \cup \{+\infty\} \). Such a function \( f \) is called a DC function, and \( g - h \) a DC decomposition of \( f \), while \( g \) and \( h \) are the DC components of \( f \). The convex constraint \( x \in C \) can be incorporated in the objective function of \( (P_{dc}) \) by using the indicator function of \( C \) denoted by \( \chi_C \), which is defined by \( \chi_C(x) = 0 \) if \( x \in C \), and \( +\infty \) otherwise:

\[
\inf \{ f(x) := g(x) - h(x) : x \in C \} = \inf \{ \chi_C(x) + g(x) - h(x) : x \in \mathbb{R}^p \}.
\]
For a convex function θ, the subdifferential of θ at \( x_0 \in \text{dom} \theta := \{x \in \mathbb{R}^p : \theta(x_0) < +\infty \} \), denoted by \( \partial \theta(x_0) \), is defined by

\[
\partial \theta(x_0) := \{y \in \mathbb{R}^p : \theta(x) \geq \theta(x_0) + \langle x - x_0, y \rangle, \forall x \in \mathbb{R}^p\}.
\] (3.1)

The subdifferential \( \partial \theta(x_0) \) generalizes the derivative in the sense that \( \theta \) is differentiable at \( x_0 \) if and only if \( \partial \theta(x_0) \equiv \{ \nabla_x \theta(x_0) \} \).

The complexity of DC programs resides, of course, in the lack of practical optimal globality conditions. Local optimality conditions are then useful in DC programming.

A point \( x^* \) is said to be a local minimizer of \( g - h \) if \( g(x^*) - h(x^*) \) is finite and there exists a neighborhood \( \mathcal{U} \) of \( x^* \) such that

\[
g(x^*) - h(x^*) \leq g(x) - h(x), \quad \forall x \in \mathcal{U}.
\] (3.2)

The necessary local optimality condition for (primal) DC program \((P_{dc})\) is given by

\[
\emptyset \neq \partial h(x^*) \subset \partial g(x^*).
\] (3.3)

Condition 3.3 is also sufficient (for local optimality) in many important classes of DC programs (see Le Thi & Pham Dinh, 1997, 2005).

A point \( x^* \) is said to be a critical point of \( g - h \) if

\[
\partial h(x^*) \cap \partial g(x^*) \neq \emptyset.
\] (3.4)

Relation 3.4 is in fact the generalized Karush-Kuhn-Tucker (KTT) condition for \((P_{dc})\), and \( x^* \) is also called a generalized KKT point.

DCA is based on local optimality conditions and duality in DC programming. The main idea of DCA is simple: each iteration \( r \) of DCA approximates the concave part \( -h \) by its affine majorization (which corresponds to taking \( y^r \in \partial h(x^r) \)) and minimizes the resulting convex function.

The generic DCA scheme can be described as follows:

**DCA Scheme**

**Initialization:** Let \( x^0 \in \mathbb{R}^p \) be a guess, set \( r := 0 \).

**Repeat**

- Calculate some \( y^r \in \partial h(x^r) \)
- Calculate \( x^{r+1} \in \text{arg min}\{g(x) - [h(x^r) + \langle x - x^r, y^r \rangle] : x \in \mathbb{R}^p\} \) \((P_r)\)
- Increase \( r \) by 1

**Until** convergence of \( \{x^r\} \).
Note that \((P_r)\) is a convex optimization problem and so far is easy to solve.

Convergence properties of DCA and its theoretical basis can be found in Le Thi (1997), Le Thi and Pham Dinh (1997, 2005), and Pham Dinh and Le Thi (1998). For instance, it is important to mention that (for simplicity, we omit here the dual part of DCA):

- DCA is a descent method without line search: the sequence \(\{g(x^r) - h(x^r)\}\) is decreasing.
- If \(g(x^{r+1}) - h(x^{r+1}) = g(x^r) - h(x^r)\), then \(x^r\) is a critical point of \(g - h\). In such a case, DCA terminates at the \(r\)th iteration.
- If the optimal value \(\alpha\) of problem \((P_{dc})\) is finite and the infinite sequence \(\{x^r\}\) is bounded, then every limit point \(x^*\) of the sequence \(\{x^r\}\) is a critical point of \(g - h\).
- DCA has a linear convergence for DC programs.

For a complete study of DC programming and DCA see Le Thi (1997), Le Thi and Pham Dinh (1997, 2005), and Pham Dinh and Le Thi (1998). The solution of a nonconvex program \((P_{dc})\) by DCA must be composed of two stages: the search for an appropriate DC decomposition of \(f\) and that for a good initial point.

It should be noted that:

- The convex concave procedure (CCCP) for constructing the discrete time dynamical systems mentioned in Yuille and Rangarajan (2002) is a special case of DCA applied to smooth optimization.
- The SLA (successive linear approximation) algorithm developed in Bradley and Mangasarian (1998) is a version of DCA for concave minimization.
- The EM algorithm (Dempster, Laird, & Rubin, 1977) applied to the log-linear model is a special case of DCA.

We show below how the DCA can be applied to the equivalent block clustering problem, 2.12.

### 3.2 DC Formulation of Problem (2.12).

By directly checking, for

\[
C := \max\{\alpha^2 + 2\alpha, \beta^2 + 2\beta, 4\alpha + 2, 4\beta + 2\}, \tag{3.5}
\]

the function

\[
(x, y, z) \in \mathbb{R}^3 \mapsto \frac{C}{2} (x^2 + y^2 + z^2) - xyz^2
\]

is convex on \([0, 1] \times [0, 1] \times [\alpha, \beta]\). Therefore, for

\[
\rho_{1,P} \geq CmL, \quad \rho_{1,Q} \geq CnK, \quad \rho_V \geq Cmn, \tag{3.6}
\]
the function
\[
\frac{\rho_1}{2} \|P\|^2 + \frac{\rho_2}{2} \|Q\|^2 + \frac{\rho_3}{2} \|V\|^2 - f(P, Q, V)
\]
is convex on \( S_p \times S_q \times V \).

Likewise, for \( \rho_2 \geq 4\sqrt{2n}\gamma \) and \( \rho_3 \geq 4\sqrt{2m}\gamma \), the following functions are convex:
\[
\frac{\rho_2}{2} \|P\|^2 - \sqrt{2n}\gamma \sum_{k=1}^{K} \left( \sum_{i=1}^{n} p_{ik} - 1 \right)^2,
\]
\[
\frac{\rho_3}{2} \|Q\|^2 - \sqrt{2m}\gamma \sum_{l=1}^{L} \left( \sum_{j=1}^{m} q_{jl} - 1 \right)^2.
\]

Hence, we can recast problem 2.12 as a DC program as follows.
\[
\min \{ F(P, Q, V) = G(P, Q, V) - H(P, Q, V) : (P, Q, V) \in S_p \times S_q \times V \},
\]
where
\[
G(P, Q, V) = \frac{\rho_p}{2} \|P\|^2 + \frac{\rho_Q}{2} \|Q\|^2 + \frac{\rho_V}{2} \|V\|^2;
\]
\[
\rho_p \geq CmL + 4\sqrt{2n}\gamma; \quad \rho_Q \geq CnK + 4\sqrt{2m}\gamma; \quad \rho_V \geq Cmn;
\]
\[
H(P, Q, V) = \frac{\rho_p}{2} \|P\|^2 + \frac{\rho_Q}{2} \|Q\|^2 + \frac{\rho_V}{2} \|V\|^2 - f(P, Q, V) -
\]
\[
-\sqrt{2n}\gamma \sum_{k=1}^{K} \left( \sum_{i=1}^{n} p_{ik} - 1 \right)^2 - \sqrt{2m}\gamma \sum_{l=1}^{L} \left( \sum_{j=1}^{m} q_{jl} - 1 \right)^2 + \sqrt{2}n\gamma \sum_{k=1}^{K} \max \left\{ 0, \sum_{i=1}^{n} p_{ik} - 1 \right\}^2
\]
\[
+ \sqrt{2}m\gamma \sum_{l=1}^{L} \max \left\{ 0, \sum_{j=1}^{m} q_{jl} - 1 \right\}^2 + 2\gamma \tau \sum_{i=1}^{n} \max_{1 \leq k \leq k} \{p_{ik}\} + 2\gamma \tau \sum_{j=1}^{m} \max_{1 \leq l \leq L} \{q_{jl}\}.
\]

3.3 DCA for Solving DC Program 3.7. According to the general DCA scheme described in section 3.1, applying DCA to equation 3.7 amounts to computing two sequences \( \{ (Y^r, Z^r, W^r) \} \) and \( \{ (P^r, Q^r, V^r) \} \) in the way...
that \((Y', Z', W') \in \partial H(P', Q', V')\) and \((P'^{r+1}, Q'^{r+1}, V'^{r+1})\) solves the convex program of the form \((P_r)\).

### 3.3.1 Computing \(\partial H(P, Q, V)\)

For simplicity of notation, we set

\[
P_i = (p_{ik})_{1 \leq k \leq K}, \quad Q_j = (q_{jl})_{1 \leq l \leq L}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, m,
\]

\[
P = (P_1, P_2, \ldots, P_n)^T, \quad Q = (Q_1, Q_2, \ldots, Q_m)^T,
\]

\[
h_{ip}(P_i) := \max_{1 \leq k \leq K} \{p_{ik}\}; \quad K_{ip} = \{k \in \{1, \ldots, K\} : p_{ik} = \max_{1 \leq k \leq K} \{p_{ik}\}\},
\]

\[
h_{jq}(Q_j) := \max_{1 \leq l \leq L} \{q_{jl}\}; \quad L_{jq} = \{l \in \{1, \ldots, L\} : q_{jl} = \max_{1 \leq l \leq L} \{q_{jl}\}\}.
\]

Then,

\[
\partial h_{ip}(P_i) = \left\{ P_i' = (p_{ik}') : \text{if } k \notin K_{ip}, \quad \sum_{k=1}^{K} p_{ik}' = 1 \right\}.
\]

Similarly,

\[
\partial h_{jq}(Q_j) = \left\{ Q_j' = (q_{jl}') : \text{if } l \notin L_{jq}, \quad \sum_{l=1}^{L} q_{jl}' = 1 \right\}.
\]

Hence \((Y', Z', W') \in \partial H(P', Q', V')\) can be determined as

\[
Y'_r = (Y'_i), \quad Z'_r = (Z'_j), \quad W'_r = (W'_{kl}),
\]

where

\[
W'_{kl} = \rho Y v'_{kl} - 2 \sum_{i=1}^{n} \sum_{j=1}^{m} p'_{ik} q'_{jl} (v'_{kl} - x_{ij}), \quad (3.9)
\]

\[
Y'_r = \rho_P P'_r - \left[ \nabla P f(P', Q', V') \right]_i - 2 \sqrt{2n} \gamma \left( \sum_{i=1}^{n} (p'_{ik} - 1) \right),
\]

\[
+ 2 \sqrt{2n} \gamma \left( \max_{1 \leq k \leq K} \left\{ 0, \sum_{i=1}^{n} p'_{ik} - 1 \right\} \right) + 2 \gamma \tau \partial h_{ip}(P'_r), \quad (3.10)
\]

\[
Z'_r = \rho_Q Q'_r - \left[ \nabla Q f(P', Q', V') \right]_j - 2 \sqrt{2m} \gamma \left( \sum_{j=1}^{m} (q'_{jl} - 1) \right),
\]

\[
+ 2 \sqrt{2m} \gamma \left( \max_{1 \leq l \leq L} \left\{ 0, \sum_{j=1}^{m} q'_{jl} - 1 \right\} \right) + 2 \gamma \tau \partial h_{jq}(Q'_r). \quad (3.11)
\]
3.3.2 Computing \((P^{r+1}, Q^{r+1}, V^{r+1})\). This is equivalent to solving the convex program of the form \((P_r)\), say,
\[
\min\{G(P, Q, V) - \langle (P, Q, V), (Y', Z', W') \rangle : (P, Q, V) \in S_p \times S_q \times V\}.
\]
It is easy to see that \((\text{Proj}_C\) denotes the projection on the set \(C\))
\[
P^{r+1}_i = \text{Proj}_{\Delta_K}(Y'_i/\rho_P);
Q^{r+1}_j = \text{Proj}_{\Delta_L}(Z'_j/\rho_Q);
V^{r+1} = \text{Proj}_C(W'/\rho_V),
\]
where \(\Delta_K, \Delta_L\) are the \((K-1)\)--simplex, the \((L-1)\)--simplex defined as
\[
\Delta_K = \left\{ x \in \mathbb{R}^K : \sum_{k=1}^K x_k = 1, x_k \in [0, 1] \right\};
\]
\[
\Delta_L = \left\{ x \in \mathbb{R}^L : \sum_{l=1}^L x_l = 1, x_l \in [0, 1] \right\},
\]
and \(C = [\alpha, \beta]^{K \times L}\). Since the projections on a simplex and on a box are explicitly determined, the computation of the sequence \((P^{r+1}, Q^{r+1}, V^{r+1})\) is explicit too. The proposed DCA applied to the DC program, 3.7, then is explicit and, consequently, inexpensive. It can be summarized as follows:

**DCA**

**Initialization**: Choose the two matrices of cluster memberships \((P^0, Q^0)\) and the cluster centers \(V^0\). Let \(\epsilon > 0\) be sufficiently small, \(r \leftarrow 0\).

**Repeat**
- Set \((Y', Z', W')\) according to equations 3.9–3.11.
- Set \(P^{r+1}_i = \text{Proj}_{\Delta_K}(Y'_i/\rho_P); Q^{r+1}_j = \text{Proj}_{\Delta_L}(Z'_j/\rho_Q); V^{r+1} = \text{Proj}_C(W'/\rho_V)\).
- \(r \leftarrow r + 1\).

**Until** \(\|P^r, Q^r, V^r) - (P^{r-1}, Q^{r-1}, V^{r-1})\| \leq \epsilon\).

4 Computational Experiments

To our knowledge, there is no algorithm for globally solving problem 1.2. Since the approaches developed in Rosmalen et al. (2005) considered the same clustering criterion (the squared error 1.1), we compare our algorithm DCA with two-mode K-means and two-mode fuzzy clustering, the two best
algorithms studied in Rosmalen et al. (2005). We also compare DCA with the BCEM algorithm (Govaert & Nadif, 2003) (an EM-based algorithm for block clustering), which is an efficient standard method for block clustering. BCEM is based on probabilistic mixture models, but it has been shown (Celeux & Govaert, 1992, 1993; Govaert & Nadif, 2003) that some of the most popular heuristic clustering approaches, including K-mean, can be viewed as approximate estimations of probability models. Descriptions of these three comparative algorithms are given in the appendix. Before presenting numerical results, we give a brief description of the data, as well as the way to set up experiments.

4.1 Data. To illustrate the performance of algorithms, we performed numerical tests on three types of data: real data sets, simulated continuous data sets (Data1–Data12), and simulated binary data sets (Data13–Data17). The real data sets are available online (http://algorithmics.molgen.mpg.de/Static/Supplements/CompCancer/datasets.htm). Only the number of row clusters ($K$) is assumed to be known. We give a brief description of real data sets:

- Brain Cancer 1 contains 50 gliomas: 28 glioblastomas and 22 anaplastic oligodendrogliomas. This data set was used to build a two-class prediction of malignant gliomas (Nutt et al., 2003).
- Brain Cancer 2 contains 42 patient samples (10 medulloblastomas, 5 central nervous system (CNS) atypical teratoids or rhabdoid tumors, 5 renal and extrarenal rhabdoid tumors, and 8 supratentorial primitive neuroectodermal tumors, as well as 10 nonembryonal brain tumors—malignant glioma—and 4 normal human cerebella). Pomeroy et al. (2002) used this data set to distinguish different CNS embryonal tumours from each other.
- Multi Tissue contains 190 tumor samples, spanning 14 common tumor types. This data set was used in Ramaswamy et al. (2001) to determine whether the diagnosis of multiple common adult malignancies could be achieved purely by molecular classification.
- Lung consists of 186 lung tumor samples and 17 normal lung tissues (NL). The lung tumors included 139 adenocarcinoma (AD), 6 small-cell lung cancer (SCLC), 20 pulmonary carcinoids (COID), and 21 squamous cell lung carcinomas (SQ). Bhattacharjee et al. (2001) used this data set for studying a new molecular taxonomy of tumors and demonstrating the potential power of gene expression profiling in lung cancer diagnosis.

We use the same way as proposed in Rosmalen et al. (2005) to generate continuous data sets (Data1–Data12). For each of four data dimensions $(n, m, K, L) \in \{(20, 20, 5, 4), (60, 60, 6, 6), (100, 100, 10, 10), (150, 100, 7, 7)\),
Table 1: Simulated Continuous Data Sets.

<table>
<thead>
<tr>
<th>Name</th>
<th>N</th>
<th>M</th>
<th>K</th>
<th>L</th>
<th>E</th>
<th>Name</th>
<th>N</th>
<th>M</th>
<th>K</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data1</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>4</td>
<td>0.5</td>
<td>Data2</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Data3</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>Data4</td>
<td>60</td>
<td>60</td>
<td>6</td>
<td>6</td>
<td>0.5</td>
</tr>
<tr>
<td>Data5</td>
<td>60</td>
<td>60</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>Data6</td>
<td>60</td>
<td>60</td>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Data7</td>
<td>100</td>
<td>100</td>
<td>10</td>
<td>10</td>
<td>0.5</td>
<td>Data8</td>
<td>100</td>
<td>100</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Data9</td>
<td>100</td>
<td>100</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>Data10</td>
<td>150</td>
<td>100</td>
<td>7</td>
<td>7</td>
<td>0.5</td>
</tr>
<tr>
<td>Data11</td>
<td>150</td>
<td>100</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>Data12</td>
<td>150</td>
<td>100</td>
<td>7</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

The procedure for generating continuous data sets is as follows:

1. Randomly generate the row-classification matrix $P^* \in \{0, 1\}^{n \times K}$.
2. Randomly generate the column-classification matrix $Q^* \in \{0, 1\}^{m \times L}$.
3. Generate the matrix of prototypes $V^* \in \mathbb{R}^{K \times L}$ according to $K \times L$ independent normal distribution with mean 0 and standard deviation equal to 1.
4. Generate three noise matrices $E \in \mathbb{R}^{n \times m}$ according to $n \times m$ independent normal distribution with mean 0 and standard deviation equal to 0.5, 1 and 2. Note that standard deviations of 0.5 and 1 give a reasonable amount of noise in the simulated data, whereas a standard deviation of 2 can make clustering difficult.
5. Compute the data matrix $X$ by $X = P^* V^* Q^* + E$.

The information on these data sets is summarized in Table 1.

For generating binary data sets (Data13–Data17), we use a procedure that is similar to the one given in Govaert and Nadif (2003):

1. Generate the row-classification matrix $P^* \in \{0, 1\}^{n \times K}$ according to the mixing proportions $\pi_1 = \pi_2 = \cdots = \pi_K = 1/K$.
2. Generate the column-classification matrix $Q^* \in \{0, 1\}^{m \times L}$ according to the mixing proportions $\eta_1 = \eta_2 = \cdots = \eta_L = 1/L$.
3. Randomly generate the matrix of prototypes $V^* \in \{0, 1\}^{K \times L}$.
4. Randomly generate $\epsilon_{kl} \in [0, \frac{1}{2}]$ for $k = 1, \ldots, K, l = 1, \ldots, L$.
5. Compute $\alpha_{kl}$ according to equation A.12, for $k = 1, \ldots, K, l = 1, \ldots, L$.
6. Generate $x_{ij}$ for $i = 1, \ldots, n, j = 1, \ldots, m$ according to the Bernoulli distribution parameterized by $\alpha_{kl}$, where $(k, l)$ is the bicluster containing $x_{ij}$.

4.2 Setup for Experiments and Parameters. All clustering algorithms were implemented in the Visual C++ 2008, and performed on a PC Intel i5.
CPU650, 3.2 GHz of 4 GB RAM. We stop all algorithms with the tolerance $\epsilon = 10^{-4}$.

Rosmalen et al. (2005) stated that if the initial value of $s$ is too high, the two-mode fuzzy algorithm often reaches a saddle point. To prevent this, we set the fuzzy step size $\gamma$ equal to 0.9 and the initial value of $s$ equal to 1.2. The threshold value $s_{\text{min}}$ is set to 1.01.

Our experiments have two parts: on continuous data and on binary data. In the first experiment, we compare the performance of DCA, two-mode K-means and two-mode fuzzy on the simulated continuous data sets containing 12 problems (Data1–Data12) and on 4 real data sets.

The following criteria were used to compare the performance of three algorithms DCA, two-mode K-means and two-mode fuzzy: VAF (variance accounted for criterion), the error rate, (in comparing with true clusters in case of simulated data where true clusters are known) and the CPU time in seconds.

The VAF is defined as

$$VAF = 1 - \frac{\sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ik} q_{jl} (x_{ij} - v_{kl})^2}{\sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} - \bar{x})^2},$$

where $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$. It is clear that maximizing VAF corresponds to minimizing $f(P, Q, V)$.

The error rate of the partitions $(P, Q)$ given by an algorithm in comparing with true partitions $(P^*, Q^*)$ in block clustering is defined from one-way clustering as shown in Govaert and Nadif (2008):

$$e((P, Q), (P^*, Q^*)) = e(P, P^*) + e(Q, Q^*) - e(P, P^*) e(Q, Q^*),$$

where $e(P, P^*) = 1 - \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} p_{ik}^*$, $e(Q, Q^*) = 1 - \frac{1}{m} \sum_{j=1}^{m} \sum_{l=1}^{L} q_{jl} q_{jl}^*$.

For real data sets, we know only the true row clusters; the true column clusters are not available. Hence, the evaluation is done by the VAF criterion. For each real data set, we performed the tests with five values of $L$: $L \in \{5, 10, 15, 20, 25\}$.

In the second experiment, we added the BCEM algorithm. Here we compare the error rate and CPU times in seconds of the four algorithms on binary data.

For every data instance, we perform each algorithm 20 times from 20 random starting points and report the best, the mean, and the standard
deviation of each criterion. Bold values in the result tables (Tables 2 and 3) are the best value for each data instance.

4.3 Experimental Results and Comments

4.3.1 Experiment 1: Continuous Data. The computational results on simulated data sets (resp. real data sets) are reported in Table 2 (resp. Table 3).

From Table 2 we observe that in all experiments, among three algorithms, DCA is the best in terms of VAF and error rate. The results of DCA are quite stable with different starting points, while two-mode K-means and two-mode fuzzy are sensitive to them. Moreover, not surprisingly, the larger the standard deviation of the noise matrix $E$ is, the more difficult the problem is, and then the larger error rate given by each algorithm would be.

Similarly, for real data sets, DCA is always the best in terms of VAF criterion. For each algorithm and in all real data sets, the maximal value of VAF when $L$ varies corresponds to the case $L = 10$. In fact, the VAF value increases at first, reaches the best value at $L = 10$, and then decreases. According to this phenomenon, we can conclude that $L = 10$ is the optimal number of column clusters in these data sets.

Concerning computation time, DCA is faster than two-mode fuzzy and slightly slower than two-mode K-means: the average CPU time of two-mode K-means, DCA, two-mode fuzzy is, respectively, 1.3, 2.7, 9.9 seconds in simulated data sets and 12.5, 12.8, 14 seconds in real data sets.

4.3.2 Experiment 2: Binary Data. The computational results (error rate and CPU Time) are reported in Table 4.

We observe from Table 4 that DCA is the best in terms of error rate: DCA is better than two-mode K-means and two-mode fuzzy for all five data instances; DCA is better than BCEM on three of five data sets; and on the two other ones, DCA and BCEM give the same results. The average CPU times of two-mode K-means, two-mode fuzzy, BCEM and DCA are, respectively, 2.28, 3.02, 2.66, and 2.68 seconds.

5 Conclusion

We have rigorously studied the DC programming and DCA for biclustering. Based on an interesting result of the exact penalty, we recast the hard combinatorial optimization model of biclustering as a DC program in its elegant matrix formulation. We propose a DC decomposition of which the corresponding DCA is simple, elegant, and inexpensive: it consists of computing, at each iteration, the projection of points onto a simplex or a box that are given in the explicit form. Computational experiments show the efficiency and the superiority of DCA with respect to the two-mode K-means, two-mode fuzzy clustering, and BCEM. We are convinced that it is possible
Table 2: Comparative Results on Simulated Continuous Data Sets.

<table>
<thead>
<tr>
<th>Name</th>
<th>VAF</th>
<th>Error Rate</th>
<th>Time</th>
<th>VAF</th>
<th>Error Rate</th>
<th>Time</th>
<th>VAF</th>
<th>Error Rate</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean±SD</td>
<td>Best Mean±SD</td>
<td>Best</td>
<td>Mean±SD</td>
<td>Best Mean±SD</td>
<td>Best</td>
<td>Mean±SD</td>
<td>Best Mean±SD</td>
<td>Best</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data1</td>
<td>0.73±0.05</td>
<td>0.76</td>
<td>0.17±0.04</td>
<td>0.16</td>
<td>0.02</td>
<td>0.75±0.05</td>
<td>0.78</td>
<td>0.17±0.04</td>
<td>0.16</td>
</tr>
<tr>
<td>Data2</td>
<td>0.53±0.04</td>
<td>0.54</td>
<td>0.26±0.04</td>
<td>0.24</td>
<td>0.02</td>
<td>0.44±0.03</td>
<td>0.45</td>
<td>0.29±0.03</td>
<td>0.28</td>
</tr>
<tr>
<td>Data3</td>
<td>0.24±0.05</td>
<td>0.26</td>
<td>0.45±0.03</td>
<td>0.43</td>
<td>0.29</td>
<td>0.22±0.07</td>
<td>0.26</td>
<td>0.41±0.05</td>
<td>0.39</td>
</tr>
<tr>
<td>Data4</td>
<td>0.75±0.04</td>
<td>0.77</td>
<td>0.9±0.03</td>
<td>0.07</td>
<td>0.65</td>
<td>0.42±0.05</td>
<td>0.43</td>
<td>0.10±0.01</td>
<td>0.09</td>
</tr>
<tr>
<td>Data5</td>
<td>0.43±0.05</td>
<td>0.45</td>
<td>0.15±0.03</td>
<td>0.16</td>
<td>0.91</td>
<td>0.32±0.04</td>
<td>0.34</td>
<td>0.19±0.04</td>
<td>0.20</td>
</tr>
<tr>
<td>Data6</td>
<td>0.18±0.05</td>
<td>0.20</td>
<td>0.41±0.04</td>
<td>0.40</td>
<td>0.75</td>
<td>0.12±0.04</td>
<td>0.14</td>
<td>0.49±0.03</td>
<td>0.47</td>
</tr>
<tr>
<td>Data7</td>
<td>0.72±0.03</td>
<td>0.74</td>
<td>0.05±0.02</td>
<td>0.04</td>
<td>1.9</td>
<td>0.59±0.03</td>
<td>0.61</td>
<td>0.10±0.02</td>
<td>0.09</td>
</tr>
<tr>
<td>Data8</td>
<td>0.44±0.05</td>
<td>0.46</td>
<td>0.10±0.04</td>
<td>0.09</td>
<td>2</td>
<td>0.32±0.03</td>
<td>0.33</td>
<td>0.12±0.04</td>
<td>0.11</td>
</tr>
<tr>
<td>Data9</td>
<td>0.12±0.04</td>
<td>0.15</td>
<td>0.43±0.04</td>
<td>0.41</td>
<td>2</td>
<td>0.12±0.05</td>
<td>0.15</td>
<td>0.41±0.03</td>
<td>0.39</td>
</tr>
<tr>
<td>Data10</td>
<td>0.74±0.04</td>
<td>0.77</td>
<td>0.07±0.03</td>
<td>0.06</td>
<td>2</td>
<td>0.57±0.04</td>
<td>0.59</td>
<td>0.10±0.05</td>
<td>0.09</td>
</tr>
<tr>
<td>Data11</td>
<td>0.50±0.03</td>
<td>0.52</td>
<td>0.12±0.03</td>
<td>0.10</td>
<td>2</td>
<td>0.40±0.05</td>
<td>0.41</td>
<td>0.18±0.04</td>
<td>0.17</td>
</tr>
<tr>
<td>Data12</td>
<td>0.19±0.05</td>
<td>0.20</td>
<td>0.48±0.05</td>
<td>0.46</td>
<td>3</td>
<td>0.19±0.03</td>
<td>0.20</td>
<td>0.50±0.05</td>
<td>0.49</td>
</tr>
</tbody>
</table>
Table 3: Comparative Results on Real Data Sets.

<table>
<thead>
<tr>
<th>Name</th>
<th>N</th>
<th>M</th>
<th>K</th>
<th>L</th>
<th>Size</th>
<th>Two-Mode K-Means</th>
<th>Two-Mode Fuzzy</th>
<th>DCA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VAF</td>
<td>Best</td>
<td>Time</td>
</tr>
<tr>
<td>Brain</td>
<td>50</td>
<td>1377</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>0.65 ± 0.07</td>
<td>0.68</td>
<td>5</td>
</tr>
<tr>
<td>Cancer 1</td>
<td>10</td>
<td>0.75 ± 0.08</td>
<td>0.77</td>
<td>6</td>
<td>0.64 ± 0.07</td>
<td>0.66</td>
<td>8</td>
<td>0.82 ± 0.04</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.73 ± 0.08</td>
<td>0.76</td>
<td>8</td>
<td>0.63 ± 0.08</td>
<td>0.65</td>
<td>10</td>
<td>0.81 ± 0.04</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.66 ± 0.05</td>
<td>0.68</td>
<td>13</td>
<td>0.60 ± 0.08</td>
<td>0.62</td>
<td>18</td>
<td>0.75 ± 0.06</td>
</tr>
<tr>
<td>Brain</td>
<td>42</td>
<td>1379</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0.68 ± 0.02</td>
<td>0.69</td>
<td>4</td>
</tr>
<tr>
<td>Cancer 2</td>
<td>10</td>
<td>0.77 ± 0.04</td>
<td>0.79</td>
<td>4</td>
<td>0.65 ± 0.04</td>
<td>0.67</td>
<td>4</td>
<td>0.82 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.75 ± 0.03</td>
<td>0.76</td>
<td>6</td>
<td>0.74 ± 0.05</td>
<td>0.76</td>
<td>8</td>
<td>0.84 ± 0.03</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.70 ± 0.04</td>
<td>0.72</td>
<td>14</td>
<td>0.68 ± 0.06</td>
<td>0.70</td>
<td>12</td>
<td>0.80 ± 0.04</td>
</tr>
<tr>
<td>Multi</td>
<td>190</td>
<td>1363</td>
<td>14</td>
<td>5</td>
<td>5</td>
<td>0.75 ± 0.03</td>
<td>0.76</td>
<td>8</td>
</tr>
<tr>
<td>Tissue</td>
<td>10</td>
<td>0.75 ± 0.04</td>
<td>0.76</td>
<td>9</td>
<td>0.67 ± 0.03</td>
<td>0.69</td>
<td>13</td>
<td>0.86 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.75 ± 0.03</td>
<td>0.76</td>
<td>14</td>
<td>0.67 ± 0.04</td>
<td>0.69</td>
<td>16</td>
<td>0.81 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.71 ± 0.04</td>
<td>0.72</td>
<td>24</td>
<td>0.61 ± 0.03</td>
<td>0.62</td>
<td>21</td>
<td>0.81 ± 0.04</td>
</tr>
<tr>
<td>Lung</td>
<td>203</td>
<td>1543</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0.75 ± 0.01</td>
<td>0.76</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.67 ± 0.02</td>
<td>0.68</td>
<td>19</td>
<td>0.72 ± 0.02</td>
<td>0.73</td>
<td>22</td>
<td>0.76 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.73 ± 0.03</td>
<td>0.74</td>
<td>23</td>
<td>0.71 ± 0.03</td>
<td>0.72</td>
<td>28</td>
<td>0.84 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.77 ± 0.03</td>
<td>0.79</td>
<td>28</td>
<td>0.71 ± 0.03</td>
<td>0.72</td>
<td>30</td>
<td>0.85 ± 0.03</td>
</tr>
</tbody>
</table>
Table 4: Computational Results on Simulated Binary Data Sets.

<table>
<thead>
<tr>
<th>Name</th>
<th>n</th>
<th>m</th>
<th>K</th>
<th>L</th>
<th>Size</th>
<th>Error rate</th>
<th>Error rate</th>
<th>Error rate</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean±SD</td>
<td>Best</td>
<td>Time</td>
<td>Mean±SD</td>
<td>Best</td>
<td>Time</td>
<td>Mean±SD</td>
<td>Best</td>
<td>Time</td>
</tr>
<tr>
<td>Data13</td>
<td>60</td>
<td>90</td>
<td>2</td>
<td>3</td>
<td>0.09±0.02</td>
<td>0.07</td>
<td>0.8</td>
<td>0.12±0.03</td>
<td>0.10</td>
</tr>
<tr>
<td>Data14</td>
<td>50</td>
<td>60</td>
<td>5</td>
<td>2</td>
<td>0.13±0.02</td>
<td>0.12</td>
<td>1.2</td>
<td>0.15±0.01</td>
<td>0.14</td>
</tr>
<tr>
<td>Data15</td>
<td>100</td>
<td>60</td>
<td>4</td>
<td>3</td>
<td>0.12±0.02</td>
<td>0.11</td>
<td>1.5</td>
<td>0.18±0.03</td>
<td>0.16</td>
</tr>
<tr>
<td>Data16</td>
<td>300</td>
<td>200</td>
<td>5</td>
<td>4</td>
<td>0.09±0.02</td>
<td>0.08</td>
<td>3.7</td>
<td>0.20±0.02</td>
<td>0.19</td>
</tr>
<tr>
<td>Data17</td>
<td>400</td>
<td>180</td>
<td>8</td>
<td>5</td>
<td>0.20±0.04</td>
<td>0.18</td>
<td>4.2</td>
<td>0.25±0.04</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>0.04±0.01</td>
<td>0.03</td>
<td>1.2</td>
<td>0.10±0.01</td>
<td>0.09</td>
<td>1.3</td>
<td>0.09±0.03</td>
<td>0.08</td>
<td>2.1</td>
</tr>
</tbody>
</table>
to improve the quality of our DCA by a more sophisticated implementation of the algorithm, and we hope that it will be useful to researchers and practitioners.

Appendix: Comparative Algorithms

A.1 Two-Mode K-Means Algorithm. The K-means algorithm, described by Hartigan (1975) is one of the simplest and fastest ways to obtain a good partition in one-mode clustering. In addition, it can easily be extended to handle two-mode clustering.

The so-called two-mode K-means algorithm (Rosmalen et al., 2005) aims to solve the biclustering problem, 1.2. It is based on the fact that when two of the three matrices $P$, $Q$, and $V$ are known, the optimal value of the third matrix can be easily computed.

If $P$ and $Q$ are known, the optimal cluster prototype $V$ can be computed simply as the average of the elements of $X$ belonging to row cluster $k$ and column cluster $l$:

$$v_{kl} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ik} q_{jl} x_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ik} q_{jl}}, \quad k = 1, \ldots, K; l = 1, \ldots, L. \quad (A.1)$$

When $V$ and either $P$ or $Q$ are known, the problem of minimizing $f(P, Q, V)$ becomes a linear program, and its solution can be given in a closed-form expression. More precisely, when $V$ and $Q$ are known, the optimal matrix $P$ can be computed as

$$p_{ik} = \begin{cases} 1 & \text{if } c_{ik} = \min_{i \leq r \leq K} c_{ir}, \quad i = 1, \ldots, n; k = 1, \ldots, K, \\ 0 & \text{otherwise} \end{cases} \quad (A.2)$$

where $c_{ik} = \sum_{j=1}^{m} \sum_{l=1}^{L} q_{jl} (x_{ij} - v_{kl})^2$.

When $P$ and $V$ are known, the matrix $Q$ can be computed in a similar way:

$$q_{jl} = \begin{cases} 1 & \text{if } d_{jl} = \min_{i \leq r \leq L} d_{jr}, \quad j = 1, \ldots, m; l = 1, \ldots, L, \\ 0 & \text{otherwise} \end{cases} \quad (A.3)$$

where $d_{jl} = \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} (x_{ij} - v_{kl})^2$.

Finally, the two-mode K-means scheme is described as follows:

1. Choose initial $P$ and $Q$.
2. Repeat
   - $P$ and $Q$ are known, we update $V$ according to equation A.1.
   - Update $P$ by using equation A.2 with $Q$ and a new value of $V$. 

• Update again \( V \) with \( Q \) and a new value of \( P \) according to equation A.1.
• Update \( Q \) by using equation A.3 with a new value of \( P \) and \( V \).

3. Until the improvement of \( f(P, Q, V) \) between two steps is small enough.

It has been proved that the two-mode K-means algorithm always converges to a local optimum, as the value of the criterion \( f(P, Q, V) \) cannot increase in any step. However, updating \( P \) (resp. \( Q \)) can make one or more clusters become empty. This situation can be corrected by transferring the row or column object with the highest value of \( \sum_{k=1}^{K} p_{ik} c_{ik} \) (resp. \( \sum_{l=1}^{L} q_{jl} c_{jl} \)) to the empty cluster. Because this transfer always improves the value of the criterion, the algorithm still converges.

A.2.2 Two-Mode Fuzzy Algorithm. Like the two-mode K-means algorithm, the two-mode fuzzy algorithm introduced in Rosmalen et al. (2005) aims to solve the biclustering problem 1.2. It is composed of two phases. The first phase finds a fuzzy partition while the second phase consists of applying two-mode K-means algorithm from the fuzzy partition to get a crisp partition. In the first phase, the squared error \( f(P, Q, V) \) is approximated iteratively by a fuzzy two-mode clustering criterion \( f_s(P, Q, V) \) on which a fuzzy C-means based algorithm, two-mode fuzzy C-means algorithm, is applied.

A.2.1 Two-Mode Fuzzy C-Means Algorithm. Two-mode fuzzy C-means algorithm (Rosmalen et al., 2005) is an extension of the fuzzy C-means algorithm (Bezdek, 1981) (for one-mode clustering) to two-mode clustering. It minimizes the fuzzy two-mode clustering criterion defined by

\[
f_s(P, Q, V) := \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ik}^s q_{jl}^s (x_{ij} - v_{kl})^2
\]

with a fixed parameter \( s \geq 1 \) (called the fuzziness degree) and \( p_{ik}, q_{jl} \in [0, 1] \) being fuzzy clustering membership values.

For a given parameter \( s \), the two-mode Fuzzy algorithm (Rosmalen et al., 2005) can be described as follows:

1. Choose initial \( P \) and \( Q \), which can be either crisp or fuzzy, and calculate \( V \) by

\[
v_{kl} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ik}^s q_{jl}^s x_{i,j}}{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ik}^s q_{jl}^s} \quad \forall k = 1, \ldots, K; l = 1, \ldots, L. \quad \text{(A.6)}
\]
2. Repeat  
  • Define $c_{ik} = \sum_{j=1}^{m} \sum_{l=1}^{L} q_{jl}^s (x_{ij} - v_{kl})^2 \forall i = 1, \ldots, n; k = 1, \ldots, K$  
  and update $P$ by:
  $$p_{ik} = \frac{c_{ik}^{1/(1-s)}}{\sum_{k=1}^{K} c_{ik}^{1/(1-s)}} \quad \forall i = 1, \ldots, n; k = 1, \ldots, K.$$  
  (A.7)
  
  • Define $d_{jl} = \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik}^s (x_{ij} - v_{kl})^2 \forall j = 1, \ldots, m; l = 1, \ldots, L$  
  and update $Q$ by:
  $$q_{jl} = \frac{d_{jl}^{1/(1-s)}}{\sum_{l=1}^{L} d_{jl}^{1/(1-s)}} \quad \forall j = 1, \ldots, m; l = 1, \ldots, L.$$  
  (A.8)
  
  • Update $V$ according to equation A.6.

1. Until the decrease in $f_s(P, Q, V)$ is small.

The algorithm decreases the value of objective function in every iteration. Hence, this algorithm will always converge to a local minimum or a saddle point.

A.2.2 Fuzzy Steps. To get a good, crisp partition, Rosmalen et al. (2005) proposed a fuzzy steps procedure that starts with an initial value $s > 1$ and then iteratively decreases $s$ until a threshold value $s_{\text{min}}$ which should generally be set to a value slightly higher than 1 (when $s = 1$ one have $f_s(P, Q, V) = f(P, Q, V)$). The two-mode fuzzy C-means algorithm is used to minimize $f_s(P, Q, V)$.

Finally, the two-mode Fuzzy algorithm can be described as follow.

Phase 1: Fuzzy Steps

1. Choose a positive fuzzy step $\gamma < 1$, a threshold value $s_{\text{min}} > 1$, and an initial value of $s > s_{\text{min}}$.  
2. Choose $P_0, Q_0$, and compute $V_0$ according to equation A.6.  
3. Do  
   • Apply the two-mode fuzzy C-means algorithm on equation A.5 starting from $P_0, Q_0$, and $V_0$ to get a fuzzy partition $P_1, Q_1$, and $V_1$.  
   • Set $s = 1 + \gamma (s - 1)$, and set $P_0 = P_1, Q_0 = Q_1$, and $V_0 = V_1$.  

1. while $s > s_{\text{min}}$

Phase 2: Crisp partition

Apply the two-mode K-means algorithm on problem 1.2 starting from $(P_0, Q_0, V_0)$.

A.3 Block Mixture Models and an EM-Based Algorithm for Block Clustering. Clustering based on mixture models is a classical and powerful approach that has been extensively developed in various works. The mixture models provide a general framework that permits interpreting
some classical clustering criteria (including equation 1.1) as special cases (see Celeux & Govaert, 1992, 1993; Govaert & Nadif, 2003). The EM-based algorithms for mixture models are efficient standard approaches for clustering. For these reasons, we propose comparing our algorithm with BCEM, an efficient EM-based algorithm for block clustering in a particular model corresponding to the squared error criterion 1.1. We first briefly present a general block mixture model and then focus on a particular model and the BCEM algorithm for solving it (Govaert & Nadif, 2003). For details of EM-based algorithms for mixture models on different types of data, readers are referred to Celeux and Govaert (1992, 1993) and Govaert and Nadif (2003).

### A.3.1 Classical Mixture Models

In model-based clustering, it is assumed that the data are generated by a mixture of underlying probability distribution, where each component of the mixture represents a cluster. The distribution of the data is represented by a probability density function of the form (in the case of one-mode clustering)

$$ f(X_i, \theta) = \sum_{k=1}^{K} \pi_k \phi_k(X_i, \alpha_k), $$

where $\phi_k(X_i, \alpha_k)$ is the density of an object $X_i$ from the $k$th cluster (these densities belong to the same parametric family). The unknown parameter of this model is the vector $\theta = (\pi_1, \ldots, \pi_K, \alpha_1, \ldots, \alpha_K)$ with $\alpha_k$ being parameters of cluster $k$ and $\pi_k$, called the mixing proportions, the probability that an object belongs to the $k$th cluster. Then the density of the data matrix $X$ is given by

$$ f(X, \theta) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \phi_k(X_i, \alpha_k). $$

(A.9)

There are two standard clustering approaches based on this model. The first approach, named likelihood, computes the optimal parameter $\theta^*$ of the mixture model by maximizing the density function $f(X, \theta)$ that leads to maximizing the log-likelihood function,

$$ L(X, \theta) = \log f(X, \theta) = \sum_{i=1}^{n} \log \left( \sum_{k=1}^{K} \pi_k \phi_k(X_i, \alpha_k) \right), $$

by EM algorithm (Dempster et al., 1977). The partition is then deduced by the maximum a posteriori probability (MAP) rule. EM is an iterative method composed of two steps at each iteration: expectation (E-step) and maximization (M-step).
In the second approach (Symons, 1981), classification likelihood, the classification matrix \( P = (p_{ik})_{n \times K} \) (where \( p_{ik} = 1 \) if the object \( i \) belongs to cluster \( k \), \( p_{ik} = 0 \) otherwise) is added to the parameters to be estimated. The maximum likelihood estimation of these new parameters leads to optimizing the complete log likelihood that is

\[
L_c(X, P, \theta) = \log f(X, P, \theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log (\pi_k \phi_k(X_i, \alpha_k)).
\] (A.10)

For this model, a standard algorithm widely used is CEM (classification EM; Celeux & Govaert, 1992), a variant of EM where the classification step (C-step) to determine the classification matrix \( P \) is incorporated between the E-step and the M-step.

The mixture models are widely used in cluster analysis. For the continuous data, the gaussian mixture model is often used (see Banfield & Raftery, 1993; Celeux & Govaert, 1992). In this case, \( \phi_k(X_i, \alpha_k) \) is the density of a gaussian distribution and \( \alpha_k = (\mu_k, \Delta_k) \) with mean vectors \( \mu_k \) and covariance matrices \( \Delta_k \). When the proportions \( \pi_k \) are supposed equal and the covariance matrices \( \Delta_k \) take the form \( \Delta_k = \sigma^2 I \) where \( \sigma \) is a unknown and \( I \) is the identity matrix, the complete log-likelihood model corresponds to the inertia (sum of square) criterion used by the popular K-means method (Banfield & Raftery, 1993; Celeux & Govaert, 1992, 1993). We will see that for the binary data, if the Bernoulli distribution is used, then the density complete log-likelihood model corresponds also to the famous K-means criterion. For a fair comparison with our algorithm, we focus now on the last case: the block mixture model for binary data using the Bernoulli distribution.

A.3.2 A Block Mixture Model Corresponds to the Squared Error Criterion 1.1 for Biclustering. Block mixture models are extended from mixture models for biclustering, where each component of the mixture represents a bicluster. As in the previous sections, here the data matrix \( X = (x_{ij})_{n \times m} \) is defined on a set \( I \) of \( n \) objects (rows) and a set \( J \) of \( m \) variables (columns), and \( P = (p_{ik})_{n \times K} \) and \( Q = (q_{jl})_{m \times L} \) are, respectively, the row classification matrix and the column classification matrix. Likewise, \( \pi_k \) denotes the mixing proportions of the \( k \)th row and \( \eta_l \) the mixing proportion of the \( l \)th column, for \( k \in I, \ l \in J \). Under the assumption that the assignment of \( I \times J \) defined by a product of assignments of \( I \) and \( J \) is independent, the complete log-likelihood function is given by Govaert and Nadif (2003):

\[
L_c(X, P, Q, \theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log (\pi_k) + \sum_{j=1}^{m} \sum_{l=1}^{L} q_{jl} \log (\eta_l)
\]

\[
+ \sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{j=1}^{m} \sum_{l=1}^{L} p_{ik} q_{jl} \log \phi_{kl}(X_{ij}, \alpha_{kl}).
\] (A.11)
where $\phi_{kl}$ are the probability distributions in each component $(k, l) \in I \times J$. The parameter $\theta$ of this model is the vector $\theta = (\pi, \eta, \alpha) = (\pi_1, \ldots, \pi_K, \eta_1, \ldots, \eta_L, \alpha_{11}, \ldots, \alpha_{mn})$. It is not difficult to see that as for one-mode clustering using the Gaussian distribution function, under the hypotheses that the covariance matrices $\Delta_k = \sigma_k^2 I$, maximizing the complete log-likelihood function $L_c(X, P, Q, \theta)$ is equivalent to minimizing the squared error criterion 1.1 with $v_{kl} = \alpha_{kl}$.

We focus now on the binary data, say, $X = (x_{ij})_{n \times m}$ with $x_{ij} \in \{0, 1\}$. Govaert and Nadif (2003) showed that when $\phi_{kl}$ are Bernoulli distributions, say, $\phi_{kl}(X_{ij}, \alpha_{kl}) = (\alpha_{kl})^{x_{ij}}(1 - \alpha_{kl})^{1-x_{ij}}$, $i \in I$, $j \in J$, $k = 1, \ldots K, l = 1, \ldots L$,

then the so-called simplest model of equation A.11 corresponds to the squared error criterion, equation 1.1. More precisely, by replacing the parameters $\alpha_{kl}$ by $v_{kl} \in \{0, 1\}$ and $\epsilon_{kl} \in [1/2, 1]$ with

\begin{align*}
    v_{kl} &= 1 \quad \text{and} \quad \epsilon_{kl} = 1 - \alpha_{kl} \quad \text{if} \quad \alpha_{kl} \in [1/2, 1], \\
    v_{kl} &= 0 \quad \text{and} \quad \epsilon_{kl} = \alpha_{kl} \quad \text{if} \quad \alpha_{kl} \in [0, 1/2],
\end{align*}

(A.12)

the Bernoulli distributions become

\begin{align*}
    \phi_{kl}(X_{ij}, v_{kl}, \epsilon_{kl}) &= (\epsilon_{kl})^{x_{ij}-v_{kl}}(1 - \epsilon_{kl})^{1-x_{ij}-v_{kl}}, \\
    i \in I, j \in J, k = 1, \ldots K, l = 1, \ldots L.
\end{align*}

(A.13)

The simplest model of the mixture model equations A.9 to A.13 is characterized by the two following properties: in equation A.9, $\pi_k = 1/K$ for all $k \in \{1, \ldots K\}$ and $\eta_l = 1/L$ for all $l \in \{1, \ldots L\}$, and in equation A.13 $\epsilon_{kl} = \epsilon_{kl'} = \epsilon$ for all $l' \in \{1, \ldots L\}$. For the simplest model, the complete log-likelihood function is given by

\begin{align*}
    L_c(X, P, Q, \theta) &= \log(\epsilon/(1 - \epsilon)) \sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{j=1}^{m} \sum_{l=1}^{L} p_{ik} q_{jl} |x_{ij} - v_{kl}| + D, \\
    \text{with} \quad D = nm \log(1 - \epsilon) - n \log K - m \log L.
\end{align*}

Thus, in case of the binary data, one has

\begin{align*}
    L_c(X, P, Q, \theta) &= \log \left( \frac{\epsilon}{1 - \epsilon} \right) f(P, Q, V) + D.
\end{align*}

In our experiment, we compare our algorithm with BCEM studied in Govaert and Nadif (2003) for the binary data and on the simplest model.
We briefly describe the generic scheme of BCEM algorithm (see, Govaert & Nadif, 2003, for more details of the CEM algorithm in steps 2a and 2b):

1. Start from an initial position \((P^0, Q^0, \pi^0, \eta^0, \alpha^0)\).
2. Compute \((P^{c+1}, Q^{c+1}, \pi^{c+1}, \eta^{c+1}, \alpha^{c+1})\) starting from \((P^c, Q^c, \pi^c, \eta^c, \alpha^c)\):
   2a. Compute \((P^{c+1}, \pi^{c+1}, \bar{\alpha})\) using the CEM algorithm starting from \((P^c, \pi^c, \alpha^c)\).
   2b. Compute \((Q^{c+1}, \eta^{c+1}, \alpha^{c+1})\) using the CEM algorithm starting from \((Q^c, \eta^c, \bar{\alpha})\).
3. Repeat steps 2a and 2b until convergence.

Acknowledgments

This research has been supported by Fonds Européens de Développement Régional Lorraine via the project Innovations techniques d’optimisation pour le traitement Massif de Données (InnoMaD).

References


Received September 20, 2012; accepted March 23, 2013.