Iranian Journal of Operations Research
Vol. 5, No. 1, 2014, pp. 1-14

An Incremental DC Algorithm for the Minimum Sum-of-Squares Clustering

A. M. Bagirov[1]

Here, an algorithm is presented for solving the minimum sum-of-squares clustering problems using their difference of convex representations. The proposed algorithm is based on an incremental approach and applies the well known DC algorithm at each iteration. The proposed algorithm is tested and compared with other clustering algorithms using large real world data sets.

***Keywords**: Clustering, Nonsmooth optimization, Nonconvex optimization, Incremental algorithms.*

Manuscript was received on 27/04/2015 and accepted for publication on 16/06/2015.†

1. Introduction

Clustering is an unsupervised technique dealing with problems of organizing a collection of patterns into clusters based on similarity. Most clustering algorithms are based on hierarchical and partitional approaches. Algorithms based on an hierarchical approach generate a dendrogram representing the nested grouping of patterns and similarity levels at which groupings change [19]. Partitional clustering algorithms find the partition that optimizes a clustering criterion [19]. In this paper, we develop a partitional clustering algorithm for solving the minimum sum-of-squares clustering (MSSC) problems.

To date various heuristics such as the k-means algorithm and its variations have been developed to solve the MSSC problem (see, for example, [20, 21] and references therein). The global k-means algorithm and its various modifications are among the most efficient algorithms for solving the MSSC problem [5, 10, 12, 22, 24, 25, 27].

The MSSC can be formulated as an optimization problem [8, 11, 30] and different deterministic optimization techniques and metaheuristics have been applied to solve it. These techniques include branch and bound [15] and interior point methods [16], nonsmooth optimization algorithms [6, 9, 11], algorithms based on the hyperbolic smoothing technique [7, 32, 33], the variable neighborhood search [17], simulated annealing [28], tabu search [1] and genetic algorithms [26]. Most of these algorithms are not efficient for solving the MSSC problem with very large data sets.

The objective function of MSSC problems can be represented by a difference of convex (DC) functions. There are several papers in which this representation is used to design algorithms. In [14], the truncated codifferential method is applied to solve the MSSC problem using its DC representation. The branch and bound method was modified for such problems in [31]. In [2], an algorithm based on

† Invited Paper

[1] Faculty of Science and Technology, Federation University Australia, Victoria, Australia.
Email: a.bagirov@federation.edu.au

DC Algorithms (DCAs) was introduced. In [3], a DCA and a Gaussian kernel were applied to design a clustering algorithm.

Here, we develop a new algorithm for solving the MSSC problem using its DC representation. This algorithm is based on an incremental approach and applies the DCA to solve optimization problems at each iteration of the incremental algorithm. Computational results on some real world data sets are reported and the algorithm is compared with two other clustering algorithms. It is demonstrated that the proposed algorithm is especially efficient for solving the MSSC problems with very large data sets.

The rest of the paper is organized as follows. Section 2 provides some preliminaries on DC functions and nonsmooth analysis. DC representations of cluster functions are given in Section 3. Section 4 presents an algorithm for solving clustering problems. An incremental algorithm is discussed in Section 5. Numerical results are reported in Section 6 and Section 7 contains the concluding remarks.

2. Preliminaries

In this section we give some results on nonsmooth analysis and DC functions used throughout the paper. In what follows, we denote by $\mathbb{R}^n$ the n-dimensional Euclidean space with the inner product $\langle u, v\rangle = \sum_{i=1}^{n} u_i v_i$ and the associated norm $|u| = \langle u, u\rangle^{1/2}, u, v \in \mathbb{R}^n$. The set $B(x, \varepsilon) = \{y \in \mathbb{R}^n : |y - x| < \varepsilon\}$ is the open ball centered at x with the radius $\varepsilon > 0$.

Let $f: \mathbb{R}^n \to \mathbb{R}$ be a convex function. Its subdifferential at $x \in \mathbb{R}^n$ is defined as

$$\partial_c f(x) = \{\xi \in \mathbb{R}^n : f(y) - f(x) \geq \langle \xi, y - x\rangle \ \ \forall y \in \mathbb{R}^n\}.$$

A function $f: \mathbb{R}^n \to \mathbb{R}$ is called locally Lipschitz on $\mathbb{R}^n$ if for any bounded subset $X \subset \mathbb{R}^n$ there exists $L > 0$ such that

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

The generalized derivative of a locally Lipschitz function f at a point x with respect to a direction $u \in \mathbb{R}^n$ is defined as [13]

$$f^0(x, u) = \limsup_{\alpha \downarrow 0, \ \ y \to x} \frac{f(y + \alpha u) - f(y)}{\alpha}.$$

The subdifferential $\partial f(x)$ of the function f at x is

$$\partial f(x) = \{\xi \in \mathbb{R}^n : f^0(x, u) \geq \langle \xi, u\rangle \ \ \forall u \in \mathbb{R}^n\}.$$

Each vector $\xi \in \partial f(x)$ is called a subgradient. For a convex function $f: \mathbb{R}^n \to \mathbb{R}$, one has $\partial f(x) = \partial_c f(x), x \in \mathbb{R}^n$. From now on, we will use the notation ∂f for subdifferential of convex function.

$f: \mathbb{R}^n \to \mathbb{R}$ is called a DC function if there exist convex functions $g, h: \mathbb{R}^n \to \mathbb{R}$ such that [18]

$$f(x) = g(x) - h(x), \qquad x \in \mathbb{R}^n.$$

Here, $g - h$ is called a DC decomposition of f while g and h are DC components of f. Note that a DC function has infinitely many DC decompositions.

An unconstrained DC program is an optimization problem of the form

$$\text{minimize } f(x) = g(x) - h(x) \quad \text{subject to } x \in \mathbb{R}^n. \tag{1}$$

One can define different types of stationary points for problem (1). Such stationary points include critical, Clarke stationary and inf-stationary points. A point x^* is called inf-stationary for problem (1) if

$$\partial h(x^*) \subset \partial g(x^*). \tag{2}$$

A point x^* is called Clarke stationary for problem (1) if

$$0 \in \partial f(x^*). \tag{3}$$

Finally, a point x^* is called a critical point of problem (1) if

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset. \tag{4}$$

In general, any inf-stationary point is also Clarke stationary and a critical point. Furthermore, any Clarke stationary point is a critical point.

3. DC Programming Approach to Clustering Problems

Let A be a finite set of points in $\mathbb{R}^n$, that is, $A = \{a^1, \dots, a^m\}$, $a^i \in \mathbb{R}^n$, $i = 1, \dots, m$. The hard clustering problem is the distribution of the points of the set A into a given number k of disjoint subsets A^j, $j = 1, \dots, k$, such that

1. $A^j \neq \emptyset$ and $A^j \cap A^l = \emptyset, j, l = 1, \dots, k, j \neq l$.
2. $A = \bigcup_{j=1}^{k} A^j$.

The sets A^j, $j = 1, \dots, k$, are called clusters and each cluster A^j can be identified by its center $x^j \in \mathbb{R}^n$, $j = 1, \dots, k$. The problem of finding these centers is called the k-clustering (or k-partition) problem. In order to formulate the clustering problem one needs to define the similarity (or dissimilarity) measure. Here, the similarity measure is defined using the L_2 norm as follows:

$$d_2(x, a) = \sum_{i=1}^{n} (x_i - a_i)^2 .$$

The nonsmooth optimization formulation of the MSSC problem is [8, 11]:

$$\text{minimize } f_k(x) \text{ subject to } x = \left(x^1, \dots, x^k\right) \in \mathbb{R}^{nk}, \tag{5}$$

where

$$f_k(x^1,\dots,x^k)=\frac{1}{m}\sum_{a\in A}\min_{j=1,\dots,k} d_2(x^j,a). \tag{6}$$

The objective function f_k in problem (5) can be represented as a DC function,

$$f_k(x)=f_{k1}(x)-f_{k2}(x),\ x=(x^1,\dots,x^k)\in\mathbb{R}^{nk}, \tag{7}$$

where

$$f_{k1}(x)=\frac{1}{m}\sum_{a\in A}\sum_{j=1}^{k} d_2(x^j,a),\quad f_{k2}(x)=\frac{1}{m}\sum_{a\in A}\max_{j=1,\dots,k}\sum_{s=1,s\neq j}^{k} d_2(x^s,a).$$

Since the function d_2 is convex in x, the function f_{k1}, as a sum of convex functions, is also convex. The function f_{k2} is a sum of maxima of sum of convex functions. Since the sum of convex functions is convex, the functions under maximum are also convex. Furthermore, since the maximum of a finite number of convex functions is convex, the function f_{k2} being a sum of convex functions is also convex.

Next, we derive expressions for subdifferentials of the functions f_{k1} and f_{k2}. The function f_{k1} is differentiable and its gradient is [25]

$$\nabla f_{k1}(x)=2(x-\hat{A}). \tag{8}$$

Here, $\hat{A}=(\hat{A}_1,\dots,\hat{A}_k)$, $\hat{A}_1=\dots=\hat{A}_k=(\hat{a}_1,\dots,\hat{a}_n)$ and

$$\hat{a}_t=\frac{1}{m}\sum_{a\in A} a_t.$$

This means that the subdifferential $\partial f_{k1}(x)$ is a singleton for any $x\in\mathbb{R}^n$.

In general, the function f_{k2} is nonsmooth. To compute its subdifferential, consider the following function and a set as follows [25]:

$$\varphi_a(x)=\max_{j=1,\dots,k}\sum_{s=1,s\neq j}^{k} d_2(x^s,a), \tag{9}$$

and

$$R_a(x)=\left\{j\in\{1,\dots,k\}:\ \sum_{s=1,s\neq j}^{k} d_2(x^s,a)\ =\varphi_a(x)\right\}. \tag{10}$$

The subdifferential $\partial\varphi_a(x)$ of the function φ_a at x is

$$\partial\varphi_a(x)=\operatorname{conv}\{V\in\mathbb{R}^{nk}:\ V=2(\tilde{x}^j-\tilde{A}_i^j), j\in R_a(x)\}, \tag{11}$$

where

$$\tilde{x}^j=(x^1,\dots,x^{j-1},0_n,x^{j+1},\dots,x^k),\qquad \tilde{A}_i^j=(\tilde{A}_{i1}^j,\dots,\tilde{A}_{ik}^j)\in\mathbb{R}^{nk},$$

and

$$\tilde{A}_{it}^{j} = a^{i}, \qquad t = 1, \dots, k, t \neq j, \qquad \tilde{A}_{ij}^{j} = 0_n.$$

Then, the subdifferential $\partial f_{k2}(x)$ can be expressed by

$$\partial f_{k2}(x) = \frac{1}{m} \sum_{a \in A} \partial \varphi_a(x). \tag{12}$$

Problem (5) is a global optimization problem. It has many local minimizers and only global or near global solutions provide the best clustering structure of a data set. To find such solutions using a local search method, it is crucial to apply a special procedure to generate good starting cluster centers. We apply an algorithm introduced in [25]. This algorithm involves the solution of the so-called auxiliary clustering problem. Next, we briefly describe this problem.

Assume that the solution $x^1, \dots, x^{k-1}$, $k \geq 2$, to the $(k-1)$-clustering problem is known. Denote by r_{k-1}^{a}, the distance between the data point $a \in A$ and the closest cluster center among the $k-1$ centers $x^1, \dots, x^{k-1}$:

$$r_{k-1}^{a} = \min\{d_2(x^1, a), \dots, d_2(x^{k-1}, a)\}. \tag{13}$$

The k-th auxiliary cluster function is defined as [5, 25]

$$\bar{f}_k(y) = \frac{1}{m} \sum_{a \in A} \min\{r_{k-1}^{a}, d_2(y, a)\}, \qquad y \in \mathbb{R}^n. \tag{14}$$

This function is nonsmooth, locally Lipschitz, directionally differentiable and as a sum of minima of convex functions it is, in general, nonconvex. It is obvious that

$$\bar{f}_k(y) = f_k(x^1, \dots, x^{k-1}, y), \qquad \forall y \in \mathbb{R}^n.$$

A problem,

$$\text{minimize } \bar{f}_k(y) \text{ subject to } y \in \mathbb{R}^n, \tag{15}$$

is called *the k-th auxiliary clustering problem* [5, 25]. The DC representation of the function $\bar{f}_k$ is as follows:

$$\bar{f}_k(y) = \bar{f}_{k1}(y) - \bar{f}_{k2}(y), \tag{16}$$

where

$$\bar{f}_{k1}(y) = \frac{1}{m} \sum_{a \in A} \left(r_{k-1}^{a} + d_2(y, a)\right), \qquad \bar{f}_{k2}(y) = \frac{1}{m} \sum_{a \in A} \max\{r_{k-1}^{a}, d_2(y, a)\}.$$

To express subdifferentials of the functions $\bar{f}_{k1}$ and $\bar{f}_{k2}$ at a given point $y \in \mathbb{R}^n$, deine the following sets:

$$\bar{A}_1(y) = \{a \in A: \ r_{k-1}^{a} > d_2(y, a)\}, \qquad \bar{A}_2(y) = \{a \in A: \ r_{k-1}^{a} < d_2(y, a)\},$$

$$\bar{A}_3(y) = \{a \in A: \ r_{k-1}^{a} = d_2(y, a)\}.$$

Then, the function $\bar{f}_{k2}$ at y can be rewritten as

$$\bar{f}_{k2}(y) = \frac{1}{m}\left(\sum_{a\in\bar{A}_1(y)} r_{k-1}^a + \sum_{a\in\bar{A}_2(y)} d_2(y,a) + \sum_{a\in\bar{A}_3(y)} \max\{r_{k-1}^a, d_2(y,a)\}\right).$$

The function $\bar{f}_{k1}$ is continuously differentiable on $\mathbb{R}^n$ and its gradient at $y \in \mathbb{R}^n$ is

$$\nabla \bar{f}_{k1}(y) = \frac{2}{m}\sum_{a\in A}(y-a). \tag{17}$$

The function $\bar{f}_{k2}$, in general, is nonsmooth and its subdifferential at $y \in \mathbb{R}^n$ is

$$\partial \bar{f}_{k2}(y) = \frac{2}{m}\left(\sum_{a\in\bar{A}_2(y)} (y-a) + \sum_{a\in\bar{A}_3(y)} \operatorname{conv}\{0,(y-a)\}\right). \tag{18}$$

4. An Algorithm for Solving Optimization Problems

In this section we describe the DCA for solving both the clustering problem (5) and the auxiliary clustering problem (15). It is easy to observe that both problems can be formulated as the following unconstrained DC programming problem:

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \mathbb{R}^n, \tag{19}$$

where $f(x) = f_1(x) - f_2(x)$, the function f_1 is continuously differentiable convex and the function f_2 is, in general, a nonsmooth convex function.

DCA is an algorithm for solving DC programming problems. A detailed study of this algorithm can be found in [4, 29]. The generic DCA scheme for solving the problem (19) is shown in Algorithm 1 below.

Algorithm 1. DCA scheme for problem (19).

Step 1: Select any starting point $x^1 \in \mathbb{R}^n$ and set $j := 1$.

Step 2: Compute $\xi^j \in \partial f_2(x^j)$.

Step 3: If $\xi^j = \nabla f_1(x^j)$ then stop.

Step 4: Find the solution x^{j+1} to the following convex optimization problem:

$$\text{minimize} \quad f_1(x) - \langle \xi^j, x - x^j \rangle \ \text{subject to} \ \ x \in \mathbb{R}^n. \tag{20}$$

Step 5: Set $j := j + 1$ and go to Step 2.

Proposition 4.1. All accumulation points of the sequence $\{x^k\}$ generated by Algorithm 1 are Clarke stationary points of the problem (19).

Proof. It is well known that accumulation points of the sequence $\{x^k\}$ are critical points of the problem (19). Since the function f_1 in (19) is continuously differentiable, the sets of critical points and Clarke stationary points of the problem (19) coincide.

In order to apply DCA to solve problem (15), the subgradient ξ^j in Step 2 is computed as (see (18))

$$\xi^j = \frac{2}{m} \sum_{a \in \bar{A}_2(x^j)} (x^j - a), \qquad x^j \in \mathbb{R}^n.$$

Then, the solution x^{j+1} to the problem (20) in Step 4 can be expressed as follows:

$$x^{j+1} = \frac{1}{m} \left(\left| \bar{A}_2(x^j) \right| x^j + \sum_{a \in \bar{A}_1(x^j) \cup \bar{A}_3(x^j)} a \right).$$

Applying (17), the stopping criterion in Step 3 can be given by

$$\sum_{a \in \bar{A}_1(x^j) \cup \bar{A}_3(x^j)} (x^j - a) = 0.$$

Now, we describe an application of DCA to solve the clustering problem (5). Let $x^j = \left(x_j^1, \dots, x_j^k\right) \in \mathbb{R}^{nk}$ be a vector of cluster centers at the iteration j and A^1, ..., A^k be the cluster partition of the data set A given by these centers.

In order to compute the subgradient ξ^j in Step 2, for each $a \in A$ we compute the set $R_a(x^j)$ given by (10), take any $p \in R_a(x^j)$ and compute the subgradient $v_a^j \in \partial \varphi_a(x^j)$ using (11). Then, we apply (12) to compute the subgradient ξ^j. Thus, we get the following formula for the subgradient ξ^j:

$$\begin{aligned} \xi^j &= \frac{2}{m} \left(\sum_{a \in A \setminus A^1} \left(x_j^1 - a\right), \dots, \sum_{a \in A \setminus A^k} \left(x_j^k - a\right) \right) \\ &= \frac{2}{m} \left((m - |A^1|) x_j^1 - (m\bar{a} - |A^1| \bar{a}_1), \dots, \left(m - \left|A^k\right|\right) x_j^k - \left(m\bar{a} - \left|A^k\right| \bar{a}_k\right) \right), \end{aligned}$$

where $\bar{a}^l$ is the center of the cluster A^l, $l = 1, \dots, k$, and $\bar{a}$ is the center of the whole set A.

The solution $x^{j+1} = (x_{j+1}^1, \dots, x_{j+1}^k)$ to the problem (20) in Step 4 is given by

$$x_{j+1}^t = \left(1 - \frac{|A^t|}{m}\right) x_j^t + \frac{|A^t|}{m} \bar{a}^t, \qquad t = 1, \dots, k.$$

Finally, the stopping criterion in Step 3 can be given by

$$x_j^t = \left(1 - \frac{|A^t|}{m}\right) x_j^t + \frac{|A^t|}{m} \bar{a}_t, \qquad t = 1, \dots, k.$$

These results demonstrate we do not need to apply any optimization algorithm to solve problem (20) for both clustering problem (5) and auxiliary clustering problem (15). In both cases, solutions can be expressed explicitly.

5. Incremental Algorithm

We propose an incremental algorithm to solve problems (5) and (15). This algorithm starts with computing the center of the whole data set. Then, it solves the k-clustering problem ($k > 1$) by gradually adding one cluster center at each iteration. At the lth, $1 < l \leq k$, iteration a special procedure is applied to find the starting cluster centers using the solution of the $(l-1)$-clustering problem. One such procedure was introduced in [25] (see also [7]) which we will use in the incremental algorithm given below. At each iteration of the incremental algorithm, DCA is applied to solve the optimization problems (5) and (15). The proposed incremental algorithm is called the *Incremental DCA (IncDCA)*. It is easy to see that this algorithm, in addition to the k-partition problem, also solves all intermediate l-partition problems, for $l = 1, \ldots, k-1$.

Algorithm 2. An incremental clustering algorithm (IncDCA).

Step 1: (Initialization) Compute the center $x^1 \in \mathbb{R}^n$ of the set A. Set $l := 1$.

Step 2: (Stopping criterion) Set $l := l + 1$. If $l > k$ then stop the k-partition problem has been solved.

Step 3: (Computation of a set of starting points for the auxiliary clustering problem) Apply the procedure from [25] to find the set $S_{1l} \subset \mathbb{R}^n$ of the starting points for solving the auxiliary clustering problem (15) for $k = l$.

Step 4: (Computation of a set of starting points for the lth cluster center) Apply Algorithm 1 to solve problem (15) starting from each point $y \in S_{1l}$ and generate a set $S_{2l} \subset \mathbb{R}^n$ of starting points for the lth cluster center.

Step 5: (Computation of a set of cluster centers) For each $\bar{y} \in S_{2l}$, apply Algorithm 1 to solve problem (5) starting from the point $(x^1, \ldots, x^{l-1}, \bar{y})$ and find a solution $(\hat{y}^1, \ldots, \hat{y}^l)$. Denote by $S_{3l} \subset \mathbb{R}^{nl}$, the set of all such solutions.

Step 6: (Computation of the best solution) Compute

$$f_l^{\min} = \min\{f_l(\hat{y}^1, \ldots, \hat{y}^l) : (\hat{y}^1, \ldots, \hat{y}^l) \in S_{3l}\}$$

and the collection of cluster centers $(\bar{y}^1, \ldots, \bar{y}^l)$ such that $f_l(\bar{y}^1, \ldots, \bar{y}^l) = f_l^{\min}$.

Step 5: (Solution to the l-partition problem). Set $x^j := \bar{y}^j, j = 1, \ldots, l$, as a solution to the lth partition problem and go to Step 2.

Algorithm 2 contains a special procedure to generate starting cluster centers (Step 3) which is described in detail in [25]. The values of the parameters in this procedure were chosen according to the recommendations given in [7].

6. Numerical Results

To test IncDCA and compare it with other clustering algorithms, numerical experiments with a number of real-world data sets were carried out. Algorithms were implemented in Fortran 95 and compiled using the *gfortran* compiler. Computational results were obtained on a PC with the CPU Intel(R) Core(TM) i5-3470S 2.90 GHz and RAM 8 GB. Eight data sets were used in the numerical

experiments. Their brief descriptions are given in Table 1. The detailed descriptions can be found in [23]. All data sets contain only numeric features and they do not have missing values. The number of attributes in these data sets ranges from 3 to 128 and the number of data points ranges from tens of thousands (smallest: 13,910) to hundred of thousands (largest: 434,874).

The IncDCA's performance was compared with two other incremental algorithms: the global k-means algorithm (GKM) [24] and the multi-start modified global k-means algorithm (MS-MGKM) [25]. Implementations of the GKM and MS-MGKM algorithms were discussed in [24] and [25], respectively.

Table 1. A brief description of data sets

Data sets	Number of instances	Number of attributes
Gas Sensor Array Drift	13910	128
EEG Eye State	14980	14
Letter Recognition	20000	16
KEGG Metabolic Relation Network	53413	20
Shuttle Landing Control	58000	9
Localization Data for Person Activity	164860	3
Skin Segmentation	245057	3
3D Road Network	434874	3

We computed up to 25 clusters in all the data sets. The CPU times used by the algorithms were limited to 20 hours. Results are presented in tables 2-4. In these tables, we use the following notations:

- k is the number of clusters;
- f_{best} (multiplied by the number shown after name of the data set) is the best value of the cluster function (6) (multiplied by m) obtained by the three algorithms;
- E is the percentage error calculated as follows:

$$E = \frac{\bar{f} - f_{\text{best}}}{f_{\text{best}}} \times 100\%,$$

where $\bar{f}$ is the value of the clustering function obtained by the corresponding algorithm;

- N_{best} is the least number of distance function evaluations used by the three algorithms;
- r_N is the ratio of the number of distance function evaluations (N) used by an algorithm and the corresponding N_{best}: $r_N = N/N_{best}$;
- t_{best} is the least CPU time required by three algorithms;
- r_t is the ratio of the CPU time (t) required by an algorithm and the corresponding least CPU time: $r_t = t/t_{best}$;
- The sign "-" in the tables shows that an algorithm failed to compute clusters in the given time limit.

Results for the best cluster function values found by different algorithms are presented in Table 2. These results show that in most cases all the three algorithms achieved high accuracy. The IncDCA failed to find highly accurate solutions over the Localization Data and the GKM and MS-MGKM algorithms over the KEGG Network data set. It is well-known that both the GKM and MS-MGKM algorithms are able to find either global or near global solutions to the clustering problem. Results from Table 2 allow us to claim that the IncDCA algorithm is also able to find either global or near global solutions of the clustering problems.

Results for the number of distance function evaluations by different algorithms are shown in Table 3. One can see that the MS-MGKM algorithm uses the least number of distance function evaluations among the three algorithms in almost all cases. The IncDCA algorithm requires less distance function evaluations than the GKM algorithm. We can also see that the IncDCA and MS-MGKM algorithms use similar number of distance function evaluations over the three largest data sets: Localization Data, Skin Segmentation and 3D Road Network. In these data sets the GKM algorithm uses significantly more function evaluations than the other two algorithms.

Table 2. Cluster function values obtained by the algorithms

k	f_{best}	IncDCA E	MGKM E	GKM E	f_{best}	IncDCA E	MGKM E	GKM E
	Gas Sensor Array Drift ($\times 10^{13}$)				EEG Eye State ($\times 10^{8}$)			
2	7.91186	0.00	0.00	0.00	8178.14	0.00	0.00	0.00
3	5.02412	0.00	0.00	0.00	1833.88	0.00	0.00	0.00
5	3.22729	0.00	0.00	0.00	1.33858	0.00	0.00	0.00
10	1.65524	0.00	0.00	0.00	0.45669	0.00	0.00	0.00
15	1.13801	0.00	0.00	0.36	0.34653	0.05	0.05	0.00
20	0.87916	0.16	0.00	0.63	0.28987	1.50	0.00	0.00
25	0.72348	0.00	0.00	0.47	0.26006	0.00	0.00	0.56
	Letter Recognition ($\times 10^{6}$)				KEGG Network ($\times 10^{8}$)			
2	1.38189	0.00	0.00	0.00	11.38530	0.00	0.00	0.00
3	1.25058	0.00	0.00	0.00	4.90060	0.00	0.00	0.00
5	1.09799	0.00	0.00	0.00	1.88367	0.00	0.00	0.00
10	0.85752	0.00	0.00	0.00	0.63515	0.00	0.00	0.00
15	0.74818	0.00	0.00	0.00	0.35127	0.00	4.33	4.33
20	0.67632	0.00	0.00	0.17	0.25076	0.00	1.89	1.89
25	0.63081	0.00	0.00	0.19	0.19524	0.00	1.10	1.10
	Shuttle Landing Control ($\times 10^{8}$)				Localization Data ($\times 10^{5}$)			
2	21.34329	0.00	0.00	0.00	1.03715	0.02	0.00	0.00
3	10.85415	0.00	0.00	0.00	0.77229	0.02	0.00	0.00
5	7.24479	0.09	0.09	0.00	0.56018	0.10	0.00	0.00
10	2.83216	0.59	0.58	0.00	0.33413	0.20	0.00	0.00
15	1.53154	0.02	0.00	0.00	0.26293	1.69	0.00	0.00
20	1.06012	0.02	0.00	0.00	0.21974	0.03	0.00	0.00
25	0.79911	0.04	0.00	0.00	0.18891	1.72	0.00	0.00
	Skin Segmentation ($\times 10^{13}$)				3D Road Network ($\times 10^{13}$)			
2	1.32236	0.00	0.00	0.00	49.13298	0.00	0.00	0.00
3	0.89362	0.00	0.00	0.00	22.77818	0.00	0.00	0.00
5	0.50203	0.00	0.00	0.00	8.82574	0.01	0.00	0.00
10	0.25122	0.00	0.00	0.00	2.57247	0.00	-	-
15	0.16964	0.00	0.00	0.00	1.27855	0.00	-	-
20	0.12645	0.00	0.00	0.99	0.81257	0.00	-	-
25	0.10228	0.00	0.00	0.70	0.60679	0.00	-	-

Table 3. Number of distance function evaluations

k	N_{best}	IncDCA r_N	MGKM r_N	GKM r_N	N_{best}	IncDCA r_N	MGKM r_N	GKM r_N
	Gas Sensor Array Drift ($\times 10^8$)				EEG Eye State ($\times 10^6$)			
2	0.457	1.11	1.00	4.24	0.180	1.00	1.08	1248.83
3	1.302	1.12	1.00	2.98	0.375	1.00	1.08	1198.92
5	3.185	2.42	1.00	2.44	28.477	1.00	1.00	31.54
10	8.487	2.60	1.00	2.07	354.547	1.36	1.00	5.74
15	14.126	3.01	1.00	1.94	795.978	2.28	1.00	4.01
20	19.867	3.62	1.00	1.87	1274.649	3.09	1.00	3.45
25	25.505	3.58	1.00	1.85	1793.991	3.25	1.00	3.11
	Letter Recognition ($\times 10^9$)				KEGG Network ($\times 10^9$)			
2	0.172	1.01	1.00	2.34	0.157	1.00	1.00	18.17
3	0.341	1.01	1.00	2.35	0.389	1.11	1.00	14.68
5	0.673	1.06	1.00	2.39	1.292	1.46	1.00	8.84
10	1.518	1.18	1.00	2.40	4.63	3.32	1.00	5.55
15	2.398	1.34	1.00	2.37	9.69	3.31	1.00	4.13
20	3.319	1.52	1.00	2.34	15.127	2.98	1.00	3.60
25	4.251	1.69	1.00	2.31	20.873	3.64	1.00	3.30
	Shuttle Landing Control ($\times 10^8$)				Localization Data ($\times 10^{10}$)			
2	0.122	1.00	1.00	274.91	0.948	1.00	1.00	2.87
3	0.264	1.00	1.00	254.97	1.997	1.00	1.00	2.72
5	2.247	38.48	1.00	59.89	3.857	1.00	1.00	2.82
10	12.003	15.17	1.00	25.23	8.443	1.00	1.00	2.90
15	29.131	33.64	1.00	16.18	13.162	1.00	1.00	2.90
20	59.699	29.69	1.00	10.71	18.041	1.01	1.00	2.87
25	103.658	25.30	1.00	7.80	23.018	1.00	1.00	2.85
	Skin Segmentation ($\times 10^{13}$)				3D Road Network ($\times 10^{13}$)			
2	0.231	1.00	1.00	2.61	0.594	1.00	1.00	3.19
3	0.452	1.00	1.00	2.65	1.394	1.00	1.00	2.71
5	0.869	1.00	1.00	2.76	2.963	1.00	1.00	2.55
10	1.793	1.01	1.00	3.02	7.038	1.00	-	-
15	2.618	1.03	1.00	3.21	11.178	1.00	-	-
20	3.424	1.05	1.00	3.33	15.464	1.00	-	-
25	4.218	1.11	1.00	3.42	19.771	1.00	-	-

Results for the CPU times required by the algorithms are given in Table 4. These results show that the GKM algorithm is the most time consuming among the three algorithms as the size of a data set increases. The MS-MGKM algorithm requires the least CPU time over five data sets (except the cases, when $k = 2, 3$): Gas Sensor Array Drift, EEG Eye State, Letter Recognition, KEGG Network and Shuttle Control. However, in three largest data sets the IncDCA algorithm requires the least CPU time. Moreover, the GKM and MS-MGKM algorithms computed only 5 clusters in 3D Road Network data set within the given time limit. These results demonstrate that for largest data sets used in this paper, the IncDCA algorithm outperforms the other two algorithms, that is, the IncDCA algorithm is more efficient for solving clustering problems with hundreds of thousands of data points.

Table 4. CPU times used by the algorithms

k	t_{best}	IncDCA r_t	MGKM r_t	GKM r_t	t_{best}	IncDCA r_t	MGKM r_t	GKM r_t
	Gas Sensor Array Drift				EEG Eye State			
2	29.87	1.03	1.00	3.76	0.02	1.00	19.50	630.81
3	83.46	1.04	1.00	2.70	0.03	1.00	20.13	644.65
5	199.00	2.36	1.00	2.28	1.28	1.00	1.96	30.97
10	518.78	2.45	1.00	1.99	22.59	1.00	1.34	4.47
15	853.53	2.73	1.00	1.85	64.18	1.31	1.00	2.50
20	1192.04	3.18	1.00	1.79	94.07	1.92	1.00	2.33
25	1523.23	3.13	1.00	1.76	123.41	2.14	1.00	2.20
	Letter Recognition				KEGG Network			
2	9.45	1.00	1.44	2.55	15.62	1.00	1.28	16.99
3	18.91	1.00	1.33	2.57	44.60	1.00	1.46	12.46
5	38.66	1.00	1.30	2.53	199.76	1.00	1.50	6.11
10	98.67	1.00	1.10	2.16	947.83	1.58	1.00	2.92
15	164.08	1.07	1.00	1.94	1546.06	1.91	1.00	2.71
20	219.10	1.25	1.00	1.94	2182.74	1.87	1.00	2.62
25	276.78	1.40	1.00	1.92	2835.51	2.33	1.00	2.57
	Shuttle Landing Control				Localization Data			
2	0.66	1.00	8.84	216.50	192.07	1.00	4.54	5.60
3	1.34	1.00	8.70	211.08	405.29	1.00	3.92	4.80
5	30.89	16.29	1.00	18.17	815.17	1.00	3.70	4.71
10	129.92	7.34	1.00	9.82	1838.22	1.00	2.83	3.91
15	232.61	18.16	1.00	8.56	2788.89	1.00	2.67	3.83
20	449.99	16.25	1.00	5.98	3813.94	1.00	2.48	3.60
25	747.84	13.88	1.00	4.57	4859.43	1.00	2.41	3.48
	Skin Segmentation ($\times 10^8$)				3D Road Network ($\times 10^{13}$)			
2	562.15	1.00	3.69	4.83	1487.22	1.00	7.09	9.73
3	1103.96	1.00	3.60	4.78	3452.60	1.00	5.33	7.45
5	2088.68	1.00	2.82	4.21	7225.06	1.00	5.53	7.79
10	4207.69	1.00	2.20	3.84	16583.94	1.00	-	-
15	6150.09	1.00	1.99	3.72	25867.28	1.00	-	-
20	8137.20	1.00	1.81	3.55	35359.02	1.00	-	-
25	10309.79	1.00	1.65	3.37	44701.26	1.00	-	-

7. Conclusion

An algorithm for solving the minimum sum-of-squares clustering problems was developed. The algorithm exploited DC representation of the clustering problems, was based on the incremental approach and applied the DC algorithm to solve optimization problems. The proposed algorithm was tested using, in particular, real world data sets with hundreds of thousands of data points. Results demonstrated that the algorithm was able to find global or near global solutions of the clustering problems with very large data sets in a reasonable time, outperforming two other state-of-the-art incremental clustering algorithms.

Acknowledgement

This research by Dr. Adil Bagirov was supported under Australian Research Council's Discovery Projects funding scheme (Project No. DP140103213).

References

[1] Al-Sultan, K.S. (1995), A tabu search approach to the clustering problem, *Pattern Recognition*, 28(9), 1443-1451.
[2] An, L.T.H., Belghiti, M.T. and Tao, P.D. (2007), A new efficient algorithm based on DC programming and DCA for clustering, *Journal of Global Optimization*, 37(4), 593-608.
[3] An, L.T.H., Minh, L.H. and Tao, P.D. (2014), New and efficient DCA based algorithms for minimum sum-of-squares clustering, *Pattern Recognition*, 47, 388-401.
[4] An, L.T.H., Ngai, H.V. and Tao, P.D. (2012), Exact penalty and error bounds in DC programming, *Journal of Global Optimization*, 52(3), 509-535.
[5] Bagirov, A.M. (2008), Modified global k-means algorithm for minimum sum-of-squares clustering problems, *Pattern Recognition*, 41(10), 3192-3199.
[6] Bagirov, A.M. and Mohebi, E. (2015), Nonsmooth optimization based algorithms in cluster analysis, In Celebi, M.E. (Ed), Partitional Clustering Algorithms, Springer, pp. 99-146.
[7] Bagirov, A.M., Ordin, B., Ozturk, G. and Xavier, A.E. (2015), An incremental clustering algorithm based on hyperbolic smoothing, *Computational Optimization and Applications*, 61, 219-241.
[8] Bagirov, A.M., Rubinov, A.M., Soukhoroukova, N.V. and Yearwood, J. (2003), Unsupervised and supervised data classification via nonsmooth and global optimization, *Top*, 11, 1-93.
[9] Bagirov, A.M. and Ugon, J. (2005), An algorithm for minimizing clustering functions, *Optimization*, 54(4-5), 351-368.
[10] Bagirov, A.M., Ugon, J. and Webb, D. (2011), Fast modified global k-means algorithm for incremental cluster construction, *Pattern Recognition*, 44(4), 866-876.
[11] Bagirov, A.M. and Yearwood, J. (2006), A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems, *European Journal of Operational Research*, 170(2), 578-596.
[12] Bai, L., Liang, J., Sui, C. and Dang, Ch. (2013), Fast global k-means clustering based on local geometrical information, *Information Sciences*, 245, 168-180.
[13] Clarke, F.H. (1983), Optimization and Nonsmooth Analysis, Canadian Mathematical Society Series of Monographs and Advanced Texts, Wiley.

[14] Demyanov, V.F., Bagirov, A.M. and Rubinov, A.M. (2002), A method of truncated codifferential with application to some problems of cluster analysis, *Journal of Global Optimization*, 23(1), 63-80.
[15] Diehr, G. (1985), Evaluation of a branch and bound algorithm for clustering, *SIAM J. Scientific and Statistical Computing*, 6, 268-284.
[16] Du Merle, O., Hansen, P., Jaumard, B. and Mladenovic, N. (1999), An interior point algorithm for minimum sum-of-squares clustering, *SIAM Journal on Scientific Computing*, 21(4), 1485-1505.
[17] Hansen, P. and Mladenovic, N. (2001), Variable neighborhood decomposition search, *Journal of Heuristic*, 7, 335-350.
[18] Horst, R. and Thoai, N.V. (1999), DC programming: overview, *Journal of Optimization Theory and Applications*, 103(1), 1-43.
[19] Jain, A.K., Murty, M.N. and Flynn, P.J. (1999), Data clustering: a review, *ACM Comput. Surv.*, 31(3), 264-323.
[20] Jain, A.K. (2010), Data clustering: 50 years beyond k-means, *Pattern Recognition Letters*, 31(8), 651-666.
[21] Kogan, J. (2007), Introduction to Clustering Large and High-dimensional Data, Cambridge University Press.
[22] Lai, J.Z.C. and Huang, T.-J. (2010), Fast global k-means clustering using cluster membership and inequality, *Pattern Recognition*, 43(5), 1954-1963.
[23] Lichman, M. (2013), UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml.
[24] Likas, A., Vlassis, N. and Verbeek, J. (2003), The global k-means clustering algorithm, *Pattern Recognition*, 36(2), 451-461.
[25] Ordin, B. and Bagirov, A.M. (2015), A heuristic algorithm for solving the minimum sum-of-squares clustering problems, *Journal of Global Optimization*, 61, 341-361.
[26] Rahman, Md A. and Islam, Md Z. (2014), A hybrid clustering technique combining a novel genetic algorithm with k-means, *Knowledge-Based Systems*, 71, 345-365.
[27] Scitovski, R. and Scitovski, S. (2013), A fast partitioning algorithm and its application to earthquake investigation, *Computers & Geosciences*, 59, 124-131.
[28] Selim, S.Z. and Al-Sultan, K.S. (1991), A simulated annealing algorithm for the clustering, *Pattern Recognition*, 24(10), 1003-1008.
[29] Tao, P.D. and An, L.T.H. (1997), Convex analysis approach to DC programming: theory, algorithms and applications, *Acta Mathematica Vietnamica*, 22(1), 289-355.
[30] Teboulle, M. (2007), A unified continuous optimization framework for center-based clustering methods, *The Journal of Machine Learning Research*, 8, 65-102.
[31] Tuy, H., Bagirov, A.M. and Rubinov, A.M. (2001), Clustering via DC optimization, In Advances in Convex Analysis and Global Optimization, Springer, pp. 221-234.
[32] Xavier, A.E. (2010), The hyperbolic smoothing clustering method, *Pattern Recognition*, 43, 731-737.
[33] Xavier, A.E. and Xavier, V.L. (2011), Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions, *Pattern Recognition*, 44(1), 70-77.