An Efficient İncremental Clustering Algorithm for Very Large Data Sets

Adil M. Bagirov[1], Sona Taheri[2], Julien Ugon[3]
[1,2,3]Faculty of Science and Technology, Federation University Australia, Victoria, Australia
[1]*a.bagirov@federation.edu.au*

Abstract— **An algorithm for solving the minimum sum of squares clustering problems in large data sets is introduced. This algorithm uses a special structure of the clustering problem such as difference of convex representations of its objective functions. The use of such a structure allows one to design a clustering algorithm which is efficient in large and very large data sets. The proposed algorithm is tested and compared with other clustering algorithms using large real world data sets.**

Keywords— Large data sets, cluster analysis, nonsmooth optimization, difference of convex representation.

I. INTRODUCTION

Clustering is an unsupervised partitioning technique dealing with the problems of organizing a collection of patterns into clusters based on similarity. Most clustering algorithms are based on the hierarchical and partitional approaches. Algorithms based on the hierarchical approach generate a dendrogram representing the nested grouping of patterns and similarity levels at which groupings change [12]. Partitional clustering algorithms find the partition that optimizes a clustering criterion [12]. In this paper we develop a partitional clustering algorithm. More specifically, we develop an algorithm for solving the minimum sum-of-squares clustering (MSSC) problems.

The objective functions of the clustering problems, called cluster functions, can be represented as a difference of convex (DC) functions. There are several papers where the DC representation of the MSSC problems is used to design algorithms. In [10], the truncated codifferential method is applied to solve the MSSC using its DC representation. The branch and bound method was modified for such problems in [16] using their DC representation. In [1] an algorithm based on DC programming and DC Algorithms (DCA) is introduced. In [3], the authors use the hard combinatorial optimization model to formulate MSSC as a DC program and propose an algorithm based on DCA. Such an approach allows one to make simpler and less expensive computations in the resulting DCA. In [2], the DCA and a Gaussian kernel are applied to design an algorithm to solve the MSSC problem. All these algorithms are not always efficient for solving clustering problems in large data sets containing hundreds of thousands and more data points.

In this paper, we design an algorithm for solving the MSSC problem based on its DC representation. Results of numerical experiments on some real world data sets are reported and the proposed algorithm is compared with several other clustering algorithms. It is demonstrated that the proposed algorithm is especially efficient for solving the MSSC problems in very large data sets.

In what follows we denote by IR^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 $\|u\| = \langle u, u\rangle^{1/2}, u, v \in IR^n$. $B_\epsilon(x) = \{y \in IR^n : \|y - x\| < \varepsilon\}$ is the open ball centered at x with the radius $\varepsilon > 0$.

II. DC PROGRAMMING APPROACH TO CLUSTERING PROBLEMS

In this section we give a nonsmooth optimization formulation of clustering problems and their DC representations.

Definition 1. *$f : IR^n \to IR$ is called a DC function if there exist convex functions $g, h : IR^n \to IR$ such that:*

$$f(x) = g(x) - h(x), x \in IR^n.$$

Here $g - h$ is called a DC decomposition of f while g and h are DC components of f. A function f is locally DC if for any $x_0 \in IR^n$, there exist $\varepsilon > 0$ such that f is DC on the ball $B_\epsilon(x_0)$. It is well known that every locally DC function is DC [11]. Note that a DC function has infinitely many DC decompositions.

An unconstrained DC program is an optimization problem of the form:

$$minimize f(x) = g(x) - h(x) \text{ subject to } x \in IR^n. \qquad (1)$$

In cluster analysis we assume that we are given a finite set of points A in the $n-$dimensional space IR^n, that is $A = \{a^1, \dots, a^m\}$, where $ai \in IR^n$, $i = 1, \dots, m$. The hard unconstrained 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. $A^j = \emptyset_k$ and $A^j \cap A^l = \emptyset, j, l = 1, \dots, k, j = l.$
2. $A = \bigcup_{j=1}^{k} A^j$

The sets A^j , $j = 1,\ldots,k$ are called clusters and each cluster A^j can be identified by its center $x_j \in IR^n, j = 1,\ldots,k$.

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:

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

The nonsmooth optimization formulation of the MSSC problem is [7], [9]:

$$minimize\ f_k(x) \text{ subject to } x = (x^1,\ldots,x^k \in IR^{nk}), \quad (2)$$

where

$$f_k(x^1,\ldots,x^k) = \frac{1}{m}\sum_{a\in A} \min_{j=1,\ldots,k} d_2(x^j,a) \quad (3)$$

The objective function f_k in Problem (2) can be expressed as a DC function:

$$f_k(x) = f_{k1}(x) - f_{k2}(x), x = (x^1,\ldots,x^k \in IR^{nk}), \quad (4)$$

Where

$$f_{k1}(x) = \frac{1}{m}\sum_{a\in A}\sum_{j=1}^{k} d_2(x^j,a)$$

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

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 convex. Furthermore, since the maximum of a finite number of convex functions is also convex, the function f_{k2} is a sum of convex functions and therefore it is also convex.

Problem (2) is a global optimization problem, the objective function f_k in this problem has many local minimizers and only its global minimizers provide the best cluster structure of a data set with the least number of clusters. In general, conventional global optimization methods cannot be applied to solve this problem in large data sets. Therefore in such data sets heuristics and deterministic local search algorithms are the only choice. But the success of these algorithms heavily depends on the choice of starting cluster centers and the development of efficient procedures for generating starting clusters centers is crucial for the success of such algorithms. We apply an approach introduced in [15] to find starting cluster centers. This approach involves the solution of the so-called auxiliary clustering problem.

Assume that the solution $x^1,\ldots,x^{k-1}$, $k \geq 2$ 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 $k-1$ centers $x^1,\ldots,x^{k-1}$:

$$r_{k-1}^a = min\{d_2(x^1,a),\ldots,d_2(x^{k-1},a)\}. \quad (5)$$

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

$$\bar{f}_k(y) = \frac{1}{m}\sum_{a\in A} min\{r_{k-1}^a, d_2(y,a)\},\ y \in IR^n. \quad (6)$$

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 $\bar{f}_k(y) = f_k(x^1,\ldots,x^{k-1},y), \forall y \in IR^n$.

A problem:

$$\text{minimize } \bar{f}_k(y) \text{ subject to } y \in IR^n \quad (7)$$

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

$$\bar{f}_k(y) = \bar{f}_{k1}(y) - \bar{f}_{k2}(y) \quad (8)$$

Where

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

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

An algorithm for solving optimization problems for solving both Problems (2) and (7) is described in [8]. This algorithm is based on DC representations of both clustering and auxiliary clustering functions.

III. INCREMENTAL ALGORITHM

In this section we present an incremental algorithm for solving Problems (2) and (7) using the DC approach. An important part of this algorithm is a procedure for finding starting points for the l-th cluster center where $1 \leq l \leq k$. This procedure was described in detail in [15].

Algorithm 1 An incremental clustering algorithm.

1: (Initialization). Compute the center $x^1 \in IR^n$ of the set A. Set $l := 1$.

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

3: (Computation of a set of starting points for the auxiliary clustering problem). Apply the procedure from [15] to find the set $S_1 \subset IR^n$ of starting points for solving the auxiliary clustering problem (7) for $k = l$.

4: (Computation of a set of starting points for the l-th cluster center). Apply the optimization algorithm to solve Problem (7) starting from each point $y \in S_1$. This algorithm generates a set $S_2 \subset IR^n$ of starting points for the l-th cluster center.

5: (Computation of a set of cluster centers). For each $\bar{y} \in S_2$ apply the optimization to solve Problem (2) starting from the point $(x^1, \dots, x^{l-1}, \bar{y})$ and find a solution $(\hat{y}^1, \dots, \hat{y}^l)$. Denote by $S_3 \subset IR^{nl}$ a set of all such solutions.

6: (Computation of the best solution). Compute

$$f_l^{min} = min\{f_l(\hat{y}^1, \dots, \hat{y}^l): (\hat{y}^1, \dots, \hat{y}^l) \in S_3\}$$

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

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

Since the clustering Algorithm 1 applies the optimization based on the DC representation to solve clustering problems it is called the *DCClust* algorithm. It is easy to see that this algorithm in addition to the k-partition problem solves also all intermediate l-partition problems where $l = 1, \dots, k - 1$. Steps 4 and 5 are the most time-consuming steps of this algorithm as the optimization is applied repeatedly.

TABLE 1. THE BRIEF DESCRIPTION OF DATA SETS

Data sets	Number of instances	Number of attributes
Gas Sensor Array Drift	13910	128
Skin Segmentation	245057	3
3D Road Network	434874	3

We compare the DCClust with the following algorithms:

1. The global k-means algorithm (GKM) [14].
2. The Multi-start modified global k-means algorithm (MSMGKM) [15].
3. The version of the Algorithm 1 where the optimization algorithm is replaced by the DCA [4] (MS-DCA).

All these algorithms are based on the incremental approach. The DCClust algorithm contains a special procedure to generate starting cluster centers (Step 3) which is described in detail in [6], [15]. To design the version of Algorithm 1 with the DCA in Steps 4 and 5 the optimization algorithm is replaced by the DCA.

IV. NUMERICAL RESULTS

To test the DCClust algorithm and compare it with other three clustering algorithms numerical experiments with a number of real-world data sets have been 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. Three data sets have been used in numerical experiments. Their brief description is given in Table I. The detailed description can be found in [13]. All data sets contain only numeric features and they do not have missing values.

We computed up to 25 clusters in all data sets. The CPU time used by algorithms is limited to 20 hours. Since all algorithms computes clusters incrementally we present results with the maximum number of clusters obtained by an algorithm during this time limit. Results for cluster function values found by different algorithms are presented in Table II. In this table we use the following notation:

- k is the number of clusters;
- f_{best} (multiplied by the number shown after names of data sets) is the best known value of the cluster function (3) (multiplied by m) for the corresponding number of clusters;
- E_A is the error in % by an algorithm A which is calculated as follows:

$$E_A = \frac{\bar{f} - f_{best}}{f_{best}} \times 100\%$$

where $\bar{f}$ is the value of the clustering function obtained by an algorithm A;
- The sign “-” in tables shows that an algorithm failed to compute clusters in the given time frame.

Results presented in Table II show that all four algorithms are able to find global or near global solutions to clustering

TABLE 2. CLUSTER FUNCTION VALUES OBTAINED BY ALGORITHMS

k	*fbest*	*EDCClust*	*EMS−DCA*	*EMGKM*	*EGKM*
Gas Sensor Array Drift ($\times 10^{13}$)					
2	79.11857	0.00	0.00	0.00	0.00
3	5.02412	0.00	0.00	0.00	0.00
5	3.22726	0.00	0.00	0.00	0.00
10	1.65524	0.00	0.00	0.00	0.00
12	1.40655	0.00	0.01	0.01	0.00
15	1.13801	0.35	0.00	0.00	0.36
20	0.87916	0.62	0.16	0.00	0.62
25	0.72348	0.47	0.00	0.00	0.16
Skin Segmentation ($\times 10^{9}$)					
2	1.32236	0.00	0.00	0.00	0.00
3	0.89362	0.00	0.00	0.00	0.00
5	0.50203	0.00	0.00	0.00	0.00
10	0.25122	0.00	0.00	0.00	0.00
12	0.21416	0.00	0.55	0.55	0.00
15	0.16964	0.18	0.19	0.18	0.00
20	0.12770	0.14	0.17	0.17	0.00
25	0.10299	0.00	0.00	0.00	0.00
3D Road Network ($\times 10^{6}$)					
2	49.13298	0.00	0.00	0.00	0.00
3	22.77818	0.00	0.00	0.00	0.00
5	8.82574	0.00	0.01	0.00	0.00
10	2.56710	0.00	0.21	-	-
12	1.84976	0.00	0.05	-	-
15	1.27072	0.00	0.26	-	-
20	0.80872	0.00	0.73	-	-
25	0.60334	1.93	0.00	-	-

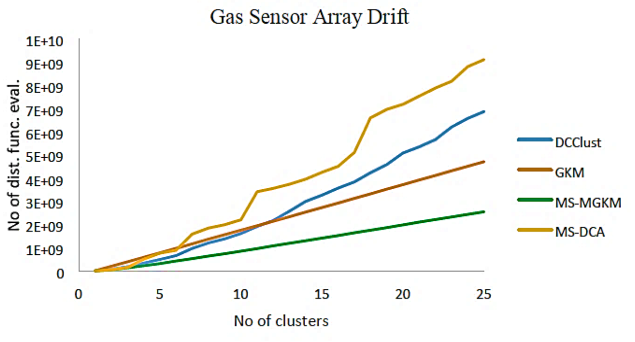
problems in large data sets, however the GKM and MSMGKM algorithms are not efficient for solving clustering problems within a given timeframe in data sets with hundreds of thousands of points.

Figures 1(a)-1(c) illustrate dependence of the number of distance function evaluations (N_d) on the number of clusters for four algorithms in all data sets. These figures demonstrate that the MS-MGKM algorithm requires the least number of distance function evaluations in two data sets. In the 3D Road Network data set this algorithm computed only 6 clusters within a 20 hours timeframe and the number N_d is similar to that of by the MS-DCA and DCClust algorithms. For the GKM algorithm N_d depends linearly on the number of clusters, however this algorithm requires significantly more distance function evaluations than other three algorithms. Comparison of the DCClust and the MS-DCA algorithms shows that the latter algorithm requires more distance function evaluations than the former algorithm.

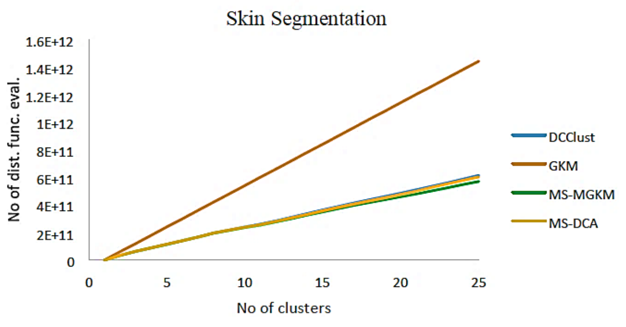
Figures 2(a)-2(c) illustrate dependence of the CPU time on the number of clusters for four algorithms in all data sets. It is obvious that the DCClust algorithm requires least CPU time among all four algorithms.

V. CONCLUSION

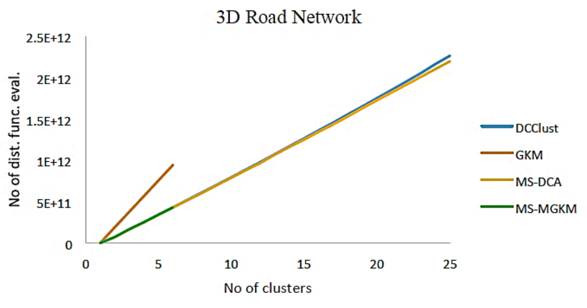
In this paper an algorithm for solving the minimum sumof-squares clustering problems is designed using their DC representation. The algorithm is an incremental algorithm and computes clusters gradually starting from one cluster which



(a)

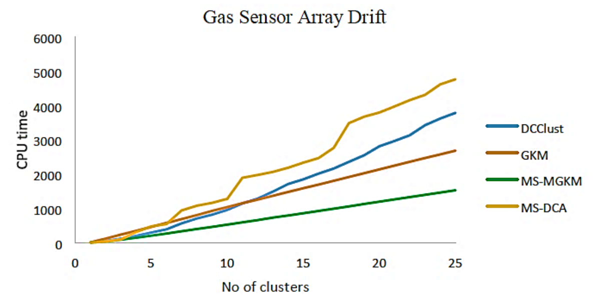


(b)

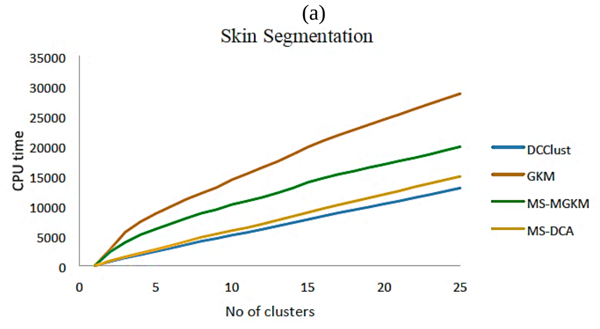


(c)

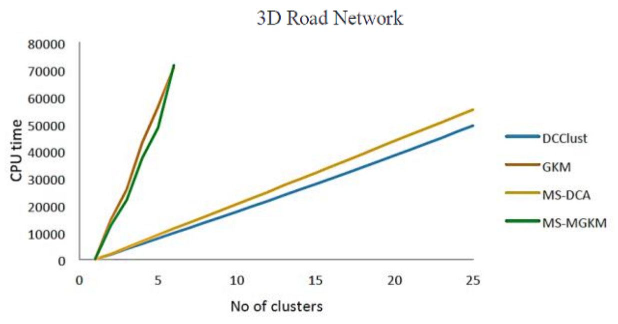
Figure 1. The number of distance function calculations vs the number of clusters.



(a)



(b)



(c)

Figure 2. The CPU time vs the number of clusters.

is the whole data set. The proposed algorithm is tested using real world data sets with the number of data points ranging from tens of thousands to hundreds of thousands and compared with other clustering algorithms. Results clearly demonstrate that the use of the DC representation of clustering problems allows one to significantly improve ability of incremental algorithms to solve clustering problems in very large data sets in a reasonable time. Furthermore, the use of nonsmooth optimization algorithms can increase this ability for even larger data sets.

ACKNOWLEDGMENT

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

REFERENCES

[1] L.T.H. An, M.T. Belghiti, and P.D. Tao. A new efficient algorithm based on DC programming and DCA for clustering. *J. of Global Optim.*, 37(4):593–608, 2007.

[2] L.T.H. An, L.H. Minh, and P.D. Tao. New and efficient DCA based algorithms for minimum sum-of-squares clustering. *Pattern Recognition*, 47:388–401, 2014.

[3] L.T.H. An and P.D. Tao. Minimum sum-of-squares clustering by DC programming and DCA. In D.-S. Huang, K.-H. Jo, H.-H. Lee, H.-J. Kang, and V. Bevilacqua, editors, *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence, ICIC 2009, LNAI-5755*, pages 327–340. Springer-Verlag, Berlin, Heidelberg.

[4] L.T.H. An and P.D. Tao. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133:23–46, 2005.

[5] A.M. Bagirov. Modified global k-means algorithm for minimum sumof-squares clustering problems. *Pattern Recognition*, 41(10):3192–3199, 2008.

[6] A.M. Bagirov, B. Ordin, G. Ozturk, and A.E. Xavier. An incremental clustering algorithm based on hyperbolic smoothing. *Computational Optimization and Applications*, 61:219–241, 2015.

[7] A.M. Bagirov, A.M. Rubinov, N.V. Soukhoroukova, and J. Yearwood. Unsupervised and supervised data classification via nonsmooth and global optimization. *Top*, 11:1–93, 2003.

[8] A.M. Bagirov, S. Taheri, and J. Ugon. Nonsmooth dc programming approach to the minimum sum-of-squares clustering problems. *Pattern Recognition, submitted*, 2015.

[9] A.M. Bagirov and J. Yearwood. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578–596, 2006.

[10] V.F. Demyanov, A.M. Bagirov, and A.M. Rubinov. A method of truncated codifferential with application to some problems of cluster analysis. *Journal of Global Optimization*, 23(1):63–80, 2002.

[11] R. Horst and N.V. Thoai. DC programming: Overview. *Journal of Optimization Theory and Applications*, 103(1):1–43, 1999.

[12] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[13] M. Lichman. UCI machine learning repository, http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences, 2013.

[14] A. Likas, N. Vlassis, and J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.

[15] B. Ordin and A.M. Bagirov. A heuristic algorithm for solving the minimum sum-of-squares clustering problems. *Journal of Global Optimization*, 61:341–361, 2015.

[16] Hoang Tuy, A.M. Bagirov, and A.M. Rubinov. Clustering via DC optimization. In *Advances in Convex Analysis and Global Optimization*, pages 221–234. Springer, 2001.