Clustering in large data sets with the limited memory bundle method

Napsu Karmitsa[a,*], Adil M. Bagirov[b], Sona Taheri[b]

[a] *Department of Mathematics and Statistics, University of Turku, FI-20014 Turku, Finland.*
E-mail: napsu@karmitsa.fi
[b] *Faculty of Science and Technology, Federation University Australia, Victoria, Australia.*
Email: a.bagirov@federation.edu.au, s.taheri@federation.edu.au

Abstract

The aim of this paper is to design an algorithm based on nonsmooth optimization techniques to solve the minimum sum-of-squares clustering problems in very large data sets. First, the clustering problem is formulated as a nonsmooth optimization problem. Then the limited memory bundle method [Haarala et.al. *Math. Prog.*, Vol. 109, No. 1, pp. 181–205, 2007] is modified and combined with an incremental approach to design a new clustering algorithm. The algorithm is evaluated using real world data sets with both the large number of attributes and the large number of data points. It is also compared with some other optimization based clustering algorithms. The numerical results demonstrate the efficiency of the proposed algorithm for clustering in very large data sets.

Keywords: Cluster analysis, Nonsmooth optimization, Nonconvex optimization, Bundle methods, Limited memory methods

1. Introduction

Clustering is one of the most important tasks in data mining and machine learning. It deals with a problem of organizing a collection of patterns into clusters based on similarity. In this paper, we consider the so-called *hard clustering problem* where each data point belongs only to one cluster. The similarity mea-

*Corresponding author.
Email address: napsu@karmitsa.fi (Napsu Karmitsa)

sure is a key notion in the cluster analysis and it can be defined using different norms. Here we use a squared Euclidean norm. The squared Euclidean norm is probably the most common choice as a similarity measure and the clustering problems with this norm are known as *the minimum sum-of-squares clustering (MSSC) problem*.

The MSSC can be formulated as a global optimization problem and various optimization techniques have been used to design clustering algorithms. These techniques include

- nonsmooth optimization (NSO, not necessarily differentiable optimization [1]) methods [2, 3, 4, 5];
- methods based on difference of convex representation of clustering functions [6, 7, 8, 9, 10] and global optimization [11];
- the hyperbolic smoothing technique [12, 13, 14];
- the variable neighborhood search algorithm [15]; and
- metaheuristics such as evolutionary algorithms [16], simulated annealing [17], tabu search [18], and genetic algorithm [19].

In addition, heuristics such as the k-means algorithm and its variations are widely used to solve this problem (see, [20, 21, 22] and references therein).

Recent advances in computer hardware allow one to store data sets containing hundreds of thousands or millions of records and/or hundreds or thousands of attributes in the random access memory (RAM). This creates opportunities to consider a huge amount of data at once and to achieve accurate clustering results in such data sets. However, most existing clustering algorithms are not applicable in such data sets because either they can only generate suboptimal solutions or they require a prohibitively large computational effort. It is, therefore, imperative to develop algorithms which can generate accurate solutions to clustering problems in very large data sets in a reasonable time. When using the nonsmooth formulation of the clustering problem the number of variables in the underlying optimization problem does not depend on the number of instances but only on numbers of attributes and clusters. By appending powerful

nonsmooth optimization methods that are efficient in large-scale problems, we are able to design algorithms to solve very large clustering problems.

In this paper, we introduce a new LMBM-Clust method for solving the MSSC problems. This method consists of two algorithms: an incremental algorithm by Ordin and Bagirov [23] and the limited memory bundle algorithm (LMBM) by Karmitsa (née Haarala) et al. [24, 25, 26]. The first algorithm is used to solve clustering problems globally. The second algorithm is applied to solve both the clustering and the so-called auxiliary clustering problems at each iteration of the incremental algorithm.

The LMBM is known to be an efficient method for solving large scale NSO problems (both convex and nonconvex) [24, 26, 27]. This important property of the method allows one to design an algorithm based on it to efficiently solve clustering problems in very large data sets. Its combination with the incremental algorithm leads to the design of an accurate clustering method. The main contributions of this paper are:

(i) the development of an algorithm which is accurate and efficient for solving clustering problems in data set containing hundreds of thousands of points and/or hundreds of attributes;

(ii) numerical evaluation of the proposed LMBM-Clust method and its comparison with other algorithms using very large data sets.

This paper is organized as follows. In Section 2 we introduce our notations and recall some basic definitions from nonsmooth analysis. The formulation of the nonsmooth clustering problem is given in Section 3. In Section 4, we first give the basic ideas of the LMBM and its convergence properties. Then, we describe the new incremental clustering algorithm. The results of the numerical experiments are presented and discussed in Section 5. Finally, Section 6 concludes the paper.

2. Notations and Background

We denote by $\|\cdot\|$ the Euclidean norm in $\mathbb{R}^n$ and by $\boldsymbol{a}^T\boldsymbol{b}$ the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ (bolded symbols are used for vectors).

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

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\| \text{ for all } \boldsymbol{x}, \boldsymbol{y} \in X.$$

The *subdifferential* $\partial f(\boldsymbol{x})$ of a LLC function $f : \mathbb{R}^n \to \mathbb{R}$ at any point $\boldsymbol{x} \in \mathbb{R}^n$ is given by [28]

$$\partial f(\boldsymbol{x}) = \text{conv}\left\{ \lim_{i \to \infty} \nabla f(\boldsymbol{x}_i) \mid \boldsymbol{x}_i \to \boldsymbol{x} \text{ and } \nabla f(\boldsymbol{x}_i) \text{ exists} \right\},$$

where "conv" denotes the convex hull of a set. A vector $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ is called a *subgradient*. The point $\boldsymbol{x}^* \in \mathbb{R}^n$ is called *stationary* if $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. Stationarity is a necessary condition for local optimality.

3. Nonsmooth formulations of clustering problems

In this section we present NSO formulations of the clustering and so-called auxiliary clustering problems. First, we recall the definition of the clustering problem.

Cluster Analysis. In the cluster analysis we assume that a finite set A of points in the n-dimensional space $\mathbb{R}^n$ is given. That is,

$$A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\}, \text{ where } \boldsymbol{a}_i \in \mathbb{R}^n, i = 1, \ldots, m.$$

The data points $\boldsymbol{a}_i$, $i = 1, \ldots, m$ are called *instances* and each instance has n *attributes*.

The *hard unconstrained clustering problem* is the distribution of points of the set A into a given number k of disjoint subsets A^j, $j = 1, \ldots, k$ with respect to predefined criteria such that

1. $A^j \neq \emptyset$, $j = 1, \ldots, k$;
2. $A^j \bigcap A^l = \emptyset$, for all $j, l = 1, \ldots, k$, $j \neq l$; and
3. $A = \bigcup_{j=1}^{k} A^j$.

The sets $A^j, j = 1, \ldots, k$ are called *clusters*. Each cluster A^j can be identified by its *center* $\boldsymbol{x}_j \in \mathbb{R}^n, j = 1, \ldots, k$. The problem of finding these centers is called the *k-clustering* (or k-partition) *problem*.

The notion of the *similarity measure* is fundamental to formulate the clustering problem. Usually a similarity measure is defined as an inverse of some distance metrics. Here, we define it using the squared Euclidean norm (the L_2-norm)

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

Clustering Problem. The *NSO formulation of the MSSC problem* is given by [4, 29]

$$\begin{cases} \text{minimize} & f_k(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \in \mathbb{R}^{nk}, \end{cases} \tag{1}$$

where

$$f_k(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) = \frac{1}{m} \sum_{\boldsymbol{a} \in A} \min_{j=1,\ldots,k} d_2(\boldsymbol{x}_j, \boldsymbol{a}). \tag{2}$$

The function f_k is called the *k-th cluster function*. It is LLC for any k, and nonconvex and nonsmooth for $k > 1$.

In many real-world data sets, the number of instances m is substantially greater than the number of attributes n. One advantage in using the NSO formulation of the clustering problem is that the number of variables in Problem (1) is only $n \times k$ and it does not depend on the number of instances m.

Auxiliary Clustering Problem. The clustering is a global optimization problem, but conventional global optimization methods are not applicable to solve it in large data sets. Therefore, we apply a local search algorithm. Nevertheless, the quality of a solution found by a local search algorithm depends on the choice of starting cluster centers. Special procedures have been designed to find such centers. In this paper, we apply a procedure introduced in [23]. This procedure involves the solution of the so-called auxiliary clustering problem.

Given the solution $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}$, $k \geq 2$ to the $(k-1)$-clustering problem define

$$r_{k-1}^{\boldsymbol{a}} = \min\left\{d_2(\boldsymbol{x}_1, \boldsymbol{a}), \ldots, d_2(\boldsymbol{x}_{k-1}, \boldsymbol{a})\right\}.$$

The *k-th auxiliary cluster function* is defined as [20]

$$\bar{f}_k(\boldsymbol{y}) = \frac{1}{m} \sum_{\boldsymbol{a} \in A} \min\left\{r_{k-1}^{\boldsymbol{a}}, d_2(\boldsymbol{y}, \boldsymbol{a})\right\}, \ \boldsymbol{y} \in \mathbb{R}^n. \tag{3}$$

This function is nonsmooth LLC and as a sum of minima of convex functions it is, in general, nonconvex. The *k-th auxiliary clustering problem* [20] is formulated as

$$\begin{cases} \text{minimize} & \bar{f}_k(\boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} \in \mathbb{R}^n. \end{cases} \tag{4}$$

4. LMBM-Clust Method

In this section we introduce a new LMBM-Clust method for solving MSSC problems. As already mentioned in the introduction, the LMBM-Clust method consists of two algorithms: an incremental clustering algorithm is used to solve clustering problems globally and at each iteration of this algorithm the LMBM is applied to solve both the clustering problem (1) and the auxiliary clustering problem (4). Figure 1 illustrates the structure of this combination and basic ideas of these algorithms.

4.1. LMBM

The LMBM is originally developed for solving general nonconvex NSO problems [25, 26]. Here, the original algorithm is slightly modified to be better suited for solving clustering problems. In this section our notation differs a little bit from that before: we always use f to denote the objective function and n to denote the size of the optimization problem. That is, $f = \bar{f}_l$ and $n = n$ for the auxiliary clustering problem (4), and $f = f_l$ and $n = nl$ for the clustering problem (1), where $l \in [1, k]$ is the current number of clusters and k is the predefined maximum number of clusters for the incremental algorithm. To

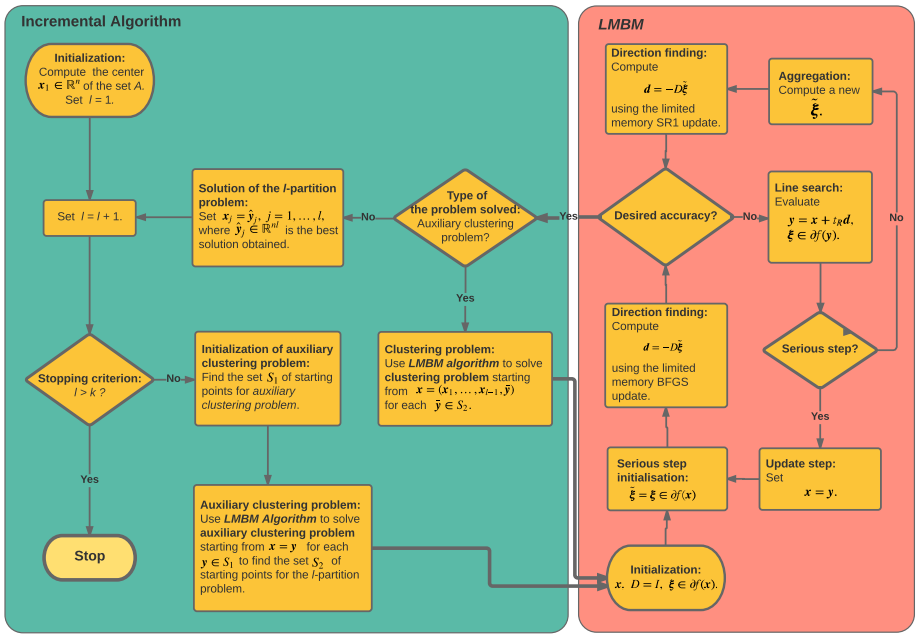


Figure 1: LMBM-Clust method.

apply the LMBM we need to assume that the objective function f is LLC and at every point $\boldsymbol{x}$ both the value of the objective function $f(\boldsymbol{x})$ and one arbitrary subgradient $\boldsymbol{\xi}$ from the subdifferential $\partial f(\boldsymbol{x})$ can be evaluated. For (auxiliary) clustering problems these assumptions are trivially satisfied.

The most essential feature of the LMBM is the usage of null steps together with a simple aggregation of subgradients. In addition, the limited memory approach (see e.g. [30]) is utilized in calculations of search directions and aggregated values. The L-BFGS update formula is used after a serious step and the L-SR1 update formula after a null step. The usage of null steps gives further information about the nonsmooth objective function whenever the search direction is not "good enough". On the other hand, a simple aggregation of subgradients guarantees the global convergence of the method.

Algorithm. We now give some more details of the LMBM in a form of a pseudocode. Later, we will describe the incremental clustering algorithm to find suitable starting points for this algorithm when solving (auxiliary) clustering prob-

lems. Here the following input is needed:

- $\boldsymbol{x}_1 \in \mathbb{R}^n$ — starting point;
- $\varepsilon > 0$ — stopping tolerance;
- $\hat{m}_c \geq 3$ — the maximum number of stored correction vectors used to form the limited memory matrix updates.

Algorithm 1: LMBM

Data: $\boldsymbol{x}_1 \in \mathbb{R}^n$, $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$, $\hat{m}_c \geq 3$, and $\varepsilon > 0$

Result: final solution $\boldsymbol{x}_h$

Compute $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$;

Set $h = 1$, $m = 1$, $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$, $\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$, and $\tilde{\beta}_1 = 0$;

while *the termination condition* $w_h = -\tilde{\boldsymbol{\xi}}_h^T \boldsymbol{d}_h + 2\tilde{\beta}_h \leq \varepsilon$ *is not met* **do**

Find step sizes t_L^h and t_R^h, and the subgradient locality measure β_{h+1};

Set $\boldsymbol{x}_{h+1} = \boldsymbol{x}_h + t_L^h \boldsymbol{d}_h$ and $\boldsymbol{y}_{h+1} = \boldsymbol{x}_h + t_R^h \boldsymbol{d}_h$;

Evaluate $f(\boldsymbol{x}_{h+1})$ and $\boldsymbol{\xi}_{h+1} \in \partial f(\boldsymbol{y}_{h+1})$;

Store the new correction vectors $\boldsymbol{s}_h = \boldsymbol{y}_{h+1} - \boldsymbol{x}_h$ and $\boldsymbol{u}_h = \boldsymbol{\xi}_{h+1} - \boldsymbol{\xi}_m$;

Set $\hat{m}_h = \min\{h, \hat{m}_c\}$;

if $t_L^h > 0$ **then** *(Serious step)*

Compute the search direction $\boldsymbol{d}_{h+1}$ using $\boldsymbol{\xi}_{h+1}$ and L-BFGS update with $\hat{m}_h$ most recent correction vectors;

Set $m = h + 1$ and $\tilde{\beta}_{h+1} = 0$;

else *(Null step)*

Determine multipliers λ_i^h satisfying $\lambda_i^h \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^3 \lambda_i^h = 1$ that minimize the function

$$(\lambda_1, \lambda_2, \lambda_3) = [\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{h+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_h]^T D^h [\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{h+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_h] + 2(\lambda_2 \beta_{h+1} + \lambda_3 \tilde{\beta}_h);$$

Compute the aggregate values

$$\tilde{\boldsymbol{\xi}}_{h+1} = \lambda_1^h \boldsymbol{\xi}_m + \lambda_2^h \boldsymbol{\xi}_{h+1} + \lambda_3^h \tilde{\boldsymbol{\xi}}_h \quad \text{and} \quad \tilde{\beta}_{h+1} = \lambda_2^h \beta_{h+1} + \lambda_3^h \tilde{\beta}_h;$$

Compute the search direction $\boldsymbol{d}_{h+1}$ using $\tilde{\boldsymbol{\xi}}_{h+1}$ and L-SR1 update with $\hat{m}_h$ most recent correction vectors;

Set $h = h + 1$;

Remark 1. When combined with incremental approach we use a nonmonotone line search to find step sizes t_L^h and t_R^h. In addition, we use different stopping tolerances for different problems. That is, we set the tolerance ε large for the auxiliary clustering problem — since this problem need not to be solved very accurately — and smaller for the clustering problem.

Global Convergence. We now recall convergence properties of the LMBM algorithm. These properties are obtained under the following assumptions that are trivially satisfied for both the clustering and the auxiliary clustering problems.

Assumption 1. *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is LLC.*

Assumption 2. *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is upper semismooth (see e.g. [31]).*

Assumption 3. *The level set $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ is bounded for every starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$.*

Under these assumptions, theorems on the convergence of the LMBM are proved in [26]. These theorems can be modified for the clustering and auxiliary clustering problems as follows.

Theorem 1. *If the* LMBM *terminates after a finite number of iterations, say at iteration k, then the point $\boldsymbol{x}_k$ is a stationary point of the (auxiliary) clustering problem.*

Theorem 2. *Every accumulation point $\bar{\boldsymbol{x}}$ generated by the* LMBM *is a stationary point of the (auxiliary) clustering problem.*

Remark 2. The LMBM terminates in a finite number of steps if we choose $\varepsilon > 0$.

4.2. LMBM-Clust *Method*

Now we present the incremental clustering algorithm for solving the clustering problem (1). Since problem (1) is nonconvex, it is important to select good or promising starting points before applying a local method like the LMBM

to solve it. That way we are able to get good quality global, or near global, solutions. We use the algorithm (Algorithm 1) introduced in [23] to generate starting cluster centers in our incremental clustering algorithm given below. The LMBM is applied at each iteration of the algorithm to solve both the clustering and the auxiliary clustering problems. The incremental clustering algorithm based on the combination of the incremental approach and LMBM is called the LMBM-Clust method.

Algorithm 2: LMBM-Clust

Data: the maximum number of clusters $k \geq 1$

Result: the solution to the k-partition problem

Compute the center $\boldsymbol{x}_1 \in \mathbb{R}^n$ of the set A;

Set $l = 1$;

while $l \leq k$ **do**

- Set $l = l + 1$;
- Apply Algorithm 1 from [23] to find the set $S_1 \subset \mathbb{R}^n$ of starting points for the auxiliary clustering problem (4) with $k = l$;
- **Solving the auxiliary clustering problem**
  - Apply the LMBM to solve Problem (4) starting from each point $\boldsymbol{y} \in S_1$ to obtain a set $S_2 \subset \mathbb{R}^n$ of starting points for the l-th clustering problem (1)
- **Solving the clustering problem**
  - For each $\bar{\boldsymbol{y}} \in S_2$ apply the LMBM to solve Problem (1) starting from the point $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}, \bar{\boldsymbol{y}})$ and find a solution $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l)$;
  - Denote by $S_3 \subset \mathbb{R}^{nl}$ a set of all such solutions
- **Solution to the l-partition problem**
  - Compute $f_l^{min} = \min \{f_l(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l) \mid (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l) \in S_3\}$ and the collection of cluster centers $(\bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_l)$ such that $f_l(\bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_l) = f_l^{min}$;
  - Set $\boldsymbol{x}_j := \bar{\boldsymbol{y}}_j$, $j = 1, \ldots, l$ as a solution to the l-partition problem

In addition to the k-partition problem, LMBM-Clust solves also all intermediate l-partition problems where $l = 1, \ldots, k-1$.

5. Numerical Experiments

To demonstrate the performance of the proposed LMBM-Clust method (`LMBM--Clust`) we compare it with four other clustering algorithms. Namely, `DC-Clust` [9] that utilizes the nonsmooth DC-representation of the MSSC clustering problem; multi-start modified global k-means algorithm `MS-MGKM` [23] that utilizes an incremental algorithm to find starting points for the k-means algorithm; the classical multi-start k-means algorithm `MS-KM` [32]; and the `k-means++` algorithm [33] that uses a heuristic to find centroid seeds for k-means clustering. In addition, the results of the older implementation of LMBM-Clust can be found in [34]. Note that the new implementation uses more efficient distance function calculation and the better optimized code.

Solvers `LMBM-Clust`, `DC-Clust`, and `MS-MGKM` are implemented in Fortran 95 while `MS-KM` is implemented in Fortran 77. For `k-means++` the MATLAB build-in function was used. All the Fortran codes were compiled using `gfortran`, the GNU fortran compiler and the experiments were performed on iMac (macOS Sierra, 10.12.6) with Intel® Core™ i7, 4 GHz and RAM 16 GB. The source codes of `LMBM-Clust` and `DC-Clust` are available at http://napsu.karmitsa.fi/clustering/. To run `k-means++` we used MATLAB2015 in a PC with Intel® Core™ i5-3470S, 2.90 GHz and RAM 8 GB.

5.1. Experiments in Large Real World Data

Data sets. Thirteen publicly available large real world data sets were used in our experiments. The brief description of these sets is given in Table 1. The more detailed description can be found in [35] and references given in Table 1. All the data sets contain only numeric features and they do not have missing values. The number of attributes ranges from very few (2) to very large (5000) and the number of instances ranges from thousands (smallest 7 797) to hundreds of thousands (largest 434 874).

Setup. `LMBM-Clust`, `DC-Clust` and `MS-MGKM` use the incremental approach to solve clustering problems. Applying these methods we compute incrementally

Table 1: The brief description of data sets

Data sets	No. instances	No. attributes	References
ISOLET	7797	617	[35]
Gisette	13500	5000	[36]
Gas Sensor Array Drift	13910	128	[37]
EEG Eye State	14980	14	[35]
D15112	15112	2	[38]
Online News Popularity	39644	58	[39]
KEGG Metabolic	53413	20	[40]
Shuttle Control	58000	9	[35]*
Sensorless Drive Diagnosis	58509	48	[35]
Pla85900	85900	2	[38]
MiniBooNE Particle Identification	130064	50	[35]
Skin Segmentation	245057	3	[41]
3D Road Network	434874	3	[42]

Thanks to NASA.

up to 25 clusters in all data sets. Since `MS-KM` and `k-means++` do not give any intermediate results, we perform different runs with different number of clusters for comparison purposes. The CPU time used by all algorithms was limited to 20 hours. For `MS-KM` the maximum number of starting points was always kept big enough but its computational time is limited to be about twice the one used by `LMBM-Clust`. Furthermore, `MS-KM` was always allowed to finish calculations from the initial point it started from even if the time limit was reached, unless in more than 20 hours it did not make any progress. In addition to CPU time, we report the number of used starting points (n_s) for `MS-KM`. Results of numerical experiments are presented in Tables 2 – 14 and illustrated in Figures 2 – 14. In all tables k stands for the number of clusters. We use the following metrics and parameters to compare different algorithms:

1. *Cluster function values,* also known as the sum of squares error (SSE). SSE is a prototype-based cohesion measure and it measures the average variation over all the clusters. In tables we present the best known value f_{best} of the cluster function (2) (multiplied by the number shown after the name of the data set and the number of points m) and the relative

errors of each algorithm. The relative error E_A by an algorithm A is calculated as:

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

where $\bar{f}$ is the value of the cluster function obtained by an algorithm A. We use the f_{best} value given in [5, 9] (for those data sets that were used also in [5, 9]) unless we get a better value in our experiments. Best values obtained in our experiments are marked with an asterisk.

2. *CPU time,* denoted as "*cpu*" in tables.
3. *The number of distance function evaluations.*
4. *Davies-Bouldin* (DBI) and *Dunn* (DI) *cluster validity indices.* Definitions of these indices are given below.

The validity indices are computed as follows: let $A^1, \ldots, A^k$ be a cluster distribution of the set A for a given $k > 0$. In addition, let $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be the Euclidean distance between cluster centers $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and let $r(\boldsymbol{x}_l)$ be the radius of the l-th cluster

$$r(\boldsymbol{x}_l) = \max_{\boldsymbol{a} \in A^l} \|\boldsymbol{x}_l - \boldsymbol{a}\|.$$

The DI is defined as [43]

$$\mathrm{DI} = \min_{i=1,\ldots,k} \left\{ \min_{j=1,\ldots,k, j \neq i} \left\{ \frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\max_{l=1,\ldots,k} r(\boldsymbol{x}_l)} \right\} \right\},$$

and it maximizes the inter cluster distances and minimizes the intra cluster distances. Therefore, the number of clusters that maximizes the DI can be taken as the optimal number of clusters.

The DBI is given by [44]

$$\mathrm{DBI} = \frac{1}{k} \sum_{i=1}^{k} \max_{j=1,\ldots,k, i \neq j} \frac{S_k(A^i) + S_k(A^j)}{d(\boldsymbol{x}_i, \boldsymbol{x}_j)}.$$

Here, $S_k(A^l)$ is the average distance of all data points from the cluster A^l to their cluster center $\boldsymbol{x}_l$. The ratio DBI is small if the clusters are compact and far from each other. Consequently, the DBI will have the smallest value for an optimal clustering. In our experiments, these indices are computed only for incremental algorithms.

Results. Numerical results on the best known values of the cluster function, errors and the CPU times used by different algorithms are presented in Tables 2 – 14. The dependence of the number of distance function calculations used by the algorithms on the number of clusters are given in Figures 2 – 14 (a), and the obtained validity indices for each data sets are summarized in Figures 2 – 14 (b) and (c).

The CPU time of `k-means++` may not be directly comparable to that of other solvers, since the tests were run on different platforms and computers. Nevertheless, we still can conclude that `k-means++` was the most efficient among tested solvers but with the data sets Sensorless Drive Diagnosis with $k = 10$, MiniBooNe Particle Identification with $k = 10$ and 3D Road Network with $k = 10$, 20 and 25, where the new solver `LMBM-Clust` succeeds in solving the clustering problem more efficiently (see Tables 10, 12 and 14). It is also worth noting that `k-means++` only computes a given number k clusters, whereas `LMBM-Clust` computes all the intermediate clusters with $l = 1, ..., k$ and, apart from very few isolated cases, `LMBM-Clust` does this far more accurately. Indeed, the accuracy of `k-means++` was not very good (although, better than `MS-KM` with the given time limits) and it worsens as the number of attributes and clusters increase. The only exceptions are MiniBooNE Particle Identification where the accuracy of `k-means++` was quite good even if the number of attributes is 50 and Skin Segmentation, where the accuracy was not good and the number of attributes is only three (see Tables 12 and 13).

The accuracy of `MS-KM` is mostly not good. In the worst case the error $E_{\texttt{MS-KM}} = 449091.75$ (see Table 5). One — but not the only — reason to this is that in larger data sets `MS-KM` only succeeded in solving the clustering problem from one random starting point within given time limits (twice the CPU time used by `LMBM-Clust`). Moreover, in MiniBooNE Particle Identification data set `MS-KM` solved the problem only with two clusters within the maximum time limit of 20 hours and that with the error $E_{\texttt{MS-KM}} = 286983.96$. For comparison purposes, we point out that `k-means++` used about 68 seconds and also the new solver `LMBM-Clust` used only 152 seconds to accurately find 25 clusters in this

data set (see Table 12).

The LMBM-Clust algorithm is clearly the most efficient among incremental clustering algorithms. Notice that in comparison with other two incremental algorithms the differences in CPU time grow in favor of LMBM-Clust as the number of instances increases. This can be best seen when comparing results in data sets with similar number of attributes. That is, for instance, data sets D15112, Pla85900, Skin Segmentation, and 3D Road Network containing 2-3 attributes. Results for these data sets are given in Tables 6, 11, 13, and 14, respectively. These results show that in the data set D15112 with the smallest number of instances LMBM-Clust was about 7 times faster than DC-Clust and 2.5 times faster than MS-MGKM ($k = 25$) while in the data set 3D Road Network with the largest number of instances the corresponding multipliers were 92 and 914. The similar kind of effect can be observed in data sets Online News Popularity, Sensorless Drive Diagnosis, and MiniBooNe Particle Identification which contain 48–58 attributes (see Tables 7, 10, and 12).

The CPU time of MS-MGKM increase rapidly as the number of instances increases while for LMBM-Clust and DC-Clust the CPU time is more dependent on the overall size of a data (i.e. both number of instances and number of attributes). In particular, all algorithms require the largest CPU time in the data set Gisette which is the one with the largest number of attributes (5000) and also the largest overall size (see Table 1 for the data set and Table 3 and Figure 3 for results). In this data set, DC-Clust succeeds in finding only six clusters in the given time limit and MS-MGKM twelve. LMBM-Clust is about 165 times faster than DC-Clust and 73 times faster than MS-MGKM for $k = 5$. Here, k-means++ failed to produce any result for this data set due to the memory restriction in its MATLAB implementation.

The accuracy of the solvers based on the incremental approach are quite similar and the solutions found are in most cases are the best or close to the best ones. The noticeable exceptions for LMBM-Clust are the MiniBooNE Particle Identification data set with $k = 3$, and the Online News Popularity and Skin Segmentation data sets with $k \geq 10$ (see Tables 12, 7 and 13). Note, how-

ever, that in the MiniBooNE Particle Identification data set `LMBM-Clust` finds accurate solutions for larger values of k.

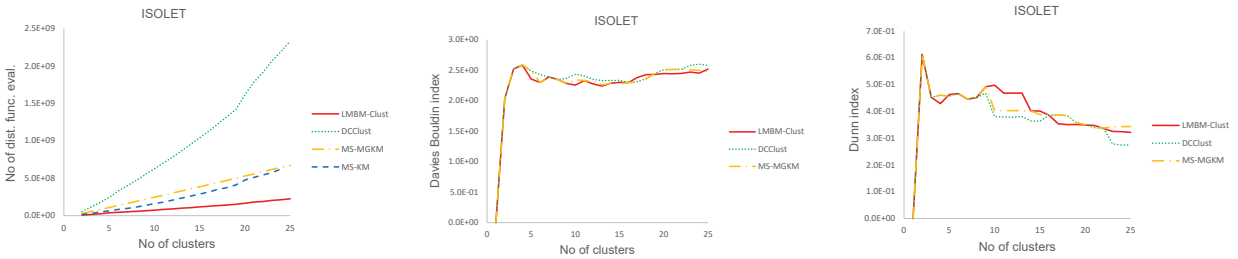
One can see from Figures 2–14 (a) that `LMBM-Clust` always use less and in some cases significantly less distance function evaluations than `DC-Clust` and `MS-MGKM`.

Results for the cluster validity index DBI, presented in Figures 2–14 (b), show that in seven out of 13 data sets all three incremental algorithms produce quite similar values of DBI. In five data sets, Gas-Sensor Array Drift, Shuttle Control, Sensorless Drive Diagnosis, MiniBooNE and Skin Segmentation, `LMBM-Clust` gives smaller values of DBI than the other two algorithms for most k. Only in one data set: Online News Popularity, DBI values of `LMBM-Clust` are larger than those of `DC-Clust` and `MS-MGKM`. These results confirm that according to the DBI index in some large data sets the LMBM-Clust algorithm is able to find better separated clusters than the other two algorithms.

Results for the DI (see Figures 2–14 (c)) are not always in agreement with those for the DBI. Nevertheless, it is well-known fact that even if there are several validity indices, most of them succeed only in certain situations. In most data sets (10 out of 13) all three algorithms give very similar values of the DI. In two data sets, ISOLET and Skin Segmentation, for most k `LMBM-Clust` finds larger values of the DI than the other two algorithms. In the 3D Road Network data set according to the DI the performance of `DC-Clust` and `MS-MGKM` is better than that of `LMBM-Clust`. These results also demonstrate that in large data sets the DBI is more accurate than the DI to identify the optimal number of clusters. This result is in line with the known weakness of DI — its sensitivity to noise. Nevertheless, in some data sets, the DBI and the DI clearly point out the optimal number of clusters. More specifically, according to these indicators there are two clusters in MiniBooNE Particle Identification, Skin Segmentation and 3D Road Network, three in Online News Popularity, KEGG Metabolic, and Sensorless Drive Diagnosis, and four in Gas Sensor Array Drift, EEG Eye State, and Shuttle Control. With incremental algorithms we could easily use the intermediate results obtained in these cases.

Table 2: Summary of the results with ISOLET ($\times 10^5$).

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	7.21940	0.00	6.01	0.00	72.88	0.00	41.17	0.00	12.40 (28)	35.92	0.56
3	6.78782	0.00	11.53	0.00	148.74	0.00	78.87	0.00	23.23 (28)	27.24	0.81
5	6.13651	0.00	24.68	1.04	311.33	1.01	153.36	0.00	50.45 (47)	18.53	1.61
10	5.28577	2.84	46.58	1.14	766.54	1.30	343.48	0.00	94.73 (47)	15.96	2.83
15	4.87391	0.54	73.75	0.09	1248.91	0.00	542.15	1.04	149.85 (33)	13.95	2.72
20	4.60857	0.03	103.67	0.00	1864.83	0.00	737.50	1.25	207.46 (26)	13.26	3.80
25	4.45658	0.00	141.26	0.00	2571.78	0.39	930.94	0.81	285.6111 (23)	5.94	8.97



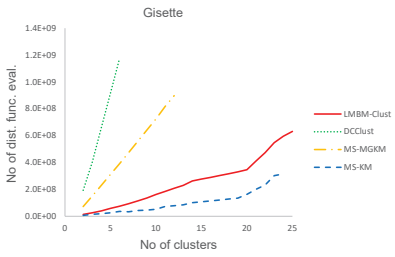
(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 2: ISOLET: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 3: Summary of the results with Gisette ($\times 10^{12}$).

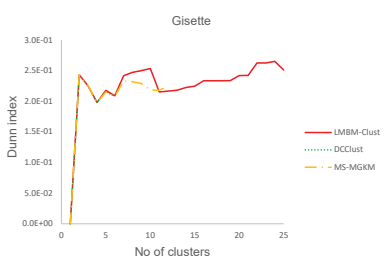
k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	4.19944	0.00	89.47	0.00	11755.17	0.00	5521.84	0.00	180.46 (4)	–*	–
3	4.11596	0.00	152.00	0.00	24362.95	0.00	11476.80	0.00	324.13 (4)	–	–
5	4.02303	0.00	321.57	0.00	53085.36	0.00	23481.63	0.00	830.80 (3)	–	–
10	3.87672	0.19	846.59	–	–	0.03	53778.74	0.00	1806.25 (6)	–	–
15	3.81766	0.00	1425.85	–	–	–	–	0.86	3102.71 (8)	–	–
20	3.81436	0.00	1775.03	–	–	–	–	3.38	3592.28 (8)	–	–
25	3.74937	0.00	3204.84	–	–	–	–	3.39	7588.09 (12)	–	–

MATLAB stock due to the memory problems when the data set "Gisette" was imported to it as an input data. So, we could not get any result with k-means++ for this data set.



(a) Distance function evaluations

(b) Davies-Bouldin index

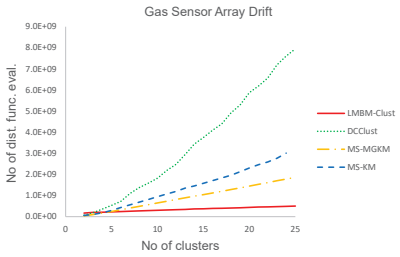


(c) Dunn index

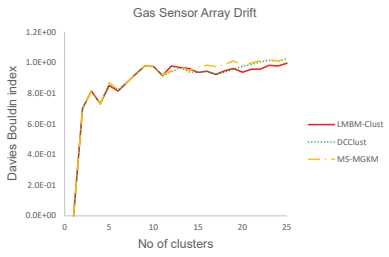
Figure 3: Gisette: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 4: Summary of the results with Gas Sensor Array Drift ($\times 10^{13}$).

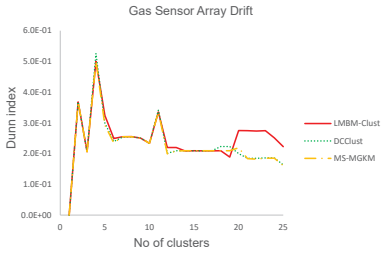
k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	7.91186	0.00	30.05	0.00	14.73	0.00	8.64	0.00	61.13 (50)	0.00	0.80
3	5.02412	0.00	35.12	0.00	39.43	0.00	23.15	0.00	70.84 (65)	0.00	0.41
5	3.22394	0.00	37.84	0.10	101.97	0.10	52.75	0.00	77.43 (81)	0.00	0.38
10	1.65524	0.00	49.34	0.00	321.07	0.00	131.95	5.36	100.37 (124)	6.63	0.71
15	1.13801	0.36	60.11	0.36	632.26	0.00	207.87	17.34	120.51 (124)	4.24	1.12
20	0.87916	2.69	69.23	0.62	973.90	0.01	284.43	30.24	139.07 (95)	14.17	1.82
25	0.72274*	2.96	78.10	0.58	1316.98	0.00	363.25	45.16	157.83 (112)	1.25	2.44



(a) Distance function evaluations



(b) Davies-Bouldin index

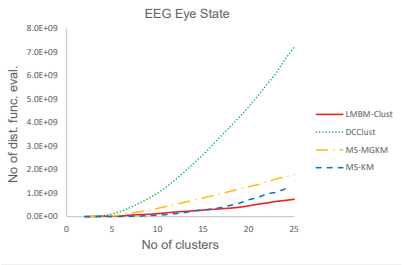


(c) Dunn index

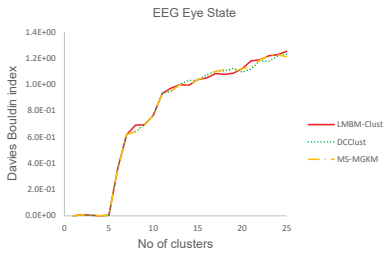
Figure 4: Gas Sensor Array Drift: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 5: Summary of the results with EEG Eye State ($\times 10^8$).

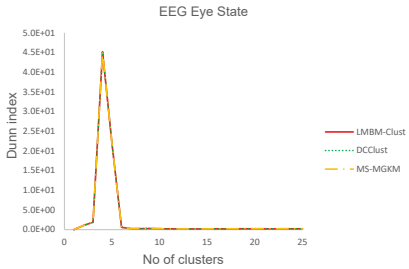
k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	7845.09934*	4.24	0.21	4.24	0.19	4.24	0.12	0.00	0.50 (1)	0.00	0.02
3	1833.88058	0.00	0.23	0.00	0.40	0.00	0.24	227.91	0.88 (1)	0.00	0.01
4	2.23605	0.00	0.34	0.00	0.69	0.00	0.35	268809.80	2.45 (1)	0.00	0.02
5	1.33858	0.00	0.39	0.00	1.45	0.00	0.72	449091.75	2.45 (1)	29.90	0.01
10	0.45310*	0.79	2.43	0.80	10.80	0.79	8.10	195.73	6.47 (4)	0.00	0.21
15	0.34653	0.05	4.82	0.26	28.42	0.05	15.60	4.78	11.66 (11)	0.63	0.31
20	0.28986*	0.00	7.73	0.96	51.15	0.00	21.49	2.83	15.91 (12)	0.71	0.52
25	0.25989*	0.16	11.82	0.00	79.77	0.07	27.19	2.12	24.29 (24)	0.14	1.20



(a) Distance function evaluations



(b) Davies-Bouldin index

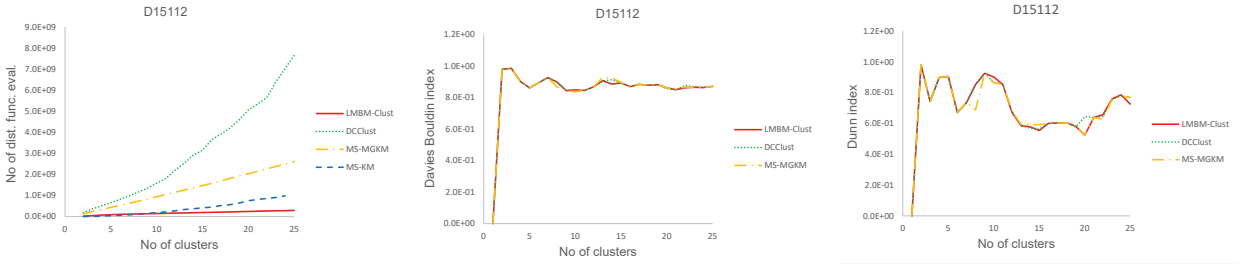


(c) Dunn index

Figure 5: EEG Eye State: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 6: Summary of the results with D15112 ($\times 10^{11}$).

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	3.68403	0.00	0.98	0.00	0.52	0.00	1.84	0.00	2.00 (3)	0.00	0.57
3	2.53240	0.00	1.37	0.00	1.10	0.00	4.10	0.00	3.13 (3)	0.00	0.13
5	1.32707	0.00	1.99	0.00	1.88	0.00	4.93	0.00	4.11 (12)	0.00	0.09
10	0.64491	1.41	2.81	0.62	4.75	0.62	6.67	0.00	6.01 (20)	0.43	0.24
15	0.43136	0.23	3.39	0.25	10.44	0.00	8.16	0.00	7.02 (18)	3.31	0.19
20	0.32177	0.24	3.99	0.00	18.47	0.25	9.88	6.55	8.16 (13)	2.18	0.88
25	0.25308*	0.48	4.60	0.00	30.72	0.01	11.53	3.81	9.77 (14)	1.56	1.73

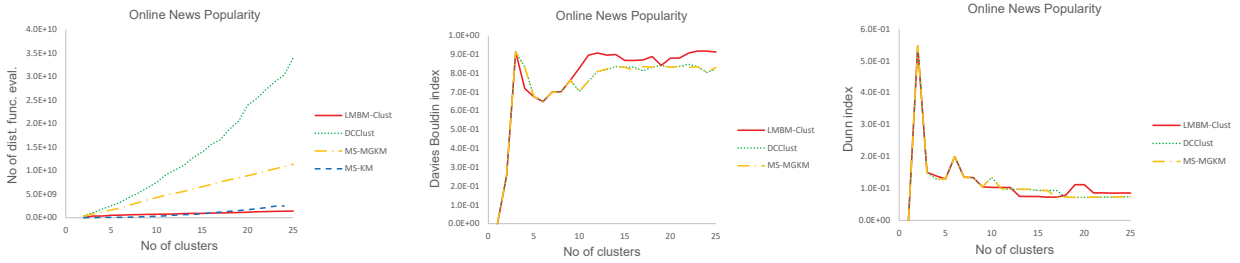


(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 6: D15112: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 7: Summary of the results with Online News Popularity ($\times 10^{14}$).

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	9.53913	0.00	2.31	0.00	40.01	0.00	33.02	0.00	5.74 (1)	0.00	0.08
3	5.91077	0.00	29.54	0.00	102.43	0.00	96.24	0.13	62.11 (5)	34.58	0.16
5	3.09885	0.00	43.06	0.00	220.98	0.00	185.42	0.35	90.12 (14)	30.99	0.47
10	1.17247	2.57	59.38	0.00	615.96	0.00	443.10	82.56	121.24 (15)	17.22	0.95
15	0.77637	14.77	72.32	0.00	1094.05	0.00	656.08	33.17	145.15 (11)	1.79	3.60
20	0.59809	7.41	89.26	0.00	1775.80	0.00	852.95	34.48	183.89 (20)	15.46	9.34
25	0.49616	7.01	105.72	0.00	2455.11	0.82	1053.71	45.11	211.52 (16)	5.62	7.36



(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 7: Online News Popularity: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 8: Summary of the results with KEGG Metabolic ($\times 10^8$).

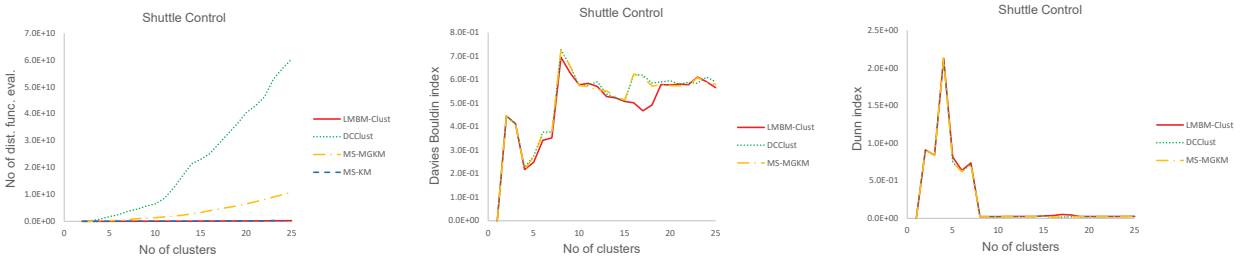
k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	11.38530	0.00	0.44	0.00	3.41	0.00	3.62	18.85	24.59 (1)	0.00	0.11
3	4.90060	0.00	0.71	0.00	9.06	0.00	15.43	124.79	39.04 (1)	0.00	0.20
5	1.88367	0.00	1.59	0.06	34.23	0.00	84.75	0.00	50.16 (1)	0.00	0.48
10	0.60513*	4.96	4.14	5.18	190.73	4.96	254.05	36.81	41.56 (1)	0.00	1.87
15	0.35393	0.70	8.83	0.25	379.39	3.54	369.39	96.64	54.89 (1)	83.70	1.79
20	0.25027	3.32	12.22	2.04	594.10	1.61	494.59	160.92	48.33 (1)	15.44	1.87
25	0.19289	1.64	17.08	0.74	828.06	0.51	636.45	221.85	49.19 (1)	5.67	3.09

(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 8: KEGG Metabolic: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 9: Summary of the results with Shuttle Control ($\times 10^8$).

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	21.34329.	0.00	0.33	0.00	0.63	0.00	1.91	51.13	35.66 (1)	5.05	0.29
3	10.85415	0.00	0.46	0.00	1.72	0.00	3.85	100.52	25.57 (1)	0.01	0.05
4	8.86910	0.00	0.59	0.77	5.69	0.00	6.27	15.24	12.28 (1)	4.08	0.32
5	7.24479	0.09	0.72	0.24	12.10	0.00	10.53	38.73	18.06 (1)	9.71	0.07
10	2.83216	0.55	1.77	0.33	43.39	0.35	31.77	145.44	23.77 (1)	55.34	0.23
15	1.53154	0.02	3.14	0.37	163.84	0.07	57.33	200.94	13.76 (1)	46.61	1.04
20	1.06012	0.03	4.91	1.16	289.39	0.00	104.28	309.78	11.51 (1)	26.54	0.97
25	0.77978*	0.00	7.65	1.09	438.74	2.48	163.07	330.52	19.62 (1)	21.37	3.32



(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 9: Shuttle Control: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 10: Summary of the results with Sensorless Drive Diagnosis ($\times 10^7$).

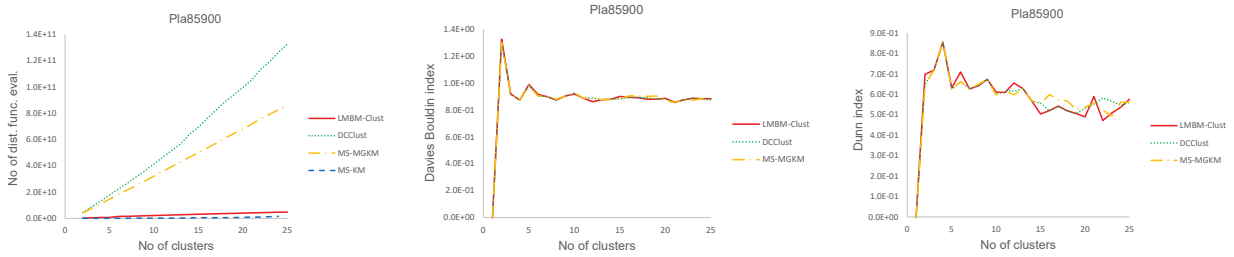
k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	3.88116	0.00	1.04	0.00	15.62	0.00	11.16	100.19	46.76 (1)	1.51	0.10
3	2.91313	0.00	1.20	4.23	118.89	0.00	37.36	155.38	43.52 (1)	12.87	0.58
5	1.93651	5.37	2.25	7.79	327.46	0.00	245.97	37.85	40.43 (1)	40.88	1.53
10	0.98472	1.89	5.81	0.00	859.33	1.89	533.82	127.20	66.97 (1)	21.19	7.36
15	0.62816	0.00	19.80	2.45	1582.09	0.00	1048.59	235.35	46.3524 (2)	28.07	6.89
20	0.49884	1.01	31.02	2.00	2509.65	0.00	1468.72	260.94	94.20 (2)	14.96	5.42
25	0.42225	0.02	44.27	1.57	3495.77	0.00	1873.696	314.82	88.74 (2)	19.48	15.13

(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 10: Sensorless Drive Diagnosis: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 11: Summary of the results with Pla85900 ($\times 10^{15}$).

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	3.74908	1.44	2.79	0.00	9.66	0.00	188.26	1.44	71.25 (1)	0.00	0.19
3	2.28057	0.00	3.87	0.00	18.52	0.00	479.33	0.00	103.35 (1)	0.00	0.71
5	1.33972	2.77	5.60	0.00	37.16	0.00	687.09	2.77	64.09 (1)	0.00	1.09
10	0.68294	0.40	13.09	0.00	87.52	0.00	1267.43	0.55	34.91 (1)	1.00	2.28
15	0.46029*	0.92	17.59	0.48	151.26	0.48	1740.19	0.17	49.26(1)	0.00	7.11
20	0.34988	0.94	22.51	0.51	225.13	0.25	2052.67	0.03	49.13 (2)	0.91	3.38
25	0.28259*	0.17	27.90	0.02	307.14	0.00	2331.13	0.32	66.93 (3)	0.17	9.85

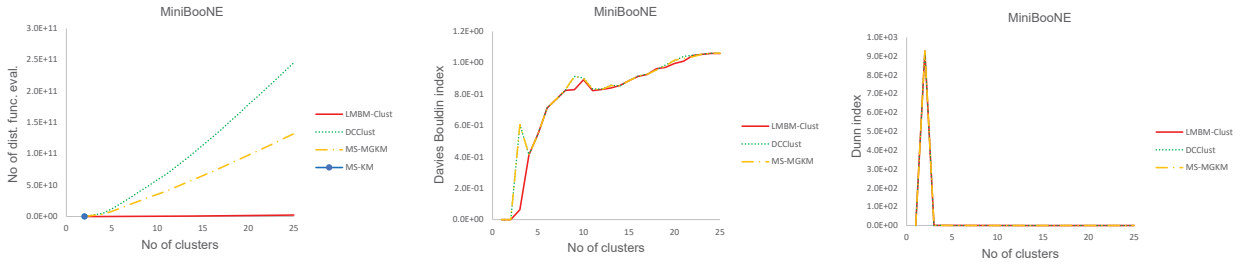


(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 11: Pla85900: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 12: Summary of the results with MiniBooNE Particle Identification ($\times 10^{10}$).

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	8.92236	0.00	2.88	0.00	7.92	0.00	16.60	286983.96	218.36 (1)	0.00	0.25
3	5.22601	21.68	3.55	0.00	177.89	0.00	309.64	–	–	0.00	1.55
5	1.82252	0.00	8.78	0.00	843.69	0.00	1053.25	–	–	0.00	3.74
10	0.90920*	1.63	33.04	1.65	3936.03	1.64	3720.18	–	–	0.00	55.81
15	0.63506	0.00	62.40	0.02	7567.37	0.00	6332.97	–	–	3.88	19.42
20	0.50863	1.17	106.35	0.02	11745.10	0.36	8823.19	–	–	3.25	25.91
25	0.44425	0.00	152.15	0.02	16042.13	0.01	11308.93	–	–	0.00	67.89



(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 12: MiniBooNE particle identification: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 13: Summary of the results with Skin Segmentation ($\times 10^9$).

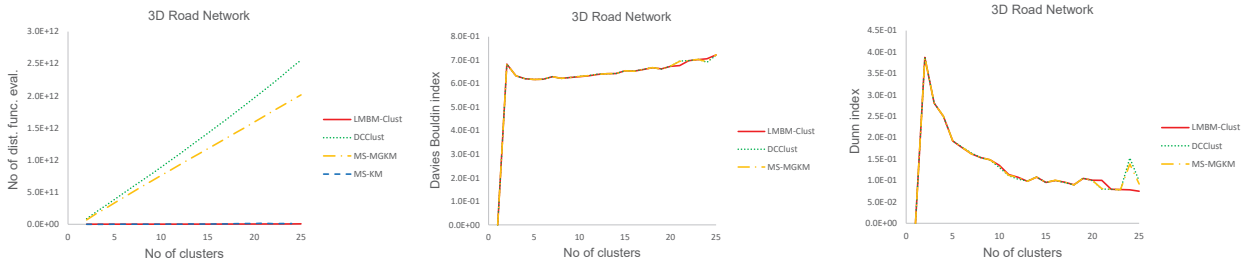
k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	1.32236	0.00	0.82	0.00	93.17	0.00	523.73	0.00	407.08 (1)	0.00	0.31
3	0.89362	0.00	1.44	0.00	168.17	0.00	853.79	0.00	359.83 (1)	0.00	0.58
5	0.50203	0.00	3.01	0.00	303.59	0.00	1228.74	3.28	191.71 (1)	1.65	0.51
10	0.25121	13.37	6.35	0.00	632.74	4.26	1750.23	9.10	123.13 (1)	17.14	0.83
15	0.16964	3.83	11.13	0.18	967.53	0.18	2342.50	14.03	277.55 (1)	6.66	2.15
20	0.12615*	4.50	15.89	1.37	1307.82	0.00	2765.46	30.88	244.93 (1)	15.15	2.96
25	0.10228*	5.78	21.52	0.69	1675.50	0.00	3151.39	44.59	189.80 (1)	8.23	3.44

(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 13: Skin Segmentation: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

Table 14: Summary of the results with 3D Road Network ($\times 10^6$).

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM		k-means++	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu (n_s)	E_A	cpu
2	49.13298	0.00	16.37	0.00	237.26	0.00	4399.05	0.00	3884.82 (1)	0.00	1.54
3	22.77818	0.00	18.10	0.00	537.69	0.00	7332.10	0.00	3019.90 (1)	0.00	1.47
5	8.82574	0.00	22.90	0.00	1066.93	0.00	14704.09	0.00	2936.95 (1)	0.00	6.27
10	2.56661*	0.00	31.98	0.16	2404.00	0.01	40040.34	0.01	7736.89 (1)	0.01	40.41
15	1.27069	0.00	42.01	0.00	3849.80	0.00	53218.65	0.00	4234.13 (1)	0.00	20.65
20	0.80865*	0.00	56.55	0.01	5400.02	0.00	62127.24	0.00	8425.20 (1)	0.00	455.83
25	0.59259	0.00	76.02	3.77	7067.94	3.76	69467.07	1.61	5437.73 (1)	1.61	103.69



(a) Distance function evaluations (b) Davies-Bouldin index (c) Dunn index

Figure 14: 3D Road Network: number of distance function evaluations, Davies-Bouldin and Dunn validity indices vs. number of clusters.

5.2. External Validation and Simulation Study

In addition to the internal validation indices DBI and DI, we study the performance of the proposed method by using some external validity indices: namely, the proportion of objects that are correctly grouped together against the true classes and the adjusted Rand index [46]. This study follows the lines given in [45].

Validation with real world data sets. We first apply `LMBM-Clust`, `DCClust`, `MS-MGKM`, and `k-means++` to three real world data sets, Iris, Soybean, and Arcane (training set only), whose true classes are known. These data sets have 150, 47, and 100 instances, 4, 35, and 10000 attributes, and 3, 4, and 2 classes, respectively, and they are available from [35]. In Iris all three true classes have 50 instances, in Soybean the first three classes have ten instances while the last class has 17 instances, and in Arcane there are 44 instances in the first class and 56 in the second.

The number of clusters in our experiments is selected to be the number of classes in the original data. We applied algorithms to data without class information. The performance of the algorithms is measured by their accuracy, which is the proportion of objects that are correctly grouped together against the true classes.

The clustering results by different algorithms are given in Tables 15-17. The clustering accuracies against the true classes by `LMBM-Clust` are 89.3 % in Iris, 89.4% in Soybean and 63.0% in Arcane. The corresponding values for the other algorithms are 88.7%, 89.4%, and 62.0% for `DCClust` and `MS-MGKM`, and 100%, 72.3%, and 62.0% for `k-means++`. The pairwise comparison of the algorithms indicates an improved accuracy by the `LMBM-Clust` method. Notice, however, that `k-means++` clusters Iris perfectly.

Table 15: Clustering results with Iris data set.

True class	LMBM-Clust			DCClust			MS-MGKM			k-means++		
	1	2	3	1	2	3	1	2	3	1	2	3
1	50	0	0	50	0	0	50	0	0	50	0	0
2	0	48	2	0	47	3	0	47	3	0	50	0
3	0	14	36	0	14	36	0	14	36	0	0	50

Table 16: Clustering results with Soybean data set.

True class	LMBM-Clust				DCClust				MS-MGKM				k-means++			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	10	0	0	0	10	0	0	0	10	0	0	0	10	0	0	0
2	0	10	0	0	0	10	0	0	0	10	0	0	0	10	0	0
3	0	0	5	5	0	0	5	5	0	0	5	5	0	0	5	5
4	0	0	0	17	0	0	0	17	0	0	0	17	0	0	8	9

Table 17: Clustering results with Arcane data set (training set only).

True class	LMBM-Clust		DCClust		MS-MGKM		k-means++	
	1	2	1	2	1	2	1	2
1	34	10	22	22	22	22	22	22
2	27	29	16	40	16	40	16	40

Simulation study with artificial data. The performance of the proposed method is further evaluated using artificial data sets. Such data sets were generated in the two dimensional space with different portion of outliers (cf. [45]). Each data set has three clusters (clusters A, B and C) with 120 instances in each cluster. The x- and y-coordinates in the cluster A are drawn independently from normal distributions $N(\mu_x^A, \sigma^A)$ and $N(\mu_y^A, \sigma^A)$ with means μ_x^A and μ_y^A and the standard deviation σ^A. Similarly, x- and y-coordinates in clusters B and C are generated from $N(\mu_x^B, \sigma^B)$, $N(\mu_y^B, \sigma^B)$, $N(\mu_x^C, \sigma^C)$, and $N(\mu_y^C, \sigma^C)$ respectively. Nevertheless, in cluster C, a specified proportion of instances (called outliers) are assumed to have a larger standard deviation σ_O^C than the standard deviation σ^C of the rest of the instances. For example, when there are 10% of outliers, the x- and y-coordinates of 12 instances in cluster C are generated from $N(\mu_x^C, \sigma_O^C)$ and $N(\mu_y^C, \sigma_O^C)$ while the rest 108 instances are generated from $N(\mu_x^C, \sigma^C)$ and $N(\mu_y^C, \sigma^C)$. The parameters used to generate the artificial data are given in Table 18. We generate data with different number of outliers and with each number of outliers we generate ten different data sets. The results given in Table 19 are averaged over these ten data sets.

Table 18: Parameters for generating instances in artificial data.

	Cluster A	Cluster B	Cluster C
Mean μ_x	0	6	6
Mean μ_y	0	-1	2
Standard deviation σ	1.5	0.5	0.5
Outliers σ_O^C	–	–	2

As already mentioned, we use the well-known *adjusted Rand index* (ARI) [46] to compare the algorithms. Suppose that $U = \{U^1, U^2, \ldots U^r\}$ and $V = \{V^1, V^2, \ldots V^s\}$ represent two different partitions of the instances such that U is the true partition and V is a clustering result. Then the ARI for the clustering result V is calculated by

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}},$$

where n is the number of instances in data, and n_{ij}, a_i, b_j are values from the following contingency table

	V^1	V^2	...	V^s	Sums
U^1	n_{11}	n_{12}	...	n	a_1
U^2	n_{21}	n_{22}	...	n	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U^r	n_{r1}	n_{r2}	...	n_{rs}	a_r
Sums	b_1	b_2	...	b_s	

Here each entry n_{ij} denotes the number of common instances in U^i and V^j. The ARI value equal to one indicates a perfect clustering.

In Table 19, we see that ARIs for different clustering algorithms are almost similar. The only remarkable difference is `k-means++` with 20% of outliers, where the ARI is the worst. This is due to the misclassification in one of the ten data sets with 20% of outliers: `k-means++` clustered all points of the original clusters B and C to one cluster while the cluster A was divided into two clusters. All the other algorithms succeeded in the clustering of this data set with only three misclassified points.

Table 19: Adjusted Rand indices by clustering algorithms in artificial data.

Outliers (%)	`LMBM-Clust`	`DCClust`	`MS-MGKM`	`k-means++`
0	1.0000	1.0000	1.0000	1.0000
10	0.9900	0.9900	0.9900	0.9900
20	0.9818	0.9818	0.9818	0.9303
30	0.9785	0.9785	0.9785	0.9785
40	0.9666	0.9666	0.9658	0.9674
50	0.9551	0.9559	0.9512	0.9559

6. Conclusions

In this paper, a new LMBM-Clust method for solving the minimum sum-of-squares clustering problems is introduced. The LMBM-Clust method consists of two different algorithms: an incremental algorithm is used to solve clustering problems globally and at each iteration of this algorithm the Limited Memory

Bundle Method (LMBM) is used to solve both the clustering and the auxiliary clustering problems. It is shown that the LMBM-Clust method converges to a stationary point of the clustering problem.

The proposed LMBM-Clust method was tested using real world data sets with the number of data points ranging from tens of thousands to hundreds of thousands. Using different metrics, such as the sum-of-squares errors and two cluster validity indices, and also two parameters, such as the CPU time and the number of distance function evaluations, we demonstrated the superiority of the proposed method over other nonsmooth optimization based incremental algorithms: in the largest data set 3D Road Network the speed up was almost thousand fold.

In addition, the LMBM-Clust method is far more accurate than the well-known and widely used k-means++ algorithm (its MATLAB implementation). In some large data sets the LMBM-Clust method is more efficient than k-means++. Our results clearly demonstrate that the multi-start k-means cannot be considered as an alternative to any above mentioned clustering algorithms in very large data sets.

Results presented in this paper demonstrate that the LMBM-Clust method is both efficient and accurate in very large data sets. Furthermore, it can provide a real time clustering in such data sets.

Acknowledgement The authors would like to thank two anonymous referees for their valuable comments. The work was financially supported by the Academy of Finland (Project No. 289500) and Australian Research Counsil's Discovery Projects funding scheme (Project No. DP140103213). The paper was written while the corresponding author was visiting Faculty of Science and Technology, Federation University Australia.

References

[1] A. Bagirov, N. Karmitsa, M. M. Mäkelä, Introduction to Nonsmooth Optimization: Theory, Practice and Software, Springer, 2014.

[2] A. Bagirov, E. Mohebi, Nonsmooth optimization based algorithms in cluster analysis, in:

E. Celebi (Ed.), Partitional Clustering Algorithms, Springer International Publishing, 2015, pp. 99–146.

[3] A. Bagirov, J. Ugon, An algorithm for minimizing clustering functions, Optimization 54 (4–5) (2005) 351–368.

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

[5] N. Karmitsa, A. Bagirov, S. Taheri, New diagonal bundle method for clustering problems in large data sets, European Journal of Operational Research, 263 (2) (2017) 367–379.

[6] L. An, M. Belghiti, P. Tao, A new efficient algorithm based on DC programming and DCA for clustering, Journal of Global Optimization 37 (4) (2007) 593–608.

[7] L. An, L. Minh, P. Tao, New and efficient DCA based algorithms for minimum sum-of-squares clustering, Pattern Recognition 47 (2014) 388–401.

[8] L. An, P. 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 (2009) 327–340.

[9] A. Bagirov, S. Taheri, J. Ugon, Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems, Pattern Recognition 53 (2016) 12–24.

[10] W. Khalaf, A. Astorino, P. D'Alessandro, M. Gaudioso, A DC optimization-based clustering technique for edge detection, Optimization Letters (2016) 1–14doi:10.1007/s11590-016-1031-7. URL http://dx.doi.org/10.1007/s11590-016-1031-7

[11] Y. Xia, A global optimization method for semi-supervised clustering, Data Mining and Knowledge Discovery 18 (2009) 214–256.

[12] A. Bagirov, B. Ordin, G. Ozturk, A. Xavier, An incremental clustering algorithm based on hyperbolic smoothing, Computational Optimization and Applications 61 (1) (2015) 219–241.

[13] A. Xavier, The hyperbolic smoothing clustering method, Pattern Recognition 43 (2010) 731–737.

[14] A. Xavier, V. Xavier, Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions, Pattern Recognition 44 (1) (2011) 70–77.

[15] P. Hansen, N. Mladenovic, Variable neighborhood decomposition search, Journal of Heuristic 7 (2001) 335–350.

[16] K. S. Xu, M. Kliger, A. O. Hero III, Adaptive evolutionary clustering, Data Mining and Knowledge Discovery 28 (2014) 304–336.

[17] S. Selim, K. Al-Sultan, A simulated annealing algorithm for the clustering, Pattern Recognition 24 (10) (1991) 1003–1008.

[18] K. Al-Sultan, A tabu search approach to the clustering problem, Pattern Recognition 28 (9) (1995) 1443–1451.

[19] M. A. Rahman, M. Z. Islam, A hybrid clustering technique combining a novel genetic algorithm with k-means, Knowledge-Based Systems 71 (2014) 345–365.

[20] A. Bagirov, Modified global k-means algorithm for sum-of-squares clustering problems, Pattern Recognition 41 (10) (2008) 3192–3199.

[21] A. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (8) (2010) 651–666.

[22] Z. Volkovich, D. Toledano-Kitai, G.-W. Weber, Self-learning k-means clustering: a global optimization approach, Journal of Global Optimization 56 (2) (2013) 219–232.

[23] B. Ordin, A. Bagirov, A heuristic algorithm for solving the minimum sum-of-squares clustering problems, Journal of Global Optimization 61 (2) (2015) 341–361.

[24] M. Haarala, Large-scale nonsmooth optimization: Variable metric bundle method with limited memory, Ph.D. thesis, University of Jyväskylä, Department of Mathematical Information Technology (2004).

[25] M. Haarala, K. Miettinen, M. M. Mäkelä, New limited memory bundle method for large-scale nonsmooth optimization, Optimization Methods and Software 19 (6) (2004) 673–692.

[26] N. Haarala, K. Miettinen, M. M. Mäkelä, Globally convergent limited memory bundle method for large-scale nonsmooth optimization, Mathematical Programming 109 (1) (2007) 181–205.

[27] N. Karmitsa, A. Bagirov, M. M. Mäkelä, Comparing different nonsmooth optimization methods and software, Optimization Methods and Software 27 (1) (2012) 131–153.

[28] F. H. Clarke, Optimization and Nonsmooth Analysis, Wiley-Interscience, New York, 1983.

[29] A. Bagirov, A. Rubinov, N. Soukhoroukova, J. Yearwood, Unsupervised and supervised data classification via nonsmooth and global optimization, Top 11 (2003) 1–93.

[30] R. H. Byrd, J. Nocedal, R. B. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, Mathematical Programming 63 (1994) 129–156.

[31] A. Bihain, Optimization of upper semidifferentiable functions, Journal of Optimization Theory and Applications 4 (1984) 545–568.

[32] J. B. MacQueen, Some methods for classification and analysis of multivariate observations, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297, University of California Press (1967).

[33] A. David, S. Vassilvitskii, K-means++: The advantages of careful seeding, in: SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1027–1035.

[34] N. Karmitsa, A. M. Bagirov, S. Taheri, MSSC clustering of large data using the limited memory bundle method, TUCS Technical Report, No. 1164, Turku Centre for Computer Science, Turku, the report is available online at http://tucs.fi/publications/view/?pub_id=tKaBaTa16b. (2016).

[35] M. Lichman, UCI machine learning repository, Available in web page <URL: http://archive.ics.uci.edu/ml>, University of California, Irvine, School of Information and Computer Sciences, (April 8th, 2016) (2001).

[36] I. Guyon, S. R. Gunn, A. Ben-Hur, G. Dror, Result analysis of the nips 2003 feature selection challenge., In: NIPS, data set available in UCI machine learning repository <URL: http://archive.ics.uci.edu/ml> (June 11th, 2016) (2004).

[37] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, R. Huerta, Chemical gas sensor drift compensation using classifier ensembles, Sensors and Actuators B: Chemical (2012) DOI: 10.1016/j.snb.2012.01.074., data set available in UCI machine learning repository <URL: http://archive.ics.uci.edu/ml> (April 8th, 2016) (2012).

[38] B. Bixby, G. Reinel, Tsplib — a library of travelling salesman and related problem instance, available in web page <URL: http://softlib.rice.edu/tsplib.html> (April 8th, 2016) (1995).

[39] K. Fernandes, P. Vinagre, P. Cortez, A proactive intelligent decision support system for predicting the popularity of online news, Proceedings of the 17th EPIA 2015 — Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal., data set available in UCI machine learning repository <URL: http://archive.ics.uci.edu/ml> (June 11th, 2016) (2015).

[40] M. Naeem, A. Sohail, Kegg metabolic dataset, Data set available in UCI machine learning repository <URL: http://archive.ics.uci.edu/ml>, centre of Research in Data Engineering Islamabad Pakistan, *naeems.naeem@gmail.com*, *sohail.asg@gmail.com* (April 8th, 2016).

[41] R. Bhatt, A. Dhall, Skin segmentation dataset, Data set available in UCI machine learning repository <URL: http://archive.ics.uci.edu/ml>, (April 8th, 2016) (2011).

[42] M. Kaul, B. Yang, C. S. Jensen, Building accurate 3d spatial networks to enable next generation intelligent transportation systems, Proceedings of International Conference on Mobile Data Management (IEEE MDM), June 3-6 2013, Milan, Italy, data set available in UCI machine learning repository <URL: http://archive.ics.uci.edu/ml> (April 8th, 2016) (2013).

[43] J. C. Dunn, Well-separated clusters and optimal fuzzy partitions, Journal of Cybernetics 4 (1) (1974) 95–104.

[44] D. L. Davies, D. W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1 (2) (1979) 224–227.

[45] H.-S. Park, C.-H. Jun, A simple and fast algorithm for K-medoids clustering, Expert Systems with Applications 36 (2009) 3336–3341.

[46] L. Hubert, P. Arabie, Comparing partitions. Journal of Classification, 2 (1985) 193–218.