Napsu Karmitsa | Adil Bagirov | Sona Taheri

MSSC Clustering of Large Data using the Limited Memory Bundle Method

Turku Centre *for* Computer Science

TUCS Technical Report
No 1164, July 2016

MSSC Clustering of Large Data using the Limited Memory Bundle Method

Napsu Karmitsa
Department of Mathematics and Statistics
University of Turku
FI-20014 Turku, Finland
napsu@karmitsa.fi

Adil Bagirov
Faculty of Science and Technology,
Federation University Australia, University Drive, Mount Helen,
PO Box 663, Ballarat, VIC 3353, Australia
a.bagirov@federation.edu.au

Sona Taheri
Faculty of Science and Technology,
Federation University Australia, University Drive, Mount Helen,
PO Box 663, Ballarat, VIC 3353, Australia
s.taheri@federation.edu.au

TUCS Technical Report
No 1164, July 2016

Abstract

Clustering is among most important tasks in data mining. This problem in large data sets is challenging for most existing clustering algorithms. It is important to develop clustering algorithms which are accurate and can provide real time clustering in such data sets. This paper introduces one such algorithm. The clustering problem is formulated as a nonsmooth optimization problem with minimum sum-of-squares distance function. Then the limited memory bundle method [Haarala et.al. *Math. Prog.*, Vol. 109, No. 1, pp. 181–205, 2007] is modified and combined with incremental approach to solve this problem. The method is evaluated using real world data sets with both the large number of attributes and large number of data points. The new software is also compared with some other optimization based clustering softwares.

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

TUCS Laboratory
Turku Optimization Group (TOpGroup)

1 Introduction

Clustering is dealing with the problems of organization of a collection of patterns into clusters based on similarity. It has many applications in medicine, engineering and business. The similarity measure in the clustering can be defined using various distance-like functions. Clustering problems with the similarity measure defined by the squared Euclidean norm are called the minimum sum-of-squares clustering (MSSC) problem. These problems have been studied extensively, and there are various optimization algorithms for solving them (see, e.g. [8, 11, 38], for brief review of some of these algorithms). In papers [6, 11, 10], nonsmooth optimization (NSO, not necessarily differentiable optimization) algorithms were developed. Algorithms based on hyperbolic smoothing technique are presented in [7, 41, 42]. Various optimization techniques such as branch and bound [18], interior point methods [19], the variable neighborhood search algorithm [25] and metaheuristics such as the simulated annealing [37], tabu search [1], and genetic algorithms [36] have also been introduced. In addition, algorithms based on difference of convex (DC) representation of the MSSC problem are introduced in [4, 2, 3, 9].

Most of the algorithms mentioned above are not applicable or have limited capabilities for solving clustering problems in large data sets. They become time consuming as the size of data sets increase. In this paper, the phrase "large data set" means that the data set contains hundred of thousands or millions of data points (features) and hundreds or thousands of attributes. However, we assume that such data sets still can be stored in the memory of a computer and might be read many times. The aim of this paper is to develop new clustering software which is accurate and efficient in large data sets.

In this paper, we introduce the new LMBM-CLUST -method for solving MSSC problems. The LMBM-CLUST -method consist two algorithms: an incremental algorithm is used to solve clustering problems globally and at each iteration of this algorithm the limited memory bundle algorithm (LMBM) by Karmitsa (née Haarala) et al. [22, 23, 24, 27] is used to solve both the clustering problem and the so-called auxiliary clustering problem with different starting points provided by the incremental algorithm.

The LMBM is a hybrid of the variable metric bundle methods (VMBM) [31, 40] and the limited memory variable metric methods (see e.g. [16]), where the first ones have been developed for small- and medium-scale NSO and the latter ones for smooth large-scale optimization. Here, the original LMBM algorithm is slightly modified to be better suited for solving clustering problems. The LMBM combines the ideas of the VMBM with the search direction calculation of limited memory approach. Therefore, the time-consuming quadratic direction finding problem appearing in the standard bundle methods (see eq. [26, 29, 33]) does not need to be solved, nor the number of stored subgradients needs to grow with the dimension of the problem. Furthermore, the method uses only a few vectors to represent the variable metric approximation of the Hessian matrix and, thus, it avoids storing and manipulating large matrices as is

the case in the VMBM. These improvements make the LMBM suitable for large-scale optimization. Namely, the number of operations needed for the calculation of the search direction is only linearly dependent on the number of variables while, for example, with the VMBM this dependence is quadratic.

This paper is organized as follows. In Section 2 we introduce our notation and recall some basic definitions and results from nonsmooth analysis. The formulation of 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 recall the ideas of incremental approach used to solve globally the clustering problem. The results of the numerical experiments are presented and discussed in Section 5, and finally, Section 6 concludes the paper.

2 Notations and Background

In this section, we give our notations as well as some basic definitions and results from nonsmooth analysis.

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 a *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* $\partial f(\boldsymbol{x})$ [17] of a LLC function $f : \mathbb{R}^n \to \mathbb{R}$ at any point $\boldsymbol{x} \in \mathbb{R}^n$ is given by

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

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 $\boldsymbol{0} \in \partial f(\boldsymbol{x}^*)$. Stationarity is a necessary condition for local optimality and, in the convex case, it is also sufficient for global optimality. An optimization method is said to be *globally convergent* if starting from any arbitrary point $\boldsymbol{x}_1$ it generates a sequence $\{\boldsymbol{x}_k\}$ that converges to a stationary point $\boldsymbol{x}^*$, that is, $\{\boldsymbol{x}_k\} \to \boldsymbol{x}^*$ whenever $k \to \infty$.

3 Nonsmooth Formulations of Clustering Problems

In this section we present NSO formulations of the clustering and the so-called auxiliary clustering problems, but first, we recall the fundamentals of clustering.

Cluster Analysis. In 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 the 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 $j, l = 1, \ldots, k$, $j \neq l$.
3. $A = \bigcup_{j=1}^{k} A^j$.

The sets $A^j, j = 1, \ldots, k$ are called *clusters*. Data points in the same cluster are similar and data points in different clusters are dissimilar to each other. 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*.

In order to formulate the clustering problem we first need to define the *similarity* (or dissimilarity) *measure*. In this paper, the similarity measure is defined using the squared Euclidean norm (the L_2 norm)

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

As already said in introduction clustering problems with the similarity measure defined by the squared Euclidean norm are called the minimum sum-of-squares clustering (MSSC) problem. Note that the similarity measure in nonsmooth clustering problem could also be defined using other norms. That is, for instance, L_1- or L_∞-norm (see e.g. [12]).

Clustering Problem. The *NSO formulation of the MSSC problem* [8, 11] is given by

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

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*. If $k = 1$ this function is convex and, for the similarity measure using the L_2-norm, also smooth (continuously differentiable). However, for $k > 1$ it is both nonconvex and nonsmooth due to the minimum

operation used. If the similarity measure is defined using L_1- or L_∞-norms then the function f_k is nonsmooth for all $k \geq 1$. This is due to the minimum operation and the fact that both L_1- and L_∞-norms are nonsmooth functions.

In many real-world databases, the number of instances m is substantially greater than the number of attributes n. One advantage in the NSO formulation of 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 objective function (2) in Problem (1) is nonconvex and the problem itself is a global optimization problem. That is, the objective function f_k in this problem has many local minimizers and only its global minimizer provides the best cluster structure with the least number of clusters. In general, conventional global optimization methods cannot be applied to solve the clustering problem in large data sets because the size of the problem becomes large as the number of clusters increases or the number of attributes is large. For solving such problems these techniques are extremely time consuming. Therefore, in such data sets heuristics and deterministic local search algorithms are the only choice. However, the success of these algorithms strongly depends on the choice of starting cluster centers and in this case the development of efficient procedures for generating such centers is crucial.

Here we apply an approach introduced in [35] to generate starting cluster centers. This approach involves the solution of the so-called auxiliary clustering problem. Next we present this problem.

Assume that the solution $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{k-1},\ k \geq 2$ to the $(k-1)$-clustering problem is known. Denote by

$$r_{k-1}^{\boldsymbol{a}} = \min\left\{d_2(\boldsymbol{x}^1, \boldsymbol{a}), \ldots, d_2(\boldsymbol{x}^{k-1}, \boldsymbol{a})\right\} \tag{3}$$

the distance between the data point $\boldsymbol{a} \in A$ and the closest cluster center among $k-1$ centers $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{k-1}$. The *$k$-th auxiliary cluster function* is defined as [5]

$$\bar{f}_k(\boldsymbol{y}) = \frac{1}{m}\sum_{\boldsymbol{a}\in A} \min\left\{r_{k-1}^{\boldsymbol{a}}, d_2(\boldsymbol{y}, \boldsymbol{a})\right\},\ \boldsymbol{y} \in \mathbb{R}^n. \tag{4}$$

This function is nonsmooth LLC and as a sum of minima of convex functions it is, in general, nonconvex. In addition, it is clear that

$$\bar{f}_k(\boldsymbol{y}) = f_k(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{k-1}, \boldsymbol{y}),\ \text{for all } \boldsymbol{y} \in \mathbb{R}^n.$$

The *k-th auxiliary clustering problem* [5] is formulated as

$$\begin{cases} \text{minimize} & \bar{f}_k(\boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} \in \mathbb{R}^n. \end{cases} \tag{5}$$

4 LMBM-CLUST -Method

In this section we introduce the new LMBM-CLUST -method for solving MSSC problems. As already said in the introduction, the LMBM-CLUST -method consist two algorithms: an incremental algorithm is used to solve clustering problems globally and at each iteration of this algorithm the LMBM is used to solve both the clustering problem (1) and the auxiliary clustering problem (5). Figure 1 illustrates the structure of this combination and basic ideas of the two algorithms. The more detailed description of algorithms will follow.

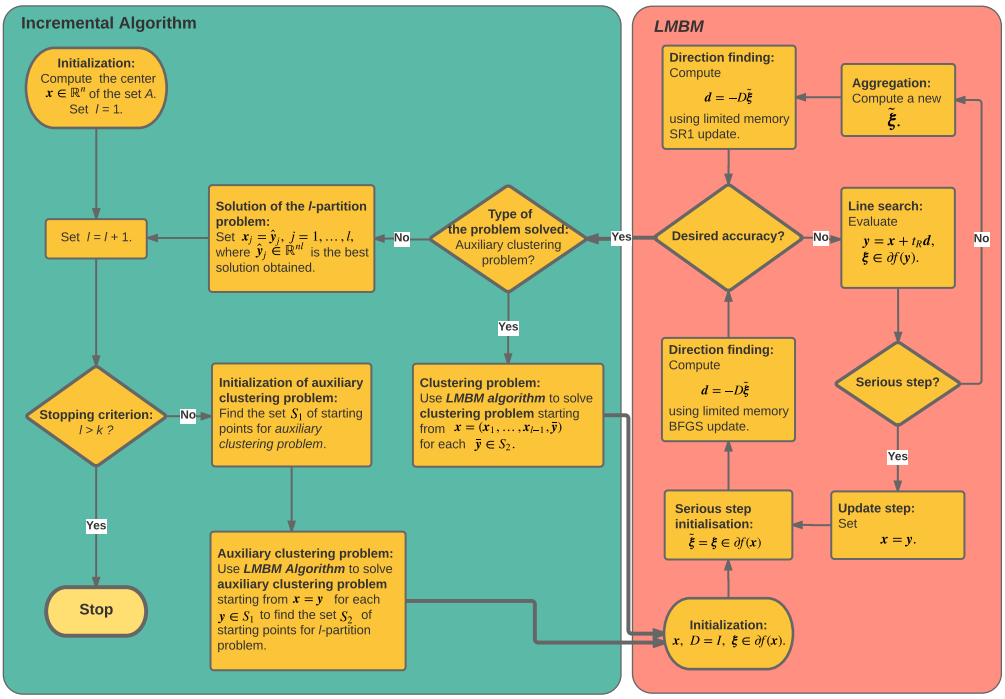


Figure 1: LMBM-CLUST -method.

4.1 LMBM

The LMBM is originally developed for solving general nonconvex nonsmooth optimization problems. Here, the original algorithm is slightly modified to be better suited for solving clustering problems. To use LMBM it is assumed that the objective function 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 this assumption is easily satisfied. In this section our notation differs a little bit from that before: n is used as a size of the optimization problem, that is, $n = n$ for auxiliary clustering problem and $n = nl$ for clustering problem, where l is the current number of clusters. In addition, k is now used as an iteration counter.

Direction Finding, Serious and Null Steps. In LMBM the search direction $\boldsymbol{d}_k$ is calculated using the classical limited memory variable metric scheme. Although, due to fact that the gradient does not need to exist for nonsmooth objective, the search direction is computed using (an aggregate) subgradient $\tilde{\boldsymbol{\xi}}_k$ instead of usual gradient. That is,

$$\boldsymbol{d}_k = -D^k \tilde{\boldsymbol{\xi}}_k,$$

where D^k is the limited memory variable metric update that, in the smooth case, would represent the approximation of the inverse of the Hessian matrix. The matrix D^k is not formed explicitly but the search direction is calculated using the limited memory approach (to be described later). The role of matrix D^k is to accumulate information about previous subgradients.

In NSO the search direction computed using a subgradient is not necessarily a descent one. Using so-called null steps gives more information about the nonsmooth objective if the current search direction is not "good enough". In order to determine a new step into the search direction $\boldsymbol{d}_k$, we use the *non-monotone two steps line search procedure*: a new iteration point $\boldsymbol{x}_{k+1}$ and a new auxiliary point $\boldsymbol{y}_{k+1}$ are produced such that

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k \qquad \text{and} \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k, \qquad \text{for } k \geq 1 \end{aligned}$$

with $\boldsymbol{y}_1 = \boldsymbol{x}_1$, where $t_R^k \in (0, t_{max}]$ and $t_L^k \in [0, t_R^k]$ are step sizes, and $t_{max} > 1$ is the upper bound for the step size.

A necessary condition for a *serious step* to be taken is to have

$$t_R^k = t_L^k > 0 \qquad \text{and} \qquad f(\boldsymbol{y}_{k+1}) \leq \max_{i \in \hat{\mathcal{K}}} f(\boldsymbol{x}_i) - \varepsilon_L^k t_R^k w_k, \tag{6}$$

where $\hat{\mathcal{K}} \subset \mathcal{K} = \{l \mid \boldsymbol{x}_l = \boldsymbol{y}_l\}$ such that $\hat{\mathcal{K}}$ includes (at most) ten last indices of $\mathcal{K}$, $\varepsilon_L^k \in (0, 1/2)$ is a line search parameter, and $w_k > 0$ represents the desirable amount of descent of f at $\boldsymbol{x}_k$. If the condition (6) is satisfied, we set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ and a serious step is taken.

If condition (6) is not satisfied, a *null step* occurs. In null steps, we have

$$t_R^k > t_L^k = 0 \qquad \text{and} \qquad -\beta_{k+1} + \boldsymbol{d}_k^T \boldsymbol{\xi}_{k+1} \geq -\varepsilon_R^k w_k,$$

where $\varepsilon_R^k \in (\varepsilon_L^k, 1/2)$ is a line search parameter, $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$, and β_{k+1} is the subgradient locality measure

$$\beta_{k+1} = \max\{|f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + (\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)^T \boldsymbol{\xi}_{k+1})|,\, \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|^2 \}.$$

Here $\gamma > 0$ is a distance measure parameter supplied by the user. In the case of a null step, we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ but information about the objective function is increased

because we store the auxiliary point $\boldsymbol{y}_{k+1}$ and the corresponding auxiliary subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ and we use them to compute new aggregate values and the limited memory update matrix.

The nonmonotone serious step condition (6) differs from that used with the original LMBM [24], where only the latest value $f(\boldsymbol{x}_k)$ of the objective function is used. Under some semismoothness assumptions the line search procedure used with the original LMBM is guaranteed to find the step sizes t_L^k and t_R^k such that exactly one of the two possibilities — a serious step or a null step — occurs [40]. The usage of non-monotone approach does not alter this result.

Aggregation. The *aggregation procedure* used in the LMBM utilizes three subgradients and two locality measures. The procedure is carried out by determining multipliers λ_i^k satisfying $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^{3} \lambda_i^k = 1$ that minimize the function

$$\begin{aligned}\varphi(\lambda_1, \lambda_2, \lambda_3) = [\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k\,]^T D^k [\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k\,] \\ + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k).\end{aligned}$$

Here $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$ is the current subgradient (m denotes the index of the iteration after the latest serious step, i.e. $\boldsymbol{x}_k = \boldsymbol{x}_m$), $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ is the auxiliary subgradient, and $\tilde{\boldsymbol{\xi}}_k$ is the current aggregate subgradient from the previous iteration ($\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$). In addition, β_{k+1} is the current subgradient locality measure and $\tilde{\beta}_k$ is the current aggregate subgradient locality measure ($\tilde{\beta}_1 = 0$). The optimal values λ_i^k, $i \in \{1, 2, 3\}$ can be calculated by using simple formulae (see [40]).

The resulting aggregate subgradient $\tilde{\boldsymbol{\xi}}_{k+1}$ and aggregate subgradient locality measure $\tilde{\beta}_{k+1}$ are computed by the formulae

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \qquad \text{and} \qquad \tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k.$$

The aggregation procedure gives us a possibility to retain the global convergence without solving the quite complicated quadratic direction finding problem (see e.g. [12]) appearing in standard bundle methods. Moreover, only one trial point $\boldsymbol{y}_{k+1}$ and the corresponding subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ need to be stored instead of $n + 3$ subgradients typically stored in standard bundle methods.. Finally, it is worth of noting that the aggregate values need to be computed only if the last step was a null step. Otherwise, we just set $\tilde{\boldsymbol{\xi}}_{k+1} = \boldsymbol{\xi}_{k+1}$ and $\tilde{\beta}_{k+1} = 0$.

Matrix Updating. The LMBM uses at most $\hat{m}_c$ correction vectors to compute updates for matrix D^k. These correction vectors are slightly modified from those in classical limited memory variable metric methods for smooth optimization (see, e.g. [16]). That is, the correction vectors are given by $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$. Note that, due to usage of null steps we may have $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and thus, we use here the auxiliary point $\boldsymbol{y}_{k+1}$ instead of $\boldsymbol{x}_{k+1}$. In addition, since the gradient does not need to exist for nonsmooth objective, the correction vectors $\boldsymbol{u}_k$ are computed using subgradients.

Let us denote by $\hat{m}_c$ the user-specified maximum number of stored correction vectors ($3 \leq \hat{m}_c$) and by $\hat{m}_k = \min\{k-1, \hat{m}_c\}$ the current number of stored correction vectors. Then the $n \times \hat{m}_k$ dimensional correction matrices S_k and U_k are defined by

$$S_k = \begin{bmatrix} \boldsymbol{s}_{k-\hat{m}_k} & \dots & \boldsymbol{s}_{k-1} \end{bmatrix} \qquad \text{and}$$
$$U_k = \begin{bmatrix} \boldsymbol{u}_{k-\hat{m}_k} & \dots & \boldsymbol{u}_{k-1} \end{bmatrix}.$$

In the LMBM both the limited memory BFGS (L-BFGS) and the limited memory SR1 (L-SR1) update formulae [16] are used in calculations of the search direction and the aggregate values. In the case of a null step, the LMBM uses the L-SR1 update formula, since this formula allows to preserve the boundedness and some other properties of generated matrices which guarantee the global convergence of the method. The inverse L-SR1 update is defined by

$$D^k = \vartheta_k I - (\vartheta_k U_k - S_k)(\vartheta_k U_k^T U_k - R_k - R_k^T + C_k)^{-1}(\vartheta_k U_k - S_k)^T,$$

where R_k is an upper triangular matrix of order $\hat{m}_k$ given by the form

$$(R_k)_{ij} = \begin{cases} (\boldsymbol{s}_{k-\hat{m}_k-1+i})^T(\boldsymbol{u}_{k-\hat{m}_k-1+j}), & \text{if } i \leq j \\ 0, & \text{otherwise,} \end{cases}$$

C_k is a diagonal matrix of order $\hat{m}_k$ such that

$$C_k = \operatorname{diag}[\boldsymbol{s}_{k-\hat{m}_k}^T \boldsymbol{u}_{k-\hat{m}_k}, \dots, \boldsymbol{s}_{k-1}^T \boldsymbol{u}_{k-1}],$$

and ϑ_k is a positive scaling parameter.

Otherwise, since these additional properties are not required after a serious step, the more efficient L-BFGS update is employed. The inverse L-BFGS update is defined by the formula

$$D^k = \vartheta_k I + \begin{bmatrix} S_k & \vartheta_k U_k \end{bmatrix} \begin{bmatrix} (R_k^{-1})^T(C_k + \vartheta_k U_k^T U_k)R_k^{-1} & -(R_k^{-1})^T \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^T \\ \vartheta_k U_k^T \end{bmatrix}.$$

In the LMBM, the individual updates that would violate positive definiteness are skipped (for more details, see [22, 23, 24]).

Stopping Criterion. For smooth functions, a necessary condition for a local minimum is that the gradient has to be zero. By continuity a norm of the gradient becomes small when we are close to an optimal point providing a good stopping criterion for algorithms. This is no longer true when we replace the gradient with an arbitrary subgradient. In LMBM the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k$ provides better approximation to the gradient but the direct test $\|\tilde{\boldsymbol{\xi}}_k\| < \varepsilon$, for some $\varepsilon > 0$, is still too uncertain as a stopping criterion. Therefore, we use the term $\tilde{\boldsymbol{\xi}}_k^T D^k \tilde{\boldsymbol{\xi}}_k = -\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d}_k$ and the aggregate subgradient locality measure $\tilde{\beta}_k$ to improve the accuracy of the sole norm $\|\tilde{\boldsymbol{\xi}}_k\|$. Hence, the stopping parameter w_k at iteration k is defined by

$$w_k = -\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d}_k + 2\tilde{\beta}_k$$

and the algorithm stops if $w_k \leq \varepsilon$ for some user specified tolerance $\varepsilon > 0$. In LMBM-CLUST we set the tolerance ε rather loose for auxiliary clustering problems, since these problems need not to be solved very accurately, and more strict to real clustering problems. The parameter w_k is also used during the line search procedure to represent the desirable amount of descent.

Algorithm. Now we give the pseudo-code of LMBM for solving the clustering problem and auxiliary clustering problem. Later, we will give an incremental algorithm to find suitable starting points for this algorithm.

```
PROGRAM LMBM
  INITIALIZE x_1 ∈ R^n, ξ_1 ∈ ∂f(x_1), m̂_c ≥ 3, and ε > 0;
  Set k = 1, m = 1, d_1 = −ξ_1, ξ̃_1 = ξ_1, and β̃_1 = 0;
  WHILE the termination condition w_k ≤ ε is not met
    Find step sizes t_L^k and t_R^k, and the subgradient locality
      measure β_{k+1};
    Set x_{k+1} = x_k + t_L^k d_k and y_{k+1} = x_k + t_R^k d_k;
    Evaluate f(x_{k+1}) and ξ_{k+1} ∈ ∂f(y_{k+1});
    Store the new correction vectors s_k = y_{k+1} − x_k and
      u_k = ξ_{k+1} − ξ_m;
    Set m̂_k = min{k, m̂_c};
    IF t_L^k > 0 THEN
      SERIOUS STEP
        Compute the search direction d_{k+1} using ξ_{k+1} and L-BFGS
          update with m̂_k most recent correction pairs;
        Set m = k + 1 and β̃_{k+1} = 0;
      END SERIOUS STEP
    ELSE
      NULL STEP
        Compute the aggregate values
          ξ̃_{k+1} = λ_1^k ξ_m + λ_2^k ξ_{k+1} + λ_3^k ξ̃_k   and
          β̃_{k+1} = λ_2^k β_{k+1} + λ_3^k β̃_k;
        Compute the search direction d_{k+1} using ξ̃_{k+1} and L-SR1
          update with m̂_k most recent correction pairs;
      END NULL STEP
    END IF
    Set k = k + 1;
  END WHILE
  RETURN final solution x_k;
END PROGRAM LMBM
```

Global Convergence. We now recall the convergence properties of the LMBM algorithm. But first, we give the assumptions needed.

ASSUMPTION 4.1. The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is LLC.

ASSUMPTION 4.2. The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is upper semismooth (see e.g. [14]).

ASSUMPTION 4.3. The level set $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ is bounded for every starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$.

Note that both for clustering and auxiliary clustering problems these assumptions are trivially satisfied.

LEMMA 4.1. *Each execution of the line search procedure is finite.*

Proof. See the proof of Lemma 2.2 in [40]. The result remains even if we use the non-monotone line search. □

THEOREM 4.2. *If the* LMBM *algorithm 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.*

Proof. See the proof of Theorem 4 in [24]. □

THEOREM 4.3. *Every accumulation point $\bar{\boldsymbol{x}}$ generated by the* LMBM *algorithm is a stationary point of the (auxiliary) clustering problem.*

Proof. See the proof of Theorem 9 in [24]. □

REMARK 4.1. The LMBM algorithm terminates in a finite number of steps If we choose $\varepsilon > 0$.

4.2 Incremental algorithm

Now we present the incremental algorithm for solving the clustering problem (1). Problem (1) is a global optimization problem and it is important to use many starting points when applying a local method like LMBM for its solution. Computation of clusters incrementally allows us to design different algorithms for generating starting cluster centers. One such algorithm is introduced in [35] and it is used in our incremental algorithm given below. The LMBM is applied to solve both the clustering and the auxiliary clustering problems at each iteration of the incremental algorithm. Together these two algorithms are a called the LMBM-CLUST -method.

```
PROGRAM Incremental Algorithm
  INITIALIZE the maximum number of clusters k ≥ 1;
  Compute the center x_1 ∈ R^n of the set A;
  Set l = 1;
  WHILE l < k
    Set l = l + 1;
    Apply the procedure from [35] to find the set S_1 ⊂ R^n of
      starting points for the auxiliary clustering problem (5)
      with k = l;
    SOLVING AUXILIARY CLUSTERING PROBLEM
      To obtain a set S_2 ⊂ R^n of starting points for the l–th
        clustering problem (1), apply LMBM to solve Problem (5)
        starting from each point y ∈ S_1;
    END SOLVING AUXILIARY CLUSTERING PROBLEM
    SOLVING CLUSTERING PROBLEM
      For each ȳ ∈ S_2 apply LMBM to solve Problem (1) starting
        from the point (x_1, ..., x_{l−1}, ȳ) and find a solution (ŷ_1, ..., ŷ_l);
      Denote by S_3 ⊂ R^{nl} a set of all such solutions;
    END SOLVING CLUSTERING PROBLEM
    SOLUTION TO THE l -PARTITION PROBLEM
      Compute f_l^min = min {f_l(ŷ_1, ..., ŷ_l) | (ŷ_1, ..., ŷ_l) ∈ S_3} and the
        collection of cluster centers (ȳ_1, ..., ȳ_l) such that
        f_l(ȳ_1, ..., ȳ_l) = f_l^min;
      Set x_j := ȳ_j, j = 1, ..., l as a solution to the l–partition problem;
    END SOLUTION TO THE l -PARTITION PROBLEM
  END WHILE
  RETURN the solution to k–partition problem;
END PROGRAM Incremental Algorithm
```

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 test the new LMBM-CLUST -method `LMBM-Clust` we compared it with three other clustering algorithms. Namely, `DC-Clust` [9] that utilizes the DC-representation of the MSSC clustering problem; modified global k-means algorithm `MS-MGKM` [35] that utilizes an incremental algorithm to find starting points for k-means, and the classical multistart k-means `MS-KM` [32].

`LMBM-Clust`, `DC-Clust`, and `MS-MGKM` methods are implemented in Fortran 95 while `MS-KM` is implemented in Fortran 77. All the softwares are compiled using `gfortran`, the GNU fortran compiler. The experiments were performed on MacBookAir (OS El Capitan 10.11.3) with Intel® Core™ i5, 1.6 GHz and RAM 4 GB.

The softwares LMBM-Clust and DC-Clust as well as the data sets used in our experiments can be downloaded from http://napsu.karmitsa.fi/clustering/.

We used thirteen large real world data sets in our experiments. The brief description of these sets is given in Table 1. The more detailed description can be found in [30] and references given in Table 1. All the data sets contain only numeric features and they do not have missing values. The numbers of attributes range from very few (2) to very large (5000) and the numbers of data points range from thousands (smallest 7 797) to hundred of thousands (largest 434 874).

Table 1: The brief description of data sets

Data sets	No. instances	No. attributes	References
ISOLET	7797	616	[30]
Gisette	13500	5000	[21]
Gas Sensor Array Drift	13910	128	[39]
EEG Eye State	14980	14	[30]
D15112	15112	2	[15]
Online News Popularity	39644	58	[20]
KEGG Metabolic	53413	20	[34]
Shuttle Control	58000	9	[30]*
Sensorless Drive Diagnosis	58509	49	[30]
Pla85900	85900	2	[15]
MiniBooNE particle identification	130064	50	[30]
Skin Segmentation	245057	3	[13]
3D Road Network	434874	3	[28]

Thanks to NASA.

LMBM-Clust, DC-Clust and MS-MGKM methods use the incremental approach to solve clustering problems globally. We computed incrementally up to 25 clusters with all data sets. MS-KM uses the simple randomised multistart scheme for starting points. Thus, it does not give any intermediate results. For comparison purposes, we made different runs for different numbers of clusters. The CPU-time used by softwares was limited to 20 hours. For MS-KM the numbers of different starting points was always kept big enough, but we limited the computational time of the method to be approximately twice of that used by LMBM-Clust (or 20 hours). We also stopped the run if the wall clock was more that 24 hours without any progress. That is, if after 24 hours MS-KM was still computing the clusters from the first starting point.

Results are given in Tables 2–14, where we have used the following notation:

- k is the number of clusters;
- f_{best} (multiplied by the number shown after the name of the data set) is the best known value for the cluster function (2) (multiplied with m) for the corresponding number of clusters. We have used the f_{best} value given in [9] (for those data sets that were used also in [9]) unless we got better value in our experiments. If the value obtained in our experiments was better than that of [9] we have marked it with asterix.

- 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 cluster function obtained by an algorithm A.

- *cpu* is the used cpu time in seconds.

In addition, in Figures 2 – 14 the computational times used by different algorithms and the number of distance function calculations versus the number of clusters for each data sets are given.

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	*cpu*	E_A	*cpu*	E_A	*cpu*	E_A	*cpu*
2	7.20988	0.00	272.31	0.00	277.20	0.00	78.72	0.00	600.61
3	6.77921	0.00	517.10	0.00	545.49	0.00	150.43	0.00	1101.18
5	6.12976	0.39	1066.26	1.01	1136.32	1.01	299.66	0.00	2203.64
10	5.27942	1.15	2419.93	1.15	2636.39	1.26	673.64	0.00	5005.88
15	4.86791	0.09	3757.33	0.09	4145.73	0.00	1066.23	0.11	8005.66
20	4.59548	0.00	5086.28	0.00	6187.10	0.01	1449.80	0.45	10225.54
25	4.43426	0.30	6314.97	0.00	8368.55	0.76	1888.43	1.24	13025.34

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	*cpu*	E_A	*cpu*	E_A	*cpu*	E_A	*cpu*
2	4.19944	0.00	20303.58	0.00	51844.82	0.00	25299.10	0.00	45111.51
3	4.11596	0.00	38420.07	–	–	0.00	49374.00	0.00	70193.29
4	4.06539	0.00	56599.01	–	–	–	–	0.00	70064.70

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	*cpu*	E_A	*cpu*	E_A	*cpu*	E_A	*cpu*
2	7.91186	0.00	18.14	0.00	24.78	0.00	17.03	0.00	40.51
3	5.02412	0.00	49.36	0.00	64.00	0.00	42.71	0.00	100.30
5	3.22394	0.10	118.28	0.10	165.43	0.10	102.51	0.00	251.05
10	1.65524	0.00	314.84	0.00	510.62	0.00	266.91	2.68	650.34
15	1.13801	0.36	528.27	0.36	988.77	0.04	422.40	12.10	1101.96
20	0.87916	0.75	740.42	0.62	1514.49	0.01	575.50	34.12	1500.01
25	0.72274*	0.55	958.77	0.58	2053.33	0.00	740.72	23.19	2002.00

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	7845.09934	4.24	0.14	4.24	0.44	4.24	0.29	0.00	1.44
3	1833.88058	0.00	0.15	0.00	0.84	0.00	0.59	227.91	2.53
5	1.33858	0.00	0.71	0.00	2.66	0.00	1.60	449091.75	7.20
10	0.45669	0.00	8.07	0.00	18.63	0.00	18.14	0.10	20.31
15	0.34653	0.04	18.74	0.26	49.07	0.05	36.39	4.78	43.15
20	0.28986*	0.00	30.58	0.96	88.35	0.00	49.12	2.83	70.94
25	0.25989*	0.14	43.15	0.00	137.86	0.07	60.22	2.12	92.12

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	3.68403	0.00	0.77	0.00	0.93	0.00	4.35	0.00	3.74
3	2.53240	0.00	1.48	0.00	1.92	0.00	10.51	0.00	4.22
5	1.32707	0.00	2.45	0.00	3.21	0.00	12.53	0.00	5.29
10	0.64490*	0.62	4.78	0.62	8.00	0.62	16.60	0.00	10.85
15	0.43136*	0.25	7.49	0.25	18.18	0.00	19.57	0.00	16.26
20	0.32177	0.24	10.39	0.00	32.58	0.25	23.01	6.55	23.51
25	0.25308*	0.00	14.17	0.00	53.89	0.01	26.26	5.55	31.47

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	9.53913	0.00	87.98	0.00	97.81	0.00	80.69	0.00	203.63
3	5.91077	0.00	236.44	0.00	253.04	0.00	249.82	0.13	516.83
5	3.09885	0.00	487.06	0.00	541.38	0.00	499.79	0.35	1026.20
10	1.17247	0.00	1354.22	0.00	1499.72	0.00	1126.97	82.56	3005.15
15	0.77637	0.00	2114.44	0.00	2653.41	0.00	1654.85	33.17	4422.03
20	0.59809	0.00	2885.10	0.00	4292.31	0.00	2274.46	34.48	6003.91
25	0.49616	0.00	3775.87	0.00	6011.97	0.82	2833.17	45.11	8003.82

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	11.38530	0.00	5.48	0.00	6.65	0.00	12.55	18.85	192.43
3	4.90060	0.00	14.03	0.00	18.25	0.00	65.41	124.79	295.71
5	1.88367	0.00	47.88	0.06	66.35	0.00	373.14	0.00	294.77
10	0.63515	0.00	194.12	0.21	358.11	0.00	1055.38	30.34	549.55
15	0.35125*	0.00	379.34	1.02	719.32	4.33	1372.31	98.14	802.76
20	0.24982*	0.00	579.81	2.23	1122.56	1.79	1736.07	161.39	1308.12
25	0.19289	1.21	782.00	0.75	1549.19	0.51	2152.38	221.85	1605.34

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	21.34329.	0.00	0.36	0.00	1.19	0.00	9.36	51.13	258.74
3	10.85415	0.00	0.85	0.00	3.33	0.00	19.80	100.52	195.08
5	7.24479	0.00	4.48	0.24	21.26	0.00	52.03	38.73	111.02
10	2.83166*	0.00	22.43	0.35	78.65	0.37	173.08	39.54	99.50
15	1.53154	0.00	57.26	0.37	290.63	0.07	255.67	200.94	140.01
20	1.06012	0.00	114.14	1.16	516.43	0.00	424.77	308.77	315.05
25	0.78727	1.49	190.86	0.13	774.88	1.50	617.82	326.43	439.92

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	3.93967	0.00	21.65	0.00	28.05	0.00	39.17	98.70	379.28
3	2.97163	0.00	80.20	0.09	109.05	0.00	110.75	152.32	352.13
5	1.99500	0.00	224.80	0.12	316.02	0.00	1034.71	36.74	598.97
10	1.01939	0.00	810.02	2.48	1319.53	4.16	2042.61	125.21	2230.44
15	0.68664	0.00	1563.78	4.18	2616.02	0.00	3469.83	214.58	3127.30
20	0.55732	0.00	2432.91	0.91	3979.53	0.00	4275.13	231.76	5029.11
25	0.47614	0.00	3306.79	0.81	5939.07	0.00	5087.14	278.80	7002.31

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	3.74908	0.00	16.26	0.00	15.63	0.00	188.26	1.44	530.39
3	2.28057	0.00	31.59	0.00	30.05	0.00	479.33	0.00	787.99
5	1.33972	0.00	62.40	0.00	60.87	0.00	687.09	2.77	306.11
10	0.68294	0.00	140.16	0.00	145.04	0.00	1267.43	0.55	352.71
15	0.46105	0.31	222.99	0.31	254.82	0.31	1740.19	0.00	462.97
20	0.34988	0.52	311.49	0.52	383.46	2.21	2052.67	0.03	666.75
25	0.28259*	0.18	399.04	0.02	529.44	0.00	2331.13	0.06	865.09

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

k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	8.92236	0.00	4.03	0.00	13.26	0.00	112.35	286908.08	1936.28
3	5.22601	0.00	220.39	0.00	258.50	0.00	1924.46	–	–
5	1.82252	0.00	1242.63	0.00	1209.26	0.00	6524.31	–	–
10	0.92406	0.00	5074.67	0.02	7196.27	0.01	21646.72	–	–
15	0.63506	0.00	9404.49	0.02	15537.69	0.01	30265.34	–	–
20	0.50863	0.00	14145.08	0.02	24363.40	0.36	37525.27	–	–
25	0.44425	0.00	18863.10	0.02	30855.61	0.01	41985.07	–	–

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

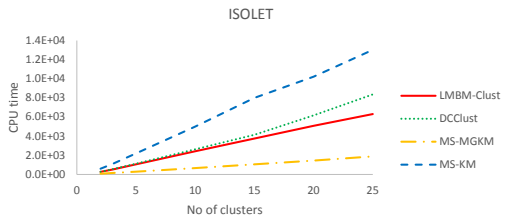
k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	1.32236	0.00	153.61	0.00	166.33	0.00	2466.95	0.00	2146.77
3	0.89362	0.00	278.13	0.00	290.68	0.00	4459.66	0.00	2709.68
5	0.50203	0.00	504.05	0.00	517.17	0.00	6408.17	3.28	1529.53
10	0.25121*	0.00	1042.00	0.00	1076.48	4.26	8521.71	0.00	2502.66
15	0.16963*	0.00	1546.04	0.18	1640.04	0.18	11051.29	13.09	3488.56
20	0.12615*	1.20	2034.39	1.38	2217.97	0.00	12614.88	26.51	4145.28
25	0.10228*	0.69	2568.40	0.70	2844.37	0.00	13941.71	32.61	5298.60

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

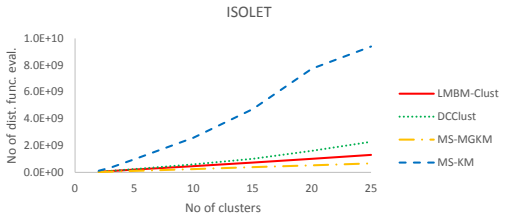
k	f_{best}	LMBM-Clust		DCClust		MS-MGKM		MS-KM	
		E_A	cpu	E_A	cpu	E_A	cpu	E_A	cpu
2	49.13298	0.00	423.92	0.00	446.96	0.00	17837.82	0.00	23821.27
3	22.77818	0.00	958.00	0.00	1004.36	0.00	31942.73	0.00	14898.22
5	8.82574	0.00	1884.02	0.00	2005.20	–	–	0.00	27531.09
10	2.56661*	0.00	4157.99	0.20	4332.89	–	–	0.01	53676.91
15	1.27069*	0.00	6465.47	0.00	6727.42	–	–	0.00	24049.60
20	0.80864*	0.00	8862.35	0.00	9300.88	–	–	0.01	35789.17
25	0.60216*	2.11	11344.29	2.11	12478.87	–	–	0.00	37730.75

The data sets can be divided into four groups with respect to numbers of attributes. First, data sets with with small number of attributes (2 or 3, see Table 1), that is, "D15112", "Pla85900", "Skin Segmentation", and "3D Road Network" data sets. Results presented in Tables 6, 11, 13, and 14 demonstrate that in these data sets the accuracies of the solvers were quite similar and the solutions found were at least close to best known solutions — mostly even with MS−KM. The only remarkable exceptions here are MS−KM in "D15112" data set with 20 and 25 clusters and in "Skin Segmentation" data set with almost all numbers of clusters (see Tables 6 and 13, respectively). In addition, in "3D Road Network" data set that has the largest number of instances within the data sets tested, MS−MGKM computed only 4 clusters in the given time limit of 20 hours (see Table 14). In this data set MS−KM usually computed clusters only from one starting point within the time limit. Nevertheless, by coincidental it was a good starting point and the results were accurate. In all these data sets MS−KM used the least number of distance function evaluations (see Figures 6, 11, 13, and 14). Nevertheless, the new software LMBM−Clust was the most efficient of the softwares tested, although, the differences were not significant with larger number of instances.

The second group consist of data sets with medium number of attributes (9 – 20, see Table 1), that is, "EEG Eye State", "KEGG Metabolic", and "Shuttle Control". Results in these data sets are given in Tables 5, 8, and 9 and Figures 5, 8, and 9, respectively. In these data sets the accuracies of the solvers were quite similar and the solutions found were at least close to best known solutions but with MS−KM. In



(a) CPU time



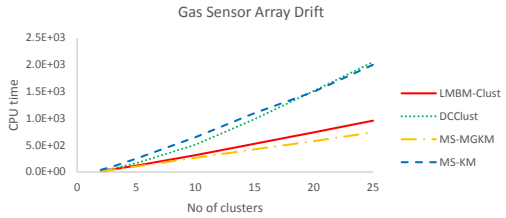
(b) No. of Norms

Figure 2: ISOLET.

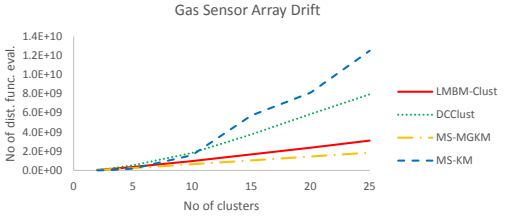
(a) CPU time

(b) No. of Norms

Figure 3: Gisette.

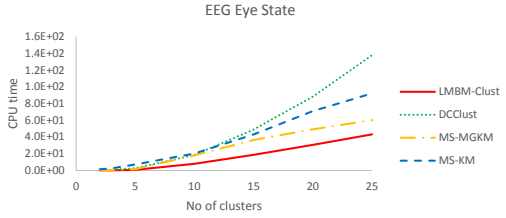


(a) CPU time

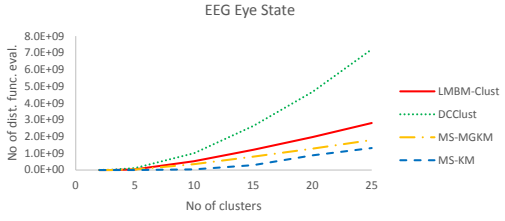


(b) No. of Norms

Figure 4: Gas Sensor Array Drift.

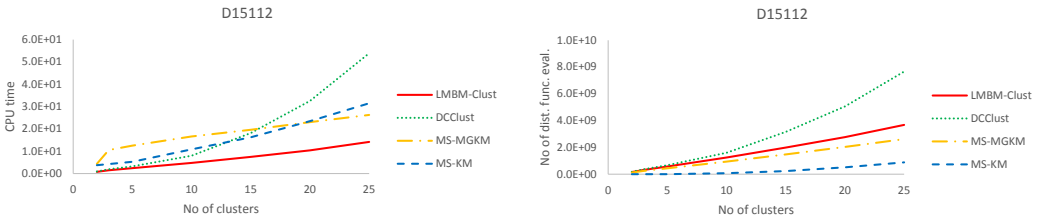


(a) CPU time



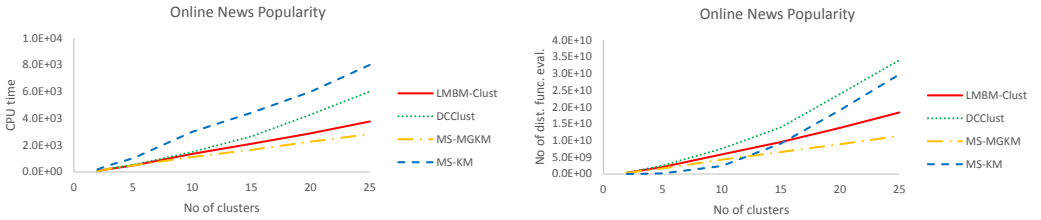
(b) No. of Norms

Figure 5: EEG Eye State.



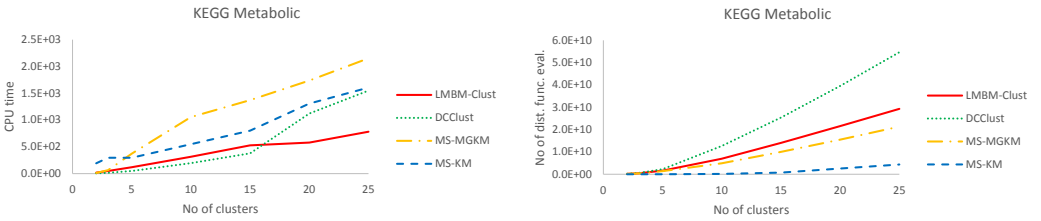
(a) CPU time (b) No. of Norms

Figure 6: D15112.



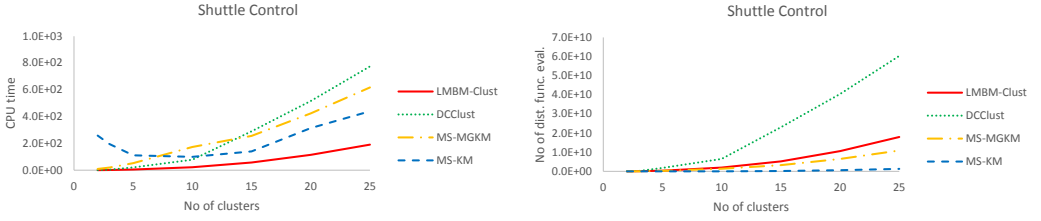
(a) CPU time (b) No. of Norms

Figure 7: Online News Popularity.



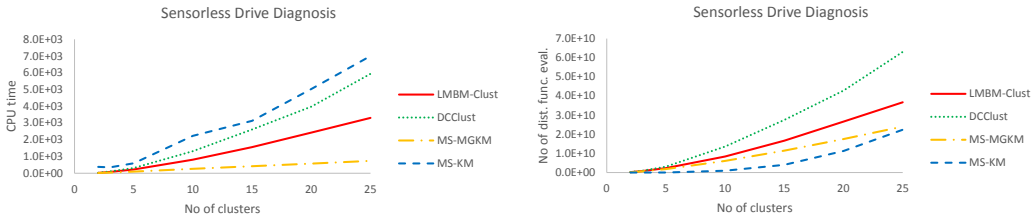
(a) CPU time (b) No. of Norms

Figure 8: KEGG Metabolic.



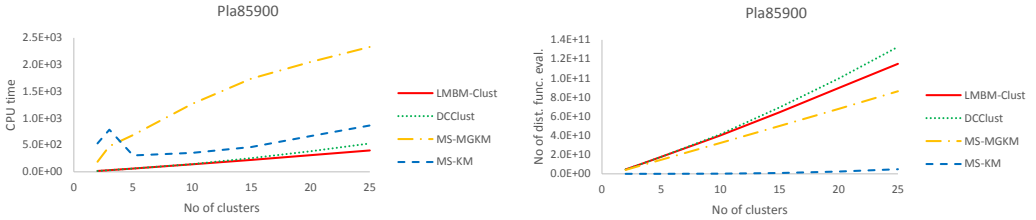
(a) CPU time (b) No. of Norms

Figure 9: Shuttle Control.



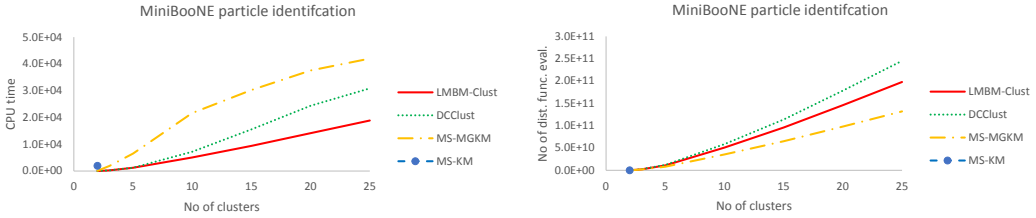
(a) CPU time (b) No. of Norms

Figure 10: Sensorless Drive Diagnostics.



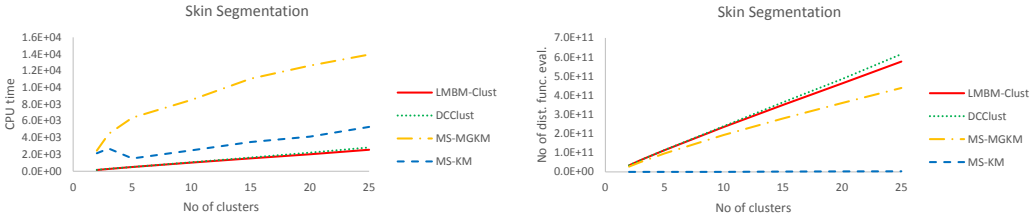
(a) CPU time (b) No. of Norms

Figure 11: Pla85900.



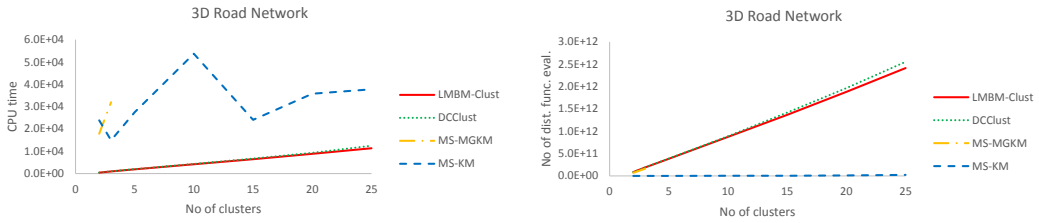
(a) CPU time (b) No. of Norms

Figure 12: MiniBooNE Particle Identification.



(a) CPU time (b) No. of Norms

Figure 13: Skin Segmentation.



(a) CPU time (b) No. of Norms

Figure 14: 3D Road Network.

almost all the cases the accuracies of the results obtained with MS−KM were highly unsatisfactory: in the worst case the error $E_{\texttt{MS}-\texttt{KM}} = 449091.75$ (see, Table 5). The other exceptions here are in "KEGG Metabolic" with 15 clusters, where MS−MGKM has $E_{\texttt{MS}-\texttt{MGKM}} = 4.33$, and "EEG Eye State" with two clusters where all the other methods but MS−KM have $E_A = 4.24$. As with small number of attributes, also in all these data sets the MS−KM used the least number of distance function calculations and the new software LMBM−Clust was the most efficient of the softwares tested. Here, the differences in computational times were bigger and in "Shuttle Control" data set that has the largest number of instances within the data sets with medium numbers of attributes, we can say that the difference in the computational times was outstanding.

In group three, there are data sets with large number of attributes (49 – 616, see Table 1): "ISOLET", "Gas Sensor Array Drift", "Online News Popularity", "Sensorless Drive Diagnosis", and "MiniBooNE particle identification". The results can be found in Tables 2, 4, 7, 10, and 12 and Figures 2, 4, 7, 10, and 12,. As before, the accuracies of the solvers were quite similar but with MS−KM. With MS−KM the accuracy was getting worse with the size of number of instances. In "MiniBooNE particle identification" data set MS−KM "solved" the problem only with two clusters within the time limit of 24 hours: the error was $E_{\texttt{MS}-\texttt{KM}} = 286908.08$ and the time 1936.28 seconds while, for instance, LMBM−Clust solved the same problem accurately in four seconds. In addition, MS−MGKM and DC−Clust had some small difficulties in "Sensorless Drive Diagnosis" data set (see Table 10). Within the group three, MS−MGKM was the most efficient of the methods when the number of instances was smaller than 50 000. It also usually used the least number of distance function evaluations. However, the computational time of MS−MGKM increased rapidly with the numbers of instances and, for instance, in "MiniBooNE particle identification" it was more than twice of that of LMBM−Clust. With larger number of instances LMBM−Clust was clearly the most efficient method tested.

The last group consist of the data set with very large number of attributes. That is "Gisette" (5000, see Table 1 for the data set and Table 3 and Figure 3 for results). In this data set, DC−Clust succeed in solving only two clusters in the given time limit, MS−MGKM three, and LMBM−Clust four. Thus, four was the maximum number of clusters computed with MS−KM as well. The accuracies of the solvers including MS−KM were similar. The computational time used by LMBM−Clust was less than

half of that used by DC−Clust, and about 75% of that used by MS−MGKM even if the number of instances in this data set was not very large (i.e. MS−MGKM should have been at its best). Although, MS−KM in this particular data set did well, it has not in general been reliable nor efficient method. Therefore, we can conclude that LMBM−Clust was the best method tested for clustering in large data sets with both large numbers of instances and attributes.

6 Conclusions

In this paper a new LMBM-CLUST -method for solving sum-of-squares clustering problems is introduced. The LMBM-CLUST -method consist two different algorithms: an incremental algorithm is used to solve clustering problems globally and at each iteration of this algorithm the LMBM algorithm is used to solve both the clustering and the auxiliary clustering problems with different starting points. The LMBM-CLUST method conveges to a stationary point of the clustering problem.

The new LMBM-CLUST -method was tested using real world data sets with the numbers of data points ranging from tens of thousands to hundreds of thousands. The new method LMBM-CLUST was clearly faster than the other methods tested with data sets including large number of instances and relatively large number of attributes. We can conclude that LMBM-CLUST -method was both efficient and accurate and it can be used to provide real time clustering in large data sets.

Acknowledgments

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 work was made while the corresponding author was visiting in Faculty of Science and Technology, Federation University Australia.

References

[1] AL-SULTAN, K. A tabu search approach to the clustering problem. *Pattern Recognition 28*, 9 (1995), 1443–1451.

[2] AN, L., BELGHITI, M., AND TAO, P. A new efficient algorithm based on DC programming and DCA for clustering. *Journal of Global Optimization 37*, 4 (2007), 593–608.

[3] AN, L., MINH, L., AND TAO, P. New and efficient DCA based algorithms for minimum sum-of-squares clustering. *Pattern Recognition 47* (2014), 388–401.

[4] AN, L., AND TAO, P. 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.

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

[6] BAGIROV, A., AND MOHEBI, E. Nonsmooth optimization based algorithms in cluster analysis. In *Partitional Clustering Algorithms*, E. Celebi, Ed. Springer International Publishing, 2015, pp. 99–146.

[7] BAGIROV, A., ORDIN, B., OZTURK, G., AND XAVIER, A. An incremental clustering algorithm based on hyperbolic smoothing. *Computational Optimization and Applications 61*, 1 (2015), 219–241.

[8] BAGIROV, A., RUBINOV, A., SOUKHOROUKOVA, N., AND YEARWOOD, J. Unsupervised and supervised data classification via nonsmooth and global optimization. *Top 11* (2003), 1–93.

[9] BAGIROV, A., TAHERI, S., AND UGON, J. Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. *Pattern Recognition 53* (2016), 12–24.

[10] BAGIROV, A., AND UGON, J. An algorithm for minimizing clustering functions. *Optimization 54*, 4–5 (2005), 351–368.

[11] BAGIROV, A., AND YEARWOOD, J. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research 170*, 2 (2006), 578–596.

[12] BAGIROV, A. M., KARMITSA, N., AND MÄKELÄ, M. M. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014.

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

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

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

[16] BYRD, R. H., NOCEDAL, J., AND SCHNABEL, R. B. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming 63* (1994), 129–156.

[17] CLARKE, F. H. *Optimization and Nonsmooth Analysis*. Wiley-Interscience, New York, 1983.

[18] DIEHR, G. Evaluation of a branch and bound algorithm for clustering. *SIAM J. Scientific and Statistical Computing*, 6 (1985), 268–284.

[19] DU MERLE, O., HANSEN, P., JAUMARD, B., AND MLADENOVIC, N. An interior point algorithm for minimum sum-of-squares clustering. *SIAM Journal on Scientific Computing 21*, 4 (1999), 1485–1505.

[20] FERNANDES, K., VINAGRE, P., AND CORTEZ, P. 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., 2015. Data set available in UCI machine learning repository <URL: http://archive.ics.uci.edu/ml> (June 11th, 2016).

[21] GUYON, I., GUNN, S. R., BEN-HUR, A., AND DROR, G. Result analysis of the nips 2003 feature selection challenge. In: NIPS, 2004. Data set available in UCI machine learning repository <URL: http://archive.ics.uci.edu/ml> (June 11th, 2016).

[22] HAARALA, M. *Large-Scale Nonsmooth Optimization: Variable Metric Bundle Method with Limited Memory*. PhD thesis, University of Jyväskylä, Department of Mathematical Information Technology, 2004.

[23] HAARALA, M., MIETTINEN, K., AND MÄKELÄ, M. M. New limited memory bundle method for large-scale nonsmooth optimization. *Optimization Methods and Software 19*, 6 (2004), 673–692.

[24] HAARALA, N., MIETTINEN, K., AND MÄKELÄ, M. M. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming 109*, 1 (2007), 181–205.

[25] HANSEN, P., AND MLADENOVIC, N. Variable neighborhood decomposition search. *Journal of Heuristic 7* (2001), 335–350.

[26] HIRIART-URRUTY, J.-B., AND LEMARÉCHAL, C. *Convex Analysis and Minimization Algorithms II*. Springer-Verlag, Berlin, 1993.

[27] KARMITSA, N., MÄKELÄ, M. M., AND ALI, M. M. Limited memory interior point bundle method for large inequality constrained nonsmooth minimization. *Applied Mathematics and Computation 198*, 1 (2008), 382–400.

[28] KAUL, M., YANG, B., AND JENSEN, C. S. 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, 2013. Data set available in UCI machine learning repository <URL: `http://archive.ics.uci.edu/ml`> (April 8th, 2016).

[29] KIWIEL, K. C. *Methods of Descent for Nondifferentiable Optimization*. Lecture Notes in Mathematics 1133. Springer-Verlag, Berlin, 1985.

[30] LICHMAN, M. 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, 2001. (April 8th, 2016).

[31] LUKŠAN, L., AND VLČEK, J. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications 102*, 3 (1999), 593–613.

[32] MACQUEEN, J. B. 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] MÄKELÄ, M. M., AND NEITTAANMÄKI, P. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific Publishing Co., Singapore, 1992.

[34] NAEEM, M., AND SOHAIL, A. 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).

[35] ORDIN, B., AND BAGIROV, A. A heuristic algorithm for solving the minimum sum-of-squares clustering problems. *Journal of Global Optimization 61*, 2 (2015), 341–361.

[36] RAHMAN, M. A., AND ISLAM, M. Z. A hybrid clustering technique combining a novel genetic algorithm with k-means. *Knowledge-Based Systems 71* (2014), 345–365.

[37] SELIM, S., AND AL-SULTAN, K. A simulated annealing algorithm for the clustering. *Pattern Recognition 24*, 10 (1991), 1003–1008.

[38] TEBOULLE, M. A unified continuous optimization framework for center-based clustering methods. *The Journal of Machine Learning Research*, 8 (2007), 65–102.

[39] VERGARA, A., VEMBU, S., AYHAN, T., RYAN, M. A., HOMER, M. L., AND HUERTA, R. Chemical gas sensor drift compensation using classifier ensembles. Sensors and Actuators B: Chemical (2012) DOI: 10.1016/j.snb.2012.01.074., 2012. Data set available in UCI machine learning repository <URL: `http://archive.ics.uci.edu/ml`> (April 8th, 2016).

[40] VLČEK, J., AND LUKŠAN, L. Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications 111*, 2 (2001), 407–430.

[41] XAVIER, A. The hyperbolic smoothing clustering method. *Pattern Recognition 43* (2010), 731–737.

[42] XAVIER, A., AND XAVIER, V. 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.

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Computer Science
- Computer Engineering

ISBN 978-952-12-3427-9
ISSN 1239-1891