New Diagonal Bundle Method for Clustering Problems in Large Data Sets

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

Clustering is one of the most important tasks in data mining. Recent developments in computer hardware allow us to store in random access memory (RAM) and repeatedly read data sets with hundreds of thousands and even millions of data points. This makes it possible to use conventional clustering algorithms in such data sets. However, these algorithms may need prohibitively large computational time and fail to produce accurate solutions. Therefore, it is important to develop clustering algorithms which are accurate and can provide real time clustering in large data sets. This paper introduces one of them. Using nonsmooth optimization formulation of the clustering problem the objective function is represented as a difference of two convex (DC) functions. Then a new diagonal bundle algorithm that explicitly uses this structure is designed and combined with an incremental approach to solve this problem. The method is evaluated using real world data sets with both large number of attributes and large number of data points. The proposed method is compared with two other clustering algorithms using numerical results.

Keywords: Data mining, Nonsmooth optimization, Nonconvex optimization, DC function, Bundle methods

1. Introduction

Clustering (unsupervised classification) deals with a problem of organizing a collection of patterns into groups based on similarity. The similarity measure can be defined using various distance-like functions. When the similarity measure is defined by the squared Euclidean norm, the clustering problem is called the minimum sum-of-squares clustering (MSSC) problem. The MSSC can be formulated as an optimization problem and there are various optimization algorithms

*Corresponding author: The work was made while the Corresponding author was visiting Faculty of Science and Technology, Federation University Australia, Victoria, Australia.
Email address: napsu@karmitsa.fi (Napsu Karmitsa)

for solving it. These include branch and bound algorithms [1, 2], interior point methods [3], variable neighborhood search algorithms [4, 5, 6], and metaheuristics such as clustering search [7], simulated annealing [8, 9], tabu search [10], and genetic algorithms [11]. A nonsmooth optimization (NSO, not necessarily differentiable optimization) formulation is used and algorithms are developed in [12, 13, 14]. In addition, algorithms based on hyperbolic smoothing technique are presented in [15, 16, 17] and algorithms based on difference of convex (DC) representation are introduced in [18, 19, 20, 21].

Recent advances in computer hardware allow us to store in random access memory (RAM) and repeatedly read data sets containing hundreds of thousands and even millions of data points. Most of the aforementioned algorithms are not applicable or have limited capabilities for solving clustering problems in such large data sets, since they become time consuming when the size of data sets increases. Therefore, it is important to develop clustering algorithms which are accurate and can provide real time clustering in these data sets. The aim of this paper is to develop one such algorithm.

The idea of the new algorithm comes from two sources. First, the possibility of splitting the MSSC problem into two different pieces — one of which is convex and smooth (continuously differentiable) and another one is generic convex nonsmooth. Second, from the fact that the convex model of the objective function is usually reasonably good except some regions where there exist the so-called concave behavior in the objective. Thus, the nonconvexity of the problem should be taken into account with the "minimum" effort.

In this paper, we introduce a new *DC diagonal bundle algorithm* (DCD-Bundle) for solving clustering problems given in a form of the DC function. The DCD-Bundle combines the ideas of the *diagonal bundle method* (D-Bundle, [22]) with a different usage of metrics depending on a convex or concave behavior of the objective at the current iteration point. The D-Bundle, in its turn, is developed for sparse, large-scale, possibly nonconvex, NSO. It is a successor of the *limited memory bundle method* (LMBM, [23, 24]) and the *variable metric bundle method* (VMBM, [25, 26]). This method is better capable of handling large dimensionality and sparsity of the objective. The idea of splitting the information on the objective obtained in previous iterations comes from [27, 28]. However, instead of splitting the bundle information (i.e. the subgradient information) as in [27, 28], the DCD-Bundle computes different variable metric approximations depending on the point. In addition, the DCD-Bundle uses the real DC structure of the problem.

The DCD-Bundle shares good properties of the D-Bundle. That is, the time-consuming quadratic direction finding problem appearing in standard bundle methods (see e.g. [29, 30, 31]) needs not 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 diagonal variable metric approximation of the Hessian matrix and, thus, it avoids storing and manipulating large matrices as is the case in the VMBM, and dense approximations to Hessian as is the case in the LMBM. The usage of different metrics in the DCD-Bundle gives us a possibility to better deal with the nonconvex-

ity of the problem than the use of only subgradient locality measure (see e.g. [32, 33]) used in the D-BUNDLE .

The rest of this paper is organized as follows. In Section 2 we introduce our notation and recall some basic definitions and results from nonsmooth analysis and DC programming. DC representations of cluster functions are given in Section 3. Optimality conditions for the auxiliary clustering problem and the clustering problem are given in Section 4. In Section 5, we discuss the basic ideas of the DCD-BUNDLE method and, in Section 6, we prove its convergence. In Section 7, we recall the ideas of incremental approach used to globally solve the clustering problem. The results of numerical experiments are presented and discussed in Section 8, and finally, Section 9 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). In addition, we denote $\text{diag}(\boldsymbol{a})$, for $\boldsymbol{a} \in \mathbb{R}^n$, the diagonal matrix such that $\text{diag}(\boldsymbol{a})_{i,i} = \boldsymbol{a}_i$. The Frobenius norm of a matrix $A \in \mathbb{R}^{n\times n}$ is denoted by $\|A\|_F$. That is, we define

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}^2}.$$

An open (closed) ball with the center $\boldsymbol{x} \in \mathbb{R}^n$ and the radius $r > 0$ is denoted by $B_r(\boldsymbol{x})$ ($\bar{B}_r(\boldsymbol{x})$).

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called a *DC function* if there exist two convex functions $f_1 : \mathbb{R}^n \to \mathbb{R}$ and $f_2 : \mathbb{R}^n \to \mathbb{R}$ such that

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}).$$

The functions f_1 and f_2 are called *DC components* of f and $f_1 - f_2$ is called a *DC decomposition* of f.

An *unconstrained DC programming problem* is an optimization problem of the form

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}), \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{1}$$

where f_1 and f_2 are convex, not necessarily differentiable functions.

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

$$\partial_c f(\boldsymbol{x}) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^n \mid \ f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq \boldsymbol{\xi}^T(\boldsymbol{y} - \boldsymbol{x}) \text{ for all } \boldsymbol{y} \in \mathbb{R}^n \right\}.$$

Each vector $\boldsymbol{\xi} \in \partial_c f(\boldsymbol{x})$ is called a *subgradient*.

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

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

The *generalized derivative* [34] of a locally Lipschitz function f at a point $\boldsymbol{x}$ with respect to a direction $\boldsymbol{u} \in \mathbb{R}^n$ is defined as

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

and the *Clarke subdifferential* (or simply subdifferential) $\partial f(\boldsymbol{x})$ of a locally Lipschitz function f at $\boldsymbol{x}$ is given by

$$\partial f(\boldsymbol{x}) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^n \mid \ f^0(\boldsymbol{x}, \boldsymbol{u}) \geq \boldsymbol{\xi}^T \boldsymbol{u} \text{ for all } \boldsymbol{u} \in \mathbb{R}^n \right\}.$$

Each vector $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ is called a *subgradient*. For a convex function, we have $\partial f(\boldsymbol{x}) = \partial_c f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$. From now on, we will use the notation ∂f also for subdifferentials of convex functions.

A function f is called *directionally differentiable* at $\boldsymbol{x}$ if the limit

$$f'(\boldsymbol{x}, \boldsymbol{u}) = \lim_{\alpha \downarrow 0} \frac{f(\boldsymbol{x} + \alpha \boldsymbol{u}) - f(\boldsymbol{x})}{\alpha}$$

exists for any $\boldsymbol{u} \in \mathbb{R}^n$. A directionally differentiable function is called *subdifferentially regular* at $\boldsymbol{x}$ if $f'(\boldsymbol{x}, \boldsymbol{u}) = f^0(\boldsymbol{x}, \boldsymbol{u})$, for all $\boldsymbol{u} \in \mathbb{R}^n$. In general, nonsmooth DC functions are not subdifferentially regular and the Clarke subdifferential calculus exists for such functions only in the form of inclusions (see e.g. [35])

$$\partial f(\boldsymbol{x}) \subseteq \partial f_1(\boldsymbol{x}) - \partial f_2(\boldsymbol{x}). \tag{2}$$

In general, such a calculus cannot be used to compute subgradients of the function f.

A point $\boldsymbol{x}^* \in \mathbb{R}^n$ is called a local minimizer of the problem (1) if there exists $r > 0$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in B_r(\boldsymbol{x}^*)$.

THEOREM 2.1. *[36] A point $\boldsymbol{x}^*$ to be a local minimizer of the problem* (1)*, it is necessary that*

$$\partial f_2(\boldsymbol{x}^*) \subseteq \partial f_1(\boldsymbol{x}^*). \tag{3}$$

If a point $\boldsymbol{x}^*$ is a local minimizer of the problem (1), then

$$0 \in \partial f(\boldsymbol{x}^*) \tag{4}$$

and

$$\partial f_2(\boldsymbol{x}^*) \cap \partial f_1(\boldsymbol{x}^*) \neq \emptyset. \tag{5}$$

Points satisfying (3) are called *inf-stationary*, points satisfying (4) are called *Clarke stationary*, and points satisfying (5) are called *critical* points of the problem (1). In general, any inf-stationary point is also a Clarke stationary and a critical point. Furthermore, any Clarke stationary point is also a critical point.

3. DC Programming Approach to Clustering Problems

In this section we recall a NSO formulation of clustering problems and their DC representations.

Cluster Analysis. Let A be a finite set of points in $\mathbb{R}^n$, that is

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

The *hard unconstrained clustering problem* is the distribution of the points of the set A into a given number k of disjoint subsets A^j, $j = 1, \ldots, k$ 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. $A = \bigcup\limits_{j=1}^{k} A^j$.

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

The *similarity* (or dissimilarity) *measure* is fundamental in cluster analysis. In this paper, the similarity measure is defined using the L_2 norm

$$d_2(\boldsymbol{x}, \boldsymbol{a}) = \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{a}_i)^2. \tag{6}$$

Clustering Problem. The *NSO formulation of the MSSC problem* is given as [13, 37]:

$$\begin{cases} \text{minimize} & f_k(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \in \mathbb{R}^{nk}, \end{cases} \tag{7}$$

where

$$f_k(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) = \frac{1}{m} \sum_{\boldsymbol{a} \in A} \min_{j=1,\ldots,k} d_2(\boldsymbol{x}^j, \boldsymbol{a}). \tag{8}$$

The DC representation of the function f_k in Problem (7) is [21]

$$f_k(\boldsymbol{x}) = f_{k1}(\boldsymbol{x}) - f_{k2}(\boldsymbol{x}), \ \boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \in \mathbb{R}^{nk}, \tag{9}$$

where

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

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

are convex functions.

Note that the similarity measure in the clustering problem can also be defined using other norms, for instance, L_1- or L_∞-norms (see, e.g. [35]). Nevertheless, with these norms both f_{k1} and f_{k2} are nonsmooth while with d_2, defined in (6), the first component f_{k1} of the DC function is smooth. Due to the fact that general nonsmooth DC functions are not subdifferentially regular, smoothness of f_{k1} is in demand in order to calculate the subdifferentials of the clustering function f_k.

Auxiliary Clustering Problem. Problem (7) is a nonconvex optimization problem and it may have a large number of local minimizers. Since we apply a local search method for solving this problem it is important to use a special procedure to generate starting cluster centers. Such an approach allows one to find high quality solutions to clustering problems using local search methods. Here we apply an algorithm introduced in [38]. This algorithm involves the so-called auxiliary clustering problem.

Let $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{k-1},\ k \geq 2$ be the solution to the $(k-1)$-clustering problem and

$$r_{k-1}^{\boldsymbol{a}} = \min\left\{d_2(\boldsymbol{x}^1, \boldsymbol{a}), \ldots, d_2(\boldsymbol{x}^{k-1}, \boldsymbol{a})\right\}. \tag{10}$$

The *k-th auxiliary cluster function* is defined as [39]

$$\bar{f}_k(\boldsymbol{y}) = \frac{1}{m}\sum_{\boldsymbol{a}\in A} \min\left\{r_{k-1}^{\boldsymbol{a}}, d_2(\boldsymbol{y}, \boldsymbol{a})\right\},\ \boldsymbol{y} \in \mathbb{R}^n. \tag{11}$$

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

$$\begin{cases} \text{minimize} & \bar{f}_k(\boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} \in \mathbb{R}^n, \end{cases} \tag{12}$$

is called the *k-th auxiliary clustering problem* [39]. The DC representation of the function $\bar{f}_k$ is given by [21]

$$\bar{f}_k(\boldsymbol{y}) = \bar{f}_{k1}(\boldsymbol{y}) - \bar{f}_{k2}(\boldsymbol{y}), \tag{13}$$

where

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

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

4. Optimality Conditions

In this section we give optimality conditions for Problems (7) and (12) using their DC representations. For more details and proofs we refer to [21].

Subgradients and Optimality Conditions for Auxiliary Clustering Problem. The function $\bar{f}_{k1}$ is smooth on $\mathbb{R}^n$ and its gradient at $\boldsymbol{y} \in \mathbb{R}^n$ is given by

$$\nabla \bar{f}_{k1}(\boldsymbol{y}) = \frac{2}{m} \sum_{\boldsymbol{a} \in A} (\boldsymbol{y} - \boldsymbol{a}). \tag{14}$$

In its turn, the function $\bar{f}_{k2}$ is nonsmooth and to write its subdifferential at a given point $\boldsymbol{y} \in \mathbb{R}^n$ we introduce the following sets

$$\begin{aligned}
\bar{A}_1(\boldsymbol{y}) &= \{\boldsymbol{a} \in A \mid \ r_{k-1}^{\boldsymbol{a}} > d_2(\boldsymbol{y}, \boldsymbol{a})\}, \\
\bar{A}_2(\boldsymbol{y}) &= \{\boldsymbol{a} \in A \mid \ r_{k-1}^{\boldsymbol{a}} < d_2(\boldsymbol{y}, \boldsymbol{a})\}, \\
\bar{A}_3(\boldsymbol{y}) &= \{\boldsymbol{a} \in A \mid \ r_{k-1}^{\boldsymbol{a}} = d_2(\boldsymbol{y}, \boldsymbol{a})\}.
\end{aligned}$$

The function $\bar{f}_{k2}$ can be rewritten as

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

Then its subdifferential at $\boldsymbol{y}$ is

$$\partial \bar{f}_{k2}(\boldsymbol{y}) = \frac{2}{m} \left(\sum_{\boldsymbol{a} \in \bar{A}_2(\boldsymbol{y})} (\boldsymbol{y} - \boldsymbol{a}) + \sum_{\boldsymbol{a} \in \bar{A}_3(\boldsymbol{y})} \operatorname{conv}\{0, (\boldsymbol{y} - \boldsymbol{a})\} \right). \tag{15}$$

Now, due to smoothness of $\bar{f}_{k1}$, the generalized subdifferential $\partial \bar{f}_k(\boldsymbol{y})$ of the function $\bar{f}_k$ at $\boldsymbol{y} \in \mathbb{R}^n$ can be given as

$$\partial \bar{f}_k(\boldsymbol{y}) = \nabla \bar{f}_{k1}(\boldsymbol{y}) - \partial \bar{f}_{k2}(\boldsymbol{y}).$$

THEOREM 4.1. *At any inf-stationary point $\boldsymbol{y}^*$ of Problem (12) the subdifferential $\partial \bar{f}_{k2}(\boldsymbol{y}^*)$ is singleton:*

$$\partial \bar{f}_{k2}(\boldsymbol{y}^*) = \frac{2}{m} \left\{ \sum_{\boldsymbol{a} \in \bar{A}_2(\boldsymbol{y}^*)} (\boldsymbol{y}^* - \boldsymbol{a}) \right\}. \tag{16}$$

Moreover, for a point $\boldsymbol{y}^*$ to be a local minimizer of Problem (12) it is necessary that

$$\sum_{\boldsymbol{a} \in \bar{A}_1(\boldsymbol{y}^*)} (\boldsymbol{y}^* - \boldsymbol{a}) = 0. \tag{17}$$

Note that any inf-stationary point of Problem (12) is also Clarke stationary and critical point of this problem. In general, the set of inf-stationary points is a strict subset of these two sets. In addition, the sets of Clarke stationary and critical points of Problem (12) coincide and they are given by

$$S = \{\boldsymbol{y} \in \mathbb{R}^n \mid \nabla \bar{f}_{k1}(y) \in \partial \bar{f}_{k2}(\boldsymbol{y})\}. \tag{18}$$

Finally, we demonstrate how two different subgradients from $\partial \bar{f}_{k2}(\boldsymbol{y}), \boldsymbol{y} \in \mathbb{R}^n$ can be computed if this subdifferential is not a singleton. This result is needed in order to develop an algorithm that can escape a Clarke stationary point and converge to an inf-stationary point. To compute different subgradients we can choose $\boldsymbol{\xi}^1,\ \boldsymbol{\xi}^2 \in \partial \bar{f}_{k2}(\boldsymbol{y})$ using (15) as follows

$$\boldsymbol{\xi}^1 = \frac{2}{m} \sum_{\boldsymbol{a} \in \bar{A}_2(\boldsymbol{y})} (\boldsymbol{y} - \boldsymbol{a}),$$

and

$$\boldsymbol{\xi}^2 = \boldsymbol{\xi}^1 + \bar{\boldsymbol{\xi}},\ \bar{\boldsymbol{\xi}} = \operatorname{argmax}_{\boldsymbol{a} \in \bar{A}_3(\boldsymbol{y})} \|\boldsymbol{y} - \boldsymbol{a}\|.$$

It is clear that if $\bar{\boldsymbol{\xi}} = \boldsymbol{0}$ then $\partial \bar{f}_{k2}(\boldsymbol{y})$ is a singleton.

Subgradients and Optimality Conditions for Clustering Problem. The function f_{k1} is smooth and its gradient is given by [35]

$$\nabla f_{k1}(\boldsymbol{x}) = 2(\boldsymbol{x} - \widehat{A}). \tag{19}$$

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

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

In general, the function f_{k2} is nonsmooth. To compute its subdifferential consider the following function and a set [38]:

$$\varphi_{\boldsymbol{a}}(\boldsymbol{x}) = \max_{j=1,\ldots,k} \sum_{s=1, s \neq j}^{k} d_2(\boldsymbol{x}^s, \boldsymbol{a}), \tag{20}$$

and

$$R_{\boldsymbol{a}}(\boldsymbol{x}) = \left\{ j \in \{1, \ldots, k\} \mid \sum_{s=1, s \neq j}^{k} d_2(\boldsymbol{x}^s, \boldsymbol{a}) = \varphi_{\boldsymbol{a}}(\boldsymbol{x}) \right\}. \tag{21}$$

The subdifferential $\partial \varphi_{\boldsymbol{a}}(\boldsymbol{x})$ of the function $\varphi_{\boldsymbol{a}}$ at $\boldsymbol{x}$ is as follows:

$$\partial \varphi_{\boldsymbol{a}}(\boldsymbol{x}) = \operatorname{conv} \left\{ V \in \mathbb{R}^{nk} \mid\ V = 2(\tilde{\boldsymbol{x}}^j - \tilde{A}_i^j), j \in R_{\boldsymbol{a}}(\boldsymbol{x}) \right\}, \tag{22}$$

where

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

and

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

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

$$\partial f_{k2}(\boldsymbol{x}) = \frac{1}{m} \sum_{\boldsymbol{a} \in A} \partial \varphi_{\boldsymbol{a}}(\boldsymbol{x}). \tag{23}$$

Similarly to the auxiliary clustering problem, the smoothness of $f_{k1}(\boldsymbol{x})$ allows us to write the generalized subdifferential $\partial f_k(\boldsymbol{x})$ of the clustering function f_k at $\boldsymbol{x} \in \mathbb{R}^{nk}$ as

$$\partial f_k(\boldsymbol{x}) = \nabla f_{k1}(\boldsymbol{x}) - \partial f_{k2}(\boldsymbol{x}).$$

Moreover, the sets of Clarke stationary and critical points of Problem (7) coincide and at these points

$$\nabla f_{k1}(\boldsymbol{x}) \in \partial f_{k2}(\boldsymbol{x}).$$

Now we are ready to give the necessary condition for a local minimum of the clustering problem.

THEOREM 4.2. *Let $\boldsymbol{x} \in \mathbb{R}^{nk}$ be a local minimizer of the problem (7). Then the objective function f_k is smooth at this point and*

$$\partial f_k(\boldsymbol{x}) = \{\nabla f_k(\boldsymbol{x})\} = \{0\},$$

where

$$\nabla f_k(\boldsymbol{x}) = \frac{2}{m} \sum_{a \in A} \sum_{j \in R_{\boldsymbol{a}}(\boldsymbol{x})} (\boldsymbol{x}^j - \boldsymbol{a}).$$

We finish this section by showing how two different subgradients from $\partial f_{k2}(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^{nk}$ can be computed if these subdifferentials are not singleton. Let us define the following two sets:

$$A_1 = \{\boldsymbol{a} \in A \mid |R_{\boldsymbol{a}}(\boldsymbol{x})| = 1\}, \qquad A_2 = \{\boldsymbol{a} \in A \mid |R_{\boldsymbol{a}}(\boldsymbol{x})| \geq 2\}.$$

If $A_1 = A$ then $\partial \bar{f}_{k2}(\boldsymbol{x})$ is a singleton. If $|A_2| \geq 1$ then $\partial \bar{f}_{k2}(\boldsymbol{x})$ is not a singleton. Take any $\boldsymbol{a} \in A_2$. Since $|R_{\boldsymbol{a}}(\boldsymbol{x})| \geq 2$ this point is attracted by at least two cluster centers. Using two cluster centers, we can compute two subgradients for the function $\varphi_{\boldsymbol{a}}$ defined by (20).

5. Diagonal Bundle Method for DC Clustering Problem

In this section, we introduce a new algorithm DCD-BUNDLE for solving clustering problems given in the DC-form. The algorithm is used to solve both the clustering problem (7) and the auxiliary clustering problem (12). It is easy to see that both these problems can be formulated as an unconstrained DC programming problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{24}$$

with f_1 and f_2 convex. In addition, in our approach we assume that f_1 is smooth. This assumption is trivially satisfied for the clustering function f_k and the auxiliary clustering function $\bar{f}_k$. Furthermore, we assume that at every point $\boldsymbol{x}$ we can evaluate the values of the DC-components f_1 and f_2 of the objective function f, the gradient $\nabla f_1(\boldsymbol{x})$ of the first component and one arbitrary subgradient $\boldsymbol{\xi}_2$ from the subdifferential $\partial f_2(\boldsymbol{x})$ of the second component.

Linearization error. The DCD-BUNDLE uses the sign of the linearization error to detect the "convex" or "concave" behavior of the objective. The linearization error α_{k+1} at a point $\boldsymbol{y}_{k+1}$ is given by

$$\alpha_{k+1} = f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + (\nabla f_1(\boldsymbol{y}_{k+1}) - \boldsymbol{\xi}_{2,k+1})^T \boldsymbol{d}_k.$$

Here $\boldsymbol{x}_k$ is the current iteration point, $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$ is a new auxiliary point, $\boldsymbol{d}_k$ is the current search direction and $\boldsymbol{\xi}_{2,k+1} \in \partial f_2(\boldsymbol{y}_{k+1})$.

Matrix Updating and Splitting of Information. The DCD-BUNDLE uses the diagonal update formula introduced in [40] for updating the matrices, since for this formula, it is easy to check and guarantee the positive definiteness of generated matrices. Moreover, using the diagonal update matrix requires a minimum amount of storage space and computations.

In practice, the DCD-BUNDLE uses at most m_c correction vectors to compute updates for matrices. These correction vectors are slightly modified from those in the classical limited memory variable metric methods for smooth optimization (see, e.g. [41]). That is, the correction vectors are given by $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k$, $\boldsymbol{u}_{1,k} = \nabla f_1(\boldsymbol{y}_{k+1}) - \nabla f_1(\boldsymbol{x}_k)$, and $\boldsymbol{u}_{2,k} = \boldsymbol{\xi}_{2,k+1} - \boldsymbol{\xi}_{2,m}$ with $\boldsymbol{\xi}_{2,k+1} \in \partial f_2(\boldsymbol{y}_{k+1})$ and $\boldsymbol{\xi}_{2,m} \in \partial f_2(\boldsymbol{x}_k)$. Due to the 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, we now compute the gradient differences separately for both DC-components. Since the gradient does not need to exist for nonsmooth component f_2, the correction vectors $\boldsymbol{u}_2$ are computed using subgradients.

Now, instead of just two correction matrices $S_k = [\boldsymbol{s}_{k-\hat{m}_k+1} \dots \boldsymbol{s}_k]$ and $U_k = [\boldsymbol{u}_{k-\hat{m}_k+1} \dots \boldsymbol{u}_k]$ used in [41] and, also in the D-BUNDLE [22], we have three correction matrices $S_k = [\boldsymbol{s}_{k-\hat{m}_k+1} \dots \boldsymbol{s}_k]$, $U_{1,k} = [\boldsymbol{u}_{1,k-\hat{m}_k+1} \dots \boldsymbol{u}_{1,k}]$ and $U_{2,k} = [\boldsymbol{u}_{2,k-\hat{m}_k+1} \dots \boldsymbol{u}_{2,k}]$, where $\hat{m}_k = \min\{k, m_c\}$. We use these matrices to compute separate approximations to both f_1 and f_2. Moreover, we use the different approximations depending on the sign of the linearization error.

The approximation of the Hessian $B_{l,k+1}$ $(l = 1, 2)$ is chosen to be a diagonal matrix and the check of positive definiteness is included as a constraint to the problem. Thus, the update matrix $B_{l,k+1}$ $(l = 1, 2)$ is defined by

$$\begin{cases} \text{minimize} & \|B_{l,k+1}S_k - U_{l,k}\|_F^2 \\ \text{subject to} & (B_{l,k+1})_{i,j} = 0 \text{ for } i \neq j \\ & (B_{l,k+1})_{i,i} \geq \mu \text{ for } i = 1, 2, \dots, n \text{ and } \mu > 0. \end{cases} \tag{25}$$

This minimization problem has a solution

$$(B_{l,k+1})_{i,i} = \begin{cases} \boldsymbol{b}_i/Q_{i,i}, & \text{if } \boldsymbol{b}_i/Q_{i,i} > \mu \\ \mu, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{b} = 2\sum_{i=\hat{1}}^{\hat{m}_k} \text{diag}(\boldsymbol{s}_i)\boldsymbol{u}_{l,i}$ and $Q = 2\sum_{i=\hat{1}}^{\hat{m}_k} [\text{diag}(\boldsymbol{s}_i)]^2$ with $\boldsymbol{s}_i \in S$ and $\boldsymbol{u}_{l,i} \in U_l$, $l = 1, 2$. In our computations we use the inverse of this matrix, that is, $D_{l,k} = (B_{l,k})^{-1}$. Note that in addition to the upper bound $\mu_{max} = \frac{1}{\mu}$, we also use the lower bound μ_{min} $(0 < \mu_{min} < \mu_{max})$ for the components of the matrix. We call the approximations $D_{1,k}$ and $D_{2,k}$ the "*convex approximation*" and the "*concave approximation*", respectively.

Direction Finding, Serious and Null Steps. The DCD-Bundle uses the above mentioned diagonal approximations to compute a search direction. If the previous step was a serious step, we suppose that the convex model of the function is good enough and we use directly the "convex approximation" of the Hessian and the current subgradient of the objective function. That is, the search direction is computed by the formula

$$\boldsymbol{d}_k = -D_{1,k}\boldsymbol{\xi}_k,$$

where $\boldsymbol{\xi}_k = \nabla f_1(\boldsymbol{x}_k) - \boldsymbol{\xi}_{2,k} \in \partial f(\boldsymbol{x}_k)$ and $\boldsymbol{\xi}_{2,k} \in \partial f_2(\boldsymbol{x}_k)$. Otherwise, if the linearization error α_{k+1} is positive, we still use the "convex approximation" of the Hessian, but the subgradient of the objective is taken in a form of an *aggregate subgradient* $\tilde{\boldsymbol{\xi}}_k$ (to be described later). Thus, the search direction is given by

$$\boldsymbol{d}_k = -D_{1,k}\tilde{\boldsymbol{\xi}}_k.$$

In case of negative α, we first compute the convex combination of the "convex" and negative "concave" approximations such that the combination still remains positive definite and then use this combination to compute the search direction. In other words, we compute the smallest $p_k \in [0, 1]$ such that $p_k D_{1,k} - (1 - p_k)D_{2,k}$ is positive definite. Since diagonal matrices are used, this value is very easy to compute. The search direction is then computed by the formula

$$\boldsymbol{d}_k = -(p_k D_{1,k} - (1 - p_k)D_{2,k})\tilde{\boldsymbol{\xi}}_k. \tag{26}$$

When the search direction is computed, we next compute a new auxiliary point: $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$. A necessary condition for a *serious step* to be taken is to have

$$f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L w_k, \tag{27}$$

where $\varepsilon_L^k \in (0, 1/2)$ is a given descent parameter and $w_k > 0$ represents the desirable amount of descent of f at $\boldsymbol{x}_k$. If condition (27) is satisfied, we set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ and a serious step is taken. Note that in the case of a serious

step we consider the current "convex approximation" to be good enough and continue with this metric even if the linearization error was negative.

If the condition (27) is not satisfied, a *null step* occurs. In null steps, we search for a scalar $t \in (0, 1]$ such that $\boldsymbol{\xi}_{k+1}^t = \nabla f_1(\boldsymbol{x}_k + t\boldsymbol{d}_k) - \boldsymbol{\xi}_{2,k+1}^t$, with $\boldsymbol{\xi}_{2,k+1}^t \in \partial f_2(\boldsymbol{x}_k + t\boldsymbol{d}_k)$, satisfies the condition

$$-\beta_{k+1} + \boldsymbol{d}_k^T \boldsymbol{\xi}_{k+1}^t \geq -\varepsilon_R w_k, \tag{28}$$

where $\varepsilon_R \in (\varepsilon_L, 1/2)$ is a given parameter and β_{k+1} is the subgradient locality measure [32, 33], similar to bundle methods. That is,

$$\beta_{k+1} = \max\left\{|f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\boldsymbol{d}_k) + t(\boldsymbol{\xi}_{k+1}^t)^T \boldsymbol{d}_k|, \gamma\|t\boldsymbol{d}_k\|^2\right\}, \tag{29}$$

where $\gamma \geq 0$ is a distance measure parameter supplied by the user. This kind of t always exists [28]. In the case of a null step, we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ however, information about the objective function is increased as we utilize the auxiliary point $\boldsymbol{y}_{k+1}^t = \boldsymbol{x}_k + t\boldsymbol{d}_k$ and the corresponding auxiliary subgradient $\boldsymbol{\xi}_{k+1}^t \in \partial f(\boldsymbol{y}_{k+1}^t)$ in the computation of next aggregate values.

To ensure the global convergence of the DCD-Bundle, we have to assume that the matrices $D_{1,k}$ and $D_{2,k}$ are bounded. This assumption is trivially satisfied according to the diagonal update formula. In addition, the condition

$$\tilde{\boldsymbol{\xi}}_k^T D_{1,k} \tilde{\boldsymbol{\xi}}_k \leq \tilde{\boldsymbol{\xi}}_k^T D_{1,k-1} \tilde{\boldsymbol{\xi}}_k \tag{30}$$

has to be satisfied each time when more than one consecutive null steps occur. In the DCD-Bundle this is guaranteed simply by *skipping the convex updates*.

Aggregation. The aggregation procedure used in the DCD-Bundle is quite similar to that of the original LMBM [23, 24]. We determine multipliers $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function

$$\begin{aligned}\varphi(\lambda_1, \lambda_2, \lambda_3) = \; & (\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1}^t + \lambda_3 \tilde{\boldsymbol{\xi}}_k)^T D_{1,k} (\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1}^t + \lambda_3 \tilde{\boldsymbol{\xi}}_k) \\ & + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k),\end{aligned} \tag{31}$$

where we denoted by $\boldsymbol{\xi}_i = \nabla f_{1,i} - \boldsymbol{\xi}_{2,i}$ (possible with upper index t) and m is the index after the latest serious step. Set

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1}^t + \lambda_3^k \tilde{\boldsymbol{\xi}}_k, \tag{32}$$

$$\tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \tag{33}$$

Algorithms. Now we present the algorithm DCD-Bundle to compute a Clarke stationary point of the clustering problems.

Algorithm 5.1. DCD-Bundle.

Input: Select positive line search parameters $\varepsilon_L \in (0, 1/2)$ and $\varepsilon_R \in (\varepsilon_L, 1)$ and the distance measure parameter $\gamma > 0$. Choose the final accuracy tolerance $\varepsilon > 0$, safeguard parameters $\mu_{max} > \mu_{min} > 0$, and the number of stored corrections $m_c \geq 1$.

Step 0: (*Initialization*) Choose a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$. Set $D_{1,1} = I$. Compute $f(\boldsymbol{x}_1)$, $\nabla f_{1,1} = \nabla f_1(\boldsymbol{x}_1)$, and $\boldsymbol{\xi}_{2,1} \in \partial f_2(\boldsymbol{x}_1)$. Set the iteration counter $k = 1$.

Step 1: (*Serious Step Initialization*) Set the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k = \nabla f_{1,k} - \boldsymbol{\xi}_{2,k}$ and the aggregate subgradient locality measure $\tilde{\beta}_k = 0$. Set an index for the serious step $m = k$.

Step 2: (*"Convex Direction"*) Compute

$$\boldsymbol{d}_k = -D_{1,k}\tilde{\boldsymbol{\xi}}_k.$$

Step 3: (*Stopping Criterion*) Calculate $w_k = \tilde{\boldsymbol{\xi}}_k^T D_{1,k}\tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k$. If $w_k < \varepsilon$, then stop with $\boldsymbol{x}_k$ as the final solution.

Step 4: (*Auxiliary Step*) Evaluate

$$\begin{aligned} \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{d}_k, \\ \nabla f_{1,k+1} &= \nabla f_1(\boldsymbol{y}_{k+1}), \text{ and} \\ \boldsymbol{\xi}_{2,k+1} &\in \partial f_2(\boldsymbol{y}_{k+1}). \end{aligned}$$

Set $\boldsymbol{s}_k = \boldsymbol{d}_k$, $\boldsymbol{u}_{1,k} = \nabla f_{1,k+1} - \nabla f_{1,m}, \boldsymbol{u}_k = \boldsymbol{\xi}_{2,k+1} - \boldsymbol{\xi}_{2,m}$ and add these values to S_k, $U_{1,k}$, and $U_{2,k}$, respectively.

Step 5 (*Serious Step*) If

$$f(\boldsymbol{y}_{k+1}) - f(\boldsymbol{x}_k) \leq -\varepsilon_L w_k,$$

then compute $D_{1,k+1}$ using S_k and $U_{1,k}$, set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$, $f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{y}_{k+1})$, and go to Step 1.

Step 6: (*Aggregation*) Compute

$$\alpha_{k+1} = f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + (\nabla f_{1,k+1} - \boldsymbol{\xi}_{2,k+1})^T \boldsymbol{d}_k. \tag{34}$$

Compute $t \in (0, 1]$ such that $\boldsymbol{\xi}_{k+1}^t = \nabla f_1(\boldsymbol{x}_k + t\boldsymbol{d}_k) - \boldsymbol{\xi}_{2,k+1}^t$, with $\boldsymbol{\xi}_{2,k+1}^t \in \partial f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$, satisfies condition (28) with β_{k+1} computed as in (29). Determine multipliers $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function $\varphi(\lambda_1, \lambda_2, \lambda_3)$ given in (31).

Set

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

Step 7: (*Null Step*) If $m = k$, then compute $D_{1,k+1}$ using S_k and $U_{1,k}$. Otherwise, set $D_{1,k+1} = D_{1,k}$. Two cases can occur:

Step 7a: $\alpha_{k+1} \geq 0$ (*Convex Null Step*): Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $k = k+1$ and go to Step 2.

Step 7b: $\alpha_{k+1} < 0$ (*Concave Null Step*): Compute $D_{2,k+1}$ using S_k and $U_{2,k}$. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $k = k+1$.

Step 8: (*"Concave Direction"*) Compute the smallest $p \in (0,1)$ such that the matrix $pD_{1,k} - (1-p)D_{2,k}$ remains positive semidefinite. Compute

$$\boldsymbol{d}_k = -\left(pD_{1,k} - (1-p)D_{2,k}\right)\tilde{\boldsymbol{\xi}}_k,$$

and go to Step 3.

As said before the above algorithm computes the Clarke stationary point of the clustering problem. If the function f_2 is smooth, then this point is also an inf-stationary point. Our main assumption in order to find inf-stationary points is the following.

ASSUMPTION 1. If the subdifferential $\partial f_2(\boldsymbol{x})$ is not a singleton at a point $\boldsymbol{x} \in \mathbb{R}^n$, then we can always compute two subgradients $\boldsymbol{\xi}_2^1, \boldsymbol{\xi}_2^2 \in \partial f_2(\boldsymbol{x})$ such that $\boldsymbol{\xi}_2^1 \neq \boldsymbol{\xi}_2^2$.

This assumption is satisfied for both the clustering and the auxiliary clustering problems as is shown in Section 4.

ALGORITHM 5.2. Finding inf-stationary points.

Step 0: (*Initialization*) Choose a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$, the line search parameter $\varepsilon_T \in (0, 1/2]$, and the final accuracy tolerance $\varepsilon > 0$. Set $j = 1$.

Step 1: (*Clarke Stationary Point*) Apply Algorithm 5.1 starting from the point $\boldsymbol{x}_j$ to find a Clarke stationary point $\boldsymbol{x}^*$ with the optimality tolerance ε.

Step 2: (*Stopping Criterion*) If $\partial f_2(\boldsymbol{x}^*) \subset \nabla f_1(\boldsymbol{x}^*) + B_\varepsilon(0)$ then stop. The point $\boldsymbol{x}^*$ is an inf-stationary point of the problem.

Step 3: (*Descent Direction*) Compute subgradients $\boldsymbol{\xi}_2^1, \boldsymbol{\xi}_2^2 \in \partial f_2(\boldsymbol{x}^*)$ such that

$$r = \max_{i=1,2} \|\nabla f_1(\boldsymbol{x}^*) - \boldsymbol{\xi}_2^i\| \geq \varepsilon,$$

and the direction $\bar{\boldsymbol{u}}_j = -\boldsymbol{v}/\|\boldsymbol{v}\|$, where

$$\boldsymbol{v} = \operatorname{argmax}\left\{\|\nabla f_1(\boldsymbol{x}^*) - \boldsymbol{\xi}_2^i\| \mid i = 1, 2\right\}.$$

Step 4: (*Step Size*) Compute $\boldsymbol{x}_{j+1} = \boldsymbol{x}^* + t_j \bar{\boldsymbol{u}}_j$ where

$$t_j = \operatorname{argmax}\left\{t > 0 \mid f(\boldsymbol{x}^* + t\bar{\boldsymbol{u}}_j) - f(\boldsymbol{x}^*) \leq -\varepsilon_T t r\right\}.$$

Set $j = j+1$ and go to Step 1.

6. Global Convergence

We now study the convergence properties of Algorithms 5.1 and 5.2 given in the previous section. We first prove that a point generated by Algorithm 5.1 is a Clarke stationary point of the problem (24). In order to do this, we assume that the final accuracy tolerance ε is equal to zero. Then we prove that Algorithm 5.2 terminates after a finite number of iterations at an inf-stationary point of the problem. For the simplicity of the presentation from now on, we drop out the index t from $\boldsymbol{y}_k^t$ and $\boldsymbol{\xi}_k^t$ even if $t \neq 1$ and use the notation $\boldsymbol{\xi} = \nabla f_1 - \boldsymbol{\xi}_2$.

The following assumptions are needed to prove the global convergence.

Assumption 2. The level set $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ is bounded for every starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$.

Assumption 3. The objective function f is bounded from below.

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

Remark 6.1. The sequence $(\boldsymbol{x}_k)$ generated by Algorithm 5.1 is bounded by Assumption 2 and the monotonicity of the sequence (f_k) which, in turn, is obtained due to the condition (27) being satisfied for serious steps and the fact that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps. By the local boundedness and the upper semi-continuity of the subdifferential, we obtain the boundedness of subgradients $\boldsymbol{\xi}_k$ and their convex combinations [34]. The matrices D_1 and D_2 are bounded due to the fact that all their components are in a closed interval $[\mu_{min}, \mu_{max}]$. Therefore, the search direction $\boldsymbol{d}_k$ and the sequence $(\boldsymbol{y}_k)$ are also bounded.

Lemma 6.1. *At the kth iteration of Algorithm 5.1, we have*

$$w_k = \tilde{\boldsymbol{\xi}}_k^T D_{1,k} \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \mu_{min} \|\tilde{\boldsymbol{\xi}}_k\|^2,$$

and

$$\beta_{k+1} \geq \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\|^2. \tag{35}$$

Proof. First, we point out that $\tilde{\beta}_k \geq 0$ for all k by equations (29), (33), and Step 1 in Algorithm 5.1. The relations

$$w_k = \tilde{\boldsymbol{\xi}}_k^T D_{1,k} \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \mu_{min} \|\tilde{\boldsymbol{\xi}}_k\|^2$$

follow immediately from Step 3 in Algorithm 5.1 and the lower bound μ_{min} used for the matrices.

Since $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps, $\beta_{k+1} = 0$ and $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| = 0$ for serious steps it follows from (29) that the condition (35) always holds for some $\gamma > 0$. □

LEMMA 6.2. *Suppose that Algorithm 5.1 is not terminated before the kth iteration. Then, there exist numbers* $\lambda^{k,j} \geq 0$ *for* $j = 1, \ldots, k$ *and* $\tilde{\sigma}_k \geq 0$ *such that*

$$(\tilde{\boldsymbol{\xi}}_k, \tilde{\sigma}_k) = \sum_{j=1}^{k} \lambda^{k,j} (\boldsymbol{\xi}_j, \|\boldsymbol{y}_j - \boldsymbol{x}_k\|), \qquad \sum_{j=1}^{k} \lambda^{k,j} = 1, \quad \textit{and} \quad \tilde{\beta}_k \geq \gamma \tilde{\sigma}_k^2.$$

Proof. The proof is similar to that of Lemma 3.2 in [26]. □

LEMMA 6.3. *Let* $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ *be a given point and suppose that there exist vectors* $\bar{\boldsymbol{g}}$, $\bar{\boldsymbol{\xi}}_i$, $\bar{\boldsymbol{y}}_i$, *and numbers* $\bar{\lambda}_i \geq 0$ *for* $i = 1, \ldots, l$, $l \geq 1$, *such that*

$$\begin{aligned} &(\bar{\boldsymbol{g}}, 0) = \sum_{i=1}^{l} \bar{\lambda}_i (\bar{\boldsymbol{\xi}}_i, \|\bar{\boldsymbol{y}}_i - \bar{\boldsymbol{x}}\|), \\ &\bar{\boldsymbol{\xi}}_i \in \partial f(\bar{\boldsymbol{y}}_i), \quad i = 1, \ldots, l, \qquad \textit{and} \\ &\sum_{i=1}^{l} \bar{\lambda}_i = 1. \end{aligned}$$

Then $\bar{\boldsymbol{g}} \in \partial f(\bar{\boldsymbol{x}})$.

Proof. See the proof of Lemma 3.3 in [26]. □

THEOREM 6.4. *If Algorithm 5.1 terminates at the kth iteration, then the point* $\boldsymbol{x}_k$ *is Clarke stationary for* f.

Proof. If Algorithm 5.1 terminates at Step 3, then the fact $\varepsilon = 0$ implies that $w_k = 0$. Thus, $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{0}$ and $\tilde{\beta}_k = \tilde{\sigma}_k = 0$ by Lemma 6.1 and Lemma 6.2.

Using Lemma 6.3 with

$$\begin{aligned} &\bar{\boldsymbol{x}} = \boldsymbol{x}_k, \qquad l = k, \qquad \bar{\boldsymbol{g}} = \tilde{\boldsymbol{\xi}}_k, \\ &\bar{\boldsymbol{\xi}}_i = \boldsymbol{\xi}_i, \qquad \bar{\boldsymbol{y}}_i = \boldsymbol{y}_i, \quad \bar{\lambda}_i = \lambda^{k,i} \quad \text{for } i \leq k, \end{aligned}$$

and also Lemma 6.2 we obtain that $\boldsymbol{0} = \tilde{\boldsymbol{\xi}}_k \in \partial f(\boldsymbol{x}_k)$ and, therefore, $\boldsymbol{x}_k$ is Clarke stationary for f. □

From now on, we suppose that Algorithm 5.1 does not terminate, that is, $w_k > 0$ for all k.

LEMMA 6.5. *Suppose that the level set* $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ *is bounded. If there exist a point* $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ *and an infinite set* $\mathcal{K} \subset \{1, 2, \ldots\}$ *such that* $(\boldsymbol{x}_k)_{k \in \mathcal{K}} \to \bar{\boldsymbol{x}}$ *and* $(w_k)_{k \in \mathcal{K}} \to 0$, *then* $\boldsymbol{0} \in \partial f(\bar{\boldsymbol{x}})$.

Proof. The proof is similar to the proof of Lemma 3.4 in [26]. □

LEMMA 6.6. *Suppose that the level set* $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ *is bounded, the number of serious steps is finite, and the last serious step occurred at the iteration* $m-1$*. Then*

$$\tilde{\boldsymbol{\xi}}_{k+1}^T D_{1,k+1} \tilde{\boldsymbol{\xi}}_{k+1} = \tilde{\boldsymbol{\xi}}_{k+1}^T D_{1,k} \tilde{\boldsymbol{\xi}}_{k+1} \qquad \textit{and} \tag{36}$$

$$\operatorname{tr}(D_{1,k}) \leq \mu_{max} n \tag{37}$$

for all $k > m$*, where* $\operatorname{tr}(D_{1,k})$ *denotes the trace of the matrix* $D_{1,k}$*.*

Proof. If $m < k$, then we always have $D_{1,k+1} = D_{1,k}$ due to Step 7 of Algorithm 5.1. Thus, the condition (36) is valid. Furthermore, in each case for all k we have

$$\begin{aligned} \operatorname{tr}(D_{1,k}) - \mu_{max} n &= \operatorname{tr}(D_{1,k}) - \mu_{max} \operatorname{tr}(I) \\ &= \operatorname{tr}(D_{1,k}) - \operatorname{tr}(\mu_{max} I) \\ &= \operatorname{tr}(D_{1,k} - \mu_{max} I) \\ &\leq 0 \end{aligned}$$

since $D_{1,k}$ is a diagonal matrix with the largest diagonal element equal to μ_{max}. Therefore, the condition (37) is also valid for all $k > m$. □

LEMMA 6.7. *Suppose that the level set* $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ *is bounded, the number of serious steps is finite, and the last serious step occurred at the iteration* $m-1$*. Then, the point* $\boldsymbol{x}_m$ *is Clarke stationary for* f*.*

Proof. From (31), (32), (33), Lemma 6.1, and Lemma 6.6 we obtain

$$\begin{aligned} w_{k+1} &= \tilde{\boldsymbol{\xi}}_{k+1}^T D_{1,k+1} \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\ &= \tilde{\boldsymbol{\xi}}_{k+1}^T D_{1,k} \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\ &= \varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) \\ &\leq \varphi(0,0,1) \\ &= \tilde{\boldsymbol{\xi}}_k^T D_{1,k} \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k \\ &= w_k \end{aligned} \tag{38}$$

for $k \geq m$.

Let us denote $D_{1,k} = W_k^T W_k$. Then, the function φ (see (31)) can be given in the form

$$\varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) = \|\lambda_1^k W_k \boldsymbol{\xi}_m + \lambda_2^k W_k \boldsymbol{\xi}_{k+1} + \lambda_3^k W_k \tilde{\boldsymbol{\xi}}_k\|^2 + 2(\lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k).$$

From (38) we obtain the boundedness of the sequences (w_k), $(W_k \tilde{\boldsymbol{\xi}}_k)$, and $(\tilde{\beta}_k)$. Furthermore, Lemma 6.6 assures the boundedness of (D_k) and (W_k). By Lemma 6.5, we obtain the boundedness of $(\boldsymbol{y}_k)$, $(\boldsymbol{\xi}_k)$, and $(W_k \boldsymbol{\xi}_{k+1})$.

Now, by noticing that $-\beta_{k+1} + \boldsymbol{\xi}_{k+1}^T \boldsymbol{d}_k \geq -\varepsilon_R w_k$ in null steps the last part of the proof proceeds similar to the proof (part (ii)) of Lemma 3.6 in [26]. □

THEOREM 6.8. *Suppose that the level set* $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ *is bounded. Then, every accumulation point of the sequence* $(\boldsymbol{x}_k)$ *is Clarke stationary for* f.

Proof. Let $\bar{\boldsymbol{x}}$ be an accumulation point of $(\boldsymbol{x}_k)$, and let $\mathcal{K} \subset \{1, 2, \ldots\}$ be an infinite set such that $(\boldsymbol{x}_k)_{k\in\mathcal{K}} \to \bar{\boldsymbol{x}}$. In view of Lemma 6.7, we can restrict our consideration to the case where the number of serious steps is infinite. We denote

$$\mathcal{K}' = \{k \mid \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k \text{ and there exists } i \in \mathcal{K},\ i \leq k \text{ such that } \boldsymbol{x}_i = \boldsymbol{x}_k\}.$$

Obviously, $\mathcal{K}'$ is infinite and $(\boldsymbol{x}_k)_{k\in\mathcal{K}'} \to \bar{\boldsymbol{x}}$. The continuity of f implies that $(f(\boldsymbol{x}_k))_{k\in\mathcal{K}'} \to f(\bar{\boldsymbol{x}})$ and, thus, $f(\boldsymbol{x}_k) \downarrow f(\bar{\boldsymbol{x}})$ due to the monotonicity of the sequence $(f(\boldsymbol{x}_k))$ followed from the descent step condition (27). Using the condition (27) and the fact that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ in null steps, we obtain:

$$0 \leq \varepsilon_L w_k \leq f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \to 0 \qquad \text{for } k \geq 1. \tag{39}$$

Thus, $(w_k)_{k\in\mathcal{K}'} \to 0$ and $(\boldsymbol{x}_k)_{k\in\mathcal{K}'} \to \bar{\boldsymbol{x}}$. Then Lemma 6.5 implies that $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$. □

REMARK 6.2. If we choose $\varepsilon > 0$, then Algorithm 5.1 terminates in a finite number of steps. In addition, the proofs remain correct if we have f_1 convex nonsmooth and f_2 convex smooth, or if both f_1 and f_2 are convex nonsmooth functions with a condition that we can compute the subgradient $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ at any x.

Now we prove that Algorithm 5.2 terminates after a finite number of iterations at an inf-stationary point of the problem (24). In addition to Assumptions 1 – 3 we need the following assumption.

ASSUMPTION 4. The gradient $\nabla f_1 : \mathbb{R}^n \to \mathbb{R}^n$ of the function f_1 satisfies Lipschitz condition.

At the end of this section, we will prove that Assumption 4 is satisfied for the first DC component f_{k1} of the clustering function f_k and for the first DC component $\bar{f}_{k1}$ of the auxiliary clustering function $\bar{f}_k$.

LEMMA 6.9. *Assume that the objective function is bounded from below and the gradient* $\nabla f_1 : \mathbb{R}^n \to \mathbb{R}^n$ *satisfies Lipschitz condition. Then Algorithm 5.2 terminates after a finite number of iterations at an* inf*-stationary point of Problem (24).*

Proof. For simplicity assume that at the j-th iteration $\bar{\boldsymbol{u}}_j = -\|\boldsymbol{v}\|^{-1}\boldsymbol{v}$ and $\boldsymbol{v} = \nabla f_1(\boldsymbol{x}_j) - \boldsymbol{\xi}_2^1$, where $\boldsymbol{\xi}_2^1 \in \partial f_2(\boldsymbol{x}_j)$. The mean value theorem implies that for some $\sigma_j \in (0, 1)$

$$\begin{aligned} f(\boldsymbol{x}_j + t\bar{\boldsymbol{u}}_j) - f(\boldsymbol{x}_j) =& [f_1(\boldsymbol{x}_j + t\bar{\boldsymbol{u}}_j) - f_1(\boldsymbol{x}_j)] - [f_2(\boldsymbol{x}_j + t\bar{\boldsymbol{u}}_j) - f_2(\boldsymbol{x}_j)] \\ \leq& t\bar{\boldsymbol{u}}_j^T \nabla f_1(\boldsymbol{x}_j + t\sigma_j\bar{\boldsymbol{u}}_j) - t\bar{\boldsymbol{u}}_j^T \boldsymbol{\xi}_2^1 \\ \leq& t\bar{\boldsymbol{u}}_j^T \left(\nabla f_1(\boldsymbol{x}_j) - \boldsymbol{\xi}_2^1\right) + t\bar{\boldsymbol{u}}_j^T \left(\nabla f_1(\boldsymbol{x}_j + t\sigma_j\bar{\boldsymbol{u}}_j) - \nabla f_1(\boldsymbol{x}_j)\right). \end{aligned}$$

Let $L > 0$ be a Lipschitz constant of the gradient ∇f_1. Then

$$\|f_1(\boldsymbol{x}_j + t\sigma_j \bar{\boldsymbol{u}}_j) - \nabla f_1(\boldsymbol{x}_j)\| \leq Lt\sigma_j \|\bar{\boldsymbol{u}}_j\| = Lt\sigma_j.$$

Since $\|\boldsymbol{v}\| \geq \varepsilon$, we obtain

$$\begin{aligned} f(\boldsymbol{x}_j + t\bar{\boldsymbol{u}}_j) - f(\boldsymbol{x}_j) \leq & t\bar{\boldsymbol{u}}_j^T \left(\nabla f_1(\boldsymbol{x}_j) - \boldsymbol{\xi}_2^1\right) + Lt^2\sigma_j \\ = & -t\|\nabla f_1(\boldsymbol{x}_j) - \boldsymbol{\xi}_2^1\| + Lt^2\sigma_j \\ < & t(-r + Lt). \end{aligned}$$

For $\bar{t} = r/2L$ we have

$$f(\boldsymbol{x}_j + \bar{t}\bar{\boldsymbol{u}}_j) - f(\boldsymbol{x}_j) < -\frac{r^2}{4L} \leq -\varepsilon_T \bar{t} r \leq -\varepsilon_T \bar{t}\varepsilon.$$

This means that at each iteration we have $t_j \geq \bar{t} \geq \varepsilon/2L$ and the function f decreases by at least $\varepsilon_T \varepsilon^2/2L > 0$ at each iteration. Since the function is bounded from below, Algorithm 5.2 must stop after a finite number of iterations. □

Finally, we show that the gradients of the functions $\bar{f}_{k1}$ and f_{k1} are Lipschitz.

LEMMA 6.10. *The gradient of the function $\bar{f}_{k1}$ satisfies Lipschitz condition with the constant $L = 2$.*

Proof. It follows from (14) that for any $y^1, y^2 \in \mathbb{R}^n$

$$\nabla \bar{f}_{k1}(y^1) - \nabla \bar{f}_{k1}(y^2) = 2(y^1 - y^2).$$

Then $\|\nabla \bar{f}_{k1}(y^1) - \nabla \bar{f}_{k1}(y^2)\| = 2\|y^1 - y^2\|$ that is the gradient $\nabla \bar{f}_{k1}$ satisfies the Lipschitz condition on $\mathbb{R}^n$ with the constant $L = 2$. □

LEMMA 6.11. *The gradient of the function f_{k1} satisfies Lipschitz condition with the constant $L = 2$.*

Proof. The proof is similar to that of Lemma 6.10. □

7. Incremental algorithm

In this section we present an incremental algorithm for solving Problem (7) using the DC approach. Since the problem (7) is nonconvex, the use of an incremental approach allows to generate starting cluster centers and to apply a local method to find high quality solutions to the clustering problem. We apply an algorithm introduced in [38] to generate starting cluster centers. The incremental algorithm proceeds as follows.

ALGORITHM 7.1. An incremental clustering algorithm.

Step 1: (*Initialization*) Compute the center $\boldsymbol{x}_1 \in \mathbb{R}^n$ of the set A. Set $l = 1$.

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

Step 3: (*Computation of a set of starting points for the auxiliary clustering problem*) Apply the procedure from [38] to find the set $S_1 \subset \mathbb{R}^n$ of starting points for solving the auxiliary clustering problem (12) with $k = l$.

Step 4: (*Computation of a set of starting points for the l-th cluster center*). Apply Algorithm 5.2 to solve Problem (12) starting from each point $\boldsymbol{y} \in S_1$. This algorithm generates a finite point set $S_2 \subset \mathbb{R}^n$.

Step 5: (*Computation of a set of cluster centers*) For each $\bar{\boldsymbol{y}} \in S_2$ apply Algorithm 5.2 to solve Problem (7) starting from the point $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}, \bar{\boldsymbol{y}})$ and to 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.

Step 6: (*Computation of the best solution*). 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 $f_l(\bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_l) = f_l^{min}$.

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

In addition to the k-partition problem, Algorithm 7.1 solves also all intermediate l-partition problems where $l = 1, \ldots, k-1$. Algorithm 5.2 is applied to solve both the clustering and the auxiliary clustering problems at each iteration of Algorithm 7.1. In its turn, Algorithm 5.2 uses Algorithm 5.1 to find a Clarke stationary point of these problems. Together, these three algorithms are called DCD-Bundle method.

8. Numerical Experiments

To test the new DCD-Bundle method `DCD-Bundle` we compared it to `DCClust` introduced in [21] and the classical multi-start k-means `MS-KM` [42]. Solvers `DCD-Bundle` and `DCClust` are both implemented in Fortran 95 while `MS-KM` is implemented in Fortran 77. All the software were 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 algorithms `DCD-Bundle` and `DCClust` as well as the data sets used in our experiments can be downloaded from http://napsu.karmitsa.fi/clustering/.

We used ten large real world data sets in our experiments. The brief description of these data sets is given in Table 1. The more detailed description can be found in [43] 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 large (128) and the number of data points ranges from tens of thousands (smallest 13 910) to hundred of thousands (largest 434 874).

`DCD-Bundle` and `DC-Clust` 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 randomized multistart scheme for starting points. Thus, it does not give any intermediate results. For comparison purposes, we made different runs for different number of clusters. For `MS-KM`, the maximum number of different starting points was always kept big enough, however, we limited the computational time of the solver to be approximately twice that of used by `DCD-Bundle`. We also stopped the run if the wall clock was more than 24 hours without any progress. More specifically, if after 24 hours `MS-KM` was still computing clusters from the first starting point. Note that, the maximum time spent in any data set by the other two solvers was less than nine hours. We report the number of used starting points for `MS-KM` instead of CPU time.

Results are given in Tables 2–11, where we use 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 (8) (multiplied with m) for the corresponding number of clusters. We have used the f_{best} value given in [21] (for those data sets that were used also in [21]) unless we got better value in our experiments. If the value obtained in our experiments was better than that of [21] we have marked it with an asterisk.
- E_A is the relative 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 cluster function value obtained by an algorithm A.
- cpu is the used CPU time in seconds.
- n_{sp} is the number of starting points used (with `MS-KM`).

In addition, the dependence of computational time of algorithms as well as the number of distance function calculations on the number of clusters are given in Figures 1 and 2.

The data sets can be divided into three groups. The first group contains data sets with a 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 4, 8, 10, and 11 demonstrate that in these data sets the accuracy of the solvers was quite similar and the solutions found were at least close to the best known solutions. The only remarkable exception here is `MS-KM` in Skin Segmentation data set with $k > 10$, where the results were far away from global minima (see Table 10). `MS-KM` also had some difficulties in D15112 data set with $k > 15$, in Pla85900 data set with $k = 2$ and $k = 5$, and in 3D

Table 1: The brief description of data sets

Data sets	No. instances	No. attributes	References
Gas Sensor Array Drift	13910	128	[44]
EEG Eye State	14980	14	[43]
D15112	15112	2	[45]
Online News Popularity	39644	58	[46]
KEGG Metabolic	53413	20	[47]
Shuttle Control	58000	9	[43]*
Pla85900	85900	2	[45]
MiniBooNE particle identification	130064	50	[43]
Skin Segmentation	245057	3	[48]
3D Road Network	434874	3	[49]

*Thanks to NASA.

Road Network data set with $k = 25$. In 3D Road Network data set, the value of the clustering function was clearly improved from that given in [21] with `DCD-Bundle` (1.81 % with 25 clusters).

The second group consists of data sets with medium number of attributes (9–20, see Table 1). They are EEG Eye State, KEGG Metabolic, and Shuttle Control data sets. Results in these data sets are given in Tables 3, 6, and 7, respectively. Results show that both `DCD-Bundle` and `DCClust` can find the near best known solution for almost all k in EEG Eye State, and KEGG Metabolic. Here, `DCD-Bundle` had difficulties with $k = 20$ in EEG Eye State while `DCClust` had difficulties with $k = 20$ in KEGG Metabolic. In addition, results in Shuttle Control data set show that `DCClust` had difficulties with 20 clusters. Here, `DCD-Bundle` had difficulties already with 15 clusters and they continued when the number of clusters was increased. In Shuttle Control data set, most points are very close to each other and clusters are not well separated when their number is large. Thus, this failure in accuracy with `DCD-Bundle` does not seem to be fatal. `MS-KM` failed to find even near best known solutions in almost all the cases.

The last group contains data sets with a large number of attributes (50–128, see Table 1). They are Gas Sensor Array Drift, Online News Popularity, and MiniBooNE particle identification. The results can be found in Tables 2, 5, and 9, respectively. Again, both `DCD-Bundle` and `DCClust` found the near best known solution for almost all k while `MS-KM` had some serious problems. Moreover, with MiniBooNE particle identification data set `MS-KM` only solved the problem with two clusters within given time limit of 24 hours. The error obtained was enormous and it used almost 2000 seconds to compute the result, while `DCD-Bundle` solved the same problem accurately in less than five seconds.

The new method `DCD-Bundle` was clearly faster than `DCClust` with data sets from the last group, that is, data sets with a large number of attributes (Sensor Array Drift, Online News Popularity, and MiniBooNE particle identification, see Figures 1(a), 1(d), and 1(h)). In addition, `DCD-Bundle` was more efficient than `DCClust` with data sets including the relatively small number of instances regardless of the number of attributes (EEG Eye State and D15112, see Figures 1(b) and 1(c)), and with data sets with a large number of instances but a small

number of attributes (Pla85900, Skin Segmentation, and 3D Road Network, see Figures 1(g), 1(i), and 1(j)).

When the number of instances was relatively large ($> 50\,000$) and the number of attributes was not small (that is, KEGG Metabolic and Shuttle Control, see Figures 1(e) and 1(f)) `DCD-Bundle` used about the same or a bit more cpu-time than `DCClust`. The needed number of distance function computations versus the number of clusters revealed similar trends to those of computational times with `DCD-Bundle` and `DCClust` while `MS-KM` usually used significantly less distance function evaluations than the other algorithms (see Figure 2). However, the procedure used in `MS-KM` is completely different from that of the other two and the sole number of distance function calculations does not give a right impression of the computational burden. As said at the beginning of the section, the computation times of `MS-KM` were always at least twice that used by `DCD-Bundle`. In large data sets poor results obtained by `MS-KM` are mainly due to the fact that this algorithm had time to solve the clustering problem using only one starting point.

Table 2: Summary of the results with Gas Sensor Array Drift ($\times 10^{13}$).

k	f_{best}	`DCD-Bundle`		`DCClust`		`MS-KM`	
		E_A	*cpu*	E_A	*cpu*	E_A	n_{sp}
2	7.91186	0.00	17.76	0.00	24.78	0.00	14
3	5.02412	0.00	49.42	0.00	64.00	0.00	42
5	3.22394	0.10	151.86	0.10	165.43	0.00	146
10	1.65524	0.01	467.90	0.00	510.62	2.68	531
15	1.13801	0.36	870.13	0.36	988.77	12.10	782
20	0.87916	2.61	1211.98	0.62	1514.49	31.32	899
25	0.72348	0.67	1534.75	0.47	2053.33	22.04	1002

Table 3: Summary of the results with EEG Eye State ($\times 10^{8}$).

k	f_{best}	`DCD-Bundle`		`DCClust`		`MS-KM`	
		E_A	*cpu*	E_A	*cpu*	E_A	n_{sp}
2	8178.13809	0.00	0.15	0.00	0.44	0.00	1
3	1833.88058	0.00	0.17	0.00	0.84	227.91	1
5	1.33858	0.00	0.76	0.00	2.66	449091.75	1
10	0.45669	0.00	9.07	0.00	18.63	193.40	6
15	0.34653	0.28	23.14	0.26	49.07	4.78	21
20	0.28987	1.53	40.91	0.96	88.35	2.83	25
25	0.25989*	0.04	59.31	0.00	137.86	2.12	54

Table 4: Summary of the results with D15112 ($\times 10^{11}$).

k	f_{best}	DCD-Bundle		DCClust		MS-KM	
		E_A	cpu	E_A	cpu	E_A	n_{sp}
2	3.68403	0.00	0.79	0.00	0.93	0.00	2
3	2.53240	0.00	1.54	0.00	1.92	0.00	2
5	1.32707	0.00	2.55	0.00	3.21	0.00	6
10	0.64491*	0.62	5.58	0.62	8.00	0.00	17
15	0.43136*	0.25	10.05	0.25	18.18	0.00	22
20	0.32177	0.03	16.03	0.00	32.58	2.69	23
25	0.25309	0.00	29.37	0.00	53.89	3.80	39

Table 5: Summary of the results with Online News Popularity ($\times 10^{14}$).

k	f_{best}	DCD-Bundle		DCClust		MS-KM	
		E_A	cpu	E_A	cpu	E_A	n_{sp}
2	9.53913	0.00	64.20	0.00	97.81	0.00	10
3	5.91077	0.00	167.06	0.00	253.04	0.13	14
5	3.09885	0.00	364.86	0.00	541.38	0.35	52
10	1.17247	0.00	1043.89	0.00	1499.72	82.56	146
15	0.77637	0.97	1681.39	0.00	2653.41	33.17	151
20	0.59809	0.00	2880.16	0.00	4292.31	34.48	262
25	0.49616	0.82	4782.29	0.00	6011.97	45.11	309

Table 6: Summary of the results with KEGG Metabolic ($\times 10^{8}$).

k	f_{best}	DCD-Bundle		DCClust		MS-KM	
		E_A	cpu	E_A	cpu	E_A	n_{sp}
2	11.38530	0.00	5.23	0.00	6.65	18.85	1
3	4.90060	0.00	13.41	0.00	18.25	124.79	1
5	1.88367	0.00	84.22	0.06	66.35	0.00	1
10	0.63515	0.30	388.59	0.21	358.11	30.34	7
15	0.35393*	0.00	747.86	0.26	719.32	98.14	11
20	0.25027*	0.00	1086.82	2.04	1122.56	161.39	18
25	0.19289	0.94	1632.00	0.75	1549.19	221.85	29

Table 7: Summary of the results with Shuttle Control ($\times 10^8$).

k	f_{best}	DCD-Bundle		DCClust		MS-KM	
		E_A	cpu	E_A	cpu	E_A	n_{sp}
2	21.34329.	0.00	0.40	0.00	1.19	51.13	1
3	10.85415	0.00	0.93	0.00	3.33	100.52	1
5	7.24479	0.00	39.27	0.24	21.26	38.73	1
10	2.83216	0.20	138.48	0.33	78.65	130.62	3
15	1.53154	3.78	531.97	0.37	290.63	200.94	27
20	1.06012	2.33	652.78	1.16	516.43	268.71	45
25	0.78727	4.64	778.04	0.13	774.88	326.43	44

Table 8: Summary of the results with Pla85900 ($\times 10^{15}$).

k	f_{best}	DCD-Bundle		DCClust		MS-KM	
		E_A	cpu	E_A	cpu	E_A	n_{sp}
2	3.74908	0.00	16.27	0.00	15.63	1.44	1
3	2.28057	0.00	31.52	0.00	30.05	0.00	1
5	1.33972	0.00	62.17	0.00	60.87	2.77	1
10	0.68294	0.00	141.46	0.00	145.04	0.55	3
15	0.46249	0.00	231.45	0.00	254.82	0.00	4
20	0.34988	0.52	339.97	0.52	383.46	0.03	7
25	0.28265	0.00	450.94	0.00	529.44	0.06	17

Table 9: Summary of the results with MiniBooNE particle identification ($\times 10^{10}$).

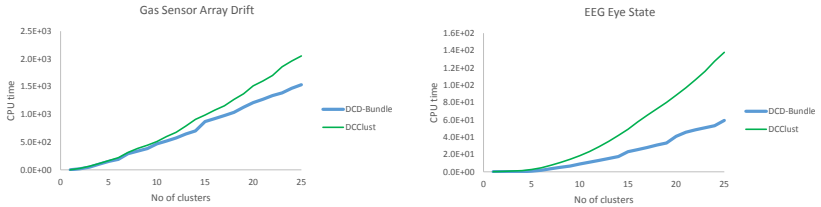
k	f_{best}	DCD-Bundle		DCClust		MS-KM	
		E_A	cpu	E_A	cpu	E_A	n_{sp}
2	8.92236	0.00	4.25	0.00	13.26	286908.08	1
3	5.22601	0.00	246.56	0.00	258.50	–	–
5	1.82252	0.00	1245.83	0.00	1209.26	–	–
10	0.92406	0.00	6209.16	0.02	7196.27	–	–
15	0.63506	0.00	12229.20	0.02	15537.69	–	–
20	0.50863	0.00	18696.58	0.02	24363.40	–	–
25	0.44425	0.00	25132.90	0.02	30855.61	–	–

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

k	f_{best}	DCD-Bundle		DCClust		MS-KM	
		E_A	cpu	E_A	cpu	E_A	n_{sp}
2	1.32236	0.00	167.97	0.00	166.33	0.00	1
3	0.89362	0.00	303.41	0.00	290.68	0.00	1
5	0.50203	0.00	544.13	0.00	517.17	3.28	1
10	0.25121	0.00	1094.79	0.00	1076.48	0.00	4
15	0.16964	0.18	1646.77	0.18	1640.04	13.09	7
20	0.12770	0.15	2164.16	0.15	2217.97	26.51	10
25	0.10299	0.01	2719.04	0.01	2844.37	32.61	7

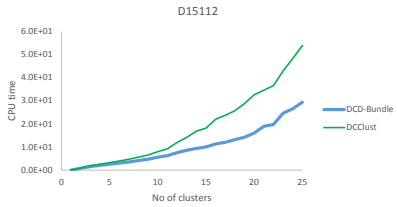
Table 11: Summary of the results with 3D Road Network ($\times 10^6$).

k	f_{best}	DCD-Bundle		DCClust		MS-KM	
		E_A	cpu	E_A	cpu	E_A	n_{sp}
2	49.13298	0.00	398.79	0.00	446.96	0.00	1
3	22.77818	0.03	897.46	0.00	1004.36	0.00	1
5	8.82574	0.00	1767.64	0.00	2005.20	0.00	1
10	2.56662*	0.00	3994.52	0.20	4332.89	0.01	1
15	1.27069*	0.00	6354.14	0.00	6727.42	0.00	1
20	0.80869*	0.00	8679.32	0.00	9300.88	0.01	1
25	0.59259*	0.00	11348.30	3.77	12478.87	1.61	1

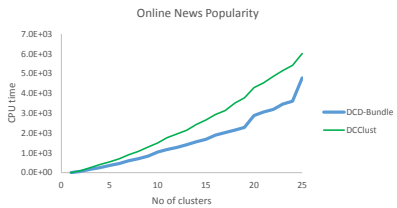


(a) Gas Sensor Array Drift

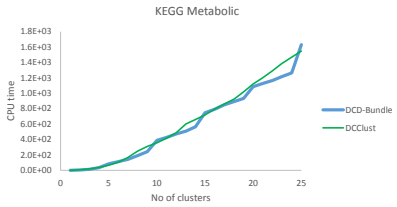
(b) EEG Eye State



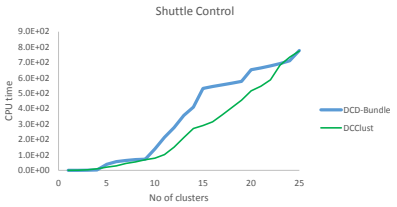
(c) D15112



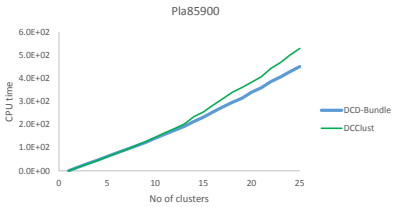
(d) Online News Popularity



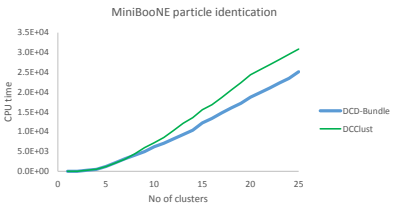
(e) KEGG Metabolic



(f) Shuttle Control

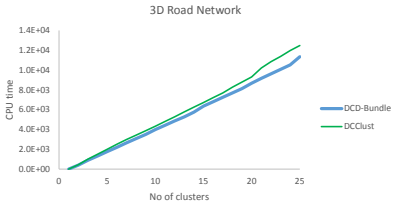


(g) Pla85900



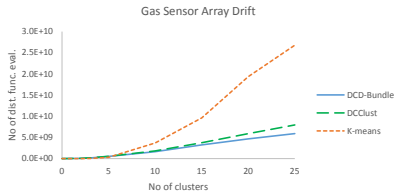
(h) MiniBoone particle identification

(i) Skin Segmentation



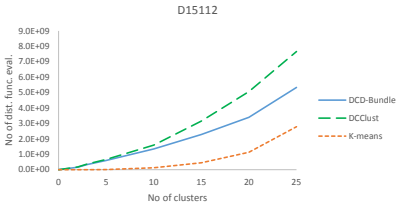
(j) 3D Road Network

Figure 1: The CPU time vs the number of clusters.

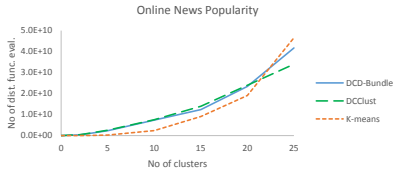


(a) Gas Sensor Array Drift

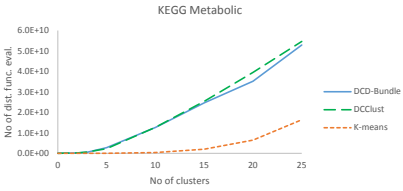
(b) EEG Eye State



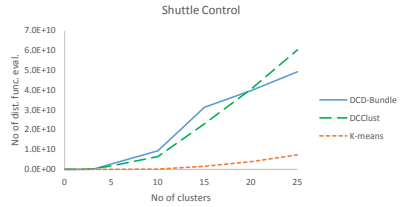
(c) D15112



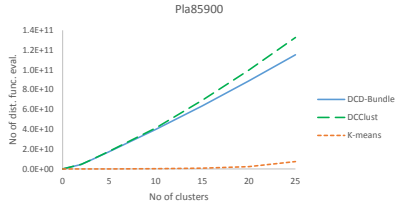
(d) Online News Popularity



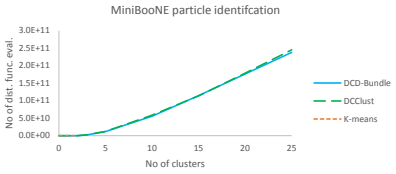
(e) KEGG Metabolic



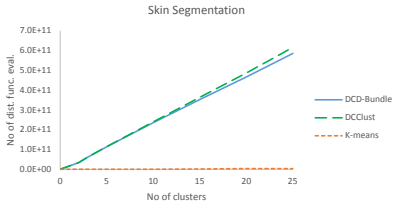
(f) Shuttle Control



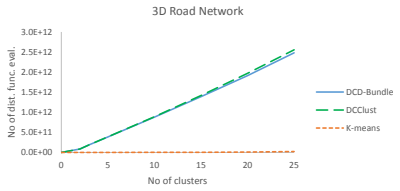
(g) Pla85900



(h) MiniBoone particle identification



(i) Skin Segmentation



(j) 3D Road Network

Figure 2: The number of distance function calculations vs the number of clusters.

9. Conclusions

In this paper a new DCD-Bundle method for solving the minimum sum-of-squares clustering problem is introduced. This problem is formulated as a nonsmooth DC programming problem. The proposed method explicitly utilizes this DC representation.

The DCD-Bundle method consists of three different algorithms: an incremental algorithm (Algorithm 7.1) is used to solve clustering problems globally. At each iteration of this algorithm an algorithm for finding an inf-stationary point (i.e. Algorithm 5.2) is used to solve both the clustering and the auxiliary clustering problems. In its turn, this algorithm utilizes a DCD-Bundle -algorithm (Algorithm 5.1) to find a Clarke stationary point of these problems. We have proved that the proposed DCD-Bundle method converges to an inf-stationary point of the clustering problem.

The DCD-Bundle method was tested using real world data sets with the number of data points ranging from tens of thousands to hundred of thousands. We can conclude that the DCD-Bundle method is 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).

References

[1] G. Diehr, Evaluation of a branch and bound algorithm for clustering, SIAM J. Scientific and Statistical Computing 6 (2) (1985) 268–284.

[2] N. Krislock, J. Malick, F. Roupin, Computational results of a semidefinite branch-and-bound algorithm for k-cluster, Computers & Operations Research, 66 (2016) 153–159.

[3] O. du Merle, P. Hansen, B. Jaumard, N. Mladenovic, An interior point algorithm for minimum sum-of-squares clustering, SIAM Journal on Scientific Computing 21 (4) (1999) 1485–1505.

[4] E. Carrizosa, N. Mladenovic, R. Todosijevic, Variable neighborhood search for minimum sum-of-squares clustering on networks, European Journal of Operational Research 230 (2) (2013) 356–363.

[5] E. Santi, D. Aloise, S. J. Blanchard, A model for clustering data from heterogeneous dissimilarities, European Journal of Operational Research 253 (3) (2016) 659–672.

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

[7] R. L. Rabello, G. R. Mauri, G. M. Ribeiro, L. A. N. Lorena, A Clustering Search metaheuristic for the Point-Feature Cartographic Label Placement Problem, European Journal of Operational Research 234 (3) (2014) 802–808.

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

[9] U. Maulik, A. Mukhopadhyay, Simulated annealing based automatic fuzzy clustering combined with ANN classification for analyzing microarray data, Computers & Operations Research, 37 (8) (2010) 1369–1380.

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

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

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

[22] N. Karmitsa, Diagonal bundle method for nonsmooth sparse optimization, Journal of Optimization Theory and Applications 166 (3) (2015) 889–905, DOI 10.1007/s10957-014-0666-8.

[23] 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.

[24] 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.

[25] L. Lukšan, J. Vlček, Globally convergent variable metric method for convex nonsmooth unconstrained minimization, Journal of Optimization Theory and Applications 102 (3) (1999) 593–613.

[26] J. Vlček, L. Lukšan, Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization, Journal of Optimization Theory and Applications 111 (2) (2001) 407–430.

[27] A. Fuduli, M. Gaudioso, G. Giallombardo, A DC piecewise affine model and a bundling technique in nonconvex nonsmooth minimization, Optimization Methods and Software 19 (1) (2004) 89–102.

[28] M. Gaudioso, E. Gorgone, Gradient set splitting in nonconvex nonsmooth numerical optimization, Optimization Methods and Software 25 (2010) 59–74.

[29] J.-B. Hiriart-Urruty, C. Lemaréchal, Convex Analysis and Minimization Algorithms II, Springer-Verlag, Berlin, 1993.

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

[31] M. M. Mäkelä, P. Neittaanmäki, Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control, World Scientific Publishing Co., Singapore, 1992.

[32] C. Lemaréchal, J.-J. Strodiot, A. Bihain, On a bundle algorithm for nonsmooth optimization, in: O. L. Mangasarian, R. R. Mayer, S. M. Robinson (Eds.), Nonlinear Programming, Academic Press, New York, 1981, pp. 245–281.

[33] R. Mifflin, A modification and an extension of Lemaréchal's algorithm for nonsmooth minimization, Matematical Programming Study 17 (1982) 77–90.

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

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

[36] V. Demyanov, A. Rubinov, Constructive Nonsmooth Analysis, Peter Lang, Frankfurt am Main, 1995.

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

[38] 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.

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

[40] J. Herskovits, E. Goulart, Sparse quasi-Newton matrices for large scale nonlinear optimization, in: Proceedings of the 6th Word Congress on Structural and Multidisciplinary Optimization, 2005.

[41] 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.

[42] 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).

[43] 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) (2013).

[44] 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).

[45] 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).

[46] 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).

[47] 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).

[48] 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).

[49] 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).