Napsu Karmitsa | Adil Bagirov | Sona Taheri

Diagonal Bundle Method for Solving the Minimum Sum-of-Squares Clustering Problems

Turku Centre *for* Computer Science

TUCS Technical Report
No 1156, April 2016

Diagonal Bundle Method for Solving the Minimum Sum-of-Squares Clustering Problems

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 1156, April 2016

Abstract

Clustering is among most important tasks in data mining. This problem in very 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. Using nonsmooth optimization formulation of the clustering problem the objective function in this problem is represented as a difference of two convex functions. Then a new diagonal bundle method that explicitly utilizes this structure is designed to solve this problem. The method is evaluated using real world data sets with both the large number of attributes and/or large number of data points. The new algorithm is also compared with an other algorithm based on difference of convex representations.

Keywords: Cluster analysis, Nonsmooth optimization, Nondifferentiable optimization, DC-function, Nonconvex problems, Bundle 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, 9, 35], for brief review of some of these algorithms). In papers [6, 9, 12], nonsmooth optimization (NSO, not necessarily differentiable optimization) algorithms were developed. Algorithms based on hyperbolic smoothing technique are presented in [7, 37, 38]. Various optimization techniques such as branch and bound [16], interior point methods [17], the variable neighborhood search algorithm [22] and metaheuristics such as the simulated annealing [34], tabu search [1], and genetic algorithms [33] have also been introduced. In addition, algorithms based on difference of convex (DC) representation of the MSSC problem are introduced in [2, 3, 4, 11].

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. The use of the DC structure of the clustering problem and modification of powerful optimization methods for this type of problem may lead to the design of efficient clustering algorithms in very large data sets. In this paper, the phrase "very large data set" means that the data set contains hundred of thousands or millions of data points (features) and/or hundreds 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 algorithm which is accurate and efficient in very large data sets.

The idea of the new method comes from two sources: first, the possibility of splitting the clustering problem into two different pieces — one of which is convex and smooth and another one which is general convex nonsmooth — and second, from the fact that the convex model of the objective function is usually reasonable good also for nonconvex problems but in some areas where there exists the so-called concave behaviour in the objective. Thus the nonconvexity of the problem should be taken to 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 DC functions. The DCD-BUNDLE combines the ideas of the *diagonal bundle method* (D-BUNDLE, [25]) to different usage of metrics depending on the convex or concave behaviour of the objective at the current iteration point. The D-BUNDLE, in its turn, is developed for sparse large-scale nonsmooth, possible nonconvex, optimization. It is a successor of the *limited memory bundle method* (LMBM, [20, 21]) and the *variable metric bundle method* (VMBM, [29, 36]) better capable to handle large dimensionality and sparsity of the objective. The idea of splitting the information obtained in previous iterations comes

from [18, 19]. However, instead of splitting the bundle information (i.e. the subgradient information) as in [18, 19], the DCD-BUNDLE computes different variable metric approximations depending on the point. In addition, the DCD-BUNDLE uses the real structure of the problem given as a DC formulation.

The DCD-BUNDLE shares the good properties of the D-BUNDLE. That is, the time-consuming quadratic direction finding problem appearing in the standard bundle methods (see eq. [24, 26, 30]) 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 nonconvexity of the problem than only the usual subgradient locality measure (see e.g. [27, 31]) used in the D-BUNDLE does.

This paper is organized as follows. In Section 2 we introduce our notation and recall some basic definitions and results from nonsmooth analysis. DC representations of cluster functions are given in Section 3. Optimality conditions for the auxiliary clustering problem and the clustering problem are studied 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 solve globally the clustering problem. The results of the numerical experiments are presented and discussed in Section 8, and finally, Section 9 concludes the paper.

2 Notations and Background

In this section, we give our notations, recall definitions of DC function and DC programming problem, and give some basic definitions and results from nonsmooth analysis and DC programming.

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 $\mathrm{diag}(\boldsymbol{a})$, for $\boldsymbol{a} \in \mathbb{R}^n$, the diagonal matrix such that $\mathrm{diag}(\boldsymbol{a})_{i,i} = \boldsymbol{a}_i$. The Frobenius norm of 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 center $\boldsymbol{x} \in \mathbb{R}^n$ and 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 exists 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. Note that a DC function has infinitely many DC decompositions.

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 a *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* [14] 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 *subdifferential* $\partial f(\boldsymbol{x})$ of the 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 convex function we have that $\partial f(\boldsymbol{x}) = \partial_c f(\boldsymbol{x})$ 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. [10])

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

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

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

It is clear that 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 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. In cluster analysis we assume that we are given a finite set of points A in the n-dimensional space $\mathbb{R}^n$, 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. $A = \bigcup_{j=1}^{k} A^j$.

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

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

Clustering Problem. The *NSO formulation of the MSSC problem* [8, 9] is given as

$$\begin{cases} \text{minimize} & f_k(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \in \mathbb{R}^{nk}, \end{cases} \tag{6}$$

where

$$f_k(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) = \frac{1}{m} \sum_{\boldsymbol{a} \in A} \min_{j=1,\ldots,k} d_2(\boldsymbol{x}^j, \boldsymbol{a}). \tag{7}$$

The DC representation of the function f_k in Problem (6) [11] is

$$f_k(\boldsymbol{x}) = f_{k1}(\boldsymbol{x}) - f_{k2}(\boldsymbol{x}),\ \boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \in \mathbb{R}^{nk}, \tag{8}$$

where

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

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

Since the function d_2 is convex in $\boldsymbol{x}$, the function f_{k1} is also convex as a sum of convex functions. In its turn, the function f_{k2} is a sum of maxima of sum of convex functions. Since d_2 is convex and a sum of convex functions is convex, the functions under maximum are convex. Furthermore, the maximum of a finite number of convex functions is also convex. Thus, the function f_{k2} is a sum of convex functions and therefore it is also convex.

Note that the similarity measure in nonsmooth clustering problem can also be defined using other norms. That is, for instance, L_1- or L_∞-norm (see e.g. [10]). Nevertheless, with these norms both f_{k1} and f_{k2} are nonsmooth while with d_2 the first component f_{k1} of the DC function remains smooth (continuously differentiable). 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 clustering problems.

Auxiliary Clustering Problem. Problem (6) is a global optimization problem.That is, the objective function f_k in this problem has many local minimizers and only its global minimizers provide the best cluster structure of a data set with the least number of clusters. In general, conventional global optimization methods cannot be applied to solve this problem in large data sets. Therefore in such data sets heuristics and deterministic local search algorithms are the only choice. However, the success of these algorithms heavily depends on the choice of starting cluster centers and the development of efficient procedures for generating starting clusters centers is crucial for

the success of such algorithms. Here we apply an approach introduced in [32] to find starting cluster centers. This approach involves the solution of the so-called auxiliary clustering 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 $r^{\boldsymbol{a}}_{k-1}$ the distance between the data point $\boldsymbol{a} \in A$ and the closest cluster center among $k-1$ centers $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{k-1}$. That is,

$$r^{\boldsymbol{a}}_{k-1} = \min\left\{d_2(\boldsymbol{x}^1, \boldsymbol{a}), \ldots, d_2(\boldsymbol{x}^{k-1}, \boldsymbol{a})\right\}. \tag{9}$$

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

$$\bar{f}_k(\boldsymbol{y}) = \frac{1}{m}\sum_{\boldsymbol{a}\in A} \min\left\{r^{\boldsymbol{a}}_{k-1}, d_2(\boldsymbol{y}, \boldsymbol{a})\right\},\ \boldsymbol{y} \in \mathbb{R}^n. \tag{10}$$

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

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

A problem

$$\begin{cases} \text{minimize} & \bar{f}_k(\boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} \in \mathbb{R}^n, \end{cases} \tag{11}$$

is called the *k-th auxiliary clustering problem* [5]. The DC representation of the function $\bar{f}_k$ [11] is given by

$$\bar{f}_k(\boldsymbol{y}) = \bar{f}_{k1}(\boldsymbol{y}) - \bar{f}_{k2}(\boldsymbol{y}) \tag{12}$$

where

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

4 Optimality Conditions

In this section we study optimality conditions for Problems (6) and (11) using their DC representations. For more details and proofs we refer to [11].

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

$$\nabla \bar{f}_{k1}(\boldsymbol{y}) = \frac{2}{m}\sum_{\boldsymbol{a}\in A}(\boldsymbol{y} - \boldsymbol{a}). \tag{13}$$

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

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

We rewrite the function $\bar{f}_{k2}$ at $\boldsymbol{y}$ 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

$$\partial\bar{f}_{k2}(\boldsymbol{y}) = \frac{2}{m}\left(\sum_{\boldsymbol{a}\in\bar{A}_2(\boldsymbol{y})} (\boldsymbol{y} - \boldsymbol{a}) + \sum_{\boldsymbol{a}\in\bar{A}_3(\boldsymbol{y})} \operatorname{conv}\{0, (\boldsymbol{y} - \boldsymbol{a})\}\right). \tag{14}$$

Now due to smoothness of $\bar{f}_{k1}$, the generalized subdifferential $\partial\bar{f}_k(\boldsymbol{y})$ of the k-th auxiliary cluster 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 (11) the subdifferential $\partial\bar{f}_{k2}(\boldsymbol{y}^*)$ is singleton*

$$\partial\bar{f}_{k2}(\boldsymbol{y}^*) = \frac{2}{m}\left\{\sum_{\boldsymbol{a}\in\bar{A}_2(\boldsymbol{y}^*)} (\boldsymbol{y}^* - \boldsymbol{a})\right\}. \tag{15}$$

Moreover, for a point $\boldsymbol{y}^*$ to be a local minimizer of Problem (11) it is necessary that

$$\sum_{\boldsymbol{a}\in\bar{A}_1(\boldsymbol{y}^*)} (\boldsymbol{y}^* - \boldsymbol{a}) = 0. \tag{16}$$

It is obvious that any $\inf$-stationary point of Problem (11) 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 (11) coincide and they are given by

$$S = \{\boldsymbol{y} \in \mathbb{R}^n \mid \nabla\bar{f}_{k1} \in \partial\bar{f}_{k2}(\boldsymbol{y})\}. \tag{17}$$

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 subdifferentials is not singleton. This result is needed in order

to develop an algorithm that can escape Clarke stationary point and converge to inf-stationary point. To compute different subgradients we can choose $\boldsymbol{\xi}^1,\ \boldsymbol{\xi}^2 \in \partial \bar{f}_{k2}(\boldsymbol{y})$ using (14) as follows

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

and

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

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

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

$$\nabla f_{k1}(\boldsymbol{x}) = 2(\boldsymbol{x} - \widehat{A}). \tag{18}$$

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

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

This means that the subdifferential $\partial f_{k1}(\boldsymbol{x}) = \{\nabla f_{k1}(\boldsymbol{x})\}$ is a singleton for any $\boldsymbol{x} \in \mathbb{R}^n$ (see e.g. [10]).

In general, the function f_{k2} is nonsmooth. To compute its subdifferential consider the following function and a set [32]

$$\varphi_{\boldsymbol{a}}(\boldsymbol{x}) = \max_{j=1,\ldots,k} \sum_{s=1, s \neq j}^{k} d_2(\boldsymbol{x}^s, \boldsymbol{a}), \tag{19}$$

and

$$R_{\boldsymbol{a}}(\boldsymbol{x}) = \left\{ j \in \{1, \ldots, k\} \mid \sum_{s=1, s \neq j}^{k} d_2(\boldsymbol{x}^s, \boldsymbol{a}) = \varphi_{\boldsymbol{a}}(\boldsymbol{x}) \right\}. \tag{20}$$

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

$$\partial \varphi_{\boldsymbol{a}}(\boldsymbol{x}) = \text{conv} \left\{ V \in \mathbb{R}^{nk} \mid\ V = 2(\tilde{\boldsymbol{x}}^j - \tilde{A}_i^j), j \in R_{\boldsymbol{a}}(\boldsymbol{x}) \right\}, \tag{21}$$

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

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

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

$$\partial f_{k2}(\boldsymbol{x}) = \frac{1}{m} \sum_{\boldsymbol{a} \in A} \partial \varphi_{\boldsymbol{a}}(\boldsymbol{x}). \tag{22}$$

Similarly to auxiliary clustering problem, the smoothness of $f_{k1}(\boldsymbol{x})$ allows us to write the 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 (6) 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 clustering problem.

THEOREM 4.2. *Let $\boldsymbol{x} \in \mathbb{R}^{nk}$ be a local minimizer of the problem (6). 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 singleton. If $|A_2| \geq 1$ then $\partial \bar{f}_{k2}(\boldsymbol{x})$ is not singleton. Take any $\boldsymbol{a} \in A_2$. Since $|R_{\boldsymbol{a}}(\boldsymbol{x})| \geq 2$ the 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 (19).

5 Diagonal Bundle Method for DC Clustering Problem

In this section, we introduce a new algorithm DCD-BUNDLE for solving nonsmooth clustering problems given in DC-form. The algorithm is used to solve both the clustering problem (6) and the auxiliary clustering problem (11). It is easy to see that both these problems can be formulated as an unconstrained DC programming problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{23}$$

with f_1 and f_2 convex. In our approach we also assume that f_1 is smooth. For the clustering problems with the similarity measure defined by the squared Euclidean norm this assumption is trivially satisfied. In addition, we assume that at every point $\boldsymbol{x}$ we can evaluate the values of the DC-components $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ of the objective function $f(\boldsymbol{x})$, 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" behaviour of the objective. The linearization error α_{k+1} at 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 Data. The DCD-BUNDLE uses the diagonal update formula introduced in [23] for updating the matrices, since for this formula it is easy to check and guarantee the positive definiteness of generated matrices. Moreover, using diagonal update matrix requires 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. [13]). 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 $\boldsymbol{\xi}_{2,m} \in \partial f_2(\boldsymbol{x}_k)$. 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, we now compute the gradient differences separately to both DC-components and, 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 $S_k = [\boldsymbol{s}_{k-m_c+1} \dots \boldsymbol{s}_k]$ and $U_k = [\boldsymbol{u}_{k-m_c+1} \dots \boldsymbol{u}_k]$ used in [13] and, also in the D-BUNDLE [25], we have three correction matrices $S_k = [\boldsymbol{s}_{k-m_c+1} \dots \boldsymbol{s}_k]$, $U_{1,k} = [\boldsymbol{u}_{1,k-m_c+1} \dots \boldsymbol{u}_{1,k}]$, and $U_{2,k} = [\boldsymbol{u}_{2,k-m_c+1} \dots \boldsymbol{u}_{2,k}]$. We use these matrices to compute separate approximations to both f_1 and f_2 and 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 problem. Thus, the update matrix $B_{l,k+1}$ ($l = 1, 2$) is defined by

$$\begin{cases} \text{minimize} & \|B_{l,k+1}S_k - U_{l,k}\|_F^2 \\ \text{subject to} & (B_{l,k+1})_{i,j} = 0 \text{ for } i \neq j \\ & (B_{l,k+1})_{i,i} \geq \mu \text{ for } i = 1, 2, \dots, n \text{ and } \mu > 0. \end{cases} \tag{24}$$

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 $\boldsymbol{b} = 2\sum_{i=\hat{1}}^{\hat{m}_c} \mathrm{diag}(\boldsymbol{s}_i)\boldsymbol{u}_{l,i}$ and $Q = 2\sum_{i=\hat{1}}^{\hat{m}_c} [\mathrm{diag}(\boldsymbol{s}_i)]^2$ 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 the 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 account 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 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. Note that due to diagonal matrices used this value is very easy to compute. The search direction is then computed by the formula

$$\boldsymbol{d}_k = -(p_k D_{1,k} - (1 - p_k)D_{2,k})\tilde{\boldsymbol{\xi}}_k. \tag{25}$$

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

$$f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L w_k, \tag{26}$$

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 (26) 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 we continue with this metrics even if the linearization error was negative.

If condition (26) 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

$$-\beta_{k+1} + \boldsymbol{d}_k^T \boldsymbol{\xi}_{k+1}^t \geq -\varepsilon_R w_k, \tag{27}$$

where $\varepsilon_R \in (\varepsilon_L, 1/2)$ is a given parameter and β_{k+1} is the subgradient locality measure [27, 31] similar to bundle methods. That is,

$$\beta_{k+1} = \max\left\{|f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\boldsymbol{d}_k) + t(\boldsymbol{\xi}_{k+1}^t)^T \boldsymbol{d}_k|, \gamma\|t\boldsymbol{d}_k\|^2\right\}, \tag{28}$$

where $\gamma \geq 0$ is a distance measure parameter supplied by the user. This kind of t always exists [19] . 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 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 all the matrices $D_{1,k}$ and $D_{2,k}$ are bounded. Due to the diagonal update formula this assumption is trivially satisfied. In addition, the condition

$$\tilde{\boldsymbol{\xi}}_k^T D_{1,k} \tilde{\boldsymbol{\xi}}_k \leq \tilde{\boldsymbol{\xi}}_k^T D_{1,k-1} \tilde{\boldsymbol{\xi}}_k \tag{29}$$

has to be satisfied each time there occurs more than one consecutive null step. In the DCD-BUNDLE this is guaranteed simply by *skipping the convex updates* if more than one consecutive null steps occurs.

Aggregation. The aggregation procedure used in the DCD-BUNDLE is quite similar to that of the original LMBM [20, 21]. That is, 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

$$\begin{aligned}\varphi(\lambda_1, \lambda_2, \lambda_3) = \; & (\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1}^t + \lambda_3 \tilde{\boldsymbol{\xi}}_k)^T D_{1,k} (\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1}^t + \lambda_3 \tilde{\boldsymbol{\xi}}_k) \\ & + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k),\end{aligned} \tag{30}$$

where we have 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

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1}^t + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \qquad \text{and} \tag{31}$$

$$\tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \tag{32}$$

Algorithms. Now we give the detailed algorithm for DCD-BUNDLE to compute Clarke stationary point of the clustering problem. Later, we will give the algorithm to find inf-stationary points by using this algorithm.

ALGORITHM 5.1. DCD-BUNDLE.

Data: 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 $\hat{m}_c \geq 1$.

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

Step 5 (*Serious Step*) If

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

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

$$\alpha_{k+1} = f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + (\nabla f_{1,k+1} - \boldsymbol{\xi}_{2,k+1})^T \boldsymbol{d}_k. \tag{33}$$

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 (27) with β_{k+1} computed as in (28). 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 (30).

Set

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

Step 7: (*Null Step*) If $m = k$, 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 $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $k = k + 1$ and go to Step 8.

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 Clarke stationary points of the clustering problem. If the second component function f_2 is smooth then this point is also an $\inf$-stationary point of the clustering problem. Our main assumptions in order to find $\inf$-stationary points is the following.

ASSUMPTION 5.1. 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 clustering and 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 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

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

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

$$t_j = \text{argmax}\,\{t > 0 \mid f(\boldsymbol{x}^* + t\bar{\boldsymbol{u}}_j) - f(\boldsymbol{x}^*) \leq -\varepsilon_T tr\}.$$

6 Global Convergence

We now study the convergence properties of Algorithms 5.1 and 5.2 given in previous section. We first prove that a point generated by Algorithm 5.1 is a Clarke stationary point of problem (23). In order to do this, we assume that the final accuracy tolerance ε is equal to zero. After that we prove that Algorithm 5.2 terminates after finite number of iterations at an inf-stationary point of the problem. For the simplicity of the presentation we, from now on, trop out the index t from $\boldsymbol{y}_k^t$ and $\boldsymbol{\xi}_k^t$ even if $t \neq 1$ and we use the notation $\boldsymbol{\xi} = \nabla f_1 - \boldsymbol{\xi}_2$.

The assumptions needed to prove the global convergence are the following.

ASSUMPTION 6.1. 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 6.2. The objective function f is bounded from below.

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

REMARK 6.1. The sequence $\boldsymbol{x}_k$ generated by Algorithm 5.1 is bounded by assumption and the monotonicity of the sequence f_k which, in turn, is obtained due to condition (26) 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 [14]. The matrices D_1 and D_2 are bounded due to fact that all their components are in closed interval $[\mu_{min}, \mu_{max}]$. Thus, 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

$$\beta_{k+1} \geq \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\|^2. \tag{34}$$

Proof. We point out first that $\tilde{\beta}_k \geq 0$ for all k by equations (28), (32), 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.

By (28) and since we have $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps, and since we, on the other hand, have $\beta_{k+1} = 0$ and $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| = 0$ for serious steps, condition (34) 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 [36]. □

LEMMA 6.3. *Let $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ be given 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 [36]. □

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.

Now, by Lemma 6.2 and by using Lemma 6.3 with

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

we obtain $\boldsymbol{0} = \tilde{\boldsymbol{\xi}}_k \in \partial f(\boldsymbol{x}_k)$ and, thus, $\boldsymbol{x}_k$ 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 [36]. □

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*

$$\tilde{\boldsymbol{\xi}}_{k+1}^T D_{1,k+1} \tilde{\boldsymbol{\xi}}_{k+1} = \tilde{\boldsymbol{\xi}}_{k+1}^T D_{1,k} \tilde{\boldsymbol{\xi}}_{k+1} \qquad \textit{and} \tag{35}$$

$$\operatorname{tr}(D_{1,k}) \leq \mu_{max} n \tag{36}$$

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

Proof. If $m < k_0$, we always have $D_{1,k+1} = D_{1,k}$ due Step 7 of Algorithm 5.1. Thus, condition (35) is valid. Furthermore, in each case we have

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

for all k, since $D_{1,k}$ is a diagonal matrix with the largest diagonal element equal to μ_{max}. Therefore, also condition (36) is 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 (30), (31), (32), Lemma 6.1, and Lemma 6.6 we obtain

$$\begin{aligned} w_{k+1} &= \tilde{\boldsymbol{\xi}}_{k+1}^T D_{1,k+1} \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\ &= \tilde{\boldsymbol{\xi}}_{k+1}^T D_{1,k} \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\ &= \varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) \\ &\leq \varphi(0,0,1) \\ &= \tilde{\boldsymbol{\xi}}_k^T D_{1,k} \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k \\ &= w_k \end{aligned} \tag{37}$$

for $k \geq m$.

Let us denote $D_{1,k} = W_k^T W_k$. Then, function φ (see (30)) 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 (37) 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 we have $-\beta_{k+1} + \boldsymbol{\xi}_{k+1}^T \boldsymbol{d}_k \geq -\varepsilon_R w_k$ in null steps the last part of the proof proceeds similar to the proof (part (ii)) of Lemma 3.6 in [36]. □

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, $f(\boldsymbol{x}_k) \downarrow f(\bar{\boldsymbol{x}})$ by the monotonicity of the sequence $(f(\boldsymbol{x}_k))$ obtained due to the descent step condition (26). Using condition (26) and the fact that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ in null steps, we obtain

$$0 \leq \varepsilon_L w_k \leq f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \to 0 \qquad \text{for } k \geq 1. \tag{38}$$

Thus, $(w_k)_{k\in\mathcal{K}'} \to 0$ and $(\boldsymbol{x}_k)_{k\in\mathcal{K}'} \to \bar{\boldsymbol{x}}$ by (38) and, thus, by Lemma 6.5 we have $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$. □

REMARK 6.2. If we choose $\varepsilon > 0$, 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 but we can compute the subgradient $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ at every point.

Now we prove that Algorithm 5.2 terminates after finite number of iterations at an $\inf$-stationary point of the problem. In addition to Assumptions 5.1 – 6.2 we need the following assumption.

ASSUMPTION 6.3. The gradient $\nabla f_1 : \mathbb{R}^n \to \mathbb{R}^n$ of function f_1 satisfies Lipschitz condition.

At the end of this section, we will prove that for the clustering function f_{k1} and for the auxiliriary clustering function $\bar{f}_{k1}$ Assumption 6.3 is satisfied.

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 finite number of iterations at an* $\inf$*-stationary point of Problem (23).*

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 $\boldsymbol{\xi}_2^1 \in \partial f_2(\boldsymbol{x}_j)$. Applying the mean value theorem to the function f_1 we obtain that for some $\sigma_j \in (0, 1)$

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

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

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

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

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

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

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

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 finite number of iterations. □

Finally, we show that the gradient 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 (13) 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 Proposition 6.10. □

7 Incremental algorithm

In this section we present an incremental algorithm for solving Problem (6) using the DC approach. Problem (6) is a global optimization problem and it is important to use many starting points when applying a local method for its solution. Computation of clusters incrementally allows us to design different algorithms for generating starting cluster centers. Such an approach allows us to compute global or near global solutions to clustering problems. One such algorithm is introduced in [32] and it is used in our incremental algorithm given below.

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 [32] to find the set $S_1 \subset \mathbb{R}^n$ of starting points for solving the auxiliary clustering problem (11) for $k = l$.

Step 4: (*Computation of a set of starting points for the l-th cluster center*). Apply Algorithm 5.2 to solve Problem (11) starting from each point $\boldsymbol{y} \in S_1$. This algorithm generates a set $S_2 \subset \mathbb{R}^n$ of starting points for the l-th cluster center.

Step 5: (*Computation of a set of cluster centers*) For each $\bar{\boldsymbol{y}} \in S_2$ apply Algorithm 5.2 to solve Problem (6) starting from the point $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}, \bar{\boldsymbol{y}})$ and find a solution $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l)$. Denote by $S_3 \subset \mathbb{R}^{nl}$ a set of all such solutions.

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 [11] using the same real world data sets as in [11]. We refer to [11] for detailed description of `DCClust` and also for the performances of other algorithms suitable to solve these kinds of problems. Both solvers `DCD-Bundle` and `DCClust` use the incremental approach to solve clustering problems globally and they are both implemented in Fortran 95 and 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 and the data sets used in our experiments can be downloaded from http://napsu.karmitsa.fi/clustering/.

As said we used the same data sets as in [11]. That is, eight different data sets were used in our experiments. The brief description of these sets is given in Table 1. The more detailed description can be found in [28]. 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 large (128) and the numbers of data points range from tens of thousands (smallest 13 910) to hundred of thousands (largest 434 874). We computed incrementally up to 25 clusters with all data sets. Results are given in Tables 2–9, 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 (7) (multiplied with m) for the corresponding number of clusters. We have used the f_{best} value given in [11] unless we got better value in our experiments. If the value obtained in our experiments was better than that of [11] we have marked it with star.
- 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.

Table 1: The brief description of data sets

Data sets	No. instances	No. attributes
Gas Sensor Array Drift	13910	128
EEG Eye State	14980	14
D15112	15112	2
KEGG Metabolic	53413	20
Shuttle Control	58000	9
Pla85900	85900	2
Skin Segmentation	245057	3
3D Road Network	434874	3

The data sets can be divided into two groups. The first group contains 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 4 and 7 – 9 demonstrate that in these data sets the accuracy of the solvers was quite similar and the solutions found were at least near best known solutions. The only

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

k	f_{best}	DCD-Bundle		DCClust	
		E_A	cpu	E_A	cpu
2	7.91186	0.00	17.76	0.00	24.78
3	5.02412	0.00	49.42	0.00	64.00
5	3.22726	0.00	151.86	0.00	165.43
10	1.65524	0.01	467.90	0.00	510.62
15	1.13801	0.36	870.13	0.36	988.77
20	0.87916	2.61	1211.98	0.62	1514.49
25	0.72348	0.67	1534.75	0.47	2053.33

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

k	f_{best}	DCD-Bundle		DCClust	
		E_A	cpu	E_A	cpu
2	8178.13809	0.00	0.15	0.00	0.44
3	1833.88058	0.00	0.17	0.00	0.84
5	1.33858	0.00	0.76	0.00	2.66
10	0.45669	0.00	9.07	0.00	18.63
15	0.34653	0.28	23.14	0.26	49.07
20	0.28987	1.53	40.91	0.96	88.35
25	0.25989*	0.04	59.31	0.00	137.86

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

k	f_{best}	DCD-Bundle		DCClust	
		E_A	cpu	E_A	cpu
2	3.68403	0.00	0.79	0.00	0.93
3	2.53240	0.00	1.54	0.00	1.92
5	1.32707	0.00	2.55	0.00	3.21
10	0.64892	0.00	5.58	0.00	8.00
15	0.43138	0.24	10.05	0.24	18.18
20	0.32177	0.03	16.03	0.00	32.58
25	0.25309	0.00	29.37	0.00	53.89

Table 5: Summary of the results with KEGG Metabolic ($\times 10^8$).

k	f_{best}	DCD-Bundle		DCClust	
		E_A	cpu	E_A	cpu
2	11.38530	0.00	5.23	0.00	6.65
3	4.90060	0.00	13.41	0.00	18.25
5	1.88367	0.00	84.22	0.06	66.35
10	0.63515	0.30	388.59	0.21	358.11
15	0.35393*	0.00	747.86	0.26	719.32
20	0.25027*	0.00	1086.82	2.04	1122.56
25	0.19289	0.94	1632.00	0.75	1549.19

Table 6: Summary of the results with Shuttle Control ($\times 10^8$).

k	f_{best}	DCD-Bundle		DCClust	
		E_A	cpu	E_A	cpu
2	21.34329.	0.00	0.40	0.00	1.19
3	10.85415	0.00	0.93	0.00	3.33
5	7.24479	0.00	39.27	0.24	21.26
10	2.83216	0.20	138.48	0.33	78.65
15	1.53154	3.78	531.97	0.37	290.63
20	1.06012	2.33	652.78	1.16	516.43
25	0.78727	4.64	778.04	0.13	774.88

Table 7: Summary of the results with Pla85900 ($\times 10^{15}$).

k	f_{best}	DCD-Bundle		DCClust	
		E_A	cpu	E_A	cpu
2	3.74908	0.00	16.27	0.00	15.63
3	2.28057	0.00	31.52	0.00	30.05
5	1.33972	0.00	62.17	0.00	60.87
10	0.68294	0.00	141.46	0.00	145.04
15	0.46249	0.00	231.45	0.00	254.82
20	0.34988	0.52	339.97	0.52	383.46
25	0.28265	0.00	450.94	0.00	529.44

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

k	f_{best}	DCD-Bundle		DCClust	
		E_A	cpu	E_A	cpu
2	1.32236	0.00	167.97	0.00	166.33
3	0.89362	0.00	303.41	0.00	290.68
5	0.50203	0.00	544.13	0.00	517.17
10	0.25121	0.00	1094.79	0.00	1076.48
15	0.16964	0.18	1646.77	0.18	1640.04
20	0.12770	0.15	2164.16	0.15	2217.97
25	0.10299	0.01	2719.04	0.01	2844.37

Table 9: Summary of the results with 3D Road Network ($\times 10^6$).

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

remarkable exception here is DCClust in 3D Road Network data set with 25 clusters (see Table 9). More so, since with this data set the value of the clustering function was clearly improved from that given in [11] with DCD−Bundle (1.81 % with 25 clusters).

The other group consist of data sets with larger number of attributes (9–128, see Table 1). They are, "Gas Sensor Array Drift", "EEG Eye State", "KEGG Metabolic", and "Shuttle Control". Results in these data sets are given in Tables 2, 3, 6, and 7, respectively. Results shows that both the algorithms can find the near best known solution in almost all k in "Gas Sensor Array Drift", "EEG Eye State", and "KEGG Metabolic". Here, DCD−Bundle had difficulties with $k = 20$ in "Gas Sensor Array Drift" and "EEG Eye State" while DCClust had difficulties with $k = 20$ in "KEGG Metabolic". In addition, results in "Shuttle Control" data set shows 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.

The new method DCD−Bundle was clearly faster than DCClust with data sets including relatively small number of instances, that is, with data sets "Gas Sensor Array Drift", "EEG Eye State" and "D15112" (see Table 1). In addition, it used less cpu-time with data sets with large number of instances but small number of attributes. That is, "Pla85900", "Skin Segmentation", and "3D Road Network". 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") DCD−Bundle used about the same or a bit more cpu-time than DCClust.

9 Conclusions

In this paper a new DCD-BUNDLE -method for solving sum-of-squares clustering problems is introduced. The clustering problem is formulated as a nonsmooth DC programming problem and the new method utilizes this DC representation to find a minimum.

The DCD-BUNDLE -method consist 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 new DCD-BUNDLE -algorithm (Algorithm 5.1) to find a Clarke stationary point of these problems. We have proved that the new DCD-BUNDLE -method converges to an inf-stationary point of the clustering problem.

The new DCD-BUNDLE -method was tested using real world data sets with the numbers of data points ranging from tens of thousands to hundred of thousands. We can conclude that DCD-BUNDLE -method was both efficient and accurate and it can be used to solve clustering problems 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., 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.

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

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

[5] BAGIROV, A. Modified global k-means algorithm for minimum 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. *Comp. Opt. and Appl. 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., 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.

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

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

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

[13] 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.

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

[15] DEMYANOV, V., AND RUBINOV, A. *Constructive Nonsmooth Analysis*. Peter Lang, Frankfurt am Main, 1995.

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

[17] 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.

[18] FUDULI, A., GAUDIOSO, M., AND GIALLOMBARDO, G. A DC piecewise affine model and a bundling technique in nonconvex nonsmooth minimization. *Optimization Methods and Software 19*, 1 (2004), 89–102.

[19] GAUDIOSO, M., AND GORGONE, E. Gradient set splitting in nonconvex nonsmooth numerical optimization. *Optimization Methods and Software 25* (2010), 59–74.

[20] 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.

[21] 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.

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

[23] HERSKOVITS, J., AND GOULART, E. Sparse quasi-Newton matrices for large scale nonlinear optimization. In *Proceedings of the 6th Word Congress on Structurla and Multidisciplinary Optimization* (2005).

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

[25] KARMITSA, N. 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.

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

[27] LEMARÉCHAL, C., STRODIOT, J.-J., AND BIHAIN, A. On a bundle algorithm for nonsmooth optimization. In *Nonlinear Programming*, O. L. Mangasarian, R. R. Mayer, and S. M. Robinson, Eds. Academic Press, New York, 1981, pp. 245–281.

[28] 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).

[29] 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.

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

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

[32] 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.

[33] 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.

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

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

[36] 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.

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

[38] XAVIER, A. E., 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.

Turku
Centre *for*
Computer
Science

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 978-952-12-3384-5
ISSN 1239-1891