Computational Optimization and Applications (2021) 80:411–438
https://doi.org/10.1007/s10589-021-00304-4

An augmented subgradient method for minimizing nonsmooth DC functions

A. M. Bagirov[1] · N. Hoseini Monjezi[2] · S. Taheri[3]

Received: 10 September 2020 / Accepted: 9 July 2021 / Published online: 24 July 2021

Abstract
A method, called an augmented subgradient method, is developed to solve unconstrained nonsmooth difference of convex (DC) optimization problems. At each iteration of this method search directions are found by using several subgradients of the first DC component and one subgradient of the second DC component of the objective function. The developed method applies an Armijo-type line search procedure to find the next iteration point. It is proved that the sequence of points generated by the method converges to a critical point of the unconstrained DC optimization problem. The performance of the method is demonstrated using academic test problems with nonsmooth DC objective functions and its performance is compared with that of two general nonsmooth optimization solvers and five solvers specifically designed for unconstrained DC optimization. Computational results show that the developed method is efficient and robust for solving nonsmooth DC optimization problems.

Keywords Nonsmooth optimization · Nonconvex optimization · DC optimization · Subgradients

Mathematics Subject Classification 90C26 · 49J52 · 65K05

✉ S. Taheri
sona.taheri@rmit.edu.au

A. M. Bagirov
a.bagirov@federation.edu.au

N. Hoseini Monjezi
najmeh.hoseini@sci.ui.ac.ir

[1] School of Engineering, Information Technology and Physical Sciences, Federation University Australia, Ballarat, Australia

[2] Department of Applied Mathematics and Computer Science, Faculty of Mathematics and Statistics, University of Isfahan, Isfahan, Iran

[3] School of Science, RMIT University, Melbourne, Australia

1 Introduction

Consider an unconstrained optimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is, in general, nonsmooth and is expressed as a difference of two convex functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$:

$$f(x) = f_1(x) - f_2(x).$$

Some important practical problems can be formulated as a nonsmooth DC program of the form (1). They include supervised data classification [6], cluster analysis [11, 32], clusterwise linear regression analysis [10, 12] and edge detection [27] problems. Moreover, the DC optimization methods can be applied to solve a broad class of nonsmooth nonconvex optimization problems [38].

Different stationarity conditions, such as inf-stationarity (also called d-stationarity), Clarke stationarity and criticality conditions, have been studied for unconstrained DC minimization problems [25]. In the paper [33], the notion of B-stationary points of nonsmooth DC problems was defined and its relation with the criticality condition was investigated.

The special structure of DC problems stimulates design of efficient methods by utilizing this structure. To date, DC optimization problems have been considered in the context of both local and global optimization. Several methods have been developed to solve these problems globally [23, 37]. To the best of our knowledge the difference of convex algorithm (DCA) is the first local search algorithm for solving DC optimization problems. This algorithm was introduced in [35] and further studied in [4, 5].

Over the last decade the development of local search methods for unconstrained nonsmooth DC optimization has attracted a noticeable attention. A short survey of such methods can be found in [19]. Without loss of generality, methods for local DC optimization can be divided into three groups. The first group consists of methods which are extensions of the bundle methods for convex problems. They include the codifferential method [13], the proximal bundle method with the concave affine model [22], the proximal bundle method for DC optimization [17, 24] and the double bundle method [25]. In these methods piecewise linear underestimates of both DC components or subgradients of these components are used to compute search directions.

The second group consists of the DCA, its modifications and extensions. Although the DCA performs well in practice, its convergence can be fairly slow for some particular problems. Various modifications of the DCA have been developed to improve its convergence. The inertial DCA was developed in [20]. In papers [2, 3], the boosted DC algorithm (BDCA) was proposed to minimize smooth DC functions. The BDCA accelerates the convergence of the DCA using an additional line search step. More precisely, it performs a line search at the point generated by the DCA which leads to a larger decrease in the objective value at each iteration. In [1],

the BDCA is combined with a simple derivative free optimization method which allows one to force the d-stationarity (lack of descent direction) at the obtained point. To avoid the difficulty of solving the DCA's subproblem, in [18] the first DC component is replaced with a convex model and the second DC component is used without any approximation.

The third group consists of algorithms that at each iteration apply the convex piecewise linear model of the first DC component and one subgradient of the second DC component to compute search directions. These algorithms differ from the DCA-type algorithms as at each iteration they update the model of the first DC component and calculate the new subgradient of the second component. They are also distinctive from methods of the first group because they use one subgradient of the second DC component whereas the latter methods may use more than one subgradient of this component to build its piecewise linear model. A representative of this group is the aggregate subgradient method for DC optimization developed in [9]. In this algorithm the aggregate subgradient of the first DC component and one subgradient of the second component are used to compute search directions.

In this paper, we introduce a new method which belongs to the third group. First, we define augmented subgradients of the convex functions. Using them we construct the model of the first DC component. This model and one subgradient of the second DC component are utilized to compute search directions. Then we design a new method which uses these search directions and applies the Armijo-type line search. We prove that the sequence of points generated by this method converges to critical points of the DC function. We utilize nonsmooth DC optimization test problems, including those with large number of variables, to demonstrate the performance of the developed method and to compare it with two general methods of nonsmooth optimization and five methods of nonsmooth DC optimization. Our results show that, in general, the developed method is more robust than other methods used in the numerical experiments.

The rest of the paper is organized as follows. In Sect. 2, we provide necessary notations and some preliminaries on the nonsmooth analysis and DC optimization. In Sect. 3, the new method is introduced and its convergence is studied. Results of numerical experiments are reported in Sect. 4, and concluding remarks are given in Sect. 5.

2 Preliminaries

In this section we provide the notations that will be used throughout the paper and recall some concepts of nonsmooth analysis and DC optimization. For more details on nonsmooth analysis, we refer to [7, 8].

In what follows, $\mathbb{R}^n$ is the n-dimensional Euclidean space, $u^T v = \sum_{i=1}^n u_i v_i$ is the inner product of vectors $u, v \in \mathbb{R}^n$, and $\|\cdot\|$ is the associated Euclidean norm. The origin of $\mathbb{R}^n$ is denoted by 0_n. The unit sphere is denoted by $S_1 = \{d \in \mathbb{R}^n : \|d\| = 1\}$. For $x \in \mathbb{R}^n$ and $\varepsilon > 0$, $B_\varepsilon(x)$ is an open ball of the radius $\varepsilon > 0$ centred at x. The convex hull of a set is denoted by "conv" and "cl" stands for the closure of a set.

The function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous on $\mathbb{R}^n$ if for every $x \in \mathbb{R}^n$ there exist a Lipschitz constant $L > 0$ and $\varepsilon > 0$ such that $|f(y) - f(z)| \leq L\|y - z\|$ for all $y, z \in B_\varepsilon(x)$.

The subdifferential of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $x \in \mathbb{R}^n$ is the set

$$\partial f(x) = \left\{ \xi \in \mathbb{R}^n : \ f(y) \geq f(x) + \xi^T(y - x), \ \forall y \in \mathbb{R}^n \right\}.$$

Each vector $\xi \in \partial f(x)$ is called a subgradient of f at x. The subdifferential $\partial f(x)$ is a nonempty, convex and compact set such that $\partial f(x) \subseteq B_L(0)$, where $L > 0$ is the Lipschitz constant of f. In addition, the subdifferential mapping $\partial f(x)$ is upper semicontinuous.

For $\varepsilon \geq 0$, the ε-subdifferential of a convex function f at a point $x \in \mathbb{R}^n$ is defined by

$$\partial_\varepsilon f(x) = \left\{ \xi_\varepsilon \in \mathbb{R}^n : \ f(y) \geq f(x) + \xi_\varepsilon^T(y - x) - \varepsilon, \ \forall y \in \mathbb{R}^n \right\}.$$

Each vector $\xi_\varepsilon \in \partial_\varepsilon f(x)$ is called an ε-subgradient of f at x. The set $\partial_\varepsilon f(x)$ contains the subgradient information from some neighborhood of x as the following theorem shows.

Theorem 1 [8] *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function with the Lipschitz constant $L > 0$ at a point $x \in \mathbb{R}^n$. Then for $\varepsilon \geq 0$*

$$\partial f(y) \subseteq \partial_\varepsilon f(x), \quad \forall y \in B_{\frac{\varepsilon}{2L}}(x).$$

The following theorem presents a necessary condition for Problem (1).

Theorem 2 [5, 36] *If $x^* \in \mathbb{R}^n$ is a local minimizer of Problem (1), then*

$$\partial f_2(x^*) \subseteq \partial f_1(x^*). \tag{2}$$

A point x^*, satisfying the condition (2), is called inf-stationary for Problem (1). The optimality condition (2) is, in general, hard to verify and is usually relaxed to the following condition [5, 36]:

$$\partial f_1(x^*) \cap \partial f_2(x^*) \neq \emptyset. \tag{3}$$

A point $x^* \in \mathbb{R}^n$, satisfying (3), is called a critical point of the function f. In a similar way, we can define an ε-critical point. Let $\varepsilon \geq 0$, a point $x^* \in \mathbb{R}^n$ is said to be an ε -critical point of f if [34]

$$\partial_\varepsilon f_1(x^*) \cap \partial_\varepsilon f_2(x^*) \neq \emptyset. \tag{4}$$

3 Augmented subgradient method

In this section we describe the new method and study its convergence. We start with the definition of augmented subgradients of convex functions.

3.1 Augmented subgradients

Let $\varphi : \mathbb{R}^n \to \mathbb{R}$ be a convex function and $U \subset \mathbb{R}^n$ be a compact set. Take any $x, y \in U$. In particular, $y = x$. Let $\xi \in \partial\varphi(y)$ be any subgradient computed at the point y. The linearization error of this subgradient at the point x is defined as:

$$\alpha(x, y, \xi) = \varphi(x) - \Big[\varphi(y) + \xi^T(x - y)\Big] \geq 0. \tag{5}$$

It is clear that

$$\varphi(y) - \varphi(x) = -\alpha(x, y, \xi) + \xi^T(y - x). \tag{6}$$

Definition 1 A vector $u \in \mathbb{R}^{n+1}$ is called an augmented subgradient of the convex function φ at the point x with respect to the compact set $U \subset \mathbb{R}^n$ if there exists $y \in U$ such that $u = \big(\alpha(x, y, \xi), \xi\big)$, where $\xi \in \partial\varphi(y)$ and $\alpha(x, y, \xi)$ is defined by (5).

Note that there is some similarity between augmented subgradients and elements of codifferentials considered in [15, 40].

The set $D_U\varphi(x)$ of all augmented subgradients at the point x with respect to the set U is defined as:

$$D_U\varphi(x) = \operatorname{cl}\operatorname{conv}\Big\{u \in \mathbb{R}^{n+1} : \exists\big(y \in U, \xi \in \partial\varphi(y)\big),\ u = \big(\alpha(x, y, \xi), \xi\big)\Big\}.$$

For $u \in D_U\varphi(x)$ we will use the notation $u = (\alpha, \xi)$ where $\alpha \geq 0$ and $\xi \in \mathbb{R}^n$. If we take $y = x$, then in this case $\alpha = 0$. Therefore, the set $D_U\varphi(x)$ contains also elements of the form $(0, \xi)$ where $\xi \in \partial\varphi(x)$.

Proposition 1 *The set $D_U\varphi(x)$ is compact and convex for any $x \in U$.*

Proof The proof follows from the definition of the set $D_U\varphi(x)$ and from the fact that the subdifferential of the function φ is bounded on bounded sets. □

It follows from (6) that for any $x, y \in U$ we have

$$\varphi(y) - \varphi(x) \leq \max_{u \in D_U\varphi(x)} \Big[-\alpha + \xi^T(y - x)\Big]. \tag{7}$$

Consider the following two sets at the point $x \in U$:

$$A(x) = \Big\{\alpha \geq 0 : \exists\ \big(y \in U,\ \xi \in \partial\varphi(y)\big), (\alpha, \xi) \in D_U\varphi(x)\Big\},$$
$$C(x) = \Big\{\xi \in \mathbb{R}^n : \exists\ \alpha \geq 0, (\alpha, \xi) \in D_U\varphi(x)\Big\}.$$

Take any $x \in \mathbb{R}^n$, $\tau > 0$ and consider the closed ball $\bar{B}_\tau(x)$. Let $U = \bar{B}_\tau(x)$ and $D_U\varphi(x)$ be the set of augmented subgradients at the point x with respect to U.

Proposition 2 *Let $x \in \mathbb{R}^n$, $\tau > 0$, $U = \bar{B}_\tau(x)$ and $L > 0$ be a Lipschitz constant of φ on $\bar{B}_\tau(x)$. Then for $\varepsilon > 2L\tau$ we have*

$$0 \le \alpha \le 2L\tau \quad \forall \alpha \in A(x) \quad \text{and} \quad C(x) \subseteq \partial_\varepsilon \varphi(x).$$

Proof The result for α follows from its definition. Furthermore, Theorem 1 implies that if $\varepsilon > 2L\tau$, then $\partial\varphi(y) \subseteq \partial_\varepsilon \varphi(x)$ for all $y \in \bar{B}_\tau(x)$. Then the proof is obtained from the definition of the set $C(x)$. □

One interesting case is when $U = \{x\}$. In this case $\alpha = 0$, and therefore, we have

$$D_U \varphi(x) \equiv D_x \varphi(x) = \left\{ u \in \mathbb{R}^{n+1} \ : \ u = (0, \xi), \ \xi \in \partial\varphi(x) \right\}.$$

Next we consider the DC function $f(x) = f_1(x) - f_2(x)$. Again assume that $U \subset \mathbb{R}^n$ is a compact set and $x \in U$. Define the set of augmented subgradients $D_U f_1(x)$ for the function f_1 and the set $D_x f_2(x)$ for the function f_2. Construct the set

$$D_U f(x) = D_U f_1(x) - D_x f_2(x).$$

Proposition 3 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a DC function and $U \subset \mathbb{R}^n$ be a compact set. Then at a point $x \in U$ we have*

$$f(y) \le f(x) + \max_{(\alpha,\xi) \in D_U f(x)} \left\{ -\alpha + \xi^T (y - x) \right\}, \ y \in U. \tag{8}$$

Proof Let $y \in U$ be any point. Taking into account the convexity of the function f_2 and applying (6) to the function f_1 at y we get that for $\xi_1 \in \partial f_1(y)$, $\xi_2 \in \partial f_2(x)$ the following holds:

$$\begin{aligned} f(y) &= f_1(y) - f_2(y) \\ &\le f_1(x) - \alpha(x, y, \xi_1) + \xi_1^T (y - x) - \left(f_2(x) + \xi_2^T (y - x) \right) \\ &= \left(f_1(x) - f_2(x) \right) - \alpha(x, y, \xi_1) + (\xi_1 - \xi_2)^T (y - x) \\ &= f(x) + \left(-\alpha(x, y, \xi_1) + (\xi_1 - \xi_2)^T (y - x) \right). \end{aligned}$$

It is clear that $\left(\alpha(x, y, \xi_1), \xi_1 - \xi_2 \right) \in D_U f(x)$, and thus the proof is complete. □

The following corollary can be easily obtained from Proposition 3.

Corollary 1 *Let $x \in \mathbb{R}^n$ and $U = \bar{B}_\tau(x)$, $\tau > 0$. Then for any $d \in S_1$ we have*

$$f(x + \lambda d) \le f(x) + \max_{(\alpha,\xi) \in D_U f(x)} \left\{ -\alpha + \lambda \xi^T d \right\} \ \forall \lambda \in [0, \tau]. \tag{9}$$

3.2 The model

In order to find search direction in Problem (1) we use the following model of the objective function f. Take any point $x \in \mathbb{R}^n$ and the number $\tau \in (0, 1]$. Consider the closed ball $\bar{B}_\tau(x)$. Compute the set $D_U f(x)$ at x for $U = \bar{B}_\tau(x)$. Then we replace the function f on U by the following function:

$$\hat{f}(y) = f(x) + \max_{(\alpha,\xi)\in D_U f(x)} \left\{ -\alpha + \tau \xi^T (y - x) \right\}, \ y \in U. \tag{10}$$

Proposition 4 *The function $\hat{f}$ is convex, $f(y) \le \hat{f}(y)$ for all $y \in U$ and $\hat{f}(x) = f(x)$.*

Proof Convexity of the function $\hat{f}$ follows from its definition. For any $y \in \bar{B}_\tau(x)$ there exist $d \in S_1$ and $\lambda \in [0, \tau]$ such that $y = x + \lambda d$. Then by applying Corollary 1 we get that $f(y) \le \hat{f}(y)$ for all $y \in U$. Finally, for the point x we have

$$\hat{f}(x) = f(x) + \max_{(\alpha,\xi)\in D_U f(x)} \{-\alpha\}.$$

Since $(0, \xi) \in D_U f(x)$ and $\alpha \ge 0$ for all $(\alpha, \xi) \in D_U f(x)$ we have

$$\max_{(\alpha,\xi)\in D_U f(x)} \{-\alpha\} = 0.$$

Therefore, $\hat{f}(x) = f(x)$. □

Proposition 5 *For any $y \in U$ we have*

$$\partial \hat{f}(y) = \operatorname{conv} \left\{ \xi \in \mathbb{R}^n : \ \exists\, \alpha \ge 0 : (\alpha, \xi) \in \hat{R}(y) \right\},$$

where $\hat{R}(y) = \left\{ (\alpha, \xi) \in D_U f(x) : \ \hat{f}(y) = f(x) - \alpha + \xi^T (y - x) \right\}$. In particular,

$$\partial \hat{f}(x) = \operatorname{conv} \left\{ \xi \in \mathbb{R}^n : \ (0, \xi) \in D_U f(x) \right\}. \tag{11}$$

Proof The function $\hat{f}$ is convex and all functions under maximum in its expression are linear with respect to y. Then the expression for the subdifferential $\partial \hat{f}(y)$ follows from Theorem 3.23 [8]. Since $\hat{f}(x) = f(x)$ at the point x we get that $\hat{R}(x) = \left\{ (\alpha, \xi) \in D_U f(x) : \ \alpha = 0 \right\}$. Then we obtain the expression (11) for the subdifferential $\partial \hat{f}(x)$. □

Results from Propositions 4 and 5 show that the set $D_U f(x)$ can be used to find search directions of the DC function f which is demonstrated in the next proposition.

Proposition 6 *Assume that the set $D_U f(x)$ is constructed using $U = \bar{B}_\tau(x)$, $\tau \in (0, 1]$. In addition assume $0_{n+1} \notin D_U f(x)$ and*

$$\|\overline{w}\|^2 = \min\left\{\|w\|^2 : w \in D_U f(x)\right\} > 0, \quad with \quad \overline{w} = (\overline{\alpha}, \overline{\xi}). \tag{12}$$

Then

$$f(x + \tau\overline{d}) - f(x) \leq -\tau\|\overline{w}\|,$$

where $\overline{d} = -\|\overline{w}\|^{-1}\overline{\xi}$.

Proof The set $D_U f(x)$ is compact and convex. Then the necessary condition for a minimum implies that

$$\overline{w}^T(w - \overline{w}) \geq 0, \quad \forall w \in D_U f(x).$$

Taking into account that $\|\overline{w}\|^2 = \overline{\alpha}^2 + \|\overline{\xi}\|^2$ we get

$$\overline{\alpha}^2 + \|\overline{\xi}\|^2 \leq \alpha\overline{\alpha} + \xi^T\overline{\xi}, \quad \forall w = (\alpha, \xi) \in D_U f(x). \tag{13}$$

First we show that $\overline{\xi} \neq 0_n$. Assume the contrary, that is $\overline{\xi} = 0_n$. Since $\overline{w} \neq 0_{n+1}$ it follows that $\overline{\alpha} \neq 0$ and therefore, $\overline{\alpha} > 0$. It follows from (13) that $0 < \overline{\alpha} \leq \alpha$ for all $(\alpha, \xi) \in D_U f(x)$ which is a contradiction since the element $(0, \xi)$ is included in $D_U f(x)$. Therefore $\overline{\xi} \neq 0_n$ which implies that $\overline{d} \neq 0_n$.

Dividing both sides of (13) by $-\|\overline{w}\|$, we obtain

$$-\frac{\alpha\overline{\alpha}}{\|\overline{w}\|} + \xi^T\overline{d} \leq -\|\overline{w}\|, \quad \forall w = (\alpha, \xi) \in D_U f(x). \tag{14}$$

Note that $\|\overline{w}\|^{-1}\overline{\alpha} \in (0, 1)$. Since $\tau \in (0, 1]$ it is obvious that $\|\overline{w}\|^{-1}\tau\overline{\alpha} \in (0, 1)$. Furthermore, since $\alpha \geq 0$ for all $(\alpha, \xi) \in D_U f(x)$ we get

$$\begin{aligned} -\alpha + \tau\xi^T\overline{d} &\leq \frac{-\tau\alpha\overline{\alpha}}{\|\overline{w}\|} + \tau\xi^T\overline{d} \\ &= \tau\left(\frac{-\alpha\overline{\alpha}}{\|\overline{w}\|} + \xi^T\overline{d}\right) \\ &\leq -\tau\|\overline{w}\|, \end{aligned}$$

where the last inequality comes from (14). From here and by applying (9), we obtain that $f(x + \tau\overline{d}) - f(x) \leq -\tau\|\overline{w}\|$. This means that $\overline{d}$ is a descent direction of the function f at the point $x \in \mathbb{R}^n$. □

3.3 The proposed algorithm

Now we introduce the augmented subgradient method (ASM-DC) for solving DC optimization problem (1). Let the search control parameter $c_1 \in (0, 1)$, the line search parameter $c_2 \in (0, c_1]$, the decrease parameter $c_3 \in (0, 1)$ and the tolerance $\theta > 0$ for criticality condition be given.

Algorithm 1: Augmented subgradient method for solving DC optimization problems (ASM-DC).

Data: $c_1 \in (0,1)$ - search control parameter, $c_2 \in (0, c_1]$ - line search parameter, $c_3 \in (0,1)$ - decrease parameter for τ, δ, and $\theta > 0$ - tolerance for criticality condition.

Result: An approximate critical point of Problem (1).

Step 0. (Initialization). Select any starting point $x_0 \in \mathbb{R}^n$, and numbers $\tau_0 > 0$, $\delta_0 > 0$. Set $l = 0$.

Outer iteration

Inner iteration

Step 1. (Initialization). Compute subgradients $\xi_{1l}^1 \in \partial f_1(x_l)$ and $\xi_{2l} \in \partial f_2(x_l)$. Set $\tilde{D}_{1l}^1 = \{(0, \xi_{1l}^1)\}$, $\tilde{D}_{2l} = \{(0, \xi_{2l})\}$, $\hat{D}_l^1 = \tilde{D}_{1l}^1 - \tilde{D}_{2l}$ and $k = 1$.

Step 2. Compute

$$\|(\overline{\alpha}_l^k, \overline{\xi}_l^k)\|^2 = \|\overline{w}_l^k\|^2 = \min\left\{\|w\|^2,\ w \in \hat{D}_l^k\right\}.$$

If

$$\|\overline{w}_l^k\| < \delta_l, \tag{15}$$

then EXIT inner iterations and go to Step 5.

Step 3. (Search direction). Compute $\overline{d}_l^k = -\|\overline{w}_l^k\|^{-1}\overline{\xi}_l^k$.

Step 4. If

$$f(x_l + \tau_l \overline{d}_l^k) - f(x_l) > -c_1 \tau_l \|\overline{w}_l^k\|, \tag{16}$$

then set $d_l^{k+1} = \overline{d}_l^k$, compute $\xi_{1l}^{k+1} \in \partial f_1(x_l + \tau_l d_l^{k+1})$. Set

$$\begin{aligned}
\alpha_l^{k+1} &= f_1(x_l) - f_1(x_l + \tau_l d_l^{k+1}) + \tau_l \left(\xi_{1l}^{k+1}\right)^T d_l^{k+1},\\
\tilde{D}_{1l}^{k+1} &= \operatorname{conv}\left\{\tilde{D}_{1l}^k,\ \left(\alpha_l^{k+1}, \xi_{1l}^{k+1}\right)\right\},\\
\hat{D}_l^{k+1} &= \tilde{D}_{1l}^{k+1} - \tilde{D}_{2l},
\end{aligned}$$

$k = k + 1$ and go to Step 2. Otherwise, EXIT inner iterations and go to Step 6.

Step 5. (Stopping criterion). If $\tau_l \le \theta$ and $\delta_l \le \theta$, then STOP with x_l as the final solution. Otherwise, set $\tau_{l+1} = c_3\tau_l$, $\delta_{l+1} = c_3\delta_l$, $x_{l+1} = x_l$, $l = l + 1$, and go to Step 1.

Step 6. (Line search). Calculate the step-length $\sigma_l \ge \tau_l$ as follows:

$$\sigma_l = \operatorname{argmax}\left\{\sigma \ge 0 :\ f(x_l + \sigma \overline{d}_l^k) - f(x_l) \le -c_2 \sigma \|\overline{w}_l^k\|\right\}.$$

Step 7. Set $x_{l+1} = x_l + \sigma_l \overline{d}_l^k, \tau_{l+1} = \tau_l, \delta_{l+1} = \delta_l, l = l + 1$ and go to Step 1.

Some explanations to Algorithm 1 are essential. This algorithm consists of outer and inner loops. In the inner loop we compute augmented subgradients of the DC components and the subset $\hat{D}_l^k$ of the set of augmented subgradients (Steps 1 and

4). We use this subset to formulate a quadratic programming problem to find search directions (Step 2). This problem can be rewritten as:

$$\begin{cases} \text{minimize} & \|w\|^2 \\ \text{subject to} & w \in \hat{D}_l^k. \end{cases}$$

Here the set $\hat{D}_l^k$ is a polytope. There are several algorithms for solving such problems [21, 29, 39]. The algorithm exits the inner loop when either the distance between the set of augmented subgradients and the origin is sufficiently small (that is the condition (15) is satisfied), or we find the direction of sufficient decrease (Step 4). In the first case the current value of the parameter τ does not allow to improve the solution and should be updated. In the second case the line search procedure is applied.

In the outer loop we update parameters of the method (Step 5) and apply the line search procedure to find a new solution (Step 6). Since it is not easy to find the exact value of the step-length σ_l in Step 6 we propose the following scheme to estimate it in the implementation of the algorithm. It is clear that $\sigma_l \geq \tau_l$. Consider the sequence $\{\bar{\sigma}_i\},\ i = 1, 2, \ldots$ where $\bar{\sigma}_1 = \tau_l$ and $\bar{\sigma}_i = 2^{i-1}\bar{\sigma}_1,\ i = 1, 2, \ldots$. We compute the largest $i \geq 1$ satisfying the condition in Step 6. Then $\bar{\sigma}_l = 2^{i-1}\tau_l$ is the estimate of σ_l. The algorithm stops when values of its parameters become sufficiently small (Step 5). This means that an approximate criticality condition is achieved.

3.4 Global convergence

In this section we study the global convergence of Algorithm 1. First, we prove that for each $l \geq 0$ the number of inner iterations of this algorithm is finite. At the point $x_l \in \mathbb{R}^n$ for a given $\tau_l > 0,\ l > 0$ define the following numbers:

$$K_l = \max\left\{\|w\| :\ w \in D_U f(x_l)\right\},\ \ M_l = 1 - \left((1 - c_1)(2K_l)^{-1}\delta_l\right)^2. \tag{17}$$

It follows from Proposition 2 that $K_l < \infty$ for all $l > 0$.

Proposition 7 *Let $\delta_l \in (0, K_l), l > 0$. Then the inner loop at the l-th iteration of the ASM-DC stops after m_l iterations, where*

$$m_l \leq \frac{2\log_2(\delta_l/K_l)}{\log_2(M_l)} + 1.$$

Proof We proof the proposition by contradiction. This means that at the l-th iteration for any $k \geq 1$ the condition (15) is not satisfied and the condition (16) is satisfied, that is $\|\overline{w}_l^k\| \geq \delta_l$ and

$$f(x_l + \tau_l d_l^{k+1}) - f(x_l) > -c_1\tau_l\|\overline{w}_l^k\|,\ \forall k \geq 1. \tag{18}$$

First, we show that the new augmented subgradient

$$w_l^{k+1} \equiv (\alpha_l^{k+1}, \xi_l^{k+1}) = (\alpha_l^{k+1}, \xi_{1l}^{k+1} - \xi_{2l}) = (\alpha_l^{k+1}, \xi_{1l}^{k+1}) - (0, \xi_{2l}),$$

computed at Step 4 does not belong to the set $\hat{D}_l^k$. Assume the contrary, that is $w_l^{k+1} \in \hat{D}_l^k$. It follows from the definition (5) of the linearization error that

$$f_1(x_l + \tau_l d_l^{k+1}) - f_1(x_l) = -\alpha_l^{k+1} + \tau_l \left(\xi_{1l}^{k+1}\right)^T d_l^{k+1}. \tag{19}$$

Since $\xi_{2l} \in \partial f_2(x_l)$ the subgradient inequality implies that

$$f_2(x_l + \tau_l d_l^{k+1}) - f_2(x_l) \geq \tau_l \xi_{2l}^T d_l^{k+1}.$$

Subtracting this from (19) we get

$$f(x_l + \tau_l d_l^{k+1}) - f(x_l) \leq \tau_l \left(\xi_{1l}^{k+1} - \xi_{2l}\right)^T d_l^{k+1} - \alpha_l^{k+1}.$$

Applying (18) and taking into account that $\xi_l^{k+1} = \xi_{1l}^{k+1} - \xi_{2l}$, we have

$$-c_1 \tau_l \|\overline{w}_l^k\| < \tau_l \left(\xi_l^{k+1}\right)^T d_l^{k+1} - \alpha_l^{k+1},$$

or

$$-c_1 \|\overline{w}_l^k\|^2 < -\left(\xi_l^{k+1}\right)^T \overline{\xi}_l^k - \frac{\alpha_l^{k+1}}{\tau_l} \|\overline{w}_l^k\|,$$

which, by taking into account that $c_1 \in (0, 1)$, can be rewritten as

$$\left(\xi_l^{k+1}\right)^T \overline{\xi}_l^k + \frac{\alpha_l^{k+1}}{\tau_l} \|\overline{w}_l^k\| < c_1 \|\overline{w}_l^k\|^2 < \|\overline{w}_l^k\|^2. \tag{20}$$

On the other side, since $\|\overline{w}_l^k\|^2 = \min\left\{\|w\|^2 : w \in \hat{D}_l^k\right\}$ it follows from the necessary condition for a minimum that $\|\overline{w}_l^k\|^2 \leq w^T \overline{w}_l^k$ for all $w \in \hat{D}_l^k$, where $w = (\alpha, \xi)$ and $\overline{w}_l^k = (\overline{\alpha}_l^k, \overline{\xi}_l^k)$. Hence we obtain

$$\|\overline{w}_l^k\|^2 \leq \overline{\alpha}_l^k \alpha + \left(\overline{\xi}_l^k\right)^T \xi, \quad \forall w = (\alpha, \xi) \in \hat{D}_l^k.$$

In particular, for $w_l^{k+1} = (\alpha_l^{k+1}, \xi_l^{k+1}) \in \hat{D}_l^k$ we have $\|\overline{w}_l^k\|^2 \leq \overline{\alpha}_l^k \alpha_l^{k+1} + \left(\overline{\xi}_l^k\right)^T \xi_l^{k+1}$. Since $\overline{\alpha}_l^k \leq \|\overline{w}_l^k\|$ and $\alpha_l^{k+1} \geq 0$ we get $\overline{\alpha}_l^k \alpha_l^{k+1} \leq \alpha_l^{k+1} \|\overline{w}_l^k\|$. Next taking into account that $\tau_l \in (0, 1)$, we have

$$\|\overline{w}_l^k\|^2 \leq \alpha_l^{k+1} \|\overline{w}_l^k\| + \left(\overline{\xi}_l^k\right)^T \xi_l^{k+1} \leq \frac{\alpha_l^{k+1}}{\tau_l} \|\overline{w}_l^k\| + \left(\overline{\xi}_l^k\right)^T \xi_l^{k+1},$$

which contradicts (20). Thus, $w_l^{k+1} \notin \hat{D}_l^k$.

Note that for all $t \in [0, 1]$ we have

$$\|\overline{w}_l^{k+1}\|^2 \le \|tw_l^{k+1} + (1-t)\overline{w}_l^k\|^2$$
$$= \|\overline{w}_l^k\|^2 + 2t\left(\overline{w}_l^k\right)^T (w_l^{k+1} - \overline{w}_l^k) + t^2\|w_l^{k+1} - \overline{w}_l^k\|^2.$$

It follows from (20) that

$$\left(\overline{w}_l^k\right)^T w_l^{k+1} = \overline{\alpha}_l^k \alpha_l^{k+1} + \left(\overline{\xi}_l^k\right)^T \xi_l^{k+1}$$
$$< \frac{\alpha_l^{k+1}}{\tau_l}\|\overline{w}_l^k\| + \left(\overline{\xi}_l^k\right)^T \xi_l^{k+1}$$
$$< c_1\|\overline{w}_l^k\|^2.$$

In addition, the definition of K_l implies that $\|w_l^{k+1} - \overline{w}_l^k\| \le 2K_l$. Then we get

$$\|\overline{w}_l^{k+1}\|^2 < \|\overline{w}_l^k\|^2 + 2tc_1\|\overline{w}_l^k\|^2 - 2t\|\overline{w}_l^k\|^2 + 4K_l^2t^2$$
$$= \|\overline{w}_l^k\|^2 - 2t(1-c_1)\|\overline{w}_l^k\|^2 + 4K_l^2t^2.$$

Let $t_0 = (1-c_1)(2K_l)^{-2}\|\overline{w}_l^k\|^2$. Note that $t_0 \in (0,1)$, and for $t = t_0$ we have

$$\|\overline{w}_l^{k+1}\|^2 < \left(1 - \left((1-c_1)(2K_l)^{-1}\|\overline{w}_l^k\|\right)^2\right)\|\overline{w}_l^k\|^2.$$

Since $\|\overline{w}_l^k\| \ge \delta_l$ we get

$$\|\overline{w}_l^{k+1}\|^2 < \left(1 - \left((1-c_1)(2K_l)^{-1}\delta_l\right)^2\right)\|\overline{w}_l^k\|^2.$$

Furthermore, since $\delta_l \in (0, K_l)$ it follows from (17) that $M_l \in (0,1)$ and $\|\overline{w}_l^{k+1}\|^2 < M_l\|\overline{w}_l^k\|^2$. Then we have

$$\|\overline{w}_l^{m_l}\|^2 < M_l\|\overline{w}_l^{m_l-1}\|^2 < \ldots < M_l^{m_l-1}\|\overline{w}_l^1\|^2 < M_l^{m_l-1}K_l^2.$$

Hence $\|\overline{w}_l^{m_l}\| < \delta_l$ is satisfied if $M_l^{m_l-1}K_l^2 < \delta_l^2$. This inequality must happen after at most m_l iterations, where

$$m_l \le \frac{2\log_2(\delta_l/K_l)}{\log_2(M_l)} + 1.$$

This completes the proof. □

Definition 2 A point $x \in \mathbb{R}^n$ is called a (τ,δ)-critical point of Problem (1) if

$$\min\left\{\|w\| \ : \ w \in D_U f(x)\right\} \le \delta.$$

Proposition 8 *Let $\tau, \delta > 0$ be given numbers, $x \in \mathbb{R}^n$ be a (τ,δ)-critical point of Problem (1) and $L_1 > 0$ be a Lipschitz constant of the function f_1 at the point x. Then there exists $\xi_2 \in \partial f_2(x)$ such that for any $\varepsilon > 2L_1\tau$*

$$\xi_2 \in \partial_\varepsilon f_1(x) + B_\delta(0_n).$$

Proof Let $U = \bar{B}_\tau(x)$. If x is the (τ, δ)-critical point, then there exists $\overline{w} \in D_U f(x)$ such that $\|\overline{w}\| < \delta$. The element $\overline{w}$ can be represented as

$$\overline{w} = (\overline{\alpha}, \overline{\xi}_1) - (0, \overline{\xi}_2) = (\overline{\alpha}, \overline{\xi}_1 - \overline{\xi}_2),$$

for some $(\overline{\alpha}, \overline{\xi}_1) \in D_U f_1(x)$ and $(0, \overline{\xi}_2) \in D_x f_2(x)$. Hence, $\|\overline{\xi}_1 - \overline{\xi}_2\| < \delta$. It follows from Proposition 2 that $\overline{\xi}_1 \in \partial_\varepsilon f_1(x)$ for all $\varepsilon > 2L_1\tau$. In addition $\overline{\xi}_2 \in \partial f_2(x)$. Then we get the proof. □

It follows from Proposition 8 that for sufficiently small τ and δ the (τ, δ)-critical point can be interpreted as an approximate critical point of Problem (1). Furthermore, note that if at the l-th iteration ($l > 0$) of Algorithm 1 the inner loop terminates with the condition (15), then x_l is an (τ_l, δ_l)-critical point of Problem (1).

Next, we prove the convergence of Algorithm 1. For a starting point $x_0 \in \mathbb{R}^n$ consider the level set $\mathcal{L}_f(x_0)$ of the function f:

$$\mathcal{L}_f(x_0) = \left\{x \in \mathbb{R}^n \,:\, f(x) \le f(x_0)\right\}.$$

Theorem 3 *Suppose that the level set $\mathcal{L}_f(x_0)$ is bounded and $\theta = 0$ in Algorithm* 1. *Then Algorithm* 1 *generates the sequence of approximate critical points whose accumulation points are critical points of Problem* (1).

Proof The ASM-DC consists of inner and outer loops. The values of parameters τ_l and δ_l are constants in the inner loop. In addition, these parameters are also constants in Steps 6 and 7. It follows from Proposition 7 that for given τ_l and δ_l, the inner loop stops after a finite number of iterations.

Since the set $\mathcal{L}_f(x_0)$ is bounded (and compact) the function f is bounded from below. This means that $f_* = \min\left\{f(x) \,:\, x \in \mathbb{R}^n\right\} > -\infty$. For given τ_l and δ_l the number of executions of Steps 6 and 7 is finite. To prove this we assume the contrary, that is the number of executions of these steps is infinite. Then for some $k \ge 1$ the direction $\overline{d}_l^k$, computed in Step 3, is the direction of sufficient decrease at x_l satisfying the following condition:

$$f(x_l + \tau_l \overline{d}_l^k) - f(x_l) \le -c_1\tau_l\|\overline{w}_l^k\|.$$

Since $\|\overline{w}_l^k\| \ge \delta_l$, $\sigma_l \ge \tau_l$ and $c_2 \in (0, c_1]$ we have

$$f(x_{l+1}) - f(x_l) = f(x_l + \sigma_l \overline{d}_l^k) - f(x_l) \le -c_2\sigma_l\|\overline{w}_l^k\| \le -c_2\tau_l\delta_l.$$

From here we get

$$f(x_{l+1}) \le f(x_0) - (l+1)c_2\tau_l\delta_l.$$

Then we have $f(x_l) \to -\infty$ as $l \to \infty$ which contradicts to the condition $f_* > -\infty$. This means that for fixed values of τ_l and δ_l, after finite number of iterations the inner loop will terminate with the condition (15) and x_l is the (τ_l, δ_l)-critical point.

Thus parameters τ_l and δ_l are updated in Step 5 after a finite number of outer iterations. Denote these iterations by $l_j, j = 1, 2, \ldots$ and x_{l_j} are $(\tau_{l_j}, \delta_{l_j})$-critical point. It follows from Proposition 8 that for some $\xi_{2l_j} \in \partial f_2(x_{l_j})$

$$\xi_{2l_j} \in \partial_\varepsilon f_1(x_{l_j}) + B_{\delta_{l_j}}(0_n),\ \varepsilon > 2L_1\tau_{l_j}. \tag{21}$$

It is obvious that $x_{l_j} \in \mathcal{L}_f(x_0)$ for all $j = 1, 2, \ldots$. The compactness of the set $\mathcal{L}_f(x_0)$ implies that the sequence $\{x_{l_j}\}$ has at least one accumulation point. Let x^* be an accumulation point and for the sake of simplicity assume that $x_{l_j} \to x^*$ as $j \to \infty$.

Upper semicontinuity of the subdifferential and ε-subdifferential mappings implies that for any $\eta > 0$ there exist $\gamma_1 > 0$ and $\gamma_2 > 0$ such that

$$\partial_\varepsilon f_1(y) \subset \partial f_1(x^*) + B_\eta(0_n) \quad \text{and} \quad \partial f_2(y) \subset \partial f_2(x^*) + B_\eta(0_n), \tag{22}$$

for all $y \in B_{\gamma_1}(x^*)$ and $\varepsilon \in [0, \gamma_2)$. Since $x_{l_j} \to x^*$ and $\tau_{l_j} \to 0$ as $j \to \infty$ there exists $j_\eta > 0$ such that $x_{l_j} \in B_{\gamma_1}(x^*)$ and $\tau_{l_j} \in [0, \gamma_2)$ for all $j > j_\eta$. Moreover, we can choose the sequence $\{\varepsilon_j\}$ so that $\varepsilon_j \in [0, \gamma_2)$ for all $j > j_\eta$ and $\varepsilon_j \to 0$ as $j \to \infty$. Then, it follows from (21) and the first part of (22) that for any $j > j_\eta$ there exists $\xi_{2l_j} \in \partial_\varepsilon f_1(x_{l_j})$ such that

$$\xi_{2l_j} \in \partial f_1(x^*) + B_{\eta+\delta_{l_j}}(0_n).$$

On the other hand, from the second part of (22) we get that for any $\xi_{2l_j} \in \partial f_2(x^*)$ there exists $\bar{\xi}_2 \in \partial f_2(x^*)$ such that $\|\xi_{2l_j} - \bar{\xi}_2\| < \eta$. Without loss of generality assume that $\xi_{2l_j} \to \bar{\xi}_2 \in \partial f_2(x^*)$ as $j \to \infty$. Since $\delta_{l_j} \to 0$ as $j \to \infty$ we get that

$$\bar{\xi}_2 \in \partial f_1(x^*) + B_{2\eta}(0_n).$$

Taking into account that $\eta > 0$ is arbitrary we have $\bar{\xi}_2 \in \partial f_1(x^*)$, that is $\partial f_1(x^*) \cap \partial f_2(x^*) \neq \emptyset$. This completes the proof. □

Remark 1 The "escape procedure", introduced in [25], is able to escape from critical points which are not Clarke stationary. The developed algorithm can be combined with this procedure to design an algorithm for finding Clarke stationary points of Problem (1).

4 Numerical results

Using some academic test problems we evaluate the performance of the developed method and compare it with other similar nonsmooth optimization methods. We utilize 30 academic test problems which belong to 14 instance problems with various dimensions (ranging from 2 to 100): Problems 1-10 are from [24] and

Problems 11–14 are described in [26]. In addition, we use three large-scale nonsmooth DC optimization problems in our numerical experiments. We design these problems such that different types of DC components are considered:

- P_{L1}: the function f_1 is smooth and the function f_2 is nonsmooth;
- P_{L2}: the function f_1 is nonsmooth and the function f_2 is smooth;
- P_{L3}: both functions f_1 and f_2 are nonsmooth.

The DC components of these test problems, $f(x) = f_1(x) - f_2(x)$, are given below:

- P_{L1}.

$$f_1(x) = \sum_{i=1}^{n-1} \Big((x_i - 2)^2 + (x_{i+1} - 1)^2 + \exp(2x_i - x_{i+1}) \Big),$$

$$f_2(x) = \sum_{i=1}^{n-1} \max \Big((x_i - 2)^2 + (x_{i+1} - 1)^2, \exp(2x_i - x_{i+1}) \Big).$$

  Starting point: $x = (x_1, \ldots, x_n) = (3, \ldots, 3)$.
- P_{L2}.

$$f_1(x) = n \max_{i=1,\ldots,n} \Big(\sum_{j=1}^{n} \frac{x_j^2}{i+j-1} \Big),$$

$$f_2(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{x_j^2}{i+j-1}.$$

  Starting point: $x = (x_1, \ldots, x_n) = (2, \ldots, 2)$.
- P_{L3}.

$$f_1(x) = n \max_{i=1,\ldots,n} \Big| \sum_{j=1}^{n} \frac{x_j}{i+j-1} \Big|,$$

$$f_2(x) = \sum_{i=1}^{n} \Big| \sum_{j=1}^{n} \frac{x_j}{i+j-1} \Big|.$$

  Starting point: $x = (x_1, \ldots, x_n) = (2, \ldots, 2)$.

We set the number of variables to $n = 200, 500, 1000, 1500, 2000, 3000, 5000$ for these three test problems.

4.1 Solvers and parameters

We apply seven nonsmooth optimization solvers for comparison. Among these solvers five are designed for solving DC optimization problems and two of them are general solvers for nonsmooth nonconvex optimization problems. These solvers are:

– Aggregate subgradient method for DC optimization (AggSub) [9];
– Proximal bundle method for DC optimization (PBDC) [24];
– DC Algorithm (DCA) [4, 5];
– Proximal bundle method for nonsmooth DC programming (PBMDC) [17];
– Sequential DC programming with cutting-plane models (SDCA-LP) [18];
– Proximal bundle method (PBM) [30, 31];
– Hybrid Algorithm for Nonsmooth Optimization (HANSO 2.2) [14, 28].

The quadratic programming problem in Step 2 of the ASM-DC is solved using the algorithm from [39]. Parameters in the ASM-DC are chosen as follows: $c_1 = 0.2$, $c_2 = 0.05$, $c_3 = 0.1$, $\theta = 5 \cdot 10^{-7}$, $\tau_{l+1} = 0.1\tau_l$ with $\tau_1 = 1$, $\delta_l = 10^{-7}$ for all l. The algorithms ASM-DC, AggSub, PBDC, DCA, PBM are implemented in Fortran 95 and compiled using the gfortran compiler. For HANSO, we use the MATLAB implementation of HANSO version 2.2 which is available in

"https://cs.nyu.edu/overton/software/hanso/". For PBMDC and SDCA-LP, we utilize the MATLAB implementations which are available in

"http://www.oliveira.mat.br/solvers". Note that the SDCA-LP requires the feasible set to be compact. However, all test problems considered in this paper are unconstrained. In order to apply the SDCA-LP to these problems we define a large n-dimensional box around the starting point and consider only those points, generated by the SDCA-LP, that belong to this box. The parameters of all these algorithms are chosen according to the values recommended in their references. In addition, we consider a time limit for all algorithms, that is three hours for solvers in MATLAB and half an hour for implemented in Fortran.

4.2 Evaluation measures and notations

We use the number of function evaluations, the number of subgradient evaluations and the final value of the objective function obtained by algorithms to report the results of our experiments. The following notations are used:

– P is the label of a test problem;
– n is the number of variables in the problem;
– f^s_{best} is the best value of the objective function obtained by a solver "s".

We say that a solver "s" finds a solution with respect to a tolerance β_1 if

$$0 \leq \frac{f^s_{best} - f_{opt}}{|f_{opt}| + 1} \leq \beta_1, \tag{23}$$

where f_{opt} is the value of the objective function at one of the known local minimizers. Otherwise, we say that the solver fails. We set $\beta_1 = 10^{-4}$.

Performance profiles, introduced in [16], are also applied to analyze the results. Recall that in the performance profiles, the value of $\rho_s(\tau)$ at $\tau = 0$ shows the ratio of the number of test problems, for which the solver s uses the least computational time, function or subgradient evaluations, to the total number of test problems. The

value of $\rho_s(\tau)$ at the rightmost abscissa gives the ratio of the number of test problems, that the solver s can solve, to the total number of test problems and this value indicates the robustness of the solver s. Furthermore, the higher is a particular curve, the better is the corresponding solver. Note that the values of τ are scaled using the natural logarithm.

In addition, we report the CPU time — the computational time (in seconds) — required by the developed algorithm to find a solution, and also results on the quality of solutions obtained by different algorithms. The quality of solutions is determined by the values of the objective function at these solutions. More precisely, consider two different local minimizers x_1 and x_2 of Problem (1). The local minimizer x_1 is a better quality solution than the local minimizer x_2 if $f(x_1) < f(x_2)$. The notation (S, B, W) is used to report the total number of test problems that the ASM-DC finds the similar (S), the better (B) and the worse (W) quality solutions than other algorithms. The accuracy of solutions are defined using (23), where $\beta_1 = 10^{-4}$.

4.3 Results

We present the results of our numerical experiments as follows. First, we give the results obtained by algorithms for the test problems 1–14 with one starting point that are given in [24, 26]. We report the values of objective functions, the number of function evaluations, and the number of subgradient evaluations. Second, we present the results for the values of objective functions, the number of function evaluations, and the number of subgradient evaluations obtained for these test problems with 20 random starting points. Next, we give the average CPU time required by the ASM-DC on the test problems 1–14 with both one and 20 starting points, and also discuss the quality of solutions obtained by the algorithms. Finally, we present the results for three large-scale test problems introduced above and report the values of objective functions, the number of function evaluations, and the number of subgradient evaluations required by the algorithms.

Numerical results for Problems 1–14 with one starting point. We summarize these results in Tables 1–3 and Fig. 1. Table 1 presents the best values of the objective function obtained by solvers. We use "*" for those values that a solver fails to obtain the local or global minimizer. Results from this table show that the ASM-DC is more reliable to find the global minimizer than all other solvers. It is able to find the optimal solutions for all test problems considered in our experiment, whereas other solvers fail for some test problems with different number of variables. More specifically, the AggSub fails to find the optimal solution in Problem 5, and also Problem 14 with $n = 5, 10, 50, 100$. The PBDC fails in Problems 5 and 13, and also Problem 12 with $n = 50, 100$. The DCA fails in Problem 10 with $n = 5, 10, 50, 100$. The PBMDC fails in Problems 7 and 13, Problem 12 with $n = 5, 10, 50, 100$, and Problem 14 with $n = 100$. The SDCA-LP fails in Problems 1, 8, 9, 13, Problem 10 with $n = 50, 100$, and Problem 12 with $n = 10, 50, 100$. The PBM fails in Problems 10 and 12 for $n = 100$, and also Problem 14 with $n = 10, 50$. The HANSO fails in Problems 2 and 3, and also Problem 14 with $n = 2, 10, 50, 100$.

Table 1 Best values of objective functions for Problems 1–14 obtained by solvers

P	n	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP	PBM	HANSO
1	2	2.000000	2.000000	2.000000	2.000012	2.000000	20.000000*	2.000001	2.000000
2	2	0.000000	0.000001	0.000000	0.000000	0.000000	0.000000	1.000000	2.584568*
3	4	0.000008	0.000003	0.000000	0.000000	0.000000	2.000000	0.000000	42.806069*
4	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	10	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	50	0.000000	0.000007	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	100	0.000000	0.000001	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	20	0.000019	0.603319*	190.000000*	0.000004	0.000001	0.000014	0.000000	0.000000
6	2	-2.500000	-2.499999	-2.499999	-2.500000	-2.500000	-2.499999	-2.500000	-2.500000
7	2	0.500000	0.500000	0.500000	0.500007	1.001089*	0.500000	1.000000	2.514550*
8	3	3.500000	3.500000	3.500000	3.750002	3.500000	3.772742*	3.750003	3.750000
9	4	9.200000	9.200000	1.833333	9.200000	1.833333	9.206336*	9.200000	9.199999
10	2	-0.500000	-0.499999	-0.499999	-0.500000	-0.500000	-0.499992	-0.499999	-0.500000
10	5	-2.500000	-2.499999	-2.499999	-0.500000*	-2.500000	-3.499989	-2.499998	-2.499999
10	10	-8.500000	-8.499999	-8.499999	1.500000*	-8.500000	-8.499977	-8.499997	-8.499999
10	50	-48.500000	-42.499999	-48.499894	33.500000*	-48.500000	424.350000*	-44.499996	-48.499999
10	100	-98.500000	-54.500000	-98.499976	30.500000*	-98.500000	3.373600×10^{3}*	-7.080729*	-98.499999
11	3	116.333333	116.333334	116.333333	116.333333	116.333333	116.333333	116.333333	116.333334
12	2	0.618035	0.618035	1.618033	0.618034	1.618088	1.618035	0.618035	0.618035
12	5	0.618035	0.618036	1.608033	0.618034	0.620529*	0.618034	1.618039	0.618034
12	10	0.618035	0.618038	0.618034	0.618033	0.622253*	65.114310*	1.618038	0.618034
12	50	0.618043	0.618038	430.746179*	0.618040	0.627830*	8.163098×10^{5}*	0.618040	0.618033

Table 1 (continued)

P	n	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP	PBM	HANSO
12	100	0.618052	0.618040	422.561066*	0.618042	0.632189*	7.960967×10^{6}*	17.497692*	0.618033
13	10	0.000000	-0.000000	0.186079*	0.000000	56.250000*	990.000000*	0.000000	0.000000
14	2	0.000000	0.000000	0.000000	0.000000	0.000001	0.000001	0.000000	0.003192*
14	5	0.000000	0.013716*	0.000000	0.000004	0.000007	0.000013	0.000000	0.000007
14	10	0.000000	0.156217*	0.000000	0.000000	0.000006	0.000032	0.000233*	0.349793*
14	50	0.000000	0.530519*	0.000000	0.000003	0.000024	0.000051	0.000150*	0.034229*
14	100	0.000005	16.091961*	0.000000	0.000001	0.000299*	0.000057	0.000013	3.214774*

Tables 2 and 3 present the number of function and subgradient evaluations for Problems 1–14, respectively. We denote by "–" the cases that a solver fails to find a solution. It can be observed from these tables that, these numbers required by the ASM-DC are reasonable and they are comparable with those required by the AggSub and HANSO algorithms. In most cases the ASM-DC requires more number of function and subgradient evaluations than PBDC, DCA, PBMDC, SDCA-LP and PBM algorithms. This is due to the fact that the ASM-DC requires the large number of inner iterations since it does not use augmented subgradients from previous iterations.

Table 2 Number of function evaluations for Problems 1–14

P	n	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP	PBM	HANSO
1	2	189	344	22	87	10	–	12	143
2	2	109	170	21	8	3	7	26	–
3	4	149	4979	25	11	5	15	26	–
4	2	43	50	6	6	2	5	5	34
4	5	126	191	13	10	3	12	22	130
4	10	256	610	16	31	5	25	52	314
4	50	3208	2972	52	64	25	103	527	2661
4	100	10878	13986	102	152	50	204	1354	5182
5	20	436	–	–	5058	16	48	69	533
6	2	135	108	22	10	17	29	4	5
7	2	388	1459	72	190	–	49	22	–
8	3	271	165	75	18	11	–	11	9
9	4	199	157	85	3	17	–	4	42
10	2	173	108	19	6	2	30	10	3
10	5	202	147	20	–	3	292	12	8
10	10	216	142	55	–	11	1650	40	40
10	50	389	223	138	–	35	–	35	182
10	100	484	259	348	–	77	–	–	315
11	3	174	3109	10	11	5	13	20	252
12	2	290	157	20	13	7	36	69	69
12	5	457	390	58	22	–	695	88	211
12	10	831	456	1415	34	–	–	159	346
12	50	3015	1733	–	96	–	–	688	2865
12	100	6998	1855	–	116	–	–	–	5391
13	10	26	62	–	13	–	–	7	40
14	2	50	56	9	6	11	28	8	–
14	5	206	–	178	16	22	57	70	110
14	10	357	–	103565	12	33	107	–	–
14	50	672	–	100988	15	58	280	–	–
14	100	1210	–	131661	17	–	552	183	–

Table 3 Number of subgradient evaluations for Problems 1–14

P	n	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP	PBM	HANSO
1	2	51	174	16	80	17	–	12	143
2	2	41	69	15	6	5	7	26	–
3	4	53	2479	13	10	7	15	26	–
4	2	19	25	3	4	2	5	5	34
4	5	44	85	5	9	4	12	22	130
4	10	87	231	10	14	8	25	52	314
4	50	1047	1479	31	63	38	103	527	2661
4	100	3539	6704	66	151	75	204	1354	5182
5	20	145	–	–	5057	26	48	69	533
6	2	42	56	13	3	28	29	4	5
7	2	111	716	46	178	–	49	22	–
8	3	75	81	45	16	17	–	11	9
9	4	62	76	57	3	23	–	4	42
10	2	44	55	9	4	2	30	10	3
10	5	67	66	10	–	5	292	12	8
10	10	69	67	32	–	18	1650	40	40
10	50	127	88	86	–	54	–	35	182
10	100	153	102	216	–	117	–	–	315
11	3	60	1539	7	9	7	13	20	252
12	2	78	77	14	11	10	36	69	69
12	5	122	186	38	21	5	695	88	211
12	10	257	219	1256	33	–	–	159	346
12	50	978	836	–	94	–	–	688	2865
12	100	2306	907	–	115	–	–	–	5391
13	10	13	38	–	12	–	–	7	40
14	2	23	32	5	4	16	28	8	–
14	5	72	–	172	14	31	57	70	110
14	10	107	–	103533	10	50	107	–	–
14	50	211	–	100936	14	88	280	–	–
14	100	383	–	125166	16	–	552	183	–

Next, we demonstrate efficiency and robustness of the algorithms using the performance profiles. The results with one starting point are depicted in Fig. 1. We utilize the number of function evaluations required by algorithms to draw the performance profiles. The number of subgradient evaluations follow the similar trends with that of the number of function evaluations, and thus, we omit these results. For the algorithms based on the DC structure of the problem, we use the average value of the number of the function evaluations of the DC components. For the other algorithms, we use the number of function evaluations of the objective function. From Fig. 1, it can be observed that the ASM-DC is the most robust for the collection of test problems used in the numerical experiments. However, it requires more function

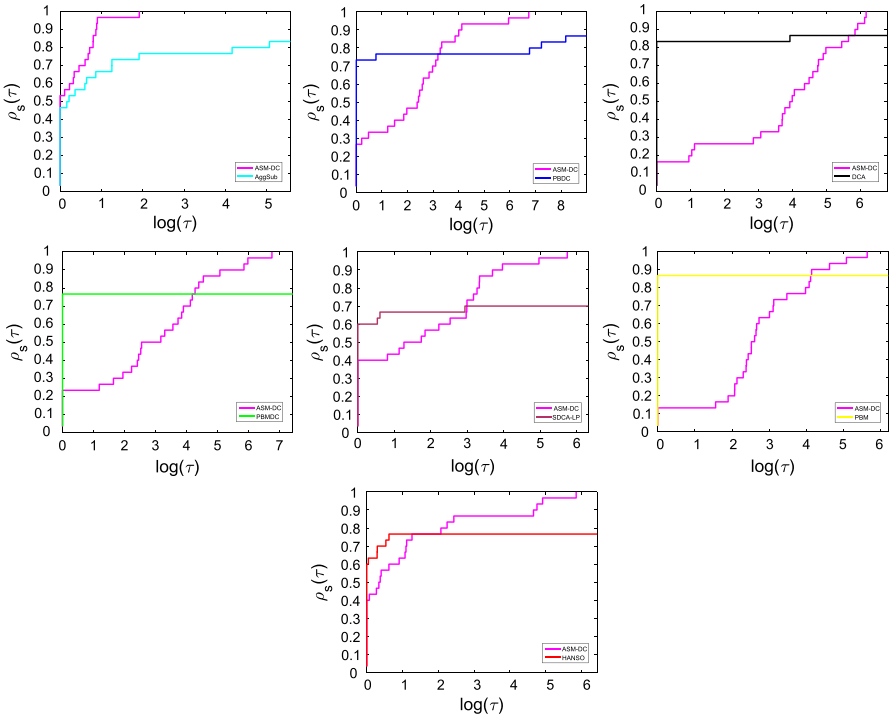

Fig. 1 Number of function evaluations for Problems 1–14 with one starting point

evaluations than others with the exception of the AggSub algorithm. This is due to the fact that the ASM-DC requires the large number of inner iterations because it does not use augmented subgradients from previous iterations.

Numerical results for Problems 1–14 with 20 random starting points. Performance profiles using Problems 1–14 with 20 random starting points and the number of function evaluations are presented in Fig. 2. The results from this figure indicate a similar trend to those of Fig. 1 which confirm the results obtained by algorithms with only one starting point. Performance profiles using the number of subgradient evaluations follow the similar trends with the number of function evaluations for all solvers, and therefore, we omit these results.

Computational time and results on the quality of solutions. In Table 4 for a given number n of variables we report the average CPU time in seconds required by the ASM-DC for Problems 1–14 (30 problems considering different variables) with one starting point and also 20 random starting points. We can see that the CPU time required by the ASM-DC for test problems with the number of variables $n = 2, 3, 4$ is zero. We do not include the CPU time required by other algorithms since they are implemented on different platforms.

Table 5 presents the comparison of the quality of solutions obtained by the ASM-DC with those obtained by other algorithms. Problems 1–14 (30 problems considering different number of variables) with 20 random starting points are used (note that the total number of cases is 600). We can see that the ASM-DC

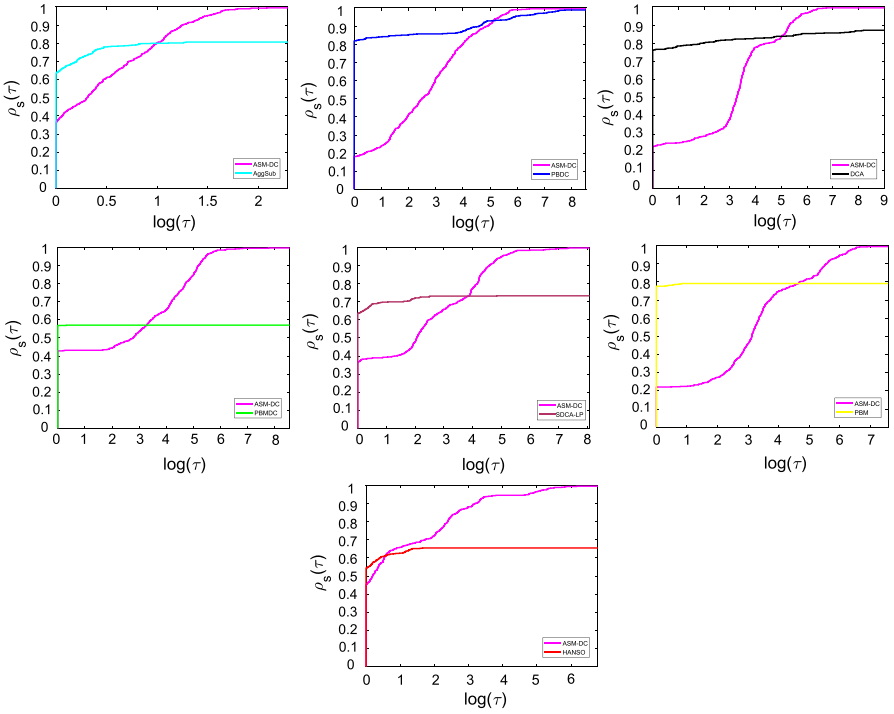

Fig. 2 Number of function evaluations for Problems 1–14 with 20 random starting points

Table 4 CPU time (in seconds) required by ASM-DC

n	2	3	4	5	10	20	50	100
One starting point	0.000	0.000	0.000	0.004	0.022	0.000	0.320	12.402
20 starting points	0.000	0.000	0.000	0.004	0.017	0.059	2.106	29.869

Table 5 Comparison of the quality of solutions obtained by ASM-DC and other algorithms

	AggSub	PBDC	DCA	PBMDC	SDCA-LP	PBM	HANSO
ASM-DC	(446,131,23)	(493,39,68)	(435,141,24)	(265,277,58)	(384,168,48)	(386,187,27)	(322,238,40)

outperforms all other solvers, except the PBDC, in terms of its ability to find high quality solutions. For instance, the ASM-DC obtains the better quality solution than the AggSub in 131 cases whereas the latter one finds the better quality solution than the former algorithm in 23 cases. They find the similar solutions in 446 cases with respect to the tolerance given above. These results demonstrate that the ASM-DC has better global search properties than the other solvers, except the PBDC.

Table 6 Best values of objective functions obtained by solvers for large-scale problems

P	n	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP	PBM	HANSO
P_{L1}	200	0.000055	0.000065	98.567212	153.290502	–	–	99.000000	533.942016
P_{L1}	500	0.000085	0.000070	249.000000	539.992947	–	–	249.000000	533.942016
P_{L1}	1000	0.000323	0.000321	499.000000	510.845806	–	–	499.000000	533.942016
P_{L1}	1500	0.000720	0.000507	749.0000000	1.907545×10^3	–	–	749.000000	533.942016
P_{L1}	2000	0.002466	0.000827	999.000000	1.002287×10^3	–	–	999.000000	533.942016
P_{L1}	3000	0.000458	0.000468	1.499000×10^3	5.760877×10^3	–	–	1.499000×10^3	533.942016
P_{L1}	5000	0.009887	0.002217	2.499000×10^3	4.584284×10^3	–	–	2.498999×10^3	533.942016
P_{L2}	200	0.000000	0.000000	0.000000	0.000000	0.000000	3.595387×10^3	0.000000	0.000000
P_{L2}	500	0.000000	0.000000	0.000000	0.000000	0.000004	1.081506×10^4	0.000000	0.000000
P_{L2}	1000	0.000000	0.000000	0.000000	0.000000	0.000012	2.439871×10^4	0.000000	0.000000
P_{L2}	1500	0.000000	0.000002	0.000001	0.000000	0.000017	3.902885×10^4	0.000000	0.000000
P_{L2}	2000	0.000000	0.000000	0.000002	0.000000	0.000026	5.433859×10^4	0.000000	0.000000
P_{L2}	3000	0.000000	0.000001	0.000005	0.000000	0.000074	8.637147×10^4	0.000000	0.000000
P_{L2}	5000	0.000000	0.000001	0.000012	0.000000	0.000160	1.541663×10^4	0.000000	0.000000
P_{L3}	200	0.000002	4.665255	0.000000	0.161240	0.000008	0.000049	0.000005	0.086629
P_{L3}	500	0.000007	7.912745	0.000000	0.083517	0.000018	0.000148	0.000003	0.313403
P_{L3}	1000	0.000373	17.321600	0.000000	0.050604	0.000009	0.000037	0.000022	0.547216
P_{L3}	1500	0.001568	10.501611	0.000000	2.787352	0.000014	0.000109	0.000008	87.486066
P_{L3}	2000	0.000564	12.630332	0.000000	26.174926	0.000015	0.000033	0.000011	105.013574
P_{L3}	3000	0.000588	36.586257	0.000000	3.551644	0.000026	0.000053	0.000015	233.241688
P_{L3}	5000	0.001377	82.407342	0.000000	1.813284	0.000021	0.000101	0.000015	129.460700

Table 7 Number of function evaluations for large-scale problems

P	n	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP	PBM	HANSO
P_{L1}	200	2328	1075	60	2508	–	–	8	33
P_{L1}	500	2083	1400	146	2006	–	–	8	33
P_{L1}	1000	2165	2127	50	3511	–	–	7	33
P_{L1}	1500	2691	1734	102	1003	–	–	7	33
P_{L1}	2000	3123	1299	49	4526	–	–	7	33
P_{L1}	3000	1239	1250	40	2508	–	–	7	33
P_{L1}	5000	1695	1373	39	1505	–	–	8	33
P_{L2}	200	1835	1604	974	5010	108	4509	201	365
P_{L2}	500	2161	2342	1355	5010	144	2264	286	574
P_{L2}	1000	3153	2277	1664	5904	219	1655	399	739
P_{L2}	1500	3966	2722	2044	4158	295	1509	499	901
P_{L2}	2000	4433	2936	2244	5511	299	955	594	1054
P_{L2}	3000	6125	3984	2624	6513	355	1093	704	1293
P_{L2}	5000	7006	4775	3410	6090	485	854	976	1384
P_{L3}	200	1044	14639	1928	3006	55	97	1564	1037
P_{L3}	500	1444	12869	2539	3507	58	355	2101	1163
P_{L3}	1000	3445	21456	2980	3006	73	370	1298	1001
P_{L3}	1500	2408	41096	2763	1503	69	1130	3262	756
P_{L3}	2000	4017	34758	3766	1503	73	1090	2532	1473
P_{L3}	3000	5690	43834	3281	2505	79	550	2677	903
P_{L3}	5000	6304	38273	4605	3507	86	601	3600	1832

Numerical results for large-scale problems. We report the results obtained by different algorithms for three large-scale test problems described above. Note that since the conventional code of HANSO is out of memory for problems P_{L2} and P_{L3} with $n \geq 2000$ and $n \geq 1500$, respectively, we use its limited memory version. The results are given in Tables 6 – 8, where the objective function values, the number of function evaluations and the number of subgradient evaluations are presented. For the test problem P_{L1}, considering different number of variables, the best values of the objective function obtained by the ASM-DC and the AggSub are much smaller than those obtained by other solvers. The solvers PBMDC and SDCA-LP fail to find any solution in a given time frame. For the test problem P_{L2}, all solvers, except SDCA-LP, are able to find the optimal solution with a given tolerance. The SDCA-LP fails to produce accurate solutions within the given time limit for all number of variables. In the test problem P_{L3} with $n = 200,\ 500$, the solvers ASM-DC, PBDC, PBMDC, SDCA-LP and PBM are able to find accurate solutions. For this test problem with $n \geq 1000$, the PBDC finds the most accurate solutions and the ASM-DC is less accurate than the PBDC, PBMDC, PBM and SDCA-LP. Other algorithms fail to find accurate solutions. Overall, the ASM-DC is the most robust method for solving large-scale problems used in the numerical experiments.

From Tables 7 and 8 we can see that the PBMDC requires the least number of both function and subgradient evaluations than other solvers when it is able to find

Table 8 Number of subgradient evaluations for large-scale test problems

P	n	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP	PBM	HANSO
P_{L1}	200	628	277	48	2502	–	–	8	33
P_{L1}	500	456	364	115	2002	–	–	8	33
P_{L1}	1000	497	607	37	3503	–	–	7	33
P_{L1}	1500	664	479	80	1001	–	–	7	33
P_{L1}	2000	793	631	34	4515	–	–	7	33
P_{L1}	3000	407	291	29	2502	–	–	7	33
P_{L1}	5000	492	386	30	1501	–	–	8	33
P_{L2}	200	593	787	780	5005	177	4509	201	365
P_{L2}	500	705	1157	1057	5005	23	2264	286	574
P_{L2}	1000	1041	1125	1416	5896	360	1655	399	739
P_{L2}	1500	1310	1344	1740	4153	482	1509	499	901
P_{L2}	2000	1466	1459	1914	5505	497	955	594	1054
P_{L2}	3000	2027	1976	2232	6506	567	1093	704	1293
P_{L2}	5000	2323	2370	2908	6083	797	854	976	1384
P_{L3}	200	333	7151	1845	3003	97	97	1564	1037
P_{L3}	500	469	6252	2513	3503	99	3555	2101	1163
P_{L3}	1000	1107	10555	2958	3003	126	370	1298	1001
P_{L3}	1500	730	20239	2747	1501	121	1130	3262	756
P_{L3}	2000	1283	17149	3741	1501	127	1090	2532	1473
P_{L3}	3000	1803	21648	3264	1502	133	550	2677	903
P_{L3}	5000	2017	18854	4560	3503	143	601	3600	1832

the solution. In average, the ASM-DC requires less function and subgradient evaluations than the AggSub and slightly less than the DCA. The ASM-DC requires more function and subgradient evaluations than PBDC, PBMDC, SDCA-LP, PBM and HANSO. However, in some cases the difference between the ASM-DC and these five methods is not significant. Summarizing results from Tables 7, 8 and taking into account the number of variables we can say that the ASM-DC requires reasonable number of function and subgradient evaluations in large-scale problems.

5 Conclusions

A new method, named the augmented subgradient method (ASM-DC), is introduced for solving nonsmooth unconstrained difference of convex (DC) optimization problems. First, we define the set of augmented subgradients using subgradients of DC components and their linearization errors. Then a finite number of elements from this set are used to compute search directions, and the new method is designed using these search directions. It is proved that the sequence of points generated by the ASM-DC converges to critical points of the DC problem.

We evaluate the performance of the ASM-DC using 14 small and medium-sized and three large-scale nonsmooth DC optimization test problems. These test problems are also used to compare the performance of the ASM-DC with that of five nonsmooth DC optimization and two nonsmooth optimization methods. We used different evaluation measures including performance profiles to compare solvers. Computational results clearly demonstrate that the ASM-DC is the most robust method for the test problems used in the numerical experiments. In comparison with most other methods the ASM-DC requires more function and subgradients evaluations. The developed method uses modest CPU time even for large-scale problems. Results demonstrate that the ASM-DC in comparison with other methods is able, as a rule, to find solutions of either similar or better quality.

Acknowledgements The research by Dr. A.M. Bagirov is supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (Project No. DP190100580) and the research by Dr. N. Hoseini Monjezi is supported by the National Elite Foundation of Iran. The authors would like to thank the handling editor and three anonymous referees for their comments that helped to improve the quality of the paper.

References

1. Artacho, F.J.A., Campoy, R., Vuong, P.T.: Using positive spanning sets to achieve d-stationarity with the boosted DC algorithm. Vietnam J. Math. **48**(2), 363–376 (2020)
2. Artacho, F.J.A., Fleming, R.M.T., Vuong, P.T.: Accelerating the DC algorithm for smooth functions. Math. Program. **169**, 95–118 (2018)
3. Artacho, F.J.A., Vuong, P.T.: The boosted difference of convex functions algorithm for nonsmooth functions. SIAM J. Optim. **30**(1), 980–1006 (2020)
4. An, L.T.H., Tao, P.D., Ngai, H.V.: Exact penalty and error bounds in DC programming. J. Glob. Optim. **52**(3), 509–535 (2012)
5. An, L.T.H., Tao, P.D.: The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. Ann. Oper. Res. **133**, 23–46 (2005)
6. Astorino, A., Fuduli, A., Gaudioso, M.: DC models for spherical separation. J. Glob. Optim. **48**(4), 657–669 (2010)
7. Bagirov, A.M., Gaudioso, M., Karmitsa, N., Mäkelä, M.M., Taheri, S. (eds.): Numerical Nonsmooth Optimization: State of the Art Algorithms. Springer, Berlin (2020)
8. Bagirov, A.M., Karmitsa, N., Mäkelä, M.M.: Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer, New York (2014)
9. Bagirov, A.M., Taheri, S., Joki, K., Karmitsa, N., Mäkelä, M.M.: Aggregate subgradient method for nonsmooth DC optimization. Optim. Lett. **15**(1), 83–96 (2020)
10. Bagirov, A.M., Taheri, S., Cimen, E.: Incremental DC optimization algorithm for large-scale clusterwise linear regression. J. Comput. Appl. Math. **389**, 113323 (2021)
11. Bagirov, A.M., Taheri, S., Ugon, J.: Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. Pattern Recognit. **53**, 12–24 (2016)
12. Bagirov, A.M., Ugon, J.: Nonsmooth DC programming approach to clusterwise linear regression: optimality conditions and algorithms. Optim. Methods Softw. **33**(1), 194–219 (2018)
13. Bagirov, A.M., Ugon, J.: Codifferential method for minimizing nonsmooth DC functions. J. Glob. Optim. **50**, 3–22 (2011)
14. Burke, J.V., Lewis, A.S., Overton, M.L.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. **15**(3), 751–779 (2005)
15. Demyanov, V.F., Rubinov, A.M.: Constructive Nonsmooth Analysis. Peter Lang, Frankfurt a. M. (1995)

16. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)
17. de Oliveira, W.: Proximal bundle methods for nonsmooth DC programming. J. Glob. Optim. **75**, 523–563 (2019)
18. de Oliveira, W.: Sequential difference-of-convex programming. J. Optim. Theory Appl. **186**(3), 936–959 (2020)
19. de Oliveira, W.: The ABC of DC programming. Set-Valued Var. Anal. **28**, 679–706 (2020)
20. de Oliveira, W., Tcheou, M.P.: An inertial algorithm for DC programming. Set-Valued Var. Anal. **27**, 895–919 (2019)
21. Frangioni, A.: Solving semidefinite quadratic problems within nonsmooth optimization algorithms. Comput. Oper. Res. **21**, 1099–1118 (1996)
22. Gaudioso, M., Giallombardo, G., Miglionico, G., Bagirov, A.M.: Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. J. Glob. Optim. **71**(1), 37–55 (2018)
23. Horst, R., Thoai, N.V.: DC programming: overview. J. Optim. Theory Appl. **103**, 1–43 (1999)
24. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M.: A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. J. Glob. Optim. **68**, 501–535 (2017)
25. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M., Taheri, S.: Double bundle method for finding Clarke stationary points in nonsmooth DC programming. SIAM J. Optim. **28**(2), 1892–1919 (2018)
26. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M., Taheri, S.: Double bundle method for finding Clarke stationary points in nonsmooth DC optimization. Technical reports 1173, Turku Center for Computer Science (TUCS), Turku (2017)
27. Khalaf, W., Astorino, A., D'Alessandro, P., Gaudioso, M.: A DC optimization-based clustering technique for edge detection. Optim. Lett. **11**(3), 627–640 (2017)
28. Lewis, A.S., Overton, M.L.: Nonsmooth optimization via quasi-Newton methods. Math. Program. **141**(1–2), 135–163 (2013)
29. Luksan, L.: Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minimax approximation. Kybernetika **20**(6), 445–457 (1984)
30. Mäkelä, M.M.: Multiobjective proximal bundle method for nonconvex nonsmooth optimization: Fortran subroutine MPBNGC 2.0. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing B 13/2003, University of Jyväskylä, Jyväskylä (2003)
31. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control. World Scientific Publishing Co., Singapore (1992)
32. Ordin, B., Bagirov, A.M.: A heuristic algorithm for solving the minimum sum-of-squares clustering problems. J. Glob. Optim. **61**, 341–361 (2015)
33. Pang, J.S., Razaviyayn, M., Alvarado, A.: Computing B-stationary points of nonsmooth DC programs. Math. Oper. Res. **42**(1), 95–118 (2017)
34. Sun, W.Y., Sampaio, R.J.B., Candido, M.A.B.: Proximal point algorithm for minimization of DC functions. J. Comput. Math. **21**(4), 451–462 (2003)
35. Tao, P.D., Souad, E.B.: Algorithms for solving a class of nonconvex optimization problems: methods of subgradient. North-Holland Mathematics Studies. Fermat Days 85: Mathematics for Optimization. **129**, 249–271 (1986)
36. Toland, J.F.: On subdifferential calculus and duality in nonconvex optimization. Bull. Soc. Math. France Mémoire. **60**, 177–183 (1979)
37. Tuy, H.: Convex Analysis and Global Optimization. Kluwer, Dordrecht (1998)
38. van Ackooij, W., de Oliveira, W.: Nonsmooth and nonconvex optimization via approximate difference-of-convex decompositions. J. Optim. Theory Appl. **182**, 49–80 (2019)
39. Wolfe, P.: Finding the nearest point in a polytope. Math. Program. **11**, 128–149 (1976)
40. Zaffaroni, A.: Continuous approximations, codifferentiable functions and minimization methods. In: Demyanov, V.F., Rubinov, A.M. (eds.) Nonconvex Optimization and Its Applications, pp. 361–391. Kluwer Academic Publishers, Dordrecht, Quasidifferentiability and Related Topics (2000)