Adil M. Bagirov | Sona Taheri | Kaisa Joki | Napsu Karmitsa | Marko M. Mäkelä

A New Subgradient Based Method for Nonsmooth DC Programming

Turku Centre for Computer Science

TUCS Technical Report
No 1201, March 2019

A New Subgradient Based Method for Nonsmooth DC Programming

Adil M. Bagirov
School of Science, Engineering and Information Technology, Federation University Australia, Ballarat, Victoria, Australia
`a.bagirov@federation.edu.au`

Sona Taheri
School of Science, Engineering and Information Technology, Federation University Australia, Ballarat, Victoria, Australia
`s.taheri@federation.edu.au`

Kaisa Joki
Department of Mathematics and Statistics, University of Turku,
FI-20014 Turku, Finland
`kjjoki@utu.fi`

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

Marko M. Mäkelä
Department of Mathematics and Statistics, University of Turku,
FI-20014 Turku, Finland
`makela@utu.fi`

TUCS Technical Report
No 1201, March 2019

Abstract

The aggregate subgradient method is developed for solving unconstrained nonsmooth difference of convex (DC) optimization problems. The proposed method shares some similarities with both the subgradient and the bundle methods. Aggregate subgradients are defined as a convex combination of subgradients computed at null steps between two serious steps. At each iteration search directions are found using only two subgradients: the aggregate subgradient and a subgradient computed at the current null step. It is proved that the proposed method converges to a critical point of the DC optimization problem and also that the number of null steps between two serious steps is finite. The new method is tested using some academic test problems and compared with several other nonsmooth DC optimization solvers.

Keywords: Nonsmooth optimization, Nonconvex optimization, DC optimization, Subgradient method

TUCS Laboratory
Turku Optimization Group (TOpGroup)

1 Introduction

Consider the unconstrained DC minimization problem

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

where $f(x) = f_1(x) - f_2(x)$, functions f_1 and f_2 are convex and in general, nonsmooth. Here $\mathbb{R}^n$ is the n-dimensional Euclidean space.

To date, several methods have been developed to locally solve Problem (1) [9, 15, 19, 21, 24, 25, 33, 34, 35]. Some of these methods are extensions of the Difference of Convex Algorithm (DCA) [24, 25, 33, 34] while others can be considered as the extension of the bundle method for convex optimization [9, 15, 19, 21]. In this paper, a different approach is proposed to develop an algorithm for solving Problem (1). The key element in this approach is the concept of *the aggregate subgradient*. Aggregate subgradients have been used to design nonsmooth optimization algorithms [3, 16, 23]. Although there are different definitions of these subgradients in all cases they are convex combinations of subgradients from the current bundle.

The subgradient method is widely used in nonsmooth optimization due to its simple structure. Its convergence has been proved only for convex problems [1, 10, 11, 29, 30, 31, 32]. This method uses one subgradient and one function evaluation at each iteration and involves no line search procedure. It is applicable to large-scale problems due to its small storage requirements.

Bundle methods are efficient methods in nonsmooth optimization (see, [2, 4, 13, 14, 18, 23, 27, 28]). The problem of finding the search direction in these methods is reduced to a certain quadratic programming subproblem whose size may increase significantly as the number of variables increases. Various approaches, in particular based on the notion of the aggregate subgradient, to reduce the size of this subproblem has been proposed in [16, 17, 26]. These approaches allow one to design bundle methods which are applicable to large-scale problems.

The method introduced in this paper shares some similarities with both the subgradient and the bundle methods. It is a descent method, uses more than one subgradient at each iteration and involves the line search procedure (similar to bundle methods). On the other hand, it does not involve any time consuming quadratic programming subproblem to find search directions (similar to the subgradient methods). The proposed method has a simple structure, and it is easy to implement. We prove the convergence of this method and demonstrate its performance using some nonsmooth unconstrained DC optimization test problems.

The paper is structured as follows. In Section 2, the new aggregate subgradient method is introduced and its convergence is studied. Results of numerical experiments are reported in Section 3. Section 4 concludes the paper.

Throughout this paper, we use following notations and definitions. The inner product in $\mathbb{R}^n$ is $\langle u,v\rangle = \sum_{i=1}^{n} u_i v_i$, and $\|\cdot\|$ is the associated norm. $S_1 = \{x \in \mathbb{R}^n : \|x\| = 1\}$ is the unit sphere, and $B(x;\varepsilon) = \{y \in \mathbb{R}^n : \|x-y\| < \varepsilon\}$ is the open ball centered at $x \in \mathbb{R}^n$. The convex hull of a set is denoted by "conv". For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, its subdifferential at a point $x \in \mathbb{R}^n$ is [5]

$$\partial f(x) = \{\xi \in \mathbb{R}^n : f(y) - f(x) \geq \langle \xi, y-x\rangle \ \ \forall\, y \in \mathbb{R}^n\}$$

and its ε-subdifferential is

$$\partial_\varepsilon f(x) = \{\xi \in \mathbb{R}^n : f(y) - f(x) \geq \langle \xi, y-x\rangle - \varepsilon \ \ \forall\, y \in \mathbb{R}^n\}.$$

A point $x^* \in \mathbb{R}^n$ is called a *critical point* of Problem (1) if

$$\partial f_1(x^*) \cap \partial f_2(x^*) \neq \emptyset.$$

2 The aggregate subgradient method

In this section, we introduce an aggregate subgradient method for solving Problem (1). For a given number $\tau > 0$ consider the set

$$Q_{1\tau}(x) = \text{conv} \bigcup_{d \in S_1} \partial f_1(x + \tau d),\ x \in \mathbb{R}^n.$$

Proposition 1. The set $Q_{1\tau}(x)$ is convex and compact for any finite $\tau > 0$.

Proof. The convexity of the set $Q_{1\tau}(x)$ follows from its definition. Since the subdifferential is bounded over any bounded set the set $Q_{1\tau}(x)$ is bounded. To prove its closedness it is sufficient to prove the closedness of the set

$$\bar{Q}_{1\tau}(x) = \bigcup_{d \in S_1} \partial f_1(x + \tau d).$$

Take any sequence $\{u_k\} \subset \bar{Q}_{1\tau}(x)$ such that $u_k \to u$ as $k \to \infty$. This means that there exists the sequence $\{d_k\} \subset S_1$ such that $u_k \in \partial f_1(x + \tau d_k)$. Since the set S_1 is compact all limit points of the sequence $\{d_k\}$ belong to this set. Without loss of generality, assume that $d_k \to \bar{d} \in S_1$ as $k \to \infty$. Then the upper semicontinuity of the subdifferential mapping implies that $u \in \partial f_1(x + \tau \bar{d})$ and therefore, $u \in \bar{Q}_{1\tau}(x)$. Since a convex hull of a closed set is also closed it follows that the set $Q_{1\tau}(x)$ is closed. $\square$

Remark 1. Note that the set $Q_{1\tau}(x)$ was introduced in [6], where it is called the spherical subdifferential.

The proof of the following theorem can be found in [5] (see Theorem 2.33).

Theorem 1. Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex with a Lipschitz constant $L > 0$ at $x \in \mathbb{R}^n$. Then for all $\varepsilon \geq 0$ we have

$$\partial f(y) \subseteq \partial_\varepsilon f(x) \ \ \forall\, y \in B\left(x; \frac{\varepsilon}{2L}\right). \tag{2}$$

Proposition 2. For the set $Q_{1\tau}(x),\ x \in \mathbb{R}^n,\ \tau > 0$ the following holds:

$$Q_{1\tau}(x) \subseteq \partial_\varepsilon f_1(x)$$

for all $\varepsilon > 2L\tau$.

Proof. It is clear that $x + \tau d \in B\left(x; \frac{\varepsilon}{2L}\right)$ for any $\varepsilon > 2L\tau$ and $d \in S_1$. Then the proof follows from Theorem 1 and the convexity of $\partial_\varepsilon f_1(x)$. $\square$

Now take any subgradient $v \in \partial f_2(x)$ and construct the set

$$Q_\tau(x) = \{\xi \in \mathbb{R}^n : \ \xi = u - v,\ u \in Q_{1\tau}(x)\}.$$

The convexity of functions f_1 and f_2 and the subgradient inequality imply that

$$f(x+\tau d) - f(x) \leq \tau \langle u - v, d\rangle \ \ \forall\, u \in \partial f_1(x+\tau d) \tag{3}$$

and therefore,

$$f(x+\tau d) - f(x) \leq \tau \max_{\xi \in Q_\tau(x)} \langle \xi, d\rangle.$$

It follows from Proposition 1 that the set $Q_\tau(x)$ is convex and compact for any finite $\tau > 0$. If $0_n \notin Q_\tau(x)$ at $x \in \mathbb{R}^n$, then one can use the set $Q_\tau(x)$ to compute descent directions of the function f [8]. However, it is not always easy to compute the whole set $Q_\tau(x)$.

Next we describe the proposed algorithm, Algorithm 1. At each iteration of this algorithm, we use only two elements of the set $Q_\tau(x)$ to compute search directions. The algorithm consists of inner and outer iterations. The inner iteration contains null steps whereas the outer iteration involves serious steps. Aggregate subgradients are calculated in the inner iterations.

In the next proposition, we prove that for each outer iteration $k \geq 0$ the number of inner iterations, null steps, in Algorithm 1 is finite.

Proposition 3. Let

$$C_k = \max\{\|\xi\| : \xi \in Q_{\tau_k}(x_k)\} < +\infty. \tag{5}$$

Assume that $\delta_k < C_k$. Then the inner loop at the k-th iteration of Algorithm 1 stops after $m_k > 0$ iterations, where

$$m_k \leq \left\lceil \frac{2\log_2(\delta_k/C_k)}{\log_2 C_{1k}} \right\rceil, \ \ C_{1k} = 1 - (1-c_1)^2(4C_k^2)^{-1}\delta_k^2.$$

Algorithm 1: Aggregate subgradient method.

```
Data: $\sigma_1 \in (0,1)$ - decrease parameter for $\tau$, $\sigma_2 \in (0,1]$ - decrease parameter for
    the inner iteration tolerance $\delta$, $c_1 \in (0,1)$ - search control parameter,
    $c_2 \in (0,c_1]$ - line search parameter, $\varepsilon > 0$ - optimality tolerance.
Result: An approximate critical point of Problem (1).
Step 0. (Initialization). Choose a starting point $x_0 \in \mathbb{R}^n$ and numbers $\tau_0 > 0,\ \delta_0 > 0$. Set
    $k = 0$.

Outer iteration
    Inner iteration
        Step 1. (Initialization). Select any $d_k^0 \in S_1$. Compute $u_k^0 \in \partial f_1(x_k + \tau_k d_k^0)$ and
            $v_k \in \partial f_2(x_k)$. Set $\xi_k^0 = u_k^0 - v_k,\ \bar{\xi}_k^0 = \xi_k^0$ and $l = 0$.
        Step 2. Solve the problem
            $\begin{cases} \text{minimize} & \varphi^l(\lambda) = \|\lambda \xi_k^l + (1-\lambda)\bar{\xi}_k^l\|^2 \\ \text{subject to} & \lambda \in [0,1]. \end{cases}$
            Let $\bar{\lambda}_l$ be a solution to this problem.
        Step 3. Calculate $\bar{\xi}_k^{l+1} = \bar{\lambda}_l \xi_k^l + (1-\bar{\lambda}_l)\bar{\xi}_k^l$. If
            $\|\bar{\xi}_k^{l+1}\| \le \delta_k,$        (4)
            then EXIT inner iterations and go to Step 6.
        Step 4. (Search direction). Compute $d_k^{l+1} = -\|\bar{\xi}_k^{l+1}\|^{-1}\bar{\xi}_k^{l+1}$.
        Step 5. If
            $f(x_k + \tau_k d_k^{l+1}) - f(x_k) > -c_1\tau_k\|\bar{\xi}_k^{l+1}\|,$
            then compute the subgradient $u_k^{l+1} \in \partial f_1(x_k + \tau_k d_k^{l+1})$, set $\xi_k^{l+1} =$
            $u_k^{l+1} - v_k,\ l = l+1$ and go to Step 2. Otherwise EXIT inner iterations
            and go to Step 7.
    Step 6. (Stopping criterion). If $\tau_k \le \varepsilon$, then STOP since an approximate critical
        point has been found. Otherwise, set $\tau_{k+1} = \sigma_1\tau_k,\ \delta_{k+1} = \sigma_2\delta_k,\ x_{k+1} =$
        $x_k,\ k = k+1$ and go to Step 1.
    Step 7. (Line search). Calculate the step-length $\alpha_k \ge \tau_k$ as follows:
        $\alpha_k = \operatorname{argmax}\left\{\alpha \ge 0 :\ f(x_k + \alpha d_k^{l+1}) - f(x_k) \le -c_2\alpha\|\bar{\xi}_k^{l+1}\|\right\}.$
    Step 8. Set $x_{k+1} = x_k + \alpha_k d_k^{l+1}$, $\tau_{k+1} = \tau_k$, $\delta_{k+1} = \delta_k,\ k = k+1$ and go to Step 1.
```

Proof. The inner loop in Algorithm 1 terminates when either the condition (4) or the condition of the sufficient descent

$$f(x_k+\tau_k d_k^{l+1}) - f(x_k) \le -c_1\tau_k\|\bar{\xi}_k^{l+1}\| \tag{6}$$

is met. Let $S_k^l = \{\xi \in \mathbb{R}^n :\ \xi = \lambda\xi_k^l + (1-\lambda)\bar{\xi}_k^l,\ \lambda \in [0,1]\}$. In order to prove this proposition it is sufficient to estimate the upper bound of the number of iterations m_k when the condition (4) occurs. If none of the conditions (4) and (6) are satisfied at the l-th inner iteration, then the approximated subgradient $\xi_k^{l+1} = u_k^{l+1} - v_k$ computed in Step 5 does not belong to the set S_k^l. Indeed, according to the definition of $\bar{\xi}_k^{l+1}$ and the necessary and sufficient condition for a minimum (see Lemma 5.2.6 in [27]) we have

$$\langle \xi, \bar{\xi}_k^{l+1}\rangle \ge \|\bar{\xi}_k^{l+1}\|^2 \ \ \forall\, \xi \in S_k^l. \tag{7}$$

On the other hand, since the condition (6) does not hold we get

$$f(x_k+\tau_k d_k^{l+1}) - f(x_k) > -c_1\tau_k\|\bar{\xi}_k^{l+1}\|.$$

In addition, it follows from (3) that

$$f(x_k+\tau_k d_k^{l+1}) - f(x_k) \le \tau_k\langle \xi_k^{l+1}, d_k^{l+1}\rangle,$$

therefore, we have

$$-c_1\tau_k\|\bar{\xi}_k^{l+1}\| < \tau_k\langle \xi_k^{l+1}, d_k^{l+1}\rangle,$$

which means that

$$\langle \xi_k^{l+1}, \bar{\xi}_k^{l+1}\rangle < c_1\|\bar{\xi}_k^{l+1}\|^2. \tag{8}$$

Then (7) and the condition $c_1 \in (0,1]$ imply that $\xi_k^{l+1} \notin S_k^l$. Therefore, it is proved that the newly calculated subgradient ξ_k^{l+1} is not on the segment between $\bar{\xi}_k^l$ and ξ_k^l, and by updating subgradients in this manner, we change the set S_k^l at each inner iteration of Algorithm 1. This means that $\bar{\xi}_k^{l+2} \ne \bar{\xi}_k^{l+1}$. Indeed, if $\bar{\xi}_k^{l+2} = \bar{\xi}_k^{l+1}$, then the optimality condition (7) would hold also for all $\xi \in S_k^{l+1}$ and since $\xi_k^{l+1} \in S_k^{l+1}$ we get

$$\langle \xi_k^{l+1}, \bar{\xi}_k^{l+1}\rangle \ge \|\bar{\xi}_k^{l+1}\|^2$$

contradicting (8).

Next, we prove that if none of the stopping criteria is satisfied, then the new aggregated subgradient $\bar{\xi}_k^{l+2}$ computed at Step 3 is better than $\bar{\xi}_k^{l+1}$ in the sense that $\varphi^{l+1}(\bar{\lambda}_{l+1}) < \varphi^l(\bar{\lambda}_l)$. Since the function φ^{l+1} is strictly convex $\bar{\lambda}_{l+1}$ is its only minimizer. Furthermore, it follows from

$$\bar{\xi}_k^{l+2} = \bar{\lambda}_{l+1}\xi_k^{l+1} + (1-\bar{\lambda}_{l+1})\bar{\xi}_k^{l+1}$$

and $\bar{\xi}_k^{l+2} \neq \bar{\xi}_k^{l+1}$ that $\bar{\lambda}_{l+1} > 0$. Then we have

$$\varphi^{l+1}(\bar{\lambda}_{l+1}) = \|\bar{\xi}_k^{l+2}\|^2 < \varphi^{l+1}(0) = \|\bar{\xi}_k^{l+1}\|^2 = \varphi^l(\bar{\lambda}_l).$$

In addition, it is clear that

$$\|\bar{\xi}_k^{l+2}\|^2 \le \|t\xi_k^{l+1} + (1-t)\bar{\xi}_k^{l+1}\|^2 \ \ \forall t \in [0,1].$$

This means that for all $t \in [0,1]$

$$\|\bar{\xi}_k^{l+2}\|^2 \le t^2\|\xi_k^{l+1} - \bar{\xi}_k^{l+1}\|^2 + 2t\langle \xi_k^{l+1} - \bar{\xi}_k^{l+1}, \bar{\xi}_k^{l+1}\rangle + \|\bar{\xi}_k^{l+1}\|^2.$$

It follows from (5) that

$$\|\xi_k^{l+1} - \bar{\xi}_k^{l+1}\| \le 2C_k.$$

This together with the inequality (8) implies that

$$\varphi^{l+1}(\bar{\lambda}_{l+1}) = \|\bar{\xi}_k^{l+2}\|^2 \quad \le \quad 4t^2C_k^2 + (1 - 2t(1-c_1))\|\bar{\xi}_k^{l+1}\|^2$$

for all $t \in [0,1]$. Let $t_0 = (1-c_1)\|\bar{\xi}_k^{l+1}\|^2(4C_k^2)^{-1}$. It is clear that $t_0 \in (0,1)$. Therefore, for $t = t_0$ we get

$$\|\bar{\xi}_k^{l+2}\|^2 \le \left(1 - (1-c_1)^2(4C_k^2)^{-1}\|\bar{\xi}_k^{l+1}\|^2\right)\|\bar{\xi}_k^{l+1}\|^2.$$

At any l-th iteration, $l > 0$, where the stopping condition (4) is not met, we have $\|\bar{\xi}_k^{l+1}\| > \delta_k$. Then it follows that

$$\|\bar{\xi}_k^{l+2}\|^2 < \left(1 - (1-c_1)^2(4C_k^2)^{-1}\delta_k^2\right)\|\bar{\xi}_k^{l+1}\|^2.$$

Let $C_{1k} = 1 - (1-c_1)^2(4C_k^2)^{-1}\delta_k^2$. It is obvious that $C_{1k} \in (0,1)$. Then we have

$$\varphi^{l+1}(\bar{\lambda}_{l+1}) < C_{1k}\varphi^l(\bar{\lambda}_l).$$

Since $\varphi^0(\bar{\lambda}_0) = \|\xi_k^0\|^2 \le C_k^2$ we get

$$\varphi^l(\bar{\lambda}_l) < C_{1k}^l C_k^2.$$

Thus, the inequality (4) is satisfied if

$$C_{1k}^{m_k} C_k^2 \le \delta_k^2,$$

which must happen after at most m_k iterations, where

$$m_k \le \left\lceil \frac{2\log_2(\delta_k/C_k)}{\log_2 C_{1k}} \right\rceil.$$

This completes the proof. $\square$

Next, we prove that Algorithm 1 generates a sequence which converges to a critical point of Problem (1). For a given $x_0 \in \mathbb{R}^n$ consider the level set

$$\mathscr{L}(x_0) = \{x \in \mathbb{R}^n : \ f(x) \leq f(x_0)\}.$$

Theorem 2. Assume that the set $\mathscr{L}(x_0)$ is bounded for any starting point $x_0 \in \mathbb{R}^n$, and $\varepsilon = 0$ in Algorithm 1. Then any limit point of the sequence $\{x_k\}$ generated by this algorithm is a critical point of Problem (1).

Proof. Algorithm 1 consists of inner and outer iterations. In all inner iterations, the values of τ_k and δ_k remain unchanged. Furthermore, these parameters remain unchanged in Steps 7 and 8 of outer iterations. Their values are updated in Step 6 when the condition (4) is satisfied. According to Proposition 3 the number of steps in inner iterations is finite for the given τ_k and δ_k. As an outcome of the inner loop either Step 6 or Steps 7 and 8 are executed.

First, we show that for each unchanged values of τ_k and δ_k the number of executions of Steps 7 and 8 is finite. Indeed, in this case the line search is carried out in Step 7 and we have

$$f(x_{k+1}) - f(x_k) \leq -c_2 \tau_k \|\bar{\xi}_k^{l+1}\|, \ \|\bar{\xi}_k^{l+1}\| > \delta_k.$$

Therefore, we get

$$f(x_{k+1}) - f(x_k) \leq -c_2 \tau_k \delta_k. \tag{9}$$

Since the set $\mathscr{L}(x_0)$ is compact and the function f is continuous it follows that

$$f_* := \inf \ \{f(x) : x \in \mathbb{R}^n\} > -\infty. \tag{10}$$

If the values of τ_k and δ_k are unchanged infinitely many times then (9) implies that $f(x_k) \to -\infty$ as $k \to \infty$. This contradicts the condition (10). Thus, τ_k and δ_k are updated in Step 6 after a finite number of outer iterations, and we denote those iterations by k_p, $p = 1, 2\ldots$. During the iteration k_p, the condition (4) is satisfied meaning that

$$\|\bar{\xi}_{k_p}\| \leq \delta_{k_p} \ \ \forall \, p = 1, 2, \ldots,$$

where $\bar{\xi}_{k_p} = \bar{\xi}_{k_p}^l$ for some $l > 0$. It is obvious that $\bar{\xi}_{k_p} \in Q_{\tau_{k_p}}(x_{k_p})$ which, in turn, implies that

$$\min_{\xi \in Q_{\tau_{k_p}}(x_{k_p})} \|\xi\| \leq \delta_{k_p},$$

or

$$\min_{u \in Q_{1\tau_{k_p}}(x_{k_p})} \|u - v_{k_p}\| \leq \delta_{k_p}. \tag{11}$$

In addition, it follows from Proposition 2 that

$$Q_{1\tau_{k_p}}(x_{k_p}) \subseteq \partial_{\bar{\varepsilon}} f_1(x_{k_p}) \tag{12}$$

for all $\bar{\varepsilon} > 2L\tau_{k_p}$, where $L > 0$ is the Lipschitz constant of the function f_1 over the set $\mathscr{L}(x_0)$. Take any $c_0 > 1$ and consider $\bar{\varepsilon}_{k_p} = 2c_0 L\tau_{k_p}$. Then the inequality (11) and the inclusion (12) imply that

$$\min_{u \in \partial_{\bar{\varepsilon}_{k_p}} f_1(x_{k_p})} \|u - v_{k_p}\| \le \delta_{k_p},$$

which means that

$$\partial_{\bar{\varepsilon}_{k_p}} f_1(x_{k_p}) \cap B(v_{k_p}; \delta_{k_p}) \neq \emptyset,$$

or

$$\partial_{\bar{\varepsilon}_{k_p}} f_1(x_{k_p}) \cap B(\partial f_2(x_{k_p}); \delta_{k_p}) \neq \emptyset. \tag{13}$$

Since the set $\mathscr{L}(x_0)$ is compact and $x_{k_p} \in \mathscr{L}(x_0)$ for any $p > 0$ the sequence $\{x_{k_p}\}$ has at least one limit point. For the sake of simplicity, assume that $x_{k_p} \to \bar{x}$ as $p \to \infty$. The upper semicontinuity of the subdifferential mapping and the fact that $\tau_{k_p}, \delta_{k_p} \downarrow 0$ as $p \to \infty$ imply that for $\gamma > 0$ there exists $p_\gamma > 0$ such that

$$\partial_{\bar{\varepsilon}_{k_p}} f_1(x_{k_p}) \subset \partial f_1(\bar{x}) + B(0_n; \gamma), \quad \partial f_2(x_{k_p}) \subset \partial f_2(\bar{x}) + B(0_n; \gamma) \ \text{ and } \ \delta_{k_p} < \gamma$$

for all $p > p_\gamma$. Then for all $p > p_\gamma$ we have

$$B(\partial f_2(x_{k_p}); \delta_{k_p}) \subseteq B(\partial f_2(x_{k_p}); \gamma) \subseteq \partial f_2(\bar{x}) + B(0_n; 2\gamma).$$

Taking into account (13) we get

$$\Big(\partial f_1(\bar{x}) + B(0_n; \gamma)\Big) \cap \Big(\partial f_2(\bar{x}) + B(0_n; 2\gamma)\Big) \neq \emptyset.$$

Since $\gamma > 0$ is arbitrary we have

$$\partial f_1(\bar{x}) \cap \partial f_2(\bar{x}) \neq \emptyset.$$

This completes the proof. $\square$

3 Numerical results

In this section, we compare the proposed aggregate subgradient algorithm with three other nonsmooth optimization methods. The collection of test problems used in our experiments consists of nonsmooth unconstrained DC optimization problems which are formulated by modifying various convex problems introduced in [16, 19, 20]. The problems are given in Appendix. We present only the results of those problems where all the solvers converge to the same critical point. Overall, this leaves us 150 test problems. The problems can be formulated with different number of variables: we use 100 (extra small), 200 (small), 500 (medium) and 1000 (large)[1] variables in our tests.

[1]We include 148 large test problems since none of four methods succeeded in solving two other problems.

Solvers and Parameters. We use the following solvers in our experiments:

- `DCA` is an implementation of the well-known difference of convex algorithm (DCA) [25]. The DCA, in general, finds critical solutions to the DC optimization problems. In the implementation of the DCA, we apply the proximal bundle method MPBNGC [27] to solve the convex subproblem.

- `PBDC` [19] is a proximal bundle method for DC optimization. Similar to the DCA, the `PBDC` is guaranteed to find critical solutions to the DC optimization problems.

- `SolvOpt` [22] is an implementation of Shor's r-algorithm [32]. Unlike the other methods considered in this paper, the `SolvOpt` is not utilizing the DC structure of the problem.

- `AggSub` is an implementation of the proposed algorithm. The following parameters were used in our experiments:

$$\begin{cases} \sigma_1 = 0.2,\ \sigma_2 = 1.0,\ \delta_0 = 10^{-7}, \\ \varepsilon = 10^{-5},\ c_1 = 0.2,\ c_2 = 0.05, \end{cases} \quad \text{and} \quad \tau_0 = \begin{cases} 10.0, & n < 200, \\ 50.0, & n \geq 200. \end{cases}$$

The Fortran 95 source code of `AggSub` is available at `http://napsu.karmitsa.fi/aggsub/`. All experiments are performed on an Intel® Core™ i5-2400 CPU (3.10GHz, 3.10GHz) running under Windows 7. To compile the codes, we use `gfortran`, the GNU Fortran compiler. All the solvers are used with the default settings of the codes.

We say that a solver finds a solution with respect to a tolerance $\varepsilon > 0$ if

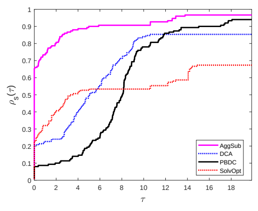
$$\frac{f_{\texttt{best}} - f_{\texttt{opt}}}{1 + |f_{\texttt{opt}}|} \leq \varepsilon,$$

where $f_{\texttt{best}}$ is the best value of the objective function obtained by the solver and $f_{\texttt{opt}}$ is the best known (or optimal) solution. Otherwise, the solver *fails*. We set $\varepsilon = 10^{-3}$. In addition to the usual stopping criteria of the solvers, we terminate the experiment if the elapsed CPU time exceeds half an hour per problem.

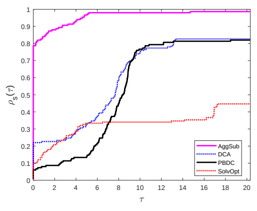
Results. The results are analyzed using the performance profiles (Figures 1 – 2) introduced in [12]. We compare the performance of the solvers both in terms of the computational time (CPU time) and the number of the subgradient evaluations (evaluations for short) — we use the sum of the number of the subgradient evaluations of the DC components in the `AggSub`, the `PBDC` and the `DCA`, while for the `SolvOpt` we use the number of the subgradient evaluations of the objective function since this solver does not utilize the DC structure of the problem. The number of function evaluations follows the similar trends with the number of the subgradient evaluations with all the solvers, and thus, we omit these results.

In the performance profiles, the value of $\rho_s(\tau)$ at $\tau = 0$ shows the ratio of the test problems for which the solver s is the best — that is, the solver s uses the least computational time or evaluations — while the value of $\rho_s(\tau)$ at the rightmost abscissa gives the ratio of the test problems that the solver s can solve — that is, the robustness of the solver s. In addition, the higher is a particular curve, the better is the corresponding solver.

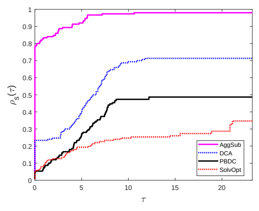
It is clear from Figure 1 that the `AggSub` is the most efficient and accurate with all sizes of the test problems — it is superior to other three methods in the larger problems. With the number of evaluations, as usual for any subgradient method, the `AggSub` has rather high numbers as can be seen in Figure 2 (a) and (b). However, this observation is faded out with the larger problems (see Figure 2 (c) and (d)). The reason for most of the failures with the `SolvOpt` is its inaccuracy while the `PBDC` usually stops far away from the solution due to the time limit. From these results, we conclude that, overall, the `AggSub` is an efficient and robust method.



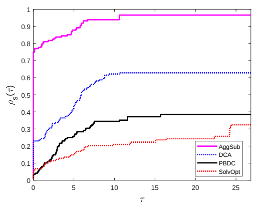
(a) $n = 100$



(b) $n = 200$

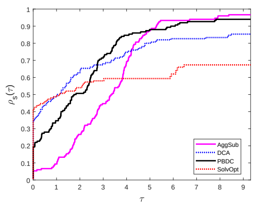


(c) $n = 500$

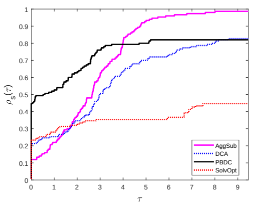


(d) $n = 1000$

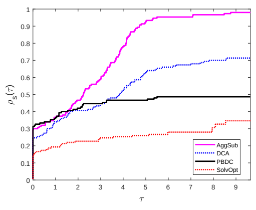
Figure 1: CPU time.



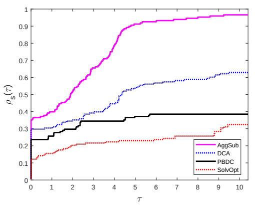
(a) $n = 100$



(b) $n = 200$



(c) $n = 500$



(d) $n = 1000$

Figure 2: Number of subgradient evaluations.

4 Concluding remarks

In this paper, an aggregated subgradient method is developed to solve unconstrained nonsmooth difference of convex (DC) optimization problems. The method combines the strengths of the well-known subgradient and bundle methods. It is a descent method, easy to implement and does not involve solving of any time-consuming quadratic programming subproblem, since only two subgradients are utilized to calculate search directions. In addition, the proposed method is globally convergent to a critical point.

The aggregated subgradient method was tested using large-scale nonsmooth unconstrained DC programming problems and compared with several other nonsmooth DC programming solvers. Results demonstrate that the method is efficient and robust for solving nonsmooth DC optimization problems, although in some cases it may require a large number of function and subgradient evaluations. The proposed method outperforms other methods when the evaluation of the objective function and its subgradient is not expensive.

Acknowledgments

This research by Dr. Adil Bagirov and Dr. Sona Taheri was supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (Project No. DP190100580). The research by Dr. Napsu Karmitsa and Dr. Sona Taheri was supported by the Academy of Finland (Projects No. 289500 and 319274).

References

[1] ANSTREICHER, K. M., AND WOLSEY, L. A. Two well-known properties of subgradient optimization. *Mathematical Programming 120*, 1 (2009), 213–220.

[2] BAGIROV, A. M., AND GANJEHLOU, A. N. A quasisecant method for minimizing nonsmooth functions. *Optimization Methods and Software 25*, 1 (2010), 3–18.

[3] BAGIROV, A. M., JIN, L., KARMITSA, N., AL NUAIMAT, A., AND SULTANOVA, N. Subgradient method for nonconvex nonsmooth optimization. *Journal of Optimization Theory and Applications 157*, 2 (2013), 416–435.

[4] BAGIROV, A. M., KARASÖZEN, B., AND SEZER, M. Discrete gradient method: Derivative-free method for nonsmooth optimization. *Journal of Optimization Theory and Applications 137*, 2 (2008), 317–334.

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

[6] BAGIROV, A. M., TAHERI, S., BAI, F., AND WU, Z. An approximate ADMM for solving linearly constrained nonsmooth optimization problems with two blocks of variables. In *International Series in Numerical Mathematics*. 2019.

[7] BAGIROV, A. M., TAHERI, S., JOKI, K., KARMITSA, N., AND MÄKELÄ, M. M. A new subgradient based method for nonsmooth DC programming, TUCS. Tech. rep., No. 1201, Turku Centre for Computer Science, Turku, 2019.

[8] 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.

[9] BAGIROV, A. M., AND UGON, J. Codifferential method for minimizing nonsmooth DC functions. *Journal of Global Optimization 50*, 1 (2011), 3–22.

[10] BELTRAN, C., AND HEREDIA, F. J. An effective line search for the subgradient method. *Journal of Optimization Theory and Applications 125*, 1 (2005), 1–18.

[11] BERTSEKAS, D. P. *Nonlinear Programming*. Athena Scientific, New York, 1999.

[12] DOLAN, E. D., AND MORÉ, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming 91*, 2 (2002), 201–213.

[13] 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.

[14] FUDULI, A., GAUDIOSO, M., AND GIALLOMBARDO, G. Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. *SIAM Journal on Optimization 14*, 3 (2004), 743–756.

[15] GAUDIOSO, M., GIALLOMBARDO, G., MIGLIONICO, G., AND BAGIROV, A. M. Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. *Journal of Global Optimization 71*, 1 (2018), 37–55.

[16] 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.

[17] 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.

[18] HARE, W., AND SAGASTIZÁBAL, C. A redistributed proximal bundle method for nonconvex optimization. *SIAM Journal on Optimization 20*, 5 (2010), 2442–2473.

[19] JOKI, K., BAGIROV, A. M., KARMITSA, N., AND MÄKELÄ, M. M. A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. *Journal of Global Optimization 68* (2017), 501–535.

[20] JOKI, K., BAGIROV, A. M., KARMITSA, N., MÄKELÄ, M. M., AND TAHERI, S. Double bundle method for nonsmooth DC optimization, TUCS. Tech. rep., No. 1173, Turku Centre for Computer Science, Turku, 2017.

[21] JOKI, K., BAGIROV, A. M., KARMITSA, N., MÄKELÄ, M. M., AND TAHERI, S. Double bundle method for finding Clarke stationary points in nonsmooth DC programming. *SIAM Journal of Optimizatiom 28*, 2 (2018), 1892–1919.

[22] KAPPEL, F., AND KUNTSEVICH, A. V. An implementation of Shor's *r*-algorithm. *Computational Optimization and Applications 15*, 2 (2000), 193–205.

[23] KIWIEL, K. C. *Methods of Descent for Nondifferentiable Optimization*. Springer-Verlag, Berlin, 1985.

[24] LE THI HOAI, A., AND PHAM DINH, T. Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *Journal of Global Optimization 11* (1997), 253–285.

[25] LE THI HOAI, A., AND PHAM DINH, T. The DC (differnece of convex functions) programming and DCA revised with DC models of real world nonconvex optimization problems. *Annals of Operations Research 133*, 1–4 (2005), 23–46.

[26] LUKŠAN, L., AND VLČEK, J. A bundle-Newton method for nonsmooth unconstrained minimization. *Mathematical Programming 83*, 1 (1998), 373–391.

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

[28] MIFFLIN, R., SUN, D., AND QI, L. Quasi-Newton bundle-type methods for nondifferentiable convex optimization. *SIAM Journal on Optimization 8*, 2 (1998), 583–603.

[29] NEMIROVSKI, A., JUDITSKY, A., LAN, G., AND SHAPIRO, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal of Optimizatiom 19*, 4 (2009), 1574–1609.

[30] NESTEROV, Y. Primal-dual subgradient methods for convex problems. *Mathematical Programming 120*, 1 (2009), 221–259.

[31] POLYAK, B. T. *Introduction to Optimization*. Optimization Software Inc., New York, 1987.

[32] SHOR, N. Z. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag, Berlin, 1985.

[33] SOUZA, J. C. O., OLIVEIRA, P. R., AND SOUBEYRAN, A. Global convergence of a proximal linearized algorithm for difference of convex functions. *Optimization Letters 10* (2016), 1529–1539.

[34] STREKALOVSKY, A. S. Global optimality conditions for nonconvex optimization. *Journal of Global Optimization 12* (1998), 415–434.

[35] TUY, H. *Convex Analysis and Global Optimization*. Kluwer Academic Publishers, Dordrescht, 1998.

Appendix: New DC Test Problems

All the test problems are unconstrained DC optimization problems, where objective functions are presented as DC functions:

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

In what follows we first present some convex functions. These functions are used as f_1 and f_2 to obtain a DC problem. Different combinations used with suitable starting points are given in Table 1. The functions and starting points are combined such that the resulting problem satisfies the following criteria:

- the problem is bounded from below,
- the function $f = f_1 - f_2$ is nonconvex,
- the starting point for the problem is not critical,
- the problem has not too many local solution. That is, algorithms converge mostly to same (or only couple of) minima from different starting points,
- the computational time with 1000 variables is more than 1 sec. with `AggSub` (i.e. the problem is not too simple),

Convex functions

1. $f(x) = (n+1)\max\left\{x_i^2 : \; i = 1,\dots,n\right\}$
2. $f(x) = \max\left\{x_i^2 : \; i = 1,\dots,n\right\}$ (*Generalization of MAXQ*)
3. $f(x) = \sum_{i=1}^n x_i^2$
4. $f(x) = n\max\left\{|x_i| : \; i = 1,\dots,n\right\}$
5. $f(x) = \sum_{i=1}^n |x_i|$
6. $f(x) = 20\max\left\{\left|\sum_{i=1}^n (x_i - x_i^*)t_j^{i-1}\right| : j = 1,\dots,20\right\}$, $t_j = 0.05j, \; j = 1,\dots,20$
7. $f(x) = \sum_{j=1}^{20}\left|\sum_{i=1}^n (x_i - x_i^*)t_j^{i-1}\right|$, $t_j = 0.05j, \; j = 1,\dots,20$
8. $f(x) = \sum_{i=2}^n |x_i - x_{i-1}|$
9. $f(x) = \max\left\{\left|\sum_{j=1}^n \frac{x_j}{i+j-1}\right| : i = 1,\dots,n\right\}$ (*Generalization of MXHILB*)

10. $f(x) = \sum_{i=1}^{n-1} \max\left\{-x_i - x_{i+1}, -x_i - x_{i+1} + (x_i^2 + x_{i+1}^2 - 1)\right\}$ (*Chained LQ*)

11. $f(x) = \sum_{i=1}^{n-1} \max\left\{x_i^4 + x_{i+1}^2, (2-x_i)^2 + (2-x_{i+1})^2, 2e^{-x_i+x_{i+1}}\right\}$ *Chained CB3 I*

12. $f(x) = \max\left\{\sum_{i=1}^{n-1}(x_i^4 + x_{i+1}^2), \sum_{i=1}^{n-1}((2-x_i)^2 + (2-x_{i+1})^2), \sum_{i=1}^{n-1}(2e^{-x_i+x_{i+1}})\right\}$ *Chained CB3 II*

13. $f(x) = \sum_{i=1}^{n} |x_i| + 10\sum_{i=1}^{n} \max\left\{2(x_i^2 - x_i - 1), 0\right\}$

14. $f(x) = n \max\left\{ \left|\sum_{j=1}^{n} \frac{x_j}{i+j-1}\right| \,:\, i = 1,\dots,n\right\}$

15. $f(x) = \sum_{i=1}^{n} \left|\sum_{j=1}^{n} \frac{x_j}{i+j-1}\right|$

16. $f(x) = (n-1)\max\left\{ \max\left\{x_i^4 + x_{i+1}^2, (2-x_i)^2 + (2-x_{i+1})^2, 2e^{-x_i+x_{i+1}}\right\} : i = 1,\dots,n-1\right\}$

17. $f(x) = \max\left\{2\sum_{i=1}^{n-1}\left(x_i^2 + (x_{i+1}-1)^2 + x_{i+1} - 1\right), 0\right\}$

18. $f(x) = \sum_{i=1}^{n-1}\left(x_i^2 + (x_{i+1}-1)^2 + x_{i+1} - 1\right)$

Starting points

1. $x_0 = (1, 1, \dots, 1)^T \in \mathbb{R}^n$
2. $x_0 = (i,\ i = 1, \dots,]n/2[,\ -i,\ i =]n/2[+1, \dots, n)^T$
3. $x_{0,i} = 1$ for odd i and $x_{0,i} = -1$ for even i, $\quad i = 1, \dots, n$
4. $x_{0,i} = -1.5$ for odd i and $x_{0,i} = 2$ for even i, $\quad i = 1, \dots, n$
5. $x_{0,i} = i, \quad i = 1, \dots, n$

New DC problems (152 different).

Table 1: Combinations of problems.

f_1	f_2	x_0
1	3	1, 2, 3, 4, 5
1	5	1, 2, 3, 4, 5
1	6	1, 2, 3, 4, 5
1	9	1, 2, 3, 4, 5
2	5	1, 2, 3, 4, 5
3	2	1, 2, 3, 4, 5
3	4	1, 2, 3, 4, 5
3	5	1, 2, 3, 4, 5
4	9	1, 2, 3, 4, 5
5	9	1,2,3,4,5
10	9	1, 2, 3, 4, 5
10	15	1, 2, 3, 4, 5
11	7	1, 2, 3, 4, 5
11	9	2, 3, 4, 5
11	14	1, 2, 3, 4, 5 (1)
11	15	2, 3, 4, 5
12	9	2, 3, 4, 5
12	14	2, 3, 4, 5 (2)
12	15	1, 2, 3, 4, 5
13	14	2, 3, 4
13	15	1, 2, 3, 4, 5
16	2	2, 3, 4, 5
16	3	2, 3, 4, 5
16	4	2, 3, 4, 5
16	5	2, 3, 4, 5
16	6	1, 3, 4, 5 (3)
16	7	2, 3, 4, 5
16	8	2, 3, 4, 5
16	9	2, 3, 4, 5
16	10	2, 3, 4, 5
16	14	2, 3, 4, 5
16	15	2, 3, 4, 5
16	18	2, 3, 4, 5
17	15	1, 2, 3, 4, 5

1 Omitted also $x_0 = 2$ with $n = 200, 500$ and 1000, and $x_0 = 5$ with $n = 200, 500$.

2 Omitted also $x_0 = 4$ with $n = 100$ and $x_0 = 5$ with $n = 200, 500$.

3 Omitted also $x_0 = 5$ with $n = 100$.

ISBN 978-952-12-3791-1
ISSN 1239-1891