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

Double Bundle Method for Nonsmooth DC Optimization

Turku Centre *for* Computer Science

TUCS Technical Report
No 1173 , January 2017

Double Bundle Method for Nonsmooth DC Optimization

Kaisa Joki
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
kaisa.joki@utu.fi
Adil M. Bagirov
Faculty of Science and Technology, Federation University Australia
University Drive, Mount Helen, PO Box 663, Ballarat, VIC 3353, Australia
a.bagirov@federation.edu.au
Napsu Karmitsa
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
napsu@karmitsa.fi
Marko M. Mäkelä
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
makela@utu.fi
Sona Taheri
Faculty of Science and Technology, Federation University Australia
University Drive, Mount Helen, PO Box 663, Ballarat, VIC 3353, Australia
s.taheri@federation.edu.au

TUCS Technical Report
No 1173 , January 2017

Abstract

The aim of this paper is to introduce a new proximal double bundle method for unconstrained nonsmooth DC optimization, where the objective function is presented as a difference of two convex (DC) functions. The novelty in our method is a new stopping procedure guaranteeing Clarke stationarity for solutions by utilizing only DC components of the objective function. This optimality condition is stronger than the criticality condition typically used in DC programming. Moreover, if a candidate solution is not Clarke stationary, then the stopping procedure yields a descent direction. With this new stopping procedure we can avoid some drawbacks, which are encountered when criticality is used. The finite convergence of the method is proved to a Clarke stationary point under mild assumptions. Finally, some encouraging numerical results are presented.

Keywords: Nonsmooth optimization, Nonconvex optimization, DC functions, Bundle methods, Cutting plane model, Clarke stationarity

TUCS Laboratory
Turku Optimization Group (TOpGroup)

1 Introduction

Although convexity is a preferred feature in optimization, it is too restrictive assumption in many practical applications, since functions involved are often nonconvex. In addition to the nonconvexity, we frequently encounter nonsmoothness in real-life optimization problems, which means that functions are not necessarily differentiable and have discontinuous gradients. Therefore, we need to handle problems which are at the same time both nonsmooth and nonconvex.

A class of functions represented as a difference of convex (DC) functions constitutes an important subclass of nonconvex functions as these functions preserve, with some modifications, important properties of convex functions. Another advantage is that the set of DC functions maintains the DC structure under simple algebraic operations frequently encountered in optimization, like scalar multiplication and lower and upper envelopes. In addition, this class is very broad. For example, every continuous function, defined on a compact convex set, can be approximated by a DC function with any desired precision [16, 36].

Mathematical programming problems with DC objective and constraint functions are called DC programming problems. DC programming is an important subclass of nonconvex optimization and many practical problems can be modelled as a DC programming problem. These problems include the production-transportation planning [15], location planning [30], engineering design [21], cluster analysis [4, 29], clusterwise linear regression analysis [5] and supervised data classification [1] to name a few.

In all the above mentioned problems, objective and/or constraint functions can be explicitly expressed in a DC form. Although, for some important functions such a representation is available, exploiting this representation for an arbitrary DC function may become a hard task, since DC representations appear often in an implicit form. It is also worth noting that DC decompositions are not unique and each DC function has an infinite number of different DC representations.

To date, DC programming has been considered as a part of global optimization and several algorithms have been designed to solve it globally (see [16, 36] and references therein), while the development of local search methods in DC programming has attracted significantly less attention. There exist also DC algorithms designed to handle nonsmoothess encountered in DC programming problems. For example, in [2] an algorithm based on quasidifferentials of DC functions and discrete gradients is developed. Some other alternatives for nonsmooth DC programming include a codifferential method [6], a proximal linearized algorithm [34] and a proximal bundle method [17] utilizing nonconvex cutting planes. In addition, a gradient splitting method introduced in [12] can be modified for minimizing DC functions.

A stopping condition in most nonsmooth DC programming algorithms guarantees only criticality of the solution point and this condition is weaker than Clarke stationarity typically used in general nonconvex nonsmooth

optimization. Unfortunately, it may happen that these DC programming algorithms stop at a point which is neither a local minimizer nor a saddle point. This nondesirable feature is often a consequence of the selected DC decomposition, since it affects the criticality condition tested. However, there is no efficient way to detect the most suitable DC decomposition among the infinite set of possible ones, in general. In addition, the very specific structure of a problem may have a strong influence on this selection.

In this paper, we introduce a new proximal double bundle method DBDC for unconstrained nonsmooth DC minimization problems. The main idea in DBDC is to combine the proximal bundle method for nonsmooth DC programming PBDC [17] with a new stopping procedure. With this combination our aim is to guarantee a stronger optimality condition than the one typically used in DC programming, but at the same time, preserve all the good features obtained from the usage of the DC structure in the PBDC method. In this way, we can be more assured about the quality of the solutions obtained and avoid in some sense arbitrary solution points.

The new DBDC method shares the good properties of PBDC, which is to our knowledge the only bundle method utilizing explicitly the DC decomposition of the objective function. Using this decomposition, the nonconvex cutting plane model is developed to capture both the convex and the concave behaviour of the objective and describe well the actual behaviour of a DC function. The new stopping procedure, in its turn, is designed to ensure Clarke stationarity for candidate solutions by using only information about the DC components. This way we are able to exploit the DC structure through the algorithm. To prove the finite convergence for the stopping procedure we assume that the subdiffrentials of DC components are polytopes. This assumption is not very restrictive and in practical applications it is nearly always satisfied.

The rest of the paper is organized as follows: In Section 2, we present some basic definitions and results from nonsmooth analysis and DC programming. In Section 3, we point out some disadvantages of critical points, which are the most natural choice for the candidate solutions in DC programming. The new stopping procedure guaranteeing Clarke stationarity is presented in Section 4 together with its convergence analysis. Section 5 describes the new minimization algorithm DBDC utilizing the new stopping procedure. Some numerical results are reported in Section 6 and, finally, in Section 7 we give some concluding remarks.

2 Preliminaries

Consider an *unconstrained DC minimization problem* of the form

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

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a difference of two convex functions f_1 and f_2. Such a function f defined on $\mathbb{R}^n$ is called a *DC function* and the representation $f_1 - f_2$ is the *DC decomposition* of the function f. The convex functions f_1 and f_2 defined on $\mathbb{R}^n$ are called *DC components* of f. Note that f is often nonconvex and it needs not to be differentiable. In addition, when f is nonsmooth then at least one of the DC components is also nonsmooth.

Next we present briefly some concepts and basic results from nonsmooth analysis and DC programming. For more details we refer to [3, 7, 13, 21, 26, 31, 32]. In what follows, $\|\cdot\|$ is the norm in the n-dimensional real Euclidean space $\mathbb{R}^n$, $B_\varepsilon(\boldsymbol{x})$ is the open ball with a center $\boldsymbol{x} \in \mathbb{R}^n$ and a radius $\varepsilon > 0$ and $\boldsymbol{x}^T\boldsymbol{y}$ is the usual inner product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

The *subdifferential* (or generalized gradient) of a convex function f at a point $\boldsymbol{x} \in \mathbb{R}^n$ is the set [32]

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

and each vector $\boldsymbol{\xi} \in \partial_c f(\boldsymbol{x})$ is called a *subgradient* of f at $\boldsymbol{x}$. In particular, if f is both convex and differentiable, then $\partial_c f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$.

The *generalized subdifferential* (or Clarke's gradient) of a locally Lipschitz continuous function f at a point $\boldsymbol{x} \in \mathbb{R}^n$ is given by [7]

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

where conv denotes the convex hull of a set. Each $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ is a *subgradient* calculated at $\boldsymbol{x}$. A point $\boldsymbol{x}^* \in \mathbb{R}^n$ is called *Clarke stationary* if $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. Clarke stationarity is a necessary condition for local optimality and, in the convex case, it guarantees the global optimality of the solution. It is also well known that for a convex function f defined on $\mathbb{R}^n$ we have $\partial f(\boldsymbol{x}) = \partial_c f(\boldsymbol{x})$ [7]. In what follows, we denote the subdifferential of a convex DC component f_i by $\partial f_i(\boldsymbol{x})$ for $i = 1, 2$.

For $\varepsilon \geq 0$, the *Goldstein ε-subdifferential* of a locally Lipschitz continuous function f at a point $\boldsymbol{x} \in \mathbb{R}^n$ is defined as [26]

$$\partial_\varepsilon^G f(\boldsymbol{x}) = \operatorname{cl}\operatorname{conv}\{\partial f(\boldsymbol{y}) \mid \boldsymbol{y} \in B_\varepsilon(\boldsymbol{x})\}.$$

The set $\partial_\varepsilon^G f(\boldsymbol{x})$ is an extension of the generalized subdifferential and $\partial f(\boldsymbol{x}) \subseteq \partial_\varepsilon^G f(\boldsymbol{x})$ for each $\varepsilon \geq 0$. In practice, we are usually satisfied with solutions fulfilling the condition $\mathbf{0} \in \partial_\varepsilon^G f(\boldsymbol{x})$, since the Goldstein ε-subdifferential approximates the set $\partial f(\boldsymbol{x})$.

Unlike a convex function, a general locally Lipschitz continuous function is not necessarily directionally differentiable. However, our objective function f is a finite valued DC function and it is directionally differentiable at any $\boldsymbol{x} \in \mathbb{R}^n$ [3]. This means that the *directional derivative* of f at $\boldsymbol{x}$ exists in every direction $\boldsymbol{d} \in \mathbb{R}^n$ being defined as

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \lim_{t\downarrow 0} \frac{f(\boldsymbol{x} + t\boldsymbol{d}) - f(\boldsymbol{x})}{t}.$$

Moreover, for a DC function we have $f'(\boldsymbol{x};\boldsymbol{d}) = f_1'(\boldsymbol{x};\boldsymbol{d}) - f_2'(\boldsymbol{x};\boldsymbol{d})$. If $f'(\boldsymbol{x};\boldsymbol{d}) < 0$ for some $\boldsymbol{d} \in \mathbb{R}^n$, then $\boldsymbol{d}$ is a descent direction meaning that there exists $\varepsilon > 0$ such that $f(\boldsymbol{x} + t\boldsymbol{d}) < f(\boldsymbol{x})$ for all $t \in (0, \varepsilon]$ [3].

Next we present for a DC function some necessary conditions for local optimality.

Theorem 2.1. [21, 35] *Let f_1 and f_2 be convex functions. If $\boldsymbol{x}^* \in \mathbb{R}^n$ is a local minimizer of $f = f_1 - f_2$, then*

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

and points satisfying (2) *are called* inf-stationary points. *Furthermore, the condition* (2) *guarantees local optimality if f_2 is a polyhedral convex function of the form $f_2(\boldsymbol{x}) = \max_{i=1,\ldots,m}\left\{\boldsymbol{a}_i^T\boldsymbol{x} - b_i\right\}$, where $\boldsymbol{a}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$.*

Unfortunately, the inf-stationarity condition (2) is not easy to verify, since in practice we usually do not know or cannot calculate the whole subdifferentials of the DC components f_1 and f_2 at some point. Therefore, in solution algorithms a relaxed form of the condition (2) is often used requiring that [14, 21, 35]

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

and a point $\boldsymbol{x}^* \in \mathbb{R}^n$ satisfying this condition is called a *critical point*. Due to Theorem 2.1, the condition (3) is also a necessary optimality condition for local optimality, and all the minimizers of f can be found among the set of critical points.

There exist some interesting relationships between inf-stationary, Clarke stationary and critical points. First of all, inf-stationarity always implies Clarke stationarity. Second, a Clarke stationary point is always also a critical point. However, for another way around these implications do not hold without some extra assumptions. One exception is the case, where f_2 is differentiable at a critical point $\boldsymbol{x}^* \in \mathbb{R}^n$, since then we have [7]

$$\partial f_2(\boldsymbol{x}^*) = \{\nabla f_2(\boldsymbol{x}^*)\} \subseteq \partial f_1(\boldsymbol{x}^*) \quad \text{and} \quad \mathbf{0} \in \partial f(\boldsymbol{x}^*) = \partial f_1(\boldsymbol{x}^*) - \partial f_2(\boldsymbol{x}^*)$$

indicating also inf-stationarity and Clarke stationarity of the point $\boldsymbol{x}^*$. If only the first DC component f_1 is differentiable at a critical point $\boldsymbol{x}^* \in \mathbb{R}^n$, then we are able to obtain only Clarke stationarity. This is due to the fact that [7]

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*) = \partial f_1(\boldsymbol{x}^*) - \partial f_2(\boldsymbol{x}^*) = \{\nabla f_1(\boldsymbol{x}^*)\} - \partial f_2(\boldsymbol{x}^*),$$

but the subdifferential of f_2 at $\boldsymbol{x}^*$ contains more than one element and, therefore, it cannot be a subset of $\partial f_1(\boldsymbol{x}^*) = \{\nabla f_1(\boldsymbol{x}^*)\}$. The relationships between different stationary points are also presented in Figure 1.

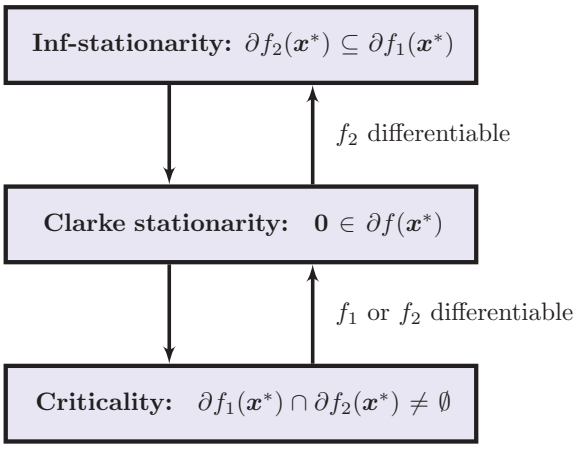

Figure 1: Relationships between different stationarities

3 Difficulties caused by criticality

Even if critical points are good candidates for the minimizer of a DC function f, they have some disadvantages. First, criticality is a weaker condition than Clarke stationarity. This follows from the subdifferential calculus which only guarantees that for a DC function $f = f_1 - f_2$ we have [7]

$$\partial f(\boldsymbol{x}) \subseteq \partial f_1(\boldsymbol{x}) - \partial f_2(\boldsymbol{x}). \tag{4}$$

Therefore, the difference of arbitrary subgradients of f_1 and f_2 does not need to belong to the set $\partial f(\boldsymbol{x})$. Nevertheless, there are some exceptions to this as we have already seen in the previous section. Second, it is possible that a critical point is neither a local optimum nor a saddle point of the original objective f. Thus, a critical point might fail to give us any useful information about the objective. In some sense this means that, in practice, solution algorithms may stop in the middle of nowhere if criticality is used as a stopping condition.

Next we present two simple examples illustrating the fact that a critical point can be easily located in an unfavourable place. The first one of these examples shows the effect of the bad selection of the DC decomposition. However, in the second example it is not easy to see if the DC decomposition of f could be selected such a way that the undesirable behaviour could be avoided.

Example 3.1. Let us consider a linear function $f(x) = x$, where $x \in \mathbb{R}$. A DC decomposition of f is obtained when we select

$$f_1(x) = \max\{-x, 2x\} \quad \text{and} \quad f_2(x) = \max\{-2x, x\}.$$

Therefore, at a point $x^* = 0$ the DC components f_1 and f_2 are not differentiable and the subdifferentials are

$$\partial f_1(0) = [-1, 2] \quad \text{and} \quad \partial f_2(0) = [-2, 1].$$

From this we obtain

$$\partial f_1(0) \cap \partial f_2(0) = [-1, 1] \neq \emptyset$$

and the point x^* is a critical point. However, since the original objective f is differentiable at $x^* = 0$ and $\partial f(0) = \{1\}$, the point x^* is not Clarke stationary and describes no interesting feature for the function f.

Example 3.2. Let us consider a simple nonlinear problem, where the objective function $f : \mathbb{R} \to \mathbb{R}$ has DC components defined as

$$f_1(x) = \max\{x^2, x\} \quad \text{and} \quad f_2(x) = \max\{0.5x^2, -x\}.$$

Let us look closer the point $x^* = 0$. First of all, the DC components f_1 and f_2 are not differentiable at x^* and their subdifferentials are

$$\partial f_1(0) = [0, 1] \quad \text{and} \quad \partial f_2(0) = [-1, 0].$$

Since

$$\partial f_1(0) \cap \partial f_2(0) = \{0\} \neq \emptyset,$$

the point $x^* = 0$ is a critical point of f. However, calculating the subdifferential of the function f at $x^* = 0$, we notice that f is actually differentiable at x^* and $\partial f(0) = \{1\}$. Therefore, the point x^* is not a local minimizer or even a saddle point and criticality does not give us any useful information. Graphs of the DC components and the objective f are illustrated in Figure 2.

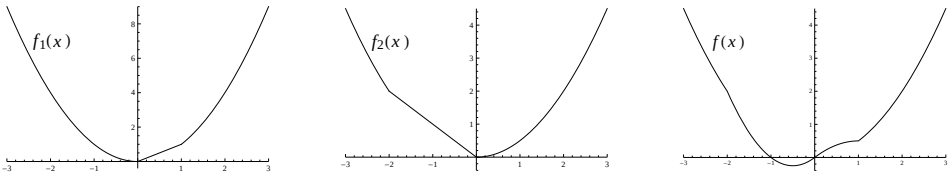

Figure 2: The DC components $f_1(x)$ and $f_2(x)$ and the objective $f(x)$

The nondesirable behaviour in critical points is often a result of the selected DC decomposition. However, since a DC function has an infinite number of different DC decompositions, it may be impossible to know which one of them should be selected and this problem is in general an open question in DC programming. One exception is a polynomial case: in [9] a special algorithm is designed to find the best DC representation of polynomials. In some easy cases, we can also see directly how a good DC decomposition should be chosen (e.g. in Example 3.1 we could set $f_1(x) = x$ and $f_2(x) = 0$). However, in real world applications the situation is rarely this simple but it depends on the overall structure of the problem. Therefore, there is no efficient way to avoid this bad feature of critical points in solution algorithms.

4 Guaranteeing Clarke stationarity

In this section, we describe the new stopping procedure which either guarantees Clarke stationarity for a point under consideration or generates a descent direction yielding a better iteration point. The novelty in the procedure is its ability to compute subgradients for a DC function utilizing only its DC components. This means that the new stopping procedure will ensure that the difference of subgradients of the DC components f_1 and f_2 belongs to the subdifferential of $f = f_1 - f_2$. As seen in the previous section, this does not hold in general. Later on, the stopping procedure is used in the 'main iteration' of the DBDC method to detect the cases where a critical point or a promising candidate solution fulfils also Clarke stationarity.

Next we consider the convex DC components $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, 2$. The *support function* of the subdifferential $\partial f_i(\boldsymbol{x})$ at $\boldsymbol{x} \in \mathbb{R}^n$ is

$$\sigma_i(\boldsymbol{d}) \equiv f_i'(\boldsymbol{x}; \boldsymbol{d}) = \max\left\{\boldsymbol{v}^T\boldsymbol{d} \mid \boldsymbol{v} \in \partial f_i(\boldsymbol{x})\right\} \quad \text{for } i = 1, 2.$$

Taking any direction $\boldsymbol{d} \in \mathbb{R}^n$ such that $\boldsymbol{d} \neq \boldsymbol{0}$, we can consider the following set

$$G_i(\boldsymbol{x}; \boldsymbol{d}) = \left\{\boldsymbol{\xi} \in \partial f_i(\boldsymbol{x}) \mid \boldsymbol{\xi}^T\boldsymbol{d} = \sigma_i(\boldsymbol{d})\right\} \quad \text{for } i = 1, 2.$$

Since f_1 and f_2 are convex, they are also weakly semismooth [27, 33]. This implies that

$$f_i'(\boldsymbol{x}; \boldsymbol{d}) = \lim_{k\to\infty} \boldsymbol{v}_k^T\boldsymbol{d} \tag{5}$$

for $i = 1, 2$ and any sequences $\{\boldsymbol{v}_k\}, \{t_k\}$ such that $\boldsymbol{v}_k \in \partial f_i(\boldsymbol{x} + t_k\boldsymbol{d})$ and $t_k \downarrow 0$ as $k \to \infty$. Consider the set

$$U_i(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi} \in \mathbb{R}^n \mid \exists\{\boldsymbol{v}_k\} \text{ and } \{t_k\}, \boldsymbol{v}_k \in \partial f_i(\boldsymbol{x}+t_k\boldsymbol{d}),\ \boldsymbol{v}_k \to \boldsymbol{\xi} \text{ and } t_k \downarrow 0 \text{ as } k \to \infty\}$$

for $i = 1, 2$. It follows from (5) that

$$U_i(\boldsymbol{x}; \boldsymbol{d}) \subseteq G_i(\boldsymbol{x}; \boldsymbol{d}) \quad \text{for } i = 1, 2. \tag{6}$$

From the definition of the set $U_i(\boldsymbol{x}; \boldsymbol{d})$ we also obtain that for any $\varepsilon > 0$ and $i = 1, 2$ there exists $t_0 > 0$ such that

$$\partial f_i(\boldsymbol{x} + t\boldsymbol{d}) \subset U_i(\boldsymbol{x}; \boldsymbol{d}) + B_\varepsilon(\boldsymbol{0}) \quad \forall t \in (0, t_0).$$

This together with (6) implies that

$$\partial f_i(\boldsymbol{x} + t\boldsymbol{d}) \subset G_i(\boldsymbol{x}; \boldsymbol{d}) + B_\varepsilon(\boldsymbol{0}) \tag{7}$$

for $i = 1, 2$ and all $t \in (0, t_0)$.

Since the support function σ_i is locally Lipschitz continuous [3], it is differentiable almost everywhere. This means that at the point $\boldsymbol{x} \in \mathbb{R}^n$ there exists a set $T_i \subset \mathbb{R}^n$ of full measure such that the set $G_i(\boldsymbol{x}; \boldsymbol{d})$ is singleton for any $\boldsymbol{d} \in T_i$. In the following, we let a set

$$T_{DC} = T_1 \cap T_2 \subset \mathbb{R}^n$$

be the set of full measure where both $G_1(\boldsymbol{x}; \boldsymbol{d})$ and $G_2(\boldsymbol{x}; \boldsymbol{d})$ are singleton for any $\boldsymbol{d} \in T_{DC}$ at $\boldsymbol{x}$.

Theorem 4.1. *Let $f = f_1 - f_2$ be a DC function, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{d} \in T_{DC}$, $G_1(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi}_1\}$ and $G_2(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi}_2\}$. Then*

$$\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x}).$$

Proof. It follows from (7) that for any $\varepsilon > 0$ there exists $t_0 > 0$ such that

$$\partial f_1(\boldsymbol{x} + t\boldsymbol{d}) \subset \{\boldsymbol{\xi}_1\} + B_\varepsilon(\boldsymbol{0}), \quad \partial f_2(\boldsymbol{x} + t\boldsymbol{d}) \subset \{\boldsymbol{\xi}_2\} + B_\varepsilon(\boldsymbol{0}) \quad \forall t \in (0, t_0).$$

This means that

$$\|\boldsymbol{v} - \boldsymbol{\xi}_1\| < \varepsilon, \quad \|\boldsymbol{w} - \boldsymbol{\xi}_2\| < \varepsilon \tag{8}$$

for all $\boldsymbol{v} \in \partial f_1(\boldsymbol{x} + t\boldsymbol{d})$, $\boldsymbol{w} \in \partial f_2(\boldsymbol{x} + t\boldsymbol{d})$ and $t \in (0, t_0)$. On the other hand, the rule (4) implies that

$$\partial f(\boldsymbol{x} + t\boldsymbol{d}) \subseteq \partial f_1(\boldsymbol{x} + t\boldsymbol{d}) - \partial f_2(\boldsymbol{x} + t\boldsymbol{d}), \ t \geq 0.$$

Therefore, for any $\boldsymbol{\xi}_t \in \partial f(\boldsymbol{x} + t\boldsymbol{d})$ there exists $\boldsymbol{v}_t \in \partial f_1(\boldsymbol{x} + t\boldsymbol{d})$ and $\boldsymbol{w}_t \in \partial f_2(\boldsymbol{x} + t\boldsymbol{d})$ such that $\boldsymbol{\xi}_t = \boldsymbol{v}_t - \boldsymbol{w}_t$ and taking into account (8) we obtain

$$\|\boldsymbol{\xi}_t - (\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2)\| \leq \|\boldsymbol{v}_t - \boldsymbol{\xi}_1\| + \|\boldsymbol{w}_t - \boldsymbol{\xi}_2\| < 2\varepsilon$$

for all $\boldsymbol{\xi}_t \in \partial f(\boldsymbol{x} + t\boldsymbol{d})$ and $t \in (0, t_0)$. This means that

$$\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x} + t\boldsymbol{d}) + B_{2\varepsilon}(\boldsymbol{0}) \quad \forall t \in (0, t_0). \tag{9}$$

Upper semicontinuity of the subdifferential $\partial f(\boldsymbol{x})$ [7] implies that for any $\varepsilon > 0$ there exists $t_1 > 0$ such that

$$\partial f(\boldsymbol{x} + t\boldsymbol{d}) \subset \partial f(\boldsymbol{x}) + B_\varepsilon(\boldsymbol{0}) \quad \forall t \in (0, t_1).$$

Then from (9) we have

$$\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x}) + B_{3\varepsilon}(\boldsymbol{0}).$$

Since $\varepsilon > 0$ is arbitrary we get that $\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x})$, which proves the theorem. □

Corollary 4.2. *Let $\boldsymbol{x} \in \mathbb{R}^n$, $f = f_1 - f_2$ be a DC function and $T_{DC} \subset \mathbb{R}^n$ be a set of full measure such that the sets $G_1(\boldsymbol{x}; \boldsymbol{d})$ and $G_2(\boldsymbol{x}; \boldsymbol{d})$ are singleton for $\boldsymbol{d} \in T_{DC}$. Consider the following set*

$$\partial_{T_{DC}} f(\boldsymbol{x}) = \text{cl conv}\left\{\boldsymbol{\xi} \in \mathbb{R}^n \,|\, \exists \boldsymbol{d} \in T_{DC}, \ \boldsymbol{\xi} = \boldsymbol{\xi}_1 - \boldsymbol{\xi}_2, \ \boldsymbol{\xi}_1 \in G_1(\boldsymbol{d}), \ \boldsymbol{\xi}_2 \in G_2(\boldsymbol{d})\right\}.$$

Then

$$\partial_{T_{DC}} f(\boldsymbol{x}) \subseteq \partial f(\boldsymbol{x}).$$

These results show that, in order to compute subgradients from the Clarke subdifferential of a DC function utilizing only subgradients of DC components, it is important to design an algorithm which allows for any direction $\boldsymbol{d} \in \mathbb{R}^n$ to find a direction $\bar{\boldsymbol{d}} \in T_{DC}$ such that $\|\boldsymbol{d} - \bar{\boldsymbol{d}}\| < \delta$ for any sufficiently small $\delta > 0$.

REMARK 4.3. In [8] directions $\boldsymbol{d} \in \mathbb{R}^n$ whose sets $G_i(\boldsymbol{x}; \boldsymbol{d})$ are singleton are used to define the so-called Demyanov difference of two convex compact sets in $\mathbb{R}^n$.

4.1 Calculation of appropriate directions

Next we will show how we can find the direction $\bar{\boldsymbol{d}} \in T_{DC}$ for any $\boldsymbol{d} \in \mathbb{R}^n$. For that reason we utilize the so-called R-sets [3] and make the following assumption:

A1 the subdifferentials $\partial f_1(\boldsymbol{x})$ and $\partial f_2(\boldsymbol{x})$ are polytopes at any $\boldsymbol{x} \in \mathbb{R}^m$.

Let $A \subset \mathbb{R}^n$ be the polytope, that is, it can be represented as:

$$A = \operatorname{conv} A_0$$

where

$$A_0 = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\},\ \boldsymbol{a}_i \in \mathbb{R}^n,\ m \geq 1.$$

Let

$$V = \{\boldsymbol{g} \in \mathbb{R}^n \mid \boldsymbol{g} = (g_1, \ldots, g_n),\ |g_i| = 1,\ i = 1, \ldots, n\}.$$

For a given $\boldsymbol{g} \in V$, introduce the following sets:

$$R_0(\boldsymbol{g}) \equiv R_0 = A_0,$$
$$R_j(\boldsymbol{g}) = \arg\max \{v_j g_j \mid \boldsymbol{v} \in R_{j-1}(\boldsymbol{g})\},\ j = 1, \ldots, n.$$

It is clear that

$$R_j(\boldsymbol{g}) \neq \emptyset,\ \forall j \in \{0, \ldots, n\} \quad \text{and} \quad R_j(\boldsymbol{g}) \subseteq R_{j-1}(\boldsymbol{g}),\ \forall j \in \{1, \ldots, n\}.$$

Moreover,

$$v_k = w_k \quad \forall \boldsymbol{v}, \boldsymbol{w} \in R_j(\boldsymbol{g}),\ k = 1, \ldots, j. \tag{10}$$

Lemma 4.4. *For any $\boldsymbol{g} \in V$, the set $R_n(\boldsymbol{g})$ is a singleton.*

Proof. It follows from (10) that for any $\boldsymbol{v}, \boldsymbol{w} \in R_n(\boldsymbol{g})$ we have $\boldsymbol{v} = \boldsymbol{w}$. □

For $\boldsymbol{g} \in V$ and $\alpha > 0$, we introduce the sequence of n vectors $\boldsymbol{e}^j(\alpha) = (\alpha g_1, \alpha^2 g_2, \ldots, \alpha^j g_j, 0, \ldots, 0)$, $j = 1, \ldots, n$. Denote by σ_A the support function of the set A, that is, $\sigma_A(\boldsymbol{d}) = \max\{\boldsymbol{v}^T \boldsymbol{d} \,|\, \boldsymbol{v} \in A\}$ and let

$$G_A(\boldsymbol{e}^i(\alpha)) = \left\{\boldsymbol{a} \in A \mid \boldsymbol{a}^T \boldsymbol{e}^i(\alpha) = \sigma_A(\boldsymbol{e}^i(\alpha))\right\},\ i = 1, \ldots, n.$$

Lemma 4.5. *Let A be a polytope with a finite number of vertices A_0. For a given $\boldsymbol{g} \in V$, there exists $\alpha_0 \in (0, 1]$ such that the set $G_A(\boldsymbol{e}^n(\alpha))$ is a singleton for all $\alpha \in (0, \alpha_0]$.*

Proof. If the set A_0 is a singleton, then the proof is obvious. Therefore, assume that A_0 is not a singleton. According to Lemma 4.4, the set $R_n(\boldsymbol{g})$ is a singleton and without loss of generality we can assume that $\boldsymbol{a} \in A_0$ is the vertex such that $R_n(\boldsymbol{g}) = \{\boldsymbol{a}\}$. Moreover, $A_0 \setminus \{\boldsymbol{a}\} \neq \emptyset$.

Next we take any $\boldsymbol{b} \in A_0 \setminus \{\boldsymbol{a}\}$. Then there exists $r \in \{1, \ldots, n\}$ such that $\boldsymbol{b} \in R_p(\boldsymbol{g})$ for all $p = 0, \ldots, r-1$, but $\boldsymbol{b} \notin R_r(\boldsymbol{g})$. Therefore,

$$a_r g_r > b_r g_r$$

and we define

$$d(\boldsymbol{b}) = a_r g_r - b_r g_r > 0.$$

Since the set A_0 is finite and $d(\boldsymbol{b}) > 0$ for all $\boldsymbol{b} \in A_0 \setminus \{\boldsymbol{a}\}$, we can determine the following number

$$\delta = \min_{\boldsymbol{b} \in A_0 \setminus \{\boldsymbol{a}\}} \left\{ d(\boldsymbol{b}) \right\} > 0.$$

From (10) we also obtain that for any $\boldsymbol{b} \in A_0 \setminus \{a\}$ we have $a_t = b_t$ for $t = 1, \ldots, r-1$ and $r \geq 2$. Thus, we get

$$\begin{aligned} \boldsymbol{a}^T \boldsymbol{e}^n(\alpha) - \boldsymbol{b}^T \boldsymbol{e}^n(\alpha) &= \sum_{t=1}^{n} (a_t - b_t)\alpha^t g_t \\ &= \alpha^r \left[a_r g_r - b_r g_r + \sum_{t=r+1}^{n} (a_t - b_t)\alpha^{t-r} g_t \right] \\ &\geq \alpha^r \left[\delta + \sum_{t=r+1}^{n} (a_t - b_t)\alpha^{t-r} g_t \right]. \end{aligned}$$

Let $D = \max \{ \|\boldsymbol{v}\| \,|\, \boldsymbol{v} \in A_0 \} < \infty$. Since $\alpha \in (0, 1]$ and $\boldsymbol{g} \in V$ we get

$$\left| \sum_{t=r+1}^{n} (a_t - b_t)\alpha^{t-r} g_t \right| \leq 2D\alpha(n-r) < 2D\alpha n$$

for any $\boldsymbol{b} \in A_0 \setminus \{a\}$. If we continue by choosing

$$\alpha_0 = \min \left\{ 1, \frac{\delta}{4Dn} \right\},$$

then for all $\alpha \in (0, \alpha_0]$

$$\left| \sum_{t=r+1}^{n} (a_t - b_t)\alpha^{t-r} g_t \right| < \frac{\delta}{2}.$$

This means that for $\boldsymbol{b} \in A_0 \setminus \{\boldsymbol{a}\}$ we have

$$\boldsymbol{a}^T \boldsymbol{e}^n(\alpha) - \boldsymbol{b}^T \boldsymbol{e}^n(\alpha) > \alpha^r \left(\delta - \frac{\delta}{2} \right) = \frac{\alpha^r \delta}{2} > 0$$

for all $\alpha \in (0, \alpha_0]$. Therefore,

$$\boldsymbol{a}^T \boldsymbol{e}^n(\alpha) - \boldsymbol{b}^T \boldsymbol{e}^n(\alpha) > 0 \quad \forall \boldsymbol{b} \in A_0 \setminus \{\boldsymbol{a}\} \tag{11}$$

and, since the set A_0 contains all the vertices of the polytope A, the inequality (11) holds also for all $\boldsymbol{b} \in A \setminus \{\boldsymbol{a}\}$. This shows our claim. □

Lemma 4.5 shows that for any polytope $A \subset \mathbb{R}^n$ there exists $\alpha_0 \in (0, 1]$ such that the sets $G_A(\boldsymbol{e}^n(\alpha))$ are singletons for any parameter $\alpha \in (0, \alpha_0]$ and the value of $\alpha_0 > 0$ depends only on a polytope.

Lemma 4.6. *Let A be a polytope with a finite number of vertices A_0 and let $\boldsymbol{d} \in \mathbb{R}^n$ be any direction such that $\boldsymbol{d} \neq \boldsymbol{0}$. Then for a given $\boldsymbol{g} \in V$ there exists $\alpha_0 \in (0, 1]$ such that for the direction $\bar{\boldsymbol{d}}(\alpha) = \boldsymbol{d} + \boldsymbol{e}^n(\alpha)$ the set $G_A(\bar{\boldsymbol{d}}(\alpha))$ is a singleton for all $\alpha \in (0, \alpha_0]$. In addition, $G_A(\bar{\boldsymbol{d}}(\alpha)) \subseteq G_A(\boldsymbol{d})$ for all $\alpha \in (0, \alpha_0]$.*

Proof. It follows from Lemma 4.5 that for a polytope $B(\boldsymbol{d})$ defined as

$$B(\boldsymbol{d}) = \operatorname{conv} G_A(\boldsymbol{d})$$

there exists $\alpha_0 \in (0, 1]$ such that the set $G_{B(\boldsymbol{d})}(\boldsymbol{e}^n(\alpha))$ is a singleton for a given $\boldsymbol{g} \in V$ and all $\alpha \in (0, \alpha_0]$. This means that there exists $\boldsymbol{v}_0 \in G_A(\boldsymbol{d}) \cap A_0$ such that

$$\boldsymbol{v}_0^T \boldsymbol{e}^n(\alpha) \geq \boldsymbol{v}^T \boldsymbol{e}^n(\alpha) + \delta_1 \quad \forall \boldsymbol{v} \in G_A(\boldsymbol{d}) \setminus \{\boldsymbol{v}_0\}$$

for some $\delta_1 > 0$. On the other hand, there exists $\delta_2 > 0$ such that

$$\boldsymbol{v}^T \boldsymbol{d} \geq \boldsymbol{w}^T \boldsymbol{d} + \delta_2 \quad \forall \boldsymbol{v} \in G_A(\boldsymbol{d}),\ \boldsymbol{w} \in A_0 \setminus G_A(\boldsymbol{d}).$$

By defining

$$\bar{\boldsymbol{d}}(\alpha) = \boldsymbol{d} + \boldsymbol{e}^n(\alpha)$$

we obtain

$$\|\bar{\boldsymbol{d}}(\alpha) - \boldsymbol{d}\| \leq n\alpha$$

for any $\alpha \in (0, 1]$. We also denote by $D = \max\{\|\boldsymbol{a}\| \mid \boldsymbol{a} \in A_0\} < \infty$. It is clear that

$$\boldsymbol{v}_0^T \bar{\boldsymbol{d}}(\alpha) \geq \boldsymbol{v}^T \bar{\boldsymbol{d}}(\alpha) + \delta_1 \quad \forall \boldsymbol{v} \in G_A(\boldsymbol{d}) \setminus \{\boldsymbol{v}_0\}. \tag{12}$$

Moreover, for any $\boldsymbol{w} \in A_0 \setminus G_A(\boldsymbol{d})$ we have

$$\begin{aligned} \boldsymbol{v}^T \bar{\boldsymbol{d}}(\alpha) - \boldsymbol{w}^T \bar{\boldsymbol{d}}(\alpha) &= (\boldsymbol{v} - \boldsymbol{w})^T \boldsymbol{d} + (\boldsymbol{v} - \boldsymbol{w})^T \boldsymbol{e}^n(\alpha) \\ &\geq \delta_2 - 2Dn\alpha \end{aligned}$$

for all $\boldsymbol{v} \in G_A(\boldsymbol{d})$. By choosing

$$\alpha_1 = \min\Big\{1, \frac{\delta_2}{4Dn}\Big\},$$

we get that for any $\alpha \in (0, \alpha_1]$

$$\boldsymbol{v}^T \bar{\boldsymbol{d}}(\alpha) - \boldsymbol{w}^T \bar{\boldsymbol{d}}(\alpha) \geq \delta_2/2 \quad \forall \boldsymbol{v} \in G_A(\boldsymbol{d}),\ \boldsymbol{w} \in A_0 \setminus G_A(\boldsymbol{d}).$$

Combining this with (12) yields

$$\boldsymbol{v}_0^T \bar{\boldsymbol{d}}(\alpha) - \boldsymbol{w}^T \bar{\boldsymbol{d}}(\alpha) \geq \min\{\delta_1, \delta_2/2\} > 0 \quad \forall \boldsymbol{w} \in A_0 \setminus \{\boldsymbol{v}_0\}$$

and, since A is a polytope, this means that the set $G_A(\bar{\boldsymbol{d}}(\alpha)) = \{\boldsymbol{v}_0\} \subset G_A(\boldsymbol{d})$ for all $\alpha \in (0, \alpha_2)$ where $\alpha_2 = \min\{\alpha_0, \alpha_1\}$. This proves the lemma. □

Now we are ready to present the main results of this section, which are utilized to guarantee the convergence of our new stopping procedure.

Theorem 4.7. *Let $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{d} \in \mathbb{R}^n$ be any direction such that $\boldsymbol{d} \neq \boldsymbol{0}$ and assume that a DC function $f = f_1 - f_2$ satisfies the assumption **A1**. Then for a given $\boldsymbol{g} \in V$ there exists $\alpha_0 \in (0, 1]$ such that for all $\alpha \in (0, \alpha_0]$:*

(i) $\bar{\boldsymbol{d}}(\alpha) = \boldsymbol{d} + \boldsymbol{e}^n(\alpha) \in T_{DC}$;

(ii) $G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha)) \subseteq G_1(\boldsymbol{x}; \boldsymbol{d})$ *and* $G_2(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha)) \subseteq G_2(\boldsymbol{x}; \boldsymbol{d})$;

(iii) $f'(\boldsymbol{x}; \boldsymbol{d}) = (\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2)^T \boldsymbol{d}$ *for* $\boldsymbol{\xi}_1 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$ *and* $\boldsymbol{\xi}_2 \in G_2(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$;

(iv) $\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x})$ *for* $\boldsymbol{\xi}_1 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$ *and* $\boldsymbol{\xi}_2 \in G_2(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$.

Proof. The properties (i) and (ii) follow immediately from Lemma 4.6. Therefore, $f_1'(\boldsymbol{x}; \boldsymbol{d}) = \boldsymbol{\xi}_1^T \boldsymbol{d}$ for $\boldsymbol{\xi}_1 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$ and $f_2'(\boldsymbol{x}; \boldsymbol{d}) = \boldsymbol{\xi}_2^T \boldsymbol{d}$ for $\boldsymbol{\xi}_2 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$. Since $f'(\boldsymbol{x}; \boldsymbol{d}) = f_1'(\boldsymbol{x}; \boldsymbol{d}) - f_2'(\boldsymbol{x}; \boldsymbol{d})$, we obtain the case (iii). The property (iv) is obtained directly from Theorem 4.1 when we take into account (i). □

4.2 Algorithm

Next we introduce our new algorithm to check Clarke stationarity of a candidate solution. This verification process is designed for a DC function $f = f_1 - f_2$ and it requires that the assumption **A1** holds. In what follows, the set U_k approximates $\partial_\varepsilon^G f(\boldsymbol{x})$, which on the other hand is an approximation of $\partial f(\boldsymbol{x})$, and we have $U_k \subset \partial_\varepsilon^G f(\boldsymbol{x})$ for all $k \geq 1$. In addition, the smaller the parameter $\varepsilon > 0$ is, the more accurate approximation of the set $\partial f(\boldsymbol{x})$ is used. Let $S_1 = \{\boldsymbol{d} \in \mathbb{R}^n \mid \|\boldsymbol{d}\| = 1\}$ be a unit sphere in $\mathbb{R}^n$.

Algorithm 1. Guaranteeing Clarke stationarity

Data: The point $\boldsymbol{x} \in \mathbb{R}^n$ under consideration, the descent parameter $m_1 \in (0, 1)$, the stopping tolerance $\delta \in (0, 1)$ and the proximity measure $\varepsilon > 0$.

Step 0. *(Initialization)* Select a direction $\boldsymbol{d}_1 \in S_1$. Set $\tilde{\boldsymbol{x}} = \boldsymbol{x}$, $U_0 = \emptyset$ and $k = 1$.

Step 1. *(New subgradient)* Find $\bar{\boldsymbol{d}}_k(\alpha) \in T_{DC}$ using $\boldsymbol{d}_k$. Compute subgradients $\boldsymbol{\xi}_{1,k} \in \partial f_1(\tilde{\boldsymbol{x}})$ and $\boldsymbol{\xi}_{2,k} \in \partial f_2(\tilde{\boldsymbol{x}})$ such that $\boldsymbol{\xi}_{1,k} \in G_1(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}_k(\alpha))$, $\boldsymbol{\xi}_{2,k} \in G_2(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}_k(\alpha))$. Set $\boldsymbol{\xi}_k = \boldsymbol{\xi}_{1,k} - \boldsymbol{\xi}_{2,k}$ and $U_k = \operatorname{conv}\{U_{k-1} \bigcup \{\boldsymbol{\xi}_k\}\}$.

Step 2. *(Clarke stationarity)* Find $\bar{\boldsymbol{u}}_k$ as the solution to the problem

$$\min_{\boldsymbol{u} \in U_k} \quad \frac{1}{2}\|\boldsymbol{u}\|^2. \tag{13}$$

If

$$\|\bar{\boldsymbol{u}}_k\| \leq \delta \tag{14}$$

then EXIT with $\boldsymbol{x}^* = \boldsymbol{x}^+$ (Clarke stationarity achieved).

Step 3. *(Search direction)* Compute $\boldsymbol{d}_{k+1} = -\bar{\boldsymbol{u}}_k / \|\bar{\boldsymbol{u}}_k\|$. If

$$f'(\boldsymbol{x}; \boldsymbol{d}_{k+1}) \leq -m_1 \|\bar{\boldsymbol{u}}_k\| \tag{15}$$

then go to Step 4. Otherwise set $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ and $k = k + 1$ and go to Step 1.

Step 4. *(Step-length)* Calculate the step-length $\beta^* \geq 0$ from the formula

$$\beta^* = \arg\max \{\beta \geq 0 \mid f(\boldsymbol{x} + \beta \boldsymbol{d}_{k+1}) - f(\boldsymbol{x}) < 0\}. \tag{16}$$

If $\beta^* \geq \varepsilon$ then set $\boldsymbol{x}^+ = \boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$ and EXIT from the algorithm (a better iteration point achieved). Otherwise set $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$ and $k = k + 1$ and go to Step 1.

REMARK 4.8. The assumption **A1** requiring that the subdifferentials of DC components f_1 and f_2 are polytopes is not really restrictive, since in practical applications this property is nearly always satisfied.

The next theorem shows the finite convergence of our new stopping procedure.

Theorem 4.9. *Let the assumption **A1** be valid. Algorithm 1 is terminated after at most*

$$N_{max} = \left\lceil \frac{4}{(1 - m_1)^2} \left(\frac{L}{\delta} \right)^4 \right\rceil$$

iterations where $\lceil \cdot \rceil$ is a ceiling of a number and $L > 0$ is the Lipschitz constant of f at a point $\boldsymbol{x} \in \mathbb{R}^n$.

Proof. Algorithm 1 is stopped if either the stopping condition (14) is satisfied or a new iteration point is found in Step 4. We prove that one of these alternatives will be fulfilled after a finite number of steps. First we show that, if none of these stopping options is satisfied during the iteration k, then the new subgradient $\boldsymbol{\xi}_{k+1}$ computed in Step 1 does not belong to the set $U_k \subset \partial_\varepsilon^G f(\boldsymbol{x})$. In fact, since $\bar{\boldsymbol{u}}_k$ is the solution to the quadratic programming problem (13), it follows from the necessary and sufficient condition for a minimum that

$$\bar{\boldsymbol{u}}_k^T \boldsymbol{u} \geq \|\bar{\boldsymbol{u}}_k\|^2 \quad \forall \boldsymbol{u} \in U_k.$$

This implies that

$$\boldsymbol{u}^T \boldsymbol{d}_{k+1} \leq -\|\bar{\boldsymbol{u}}_k\| \quad \forall \boldsymbol{u} \in U_k. \tag{17}$$

In addition, we have two options for the next $\tilde{\boldsymbol{x}}$, namely $\boldsymbol{x}$ or $\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$. If $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ this means that the condition (15) is not satisfied and we obtain

$$f'(\tilde{\boldsymbol{x}}; \boldsymbol{d}_{k+1}) = f'(\boldsymbol{x}; \boldsymbol{d}_{k+1}) > -m_1 \|\bar{\boldsymbol{u}}_k\|. \tag{18}$$

In the latter case, $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$ for $\beta^* < \varepsilon$ and, therefore, a subgradient calculated at $\tilde{\boldsymbol{x}}$ belongs to the set $\partial_\varepsilon^G f(\boldsymbol{x})$. Moreover, the step-length determination rule (16) together with the definition of the directional derivative guarantees that

$$f'(\tilde{\boldsymbol{x}}; \boldsymbol{d}_{k+1}) = f'(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}; \boldsymbol{d}_{k+1}) \geq 0 > -m_1 \delta > -m_1 \|\bar{\boldsymbol{u}}_k\|, \qquad (19)$$

where the last inequality is obtained from the fact that $\|\bar{\boldsymbol{u}}_k\| > \delta$. Moreover, the properties (iii) and (iv) of Theorem 4.7 yield that $\boldsymbol{\xi}_{1,k+1} - \boldsymbol{\xi}_{2,k+1} \in \partial f(\tilde{\boldsymbol{x}})$ and $f'(\tilde{\boldsymbol{x}}; \boldsymbol{d}_{k+1}) = \boldsymbol{\xi}_{k+1}^T \boldsymbol{d}_{k+1} = (\boldsymbol{\xi}_{1,k+1} - \boldsymbol{\xi}_{2,k+1})^T \boldsymbol{d}_{k+1}$ for $\boldsymbol{\xi}_{1,k+1} \in G_1(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}_{k+1}(\alpha))$ and $\boldsymbol{\xi}_{2,k+1} \in G_2(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}_{k+1}(\alpha))$. This together with the inequalities (18) and (19) implies that

$$\boldsymbol{\xi}_{k+1}^T \boldsymbol{d}_{k+1} > -m_1 \|\bar{\boldsymbol{u}}_k\|. \qquad (20)$$

Since $m_1 \in (0, 1)$, the inequalities (17) and (20) yield that $\boldsymbol{\xi}_{k+1} \notin U_k$. This means that, if the algorithm does not stop during one iteration, then the new subgradient allows us to significantly improve the approximation of the set $\partial_\varepsilon^G f(\boldsymbol{x})$.

In order to show that Algorithm 1 is finite convergent, it is sufficient to prove that the condition (14) will be satisfied after a finite number of iterations if a new iteration point is never found. It is obvious that $t\boldsymbol{\xi}_{k+1} + (1-t)\bar{\boldsymbol{u}}_k \in U_{k+1}$ for any $t \in (0, 1)$ and, thus, we get

$$\begin{aligned} \|\bar{\boldsymbol{u}}_{k+1}\|^2 &\leq \|t\boldsymbol{\xi}_{k+1} + (1-t)\bar{\boldsymbol{u}}_k\|^2 \\ &= \|\bar{\boldsymbol{u}}_k + t(\boldsymbol{\xi}_{k+1} - \bar{\boldsymbol{u}}_k)\|^2 \\ &= \|\bar{\boldsymbol{u}}_k\|^2 + 2t\bar{\boldsymbol{u}}_k^T(\boldsymbol{\xi}_{k+1} - \bar{\boldsymbol{u}}_k) + t^2\|\boldsymbol{\xi}_{k+1} - \bar{\boldsymbol{u}}_k\|^2. \end{aligned}$$

Since a DC function f is locally Lipschitz continuous at any point, the Goldstein subdifferential $\partial_\varepsilon^G f(\boldsymbol{x})$ is bounded at $\boldsymbol{x} \in \mathbb{R}^n$ with a Lipschitz constant [3]. Let $L > 0$ be the Lipschitz constant of f at $\boldsymbol{x}$. Then $\|\boldsymbol{\xi}\| \leq L$ for all $\boldsymbol{\xi} \in \partial_\varepsilon^G f(\boldsymbol{x})$ implying that

$$\|\boldsymbol{\xi}_{k+1} - \bar{\boldsymbol{u}}_k\| \leq 2L.$$

Moreover, from (20) we obtain $\boldsymbol{\xi}_{k+1}^T \bar{\boldsymbol{u}}_k \leq m_1 \|\bar{\boldsymbol{u}}_k\|^2$, since $\boldsymbol{d}_{k+1} = -\bar{\boldsymbol{u}}_k / \|\bar{\boldsymbol{u}}_k\|$ and, therefore,

$$\begin{aligned} \|\bar{\boldsymbol{u}}_{k+1}\|^2 &\leq \|\bar{\boldsymbol{u}}_k\|^2 + 2t\bar{\boldsymbol{u}}_k^T(\boldsymbol{\xi}_{k+1} - \bar{\boldsymbol{u}}_k) + 4t^2L^2 \\ &\leq \|\bar{\boldsymbol{u}}_k\|^2 - 2t(1-m_1)\|\bar{\boldsymbol{u}}_k\|^2 + 4t^2L^2. \end{aligned}$$

By selecting

$$t = \frac{(1-m_1)\|\bar{\boldsymbol{u}}_k\|^2}{4L^2}$$

it is obvious that $t \in (0, 1)$ and, thus, we have

$$\|\bar{\boldsymbol{u}}_{k+1}\|^2 \leq \|\bar{\boldsymbol{u}}_k\|^2 - \frac{(1-m_1)^2\|\bar{\boldsymbol{u}}_k\|^4}{4L^2}.$$

If the stopping criterion (14) is never met, then $\|\bar{\boldsymbol{u}}_k\| > \delta$ for any $k > 0$ and we get

$$\|\bar{\boldsymbol{u}}_{k+1}\|^2 \leq \|\bar{\boldsymbol{u}}_k\|^2 - \frac{(1-m_1)^2\delta^4}{4L^2}.$$

Writing this inequality for all $k > 0$ and summing up them we have

$$\|\bar{\boldsymbol{u}}_{k+1}\|^2 \leq \|\bar{\boldsymbol{u}}_1\|^2 - k\frac{(1-m_1)^2\delta^4}{4L^2}. \tag{21}$$

This yields $\|\bar{\boldsymbol{u}}_k\| \to -\infty$ which is a contradiction. Therefore, the stopping condition (14) will be satisfied after a finite number of iterations. Note that $\|\bar{\boldsymbol{u}}_1\| \leq L$ and, since

$$L^2 - k\frac{(1-m_1)^2\delta^4}{4L^2} \geq 0$$

according to (21), we have

$$k \leq \frac{4L^4}{(1-m_1)^2\delta^4}.$$

Then the maximum number N_{max} of iterations to reach the stopping condition (14) can be given by the following number

$$N_{max} = \left\lceil \frac{4}{(1-m_1)^2}\left(\frac{L}{\delta}\right)^4 \right\rceil.$$

This completes the proof. □

5 Double bundle method for DC functions

In this section, we will describe a new proximal double bundle method DBDC for solving unconstrained DC minimization problems. The DBDC method combines the ideas of the proximal bundle method PBDC [17] to the new stopping procedure Algorithm 1 designed to find Clarke stationary points. The PBDC method, in turn, utilizes the DC decomposition of the objective in the model construction and, due to this, the cutting plane model used describes quite well the actual behaviour of f. In addition, the PBDC method has fast convergence speed, but since only approximate criticality of the solutions is guaranteed, the stopping condition used can lead to "arbitrary" points where the Clarke stationarity of the original objective is not satisfied. Therefore, the aim of our hybridization is to utilize the PBDC method to obtain a promising candidate solution and after that use the new stopping procedure to guarantee Clarke stationarity of the solution obtained. In other words, our goal is to provide a more reliable method for nonsmooth DC problems, which preserves the good features of the PBDC method but at the same time improves the quality of the solutions obtained. The usage of Clarke stationarity as a stopping condition also enables us to escape from "arbitrary" solution points encountered when criticality is used.

5.1 Model for DC functions

We start with presenting the cutting plane model for nonsmooth DC functions, which is used to determine a search direction in the method DBDC. Since the main idea in the model construction is to utilize separately information about both DC components, we assume that at each point $\boldsymbol{x} \in \mathbb{R}^n$ we can evaluate the values of DC components $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ as well as arbitrary subgradients $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{x})$ and $\boldsymbol{\xi}_2 \in \partial f_2(\boldsymbol{x})$.

In order to take into account both the convex and the concave behavior of the objective f, we approximate the whole subdifferentials of both DC components. Therefore, we collect subgradients of those components into a bundle. This means that for each DC component we have its own bundle containing subgradient information gathered from the previous iterations. Thus, we maintain two completely separate bundles and the bundle for the DC component f_i at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$ is denoted by

$$\mathcal{B}_i^k = \left\{ (\boldsymbol{y}_j,\, f_i(\boldsymbol{y}_j),\, \boldsymbol{\xi}_{i,j}) \,|\, j \in J_i^k \right\} \quad \text{for } i = 1, 2,$$

where J_i^k is a nonempty set of indices and $\boldsymbol{\xi}_{i,j} \in \partial f_i(\boldsymbol{y}_j)$ is a subgradient calculated at an auxiliary point $\boldsymbol{y}_j \in \mathbb{R}^n$. Note that the index sets J_1^k and J_2^k need not to be similar and only the current iteration point $\boldsymbol{x}_k$ is always assumed to be included in both bundles $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$ with a suitable index.

Utilizing the convexity of the DC component we can easily form a convex piecewise linear model to approximate it at the iteration point $\boldsymbol{x}_k$. This model is the classical *cutting plane model* used in convex bundle methods (see, e.g., [19, 24, 26, 33]) and for the DC component f_i, $i = 1, 2$, it is constructed by

$$\hat{f}_i^k(\boldsymbol{x}) = \max_{j \in J_i^k} \left\{ f_i(\boldsymbol{x}_k) + (\boldsymbol{\xi}_{i,j})^T(\boldsymbol{x} - \boldsymbol{x}_k) - \alpha_{i,j}^k \right\}$$

with the *linearization error*

$$\alpha_{i,j}^k = f_i(\boldsymbol{x}_k) - f_i(\boldsymbol{y}_j) - (\boldsymbol{\xi}_{i,j})^T(\boldsymbol{x}_k - \boldsymbol{y}_j) \quad \text{for all } j \in J_i^k.$$

A nice feature of this model is that at every point the first order expansion supports from below the epigraph of a convex function and linearization errors are always nonnegative, that is, $\alpha_{i,j}^k \geq 0$.

The approximation for the original objective f is now obtained when we combine the previous convex cutting plane models of the DC components. Thus, the piecewise linear *nonconvex cutting plane model* of f is defined by

$$\hat{f}^k(\boldsymbol{x}) = \hat{f}_1^k(\boldsymbol{x}) - \hat{f}_2^k(\boldsymbol{x}).$$

This cutting plane model can be rewritten as follows

$$\hat{f}^k(\boldsymbol{x}_k + \boldsymbol{d}) = f(\boldsymbol{x}_k) + \Delta_1^k(\boldsymbol{d}) + \Delta_2^k(\boldsymbol{d}),$$

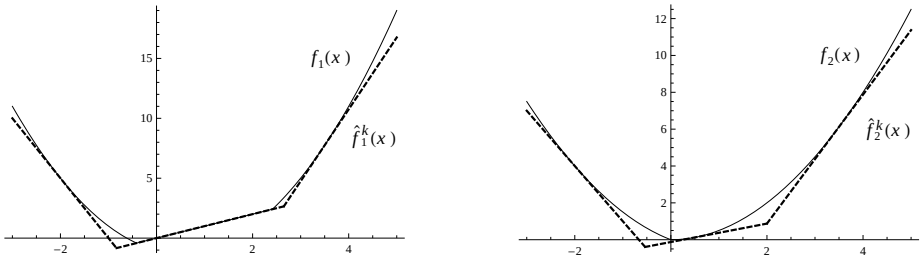

Figure 3: The convex cutting plane models $\hat{f}_1^k(x)$ and $\hat{f}_2^k(x)$ of the DC components $f_1(x)$ and $f_2(x)$

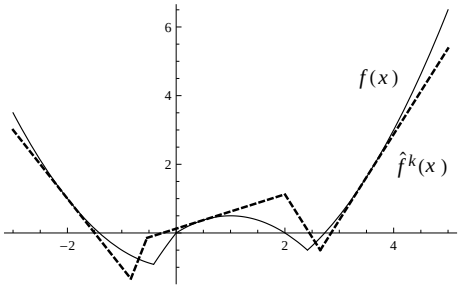

Figure 4: The nonconvex cutting plane model $\hat{f}^k(x)$ of the objective $f(x)$

when we denote by $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{x}_k$ the search direction at the current iteration point $\boldsymbol{x}_k$ and by

$$\Delta_1^k(\boldsymbol{d}) = \max_{j \in J_1^k} \left\{ (\boldsymbol{\xi}_{1,j})^T \boldsymbol{d} - \alpha_{1,j}^k \right\} \text{ and } \Delta_2^k(\boldsymbol{d}) = \min_{j \in J_2^k} \left\{ -(\boldsymbol{\xi}_{2,j})^T \boldsymbol{d} + \alpha_{2,j}^k \right\}$$

the piecewise affine functions associated with the DC components f_1 and f_2. Note that this cutting plane model is nonconvex and takes into account both the convex and the concave behavior of the objective f. Thus, we avoid the somewhat arbitrary downward shifting of first order expansions and we need not to use the so-called *subgradient locality measures* [18] commonly used in nonconvex bundle methods.

One illustration of the cutting plane model for a function $f = f_1 - f_2$ with a DC component selection $f_1(x) = \max\{x^2 - x - 1, x\}$ and $f_2(x) = 0.5x^2 + \max\{0, -x\}$ is presented in Figures 3 and 4. Linearizations of both DC components are constructed only at three points -2, 0.5 and 3.5.

5.2 Direction finding

The DBDC method uses the above presented nonconvex cutting plane model to compute the search direction. However, this model cannot be directly applied, since we cannot always guarantee the existence of the search direction for our piecewise linear model. Thus, we need to add a quadratic stabilizing term into our approximation. The search direction $\boldsymbol{d}_t^k$ is obtained by solving

globally the nonconvex DC minimization problem

$$\begin{cases} \text{minimize} & P^k(\boldsymbol{d}) = \Delta_1^k(\boldsymbol{d}) + \Delta_2^k(\boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n, \end{cases} \tag{22}$$

where $t > 0$ is a proximity parameter used in most bundle methods. Another purpose of the quadratic term is to keep our approximation local enough [24], since usually the farther away we are from the current iteration point the more unreliable the cutting plane model becomes.

Now the term $\Delta_1^k(\boldsymbol{d}) + \Delta_2^k(\boldsymbol{d})$ in the problem (22) can be seen as a predicted descent for the actual decrease in the objective function value $f(\boldsymbol{x}_k + \boldsymbol{d}) - f(\boldsymbol{x}_k)$. The following Lemma 5.1 supports this interpretation, since it guarantees that the approximation $\Delta_1^k(\boldsymbol{d}_t^k) + \Delta_2^k(\boldsymbol{d}_t^k)$ is always nonpositive and, thus, gives an estimate for a descent. Similarly the separate values $\Delta_1^k(\boldsymbol{d})$ and $\Delta_2^k(\boldsymbol{d})$ approximate the changes in the values of the DC components f_1 and $-f_2$, respectively.

Lemma 5.1. *The following properties hold:*

(*i*) $\Delta_1^k(\boldsymbol{d}) \leq f_1(\boldsymbol{x}_k + \boldsymbol{d}) - f_1(\boldsymbol{x}_k)$;

(*ii*) $\Delta_2^k(\boldsymbol{d}) \geq -f_2(\boldsymbol{x}_k + \boldsymbol{d}) - (-f_2(\boldsymbol{x}_k))$;

(*iii*) *For any* $t > 0$, *we have* $\Delta_1^k(\boldsymbol{d}_t^k) + \Delta_2^k(\boldsymbol{d}_t^k) \leq -\dfrac{1}{2t}\|\boldsymbol{d}_t^k\|^2 \leq 0$.

Proof. The properties (*i*) and (*ii*) follow directly from the features of the convex cutting plane model. For the case (*iii*) see [17]. □

We can also establish a bound for the norm $\|\boldsymbol{d}_t^k\|$ as is shown in the next lemma.

Lemma 5.2. *For any proximity parameter* $t > 0$, *it holds that*

$$\|\boldsymbol{d}_t^k\| \leq 2t\left(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|\right)$$

where $\boldsymbol{\xi}_1(\boldsymbol{x}_k) \in \partial f_1(\boldsymbol{x}_k)$ *and* $\|\boldsymbol{\xi}_{2,\max}\| = \max_{j \in J_2^k}\{\|\boldsymbol{\xi}_{2,j}\|\}$.

Proof. From the definition of $\Delta_2^k(\boldsymbol{d})$ we obtain that for all $\boldsymbol{d} \in \mathbb{R}^n$

$$\Delta_2^k(\boldsymbol{d}) = \min_{j \in J_2^k}\left\{-(\boldsymbol{\xi}_{2,j})^T\boldsymbol{d} + \alpha_{2,j}^k\right\} \geq \min_{j \in J_2^k}\left\{-(\boldsymbol{\xi}_{2,j})^T\boldsymbol{d}\right\} \geq -\|\boldsymbol{\xi}_{2,\max}\|\|\boldsymbol{d}\|,$$

since $\alpha_{2,j}^k \geq 0$ for any $j \in J_2^k$. On the other hand, the definition of $\Delta_1^k(\boldsymbol{d})$ yields

$$\Delta_1^k(\boldsymbol{d}) \geq (\boldsymbol{\xi}_{1,j})^T\boldsymbol{d} - \alpha_{1,j}^k \quad \text{for all } j \in J_1^k$$

and, when we combine this inequality with the previous one, we get

$$\Delta_1^k(\boldsymbol{d}) + \Delta_2^k(\boldsymbol{d}) \geq (\boldsymbol{\xi}_{1,j})^T\boldsymbol{d} - \alpha_{1,j}^k - \|\boldsymbol{\xi}_{2,\max}\|\|\boldsymbol{d}\| \tag{23}$$

for all $\boldsymbol{d} \in \mathbb{R}^n$ and $j \in J_1^k$. In addition, the element $(\boldsymbol{x}_k, f_1(\boldsymbol{x}_k), \boldsymbol{\xi}_1(\boldsymbol{x}_k))$, where $\boldsymbol{\xi}_1(\boldsymbol{x}_k) \in \partial f_1(\boldsymbol{x}_k)$, belongs to the bundle $\mathcal{B}_1^k$ with some index $\bar{j}$ and the corresponding linearization error is zero. Therefore, the inequality (23) holds for $\bar{j} \in J_1^k$ and for all $\boldsymbol{d} \in \mathbb{R}^n$

$$\begin{aligned}\Delta_1^k(\boldsymbol{d}) + \Delta_2^k(\boldsymbol{d}) &\geq \left(\boldsymbol{\xi}_1(\boldsymbol{x}_k)\right)^T \boldsymbol{d} - \|\boldsymbol{\xi}_{2,\max}\|\|\boldsymbol{d}\| \\ &\geq -\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\|\|\boldsymbol{d}\| - \|\boldsymbol{\xi}_{2,\max}\|\|\boldsymbol{d}\| \\ &= -\big(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|\big)\|\boldsymbol{d}\|.\end{aligned}$$

Finally, taking into account the property (iii) of Lemma 5.1 we obtain the inequality

$$-\frac{1}{2t}\|\boldsymbol{d}_t^k\|^2 \geq \Delta_1^k(\boldsymbol{d}_t^k) + \Delta_2^k(\boldsymbol{d}_t^k) \geq -\big(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|\big)\|\boldsymbol{d}_t^k\|,$$

which gives the desired bound for the solution $\boldsymbol{d}_t^k$ of the problem (22). □

The challenge in the problem (22) is to find the global solution, since even though the problem is quadratic, it is still a nonconvex nonsmooth DC minimization problem with DC components $\Delta_1^k(\boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2$ and $-\Delta_2^k(\boldsymbol{d})$. However, since in the objective function P^k the DC component $-\Delta_2^k(\boldsymbol{d})$ is polyhedral convex, the global solution can be easily obtained by utilizing a specific approach [20, 21, 31]. The main idea in the approach is based on the observation that, when the objective function P^k is reformulated as

$$P^k(\boldsymbol{d}) = \min_{i \in J_2^k} \left\{ P_i^k(\boldsymbol{d}) = \Delta_1^k(\boldsymbol{d}) - (\boldsymbol{\xi}_{2,i})^T \boldsymbol{d} + \alpha_{2,i}^k + \frac{1}{2t}\|\boldsymbol{d}\|^2 \right\},$$

then the problem (22) can be rewritten in the form

$$\min_{\boldsymbol{d} \in \mathbb{R}^n} \min_{i \in J_2^k} \left\{ P_i^k(\boldsymbol{d}) \right\} = \min_{i \in J_2^k} \min_{\boldsymbol{d} \in \mathbb{R}^n} \left\{ P_i^k(\boldsymbol{d}) \right\}.$$

Thus, we are allowed to change the order of the minimizations and solve first separately for each $i \in J_2^k$ the convex nonsmooth subproblem

$$\begin{cases} \text{minimize} & P_i^k(\boldsymbol{d}) = \Delta_1^k(\boldsymbol{d}) - (\boldsymbol{\xi}_{2,i})^T \boldsymbol{d} + \alpha_{2,i}^k + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n, \end{cases} \tag{24}$$

whose solution is denoted by $\boldsymbol{d}_t^k(i)$. After we have solved all $|J_2^k|$ convex subproblems the global solution $\boldsymbol{d}_t^k$ of the original nonconvex problem (22) is obtained by

$$\boldsymbol{d}_t^k = \boldsymbol{d}_t^k(i^*) \quad \text{where } i^* = \arg\min_{i \in J_2^k} \left\{ P_i^k(\boldsymbol{d}_t^k(i)) \right\}.$$

This means that we select the best solution from all the subproblem minimizers. Moreover, for each $i \in J_2^k$ the subproblem (24) can be reformulated in a smooth form

$$\begin{cases} \text{minimize} & v + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{subject to} & (\boldsymbol{\xi}_{1,j} - \boldsymbol{\xi}_{2,i})^T \boldsymbol{d} - (\alpha_{1,j}^k - \alpha_{2,i}^k) \leq v \quad \text{for all } j \in J_1^k \\ & v \in \mathbb{R},\ \boldsymbol{d} \in \mathbb{R}^n \end{cases} \tag{25}$$

and this way nonsmoothness does not cause any difficulties. Alternatively, instead of the problem (25), it is also possible to solve its quadratic dual problem

$$\begin{cases} \text{minimize} & \frac{1}{2}t\|\sum_{j\in J_1^k}\lambda_j\boldsymbol{\xi}_{1,j}-\boldsymbol{\xi}_{2,i}\|^2+\sum_{j\in J_1^k}\lambda_j\alpha_{1,j}^k-\alpha_{2,i}^k \\ \text{subject to} & \sum_{j\in J_1^k}\lambda_j=1 \\ & \lambda_j\geq 0 \quad \text{for all } \ j\in J_1^k, \end{cases} \tag{26}$$

which is typically easier to solve than the primal problem. The next theorem establishes the relationship between the optimal primal solution $(v_t^k(i),\, \boldsymbol{d}_t^k(i))$ and the optimal dual solutions $\lambda_{t,j}^k(i)$ for $j\in J_1^k$.

Theorem 5.3. *For each $i\in J_2^k$, the problems* (25) *and* (26) *are equivalent, and they have unique solutions $(v_t^k(i),\, \boldsymbol{d}_t^k(i))$ and $\lambda_{t,j}^k(i)$ for $j\in J_1^k$, respectively, such that*

$$\boldsymbol{d}_t^k(i)=-t\left(\sum_{j\in J_1^k}\lambda_{t,j}^k(i)\boldsymbol{\xi}_{1,j}-\boldsymbol{\xi}_{2,i}\right)$$

$$v_t^k(i)=-\frac{1}{t}\|\boldsymbol{d}_t^k(i)\|^2-\sum_{j\in J_1^k}\lambda_{t,j}^k(i)\alpha_{1,j}^k+\alpha_{2,i}^k.$$

Proof. See [26], pp. 115–117. □

5.3 Algorithm

Next we describe our new proximal double bundle algorithm DBDC for unconstrained DC minimization. This method combines the new stopping procedure presented in the previous section to the proximal bundle method PBDC [17].

To perform properly the method DBDC requires that a starting point $\boldsymbol{x}_0\in\mathbb{R}^n$ satisfies the following assumption:

A2 the level set $\mathcal{F}_0=\{\boldsymbol{x}\in\mathbb{R}^n\,|\,f(\boldsymbol{x})\leq f(\boldsymbol{x}_0)\}$ is compact.

This assumption is often made in bundle methods and it is not a restrictive one if the problem (1) is well-posed.

To make the presentation more clear we have divided our algorithm into smaller parts: the first part illustrates mainly the outline of the overall DBDC algorithm while a more significant role is on the algorithm presenting the 'main iteration', which consists of a sequence steps where the current iteration point remains unchanged. However, whenever we exit from the 'main iteration' algorithm we have either confirmed Clarke stationarity or found a new iteration point decreasing the value of the objective function. To guarantee Clarke stationarity the 'main iteration' utilizes Algorithm 1 whenever a promising candidate solution is found. If in this verification process the current iteration point $\boldsymbol{x}_k$ satisfies the required stopping conditions, then the overall bundle method DBDC terminates with $\boldsymbol{x}_k$ as the final solution.

Algorithm 2. Double bundle method for DC functions (DBDC)

Data: Choose the stopping tolerance $\delta \in (0,1)$, the proximity measure $\varepsilon > 0$, the enlargement parameter $\varepsilon_1 > 0$, the decrease parameters $r \in (0,1)$ and $c \in (0,1)$, the increase parameter $R > 1$ and the descent parameters $m_1 \in (0,1)$ and $m_2 \in (0,1)$.

Step 0. *(Initialization)* Choose a starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$, set $\boldsymbol{y}_1 = \boldsymbol{x}_0$ and compute the DC component values $f_1(\boldsymbol{x}_0)$ and $f_2(\boldsymbol{x}_0)$. Initialize the iteration counter $k = 0$. Calculate subgradients $\boldsymbol{\xi}_{1,1} \in \partial f_1(\boldsymbol{y}_1)$ and $\boldsymbol{\xi}_{2,1} \in \partial f_2(\boldsymbol{y}_1)$ and set $\alpha_{1,1}^k = \alpha_{2,1}^k = 0$. Initialize the bundles by setting

$$\mathcal{B}_1^k = \left\{(\boldsymbol{\xi}_{1,1}, \alpha_{1,1}^k)\right\} \quad \text{and} \quad \mathcal{B}_2^k = \left\{(\boldsymbol{\xi}_{2,1}, \alpha_{2,1}^k)\right\}.$$

Step 1. *(Main iteration)* Execute 'main iteration' Algorithm 3 to find $\boldsymbol{x}_{k+1}$. If $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ then Clarke stationarity is achieved and STOP with $\boldsymbol{x}^* = \boldsymbol{x}_k$ as the final solution.

Step 2. *(Bundle update)* Compute the new DC component values and subgradients

$$f_i(\boldsymbol{x}_{k+1}) \quad \text{and} \quad \boldsymbol{\xi}_i(\boldsymbol{x}_{k+1}) \in \partial f_i(\boldsymbol{x}_{k+1}) \quad \text{for } i = 1, 2.$$

Select the bundles $\mathcal{B}_1^{k+1} \subseteq \mathcal{B}_1^k$ and $\mathcal{B}_2^{k+1} \subseteq \mathcal{B}_2^k$ for the next round and update the linearization errors using the formula

$$\alpha_{i,j}^{k+1} = \alpha_{i,j}^k + f_i(\boldsymbol{x}_{k+1}) - f_i(\boldsymbol{x}_k) - (\boldsymbol{\xi}_{i,j})^T(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \qquad (27)$$

for all $i = 1, 2$ and $j \in J_i^{k+1}$. Insert also the element

$$(\boldsymbol{\xi}_1(\boldsymbol{x}_{k+1}), 0) \text{ into } \mathcal{B}_1^{k+1} \quad \text{and} \quad (\boldsymbol{\xi}_2(\boldsymbol{x}_{k+1}), 0) \text{ into } \mathcal{B}_2^{k+1}.$$

Finally, update $k = k + 1$ and go back to Step 1.

REMARK 5.4. In the bundles $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$, it suffices to store only subgradients $\boldsymbol{\xi}_{i,j} \in \partial f_i(\boldsymbol{y}_j)$ together with linearization errors $\alpha_{i,j}^k$. This is due to the fact that the linearization errors can be updated by using the formula (27) and, therefore, the new values for the next iteration round are easily obtained whenever a new iteration point $\boldsymbol{x}_{k+1}$ is found. Thus, we need not to store the auxiliary points $\boldsymbol{y}_j$ or the function values $f_i(\boldsymbol{y}_j)$.

REMARK 5.5. At the beginning of Step 2, the bundles $\mathcal{B}_1^{k+1}$ and $\mathcal{B}_2^{k+1}$ for the next round can be freely chosen and, therefore, every element stored can also be deleted at this point. Regardless of this the bundles $\mathcal{B}_1^{k+1}$ and $\mathcal{B}_2^{k+1}$ always contain at least the element corresponding to the new iteration point $\boldsymbol{x}_{k+1}$ since it is inserted into both bundles at the end of Step 2. This guarantees that the bundles are never empty when we start the execution of a new 'main iteration'.

Next we present the 'main iteration' being the core of DBDC. This algorithm uses some local parameters, which are initialized each time a new 'main iteration' is started. To simplify notations we have omitted the superscript k except for $\boldsymbol{x}_k \in \mathbb{R}^n$, since the current iteration point $\boldsymbol{x}_k$ does not change during the execution. In addition, $\boldsymbol{\xi}_1(\boldsymbol{x}_k) \in \partial f_1(\boldsymbol{x}_k)$ and $\boldsymbol{\xi}_2(\boldsymbol{x}_k) \in \partial f_2(\boldsymbol{x}_k)$ are the subgradients calculated at $\boldsymbol{x}_k$.

Algorithm 3. Main iteration

Data: The stopping tolerance $\delta \in (0,1)$, the enlargement parameter $\varepsilon_1 > 0$, the decrease parameters $r \in (0,1)$ and $c \in (0,1)$, the increase parameter $R > 1$ and the descent parameter $m_2 \in (0,1)$.

Step 0. *(Criticality)* If $\|\boldsymbol{\xi}_1(\boldsymbol{x}_k) - \boldsymbol{\xi}_2(\boldsymbol{x}_k)\| < \delta$ then go to Step 3.

Step 1. *(Initialization)* Calculate the index $j^* = \arg\max_{j \in J_2} \{\|\boldsymbol{\xi}_{2,j}\|\}$, set $\boldsymbol{\xi}_{2,\max} = \boldsymbol{\xi}_{2,j^*}$ and initialize the parameters

$$t_{\min} = r \cdot \frac{\varepsilon_1}{2(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|)} \quad \text{and} \quad t_{\max} = R\, t_{\min}.$$

Choose the value $t \in [t_{\min}, t_{\max}]$.

Step 2. *(Search direction)* Solve the search direction problem

$$\min_{\boldsymbol{d} \in \mathbb{R}^n} \left\{ \Delta_1(\boldsymbol{d}) + \Delta_2(\boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \right\} \tag{28}$$

and using the solution $\boldsymbol{d}_t$ calculate the predicted changes

$$\Delta_1(\boldsymbol{d}_t) = \max_{j \in J_1} \left\{ (\boldsymbol{\xi}_{1,j})^T \boldsymbol{d}_t - \alpha_{1,j} \right\} \quad \text{and}$$
$$\Delta_2(\boldsymbol{d}_t) = \min_{j \in J_2} \left\{ -(\boldsymbol{\xi}_{2,j})^T \boldsymbol{d}_t + \alpha_{2,j} \right\}.$$

If $\|\boldsymbol{d}_t\| < \delta$ then go to Step 3 else go to Step 4.

Step 3. *(Clarke stationarity)* Execute Algorithm 1 for the point $\boldsymbol{x}_k$. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}^+$ and EXIT from the 'main iteration'.

Step 4. *(Descent test)* Set $\boldsymbol{y} = \boldsymbol{x}_k + \boldsymbol{d}_t$. If

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) \le m_2 \left(\Delta_1(\boldsymbol{d}_t) + \Delta_2(\boldsymbol{d}_t) \right) \tag{29}$$

then choose $\boldsymbol{x}_{k+1} = \boldsymbol{y}$ and EXIT from the 'main iteration'.

Step 5. *(Bundle update)* Compute $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{y})$, $\boldsymbol{\xi}_2 \in \partial f_2(\boldsymbol{y})$ and set

$$\alpha_1 = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}) + \boldsymbol{\xi}_1^T \boldsymbol{d}_t \quad \text{and} \quad \alpha_2 = f_2(\boldsymbol{x}_k) - f_2(\boldsymbol{y}) + \boldsymbol{\xi}_2^T \boldsymbol{d}_t.$$

(a) If $f(\boldsymbol{y}) - f(\boldsymbol{x}_0) > 0$ and $\|\boldsymbol{d}_t\| > \varepsilon_1$, then set $t = t - r(t - t_{\min})$ and go back to Step 2.

(b) Otherwise insert $(\boldsymbol{\xi}_1,\, \alpha_1)$ into $\mathcal{B}_1$ and, if $\Delta_2(\boldsymbol{d}_t) \geq 0$, then insert $(\boldsymbol{\xi}_2,\, \alpha_2)$ into $\mathcal{B}_2$. If

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) \geq -m_2 \left(\Delta_1(\boldsymbol{d}_t) + \Delta_2(\boldsymbol{d}_t)\right) \tag{30}$$

then set $t = t - c(t - t_{\min})$.

Step 6. *(Parameter update)* If $\|\boldsymbol{\xi}_2\| > \|\boldsymbol{\xi}_{2,\max}\|$ then update

$$\boldsymbol{\xi}_{2,\max} = \boldsymbol{\xi}_2 \quad \text{and} \quad t_{\min} = r \cdot \frac{\varepsilon_1}{2(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|)}.$$

Go back to Step 2.

The nonconvex DC minimization problem (22) is the search direction problem (28) used in Step 2 of the 'main iteration'. Due to this, Step 2 of the algorithm is the most time-consuming part, since during each iteration we need to solve $|J_2|$ convex subproblems. However, after solving all the subproblems the global solution of the original nonconvex problem can be easily obtained by choosing the best solution from the subproblem minimizers.

REMARK 5.6. The user can control the number of subproblems solved, since the size of the bundle $\mathcal{B}_2$ can be always limited with the maximum number of stored subgradients $J_{\max} \geq 1$. The only restriction is that the element $(\boldsymbol{\xi}(\boldsymbol{x}_k), 0)$ corresponding to the current iteration point $\boldsymbol{x}_k$ cannot be deleted or substituted during the execution of the 'main iteration'. Moreover, it is possible to omit the update requirement used in Step 5(b) for the bundle $\mathcal{B}_2$ and always include the new element into $\mathcal{B}_2$.

REMARK 5.7. In Step 5(b), the condition (30) is similar to (29), but now instead of the descent we test if the decrease in the objective function is significant. This way we can detect the cases where the model of the objective function is inconsistent and fails to describe the actual behaviour of f. In this case we decrease the proximity parameter t to get a more accurate model.

REMARK 5.8. The purpose of Step 5(a) of 'main iteration' Algorithm 3 is to guarantee that the points used to constitute the elements inserted into the bundles are on the set $\mathcal{F}_{\varepsilon_1} = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, d(\boldsymbol{x}, \mathcal{F}_0) \leq \varepsilon_1\}$, where $\varepsilon_1 > 0$ is selected and $d(\boldsymbol{x}, \mathcal{F}_0) = \inf\{\|\boldsymbol{x} - \boldsymbol{z}\| \,|\, \boldsymbol{z} \in \mathcal{F}_0\}$. Moreover, all the iteration points $\boldsymbol{x}_k$ are on the set $\mathcal{F}_{\varepsilon_1}$, since each new iteration point decreases the value of the objective. In addition, the DC components f_1 and f_2 are locally Lipschitz continuous and let $L_1 > 0$ and $L_2 > 0$ be the Lipschitz constants of f_1 and f_2 on the compact set $\mathcal{F}_{\varepsilon_1}$, respectively. This implies that

$$\|\boldsymbol{\xi}_1\| \leq L_1 \text{ for each } \boldsymbol{\xi}_1 \text{ on } \mathcal{B}_1 \text{ and } \|\boldsymbol{\xi}_2\| \leq L_2 \text{ for each } \boldsymbol{\xi}_2 \text{ on } \mathcal{B}_2\,. \tag{31}$$

From this we can also deduce that the parameters t and $t_{\min}$ are bounded away from zero, since

$$t \geq t_{\min} \geq \bar{t}_{\min} = r\varepsilon_1/(2L_1 + 2L_2) > 0.$$

In addition, the parameter $t_{\max}$ is bounded from above, since

$$\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\| \geq \|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_2(\boldsymbol{x}_k)\| \geq \|\boldsymbol{\xi}_1(\boldsymbol{x}_k) - \boldsymbol{\xi}_2(\boldsymbol{x}_k)\| \geq \delta$$

whenever Step 0 is not satisfied, and therefore

$$t_{\max} \leq \bar{t}_{\max} = Rr\varepsilon_1/2\delta < \infty.$$

5.4 Convergence

In this section, we prove the convergence of the method DBDC. We especially show that our method terminates after a finite number of steps and that the solution obtained is a Clarke stationary point. As we have already seen, a Clarke stationary point is always a critical point, but the opposite does not need to hold. Therefore, a significant feature of the new DBDC method is that we can guarantee a tighter and better optimality condition than in the most of the other methods designed for DC functions.

In Theorem 4.8, we have already proved the finite convergence of Algorithm 1 guaranteeing Clarke stationarity and this required the assumption **A1** to hold. Next we show that 'main iteration' Algorithm 3 stops after a finite number of iterations and only after that we are finally ready to present the convergence result for DBDC Algorithm 2. To prove the convergence we also need to require that the assumption **A2** is valid.

We begin by showing the following auxiliary lemma which is utilized to show the finite convergence. After that we are ready to prove the termination for 'main iteration' Algorithm 3. Both proofs follow the guidelines of [17].

Lemma 5.9. *If the condition* (29) *at Step 4 of Algorithm 3 is not satisfied, then*

$$\boldsymbol{\xi}_1^T \boldsymbol{d}_t - \alpha_1 > m_2\Delta_1(\boldsymbol{d}_t) + (m_2 - 1)\Delta_2(\boldsymbol{d}_t),$$

where $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{y})$ *is a subgradient calculated at the new auxiliary point* $\boldsymbol{y} = \boldsymbol{x}_k + \boldsymbol{d}_t$ *and* $\alpha_1 = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}) + \boldsymbol{\xi}_1^T \boldsymbol{d}_t$ *is the corresponding linearization error.*

Proof. If the descent condition (29) is not satisfied at Step 4, then

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) > m_2\Big(\Delta_1(\boldsymbol{d}_t) + \Delta_2(\boldsymbol{d}_t)\Big),$$

where $\boldsymbol{y} = \boldsymbol{x}_k + \boldsymbol{d}_t$ is the new auxiliary point calculated in the 'main iteration'. Rewriting f using the DC components we obtain

$$f_1(\boldsymbol{y}) - f_1(\boldsymbol{x}_k) > m_2\Big(\Delta_1(\boldsymbol{d}_t) + \Delta_2(\boldsymbol{d}_t)\Big) - \Big(f_2(\boldsymbol{x}_k) - f_2(\boldsymbol{y})\Big)$$

and together with the property (ii) of Lemma 5.1 this ensures that we always have

$$f_1(\boldsymbol{y}) - f_1(\boldsymbol{x}_k) > m_2\Delta_1(\boldsymbol{d}_t) + (m_2 - 1)\Delta_2(\boldsymbol{d}_t).$$

The result follows from this by noticing that

$$f_1(\boldsymbol{y}) - f_1(\boldsymbol{x}_k) = f_1(\boldsymbol{x}_k + \boldsymbol{d}_t) - f_1(\boldsymbol{x}_k) = \boldsymbol{\xi}_1^T \boldsymbol{d}_t - \alpha_1$$

when $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{y})$ and $\alpha_1 = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}) + \boldsymbol{\xi}_1^T \boldsymbol{d}_t$. □

Theorem 5.10. *Let the assumption **A2** be valid. For any $\delta \in (0, 1)$, Algorithm 3 cannot pass infinitely many times through the sequence of steps from 4 to 6.*

Proof. We start with supposing, contrary to our claim, that the sequence of steps from 4 to 6 is executed infinitely many times and index by $i \in \mathcal{I}$ all the quantities referred to the ith passage. This mean that we never execute Step 3, since this would provide us either a new iteration point or Clarke stationarity after a finite number of steps. Therefore, for each $i \in \mathcal{I}$ the condition $\|\boldsymbol{d}_t^{(i)}\| \geq \delta$ is satisfied.

First, we notice that Step 5(a) cannot occur infinitely many times. If this would be the case, then the proximity parameter t would be decreased infinitely many times and it would converge to $t_{\min}$, since the safeguard parameter $t_{\min}$ is both bounded and monotonically decreasing. Moreover, the parameter $t_{\min}$ is always selected such a way that it is smaller than the threshold $\varepsilon_1/2(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}^{(i)}\|)$ implying that after a finite number of iterations also the proximity parameter t falls below this threshold. However, when this happens Step 5(a) cannot be executed anymore, since $\|\boldsymbol{d}_t^{(i)}\| \leq \varepsilon_1$ according to Lemma 5.2. Therefore, there exists an index $\hat{\imath} \in \mathcal{I}$ after which Step 5(b) is always entered.

Second, we can guarantee that the sequence $\{\boldsymbol{d}_t^{(i)}\}_{i\in\mathcal{I}}$ is bounded in norm, when we combine Lemma 5.2, the property (31) and the parameter selection rule $t \in [t_{\min}, t_{\max}]$. Hence, there exists a convergent subsequence $\{\boldsymbol{d}_t^{(i)}\}_{i\in\mathcal{I}'\subseteq\mathcal{I}}$ converging to a limit $\hat{\boldsymbol{d}}$. From Remark 5.8 we also obtain that all the auxiliary points $\boldsymbol{y}_j$ and the iteration points $\boldsymbol{x}_k$ belong to the set $\mathcal{F}_{\varepsilon_1}$. In addition, the assumption **A2** implies that the set $\mathcal{F}_{\varepsilon_1}$ is compact and, thus, there exists a constant $K > 0$ such that

$$\|\boldsymbol{x}_k - \boldsymbol{y}_j\| \leq K \quad \text{for all points } \boldsymbol{y}_j \text{ on } \mathcal{B}_1.$$

Moreover, this information together with (31) yields

$$\begin{aligned} |\alpha_{1,j}| &= |f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}_j) - (\boldsymbol{\xi}_{1,j})^T(\boldsymbol{x}_k - \boldsymbol{y}_j)| \\ &\leq |f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}_j)| + \|\boldsymbol{\xi}_{1,j}\|\|(\boldsymbol{x}_k - \boldsymbol{y}_j)\| \\ &\leq L_1\|\boldsymbol{x}_k - \boldsymbol{y}_j\| + L_1 K \leq 2L_1 K \end{aligned}$$

for all points $\boldsymbol{y}_j$ on $\mathcal{B}_1$, since we always have $\|\boldsymbol{\xi}_{1,j}\| \leq L_1$. Similar results can be shown to point $\boldsymbol{y}_j$ on $\mathcal{B}_2$ and, thus, all the subgradients and linearization errors are bounded.

The boundness results imply also that the corresponding subsequences $\{\Delta_1(\boldsymbol{d}_t^{(i)})\}_{i\in\mathcal{I}'\subseteq\mathcal{I}}$ and $\{\Delta_2(\boldsymbol{d}_t^{(i)})\}_{i\in\mathcal{I}'\subseteq\mathcal{I}}$ are bounded. Therefore, they both admit a convergent subsequence for $i \in \mathcal{I}'' \subseteq \mathcal{I}'$ and the limits are denoted by $\hat{\Delta}_1$ and $\hat{\Delta}_2$, respectively. As a consequence of the property (iii) of Lemma 5.1, we then obtain

$$\Delta_1(\boldsymbol{d}_t^{(i)}) + \Delta_2(\boldsymbol{d}_t^{(i)}) \leq -\frac{1}{2t_i}\|\boldsymbol{d}_t^{(i)}\|^2 \leq -\frac{\delta^2}{2t_i} < 0 \quad \text{for all } i \in \mathcal{I},$$

since $\|\boldsymbol{d}_t^{(i)}\| \geq \delta$ and, thus,

$$\hat{\Delta}_1 + \hat{\Delta}_2 \leq -\frac{\delta^2}{2\hat{t}} < 0,$$

where $\hat{t} = \lim_{i\to\infty} t_i > 0$. Now Remark 5.8 guarantees that $\hat{t}$ is strictly positive, since the sequence $\{t_i\}$ is bounded from below with a positive threshold. Moreover, the sequence t_i is nonincreasing and, therefore, a strictly positive limit $\hat{t}$ exists.

To complete the proof let r and s be two successive indices in $\mathcal{I}''$ and

$$\alpha_{1,r} = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{x}_k + \boldsymbol{d}_t^{(r)}) + (\boldsymbol{\xi}_{1,r})^T\boldsymbol{d}_t^{(r)}$$

with $\boldsymbol{\xi}_{1,r} \in \partial f_1(\boldsymbol{x}_k + \boldsymbol{d}_t^{(r)})$. We have

$$(\boldsymbol{\xi}_{1,r})^T\boldsymbol{d}_t^{(r)} - \alpha_{1,r} > m_2\Delta_1(\boldsymbol{d}_t^{(r)}) + (m_2 - 1)\Delta_2(\boldsymbol{d}_t^{(r)}) \tag{32}$$

and

$$\Delta_1(\boldsymbol{d}_t^{(s)}) \geq (\boldsymbol{\xi}_{1,r})^T\boldsymbol{d}_t^{(s)} - \alpha_{1,r}\,, \tag{33}$$

where the first inequality follows from Lemma 5.9 and the latter one is an immediate consequence of the definition of $\Delta_1(\boldsymbol{d})$. Finally, combining (32) and (33) gives

$$\Delta_1(\boldsymbol{d}_t^{(s)}) - m_2\Delta_1(\boldsymbol{d}_t^{(r)}) + (1 - m_2)\Delta_2(\boldsymbol{d}_t^{(r)}) > (\boldsymbol{\xi}_{1,r})^T\Big(\boldsymbol{d}_t^{(s)} - \boldsymbol{d}_t^{(r)}\Big)$$

and passing to the limit yields

$$(1 - m_2)\Big(\hat{\Delta}_1 + \hat{\Delta}_2\Big) \geq 0.$$

This is a contradiction, since $m_2 \in (0, 1)$ and, thus, $\hat{\Delta}_1 + \hat{\Delta}_2 < 0$ cannot hold. □

Finally, we are ready to show the finite convergence for the overall bundle algorithm DBDC. The proof of this result reveals similar trends than Theorem 3.10 in [28]. In addition, DBDC is globally convergent if the assumption **A2** holds for any starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$. This means that the convergence result does not depend on $\boldsymbol{x}_0$ and the method always generates a Clarke stationary point as a final solution $\boldsymbol{x}^*$ regardless of the starting point used.

Theorem 5.11. *Let the assumptions **A1** and **A2** be valid. For any parameters $\delta \in (0,1)$ and $\varepsilon > 0$, the execution of Algorithm 2 stops after a finite number of 'main iterations' at a point $\boldsymbol{x}^*$ satisfying the approximate Clarke stationarity condition $\|\boldsymbol{\xi}^*\| \leq \delta$ with $\boldsymbol{\xi}^* \in \partial_\varepsilon^G f(\boldsymbol{x}^*)$.*

Proof. The termination of DBDC can happen only if the stopping condition (14) tested at Step 2 of Algorithm 1 is satisfied and this provides the approximate Clarke stationarity. We suppose that the 'main iteration' is entered infinitely many times and index by $k \in \mathcal{K}$ all the quantities obtained from the kth passage. First of all, Theorems 4.9 and 5.10 guarantee that in the 'main iteration' we always find a new iteration point $\boldsymbol{x}_{k+1}$ after a finite number of steps and this point is obtained either from Step 3 or from Step 4. Therefore, we obtain a sequence of iteration points $\{\boldsymbol{x}_k\}$, where

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \tau \boldsymbol{d}_{k-1} \quad \text{such that} \quad \tau \geq \min\{1, \varepsilon\} > 0, \tag{34}$$

belonging to the compact set $\mathcal{F}_0$. This implies that the sequence $\{\boldsymbol{x}_k\}$ has an accummulation point $\bar{\boldsymbol{x}}$ and together with the formulation (34) of $\boldsymbol{x}_k$ this yields that the sequence $\{\boldsymbol{d}_k\} \to 0$. From this we can deduce that for any $\delta \in (0,1)$ there exists an iteration index k' such that $\|\boldsymbol{d}_k\| < \delta$ for all $k \geq k'$.

Next let us look closer the iteration k'. First of all, we notice that a new iteration point cannot be obtained from Step 4 of Algorithm 3. Due to this, the only option is Step 3 of Algorithm 3. However, during the execution of Algorithm 1 the search direction $\boldsymbol{d}_{k'}$ always fulfils $\|\boldsymbol{d}_{k'}\| = 1 > \delta$ contradicting $\|\boldsymbol{d}_{k'}\| < \delta$.

□

6 Numerical results

To verify the practical efficiency of the new method DBDC we have applied it to some academic test problems with nonsmooth nonconvex DC objective functions. In order to compare the results, we have used two proximal bundle algorithms PBDC [17] and MPBNGC [26] for nonsmooth optimization. The algorithm PBDC is the predecessor of DBDC and it also utilizes the DC decomposition of the objective. MPBNGC, in its turn, is designed for a general function and does not exploit any specific structure of the objective. In [17], PBDC is also compared to DCA [22, 21], the truncated codifferential method [6] and two other bundle methods [10, 11] and since most of our test problems are from [17] we have not used those methods in our comparisons.

The algorithm DBDC requires an unbounded storage for the bundle $\mathcal{B}_1$, but this is an impossible requirement to fullfil in practice. Thus, the implementation of DBDC slightly differs from Algorithm 2, since we have used in the 'main iteration' a subgradient aggregation strategy from [17] into the bundle $\mathcal{B}_1$. This allows us to store some information from the previous iterations, even though the size of $\mathcal{B}_1$ is bounded.

As we have already seen, the execution of Algorithm 1 is continued if at Step 4 the stepsize $\beta^* < \varepsilon$. However, in practice, it might be advantageous to stop the whole algorithm DBDC at this point, since if the stepsize β^* is really small it yields the consecutive iteration points to be very close to each other. Therefore, in the implementation of DBDC, the execution of Algorithm 1 is stopped with $\boldsymbol{x}^* = \boldsymbol{x}$ as the final solution, if $\beta^* < \varepsilon$.

`DBDC`, utilizing the aggregation scheme, is implemented in double precision Fortran 95 and it uses the subroutine `PLQDF1` [23] to solve the quadratic problems (13) and (25). The code of `PBDC` is also implemented in double precision Fortran 95 and the implementation of `MPBNGC` is done with double precision Fortran 77. Both `PBDC` and `MPBNGC` use also the subroutine `PLQDF1` [23] to solve the quadratic direction finding problems. All the codes are compiled using `f95`, the Fortran 95 compiler, and tests are performed under Linux Ubuntu system.

The codes have been tested on a set of 16 academic nonconvex test problems. Problems 1–10 are from [17] whereas Problems 11–16 are introduced in Appendix at the end of the paper. The input parameters of `DBDC` have been chosen as follows: the stopping tolerance

$$\delta = \begin{cases} 10^{-5}, & \text{if } n \leq 200 \\ 10^{-4}, & \text{if } n > 200, \end{cases}$$

the proximity measure

$$\varepsilon = \begin{cases} 10^{-6}, & \text{if } n \leq 50 \\ 10^{-5}, & \text{if } n > 50, \end{cases}$$

the enlargement parameter $\varepsilon_1 = 0.00005$, the decrease parameters $c = 0.1$ and

$$r = \begin{cases} 0.75, & \text{if } n < 10 \\ \text{the first two desimals of } n/(n+5), & \text{if } 10 \leq n < 300 \\ 0.99, & \text{if } n \geq 300, \end{cases}$$

the increase parameter $R = 10^7$ and the descent parameters $m_1 = 0.01$ and $m_2 = 0.2$. The size of the bundle $\mathcal{B}_1$ is set to $\min\{n+5, 1000\}$ and the size of $\mathcal{B}_2$ is 3. The maximum size of the set U_k in Algorithm 1 is restricted to $2n$. In `PBDC`, we have used the default settings of the code [17]. Furthermore, in `MPBNGC` we have used mostly the default values of the parameters [25], but the maximum size of the bundle is selected to be $\min\{n+3, 1000\}$ and the final accuracy is set to 10^{-10} to get about the same accuracy in solutions.

Before presenting the comprehensive analysis of the numerical results we will consider a simple example showing the most fundamental difference between the methods DBDC and PBDC. We will see how the criticality condition used in PBDC can cause serious difficulties, which however can be avoided in DBDC, when the new stopping procedure is utilized.

Example 6.1. Lets consider the DC function presented in Example 3.2. If the starting point x_0 is selected from the set $A = \{x \in \mathbb{R} \,|\, x < -2 \text{ or } x > 1\}$, then the subgradients of DC components are $\xi_1(x_0) = 2x_0$ and $\xi_2(x_0) = x_0$. Therefore, the bundles are initialized by setting

$$\mathcal{B}_1^0 = \{(\xi_1(x_0), 0)\} \quad \text{and} \quad \mathcal{B}_2^0 = \{(\xi_2(x_0), 0)\}$$

in both methods DBDC and PBDC. In addition, both solvers use the same direction finding problem (22), which during the first iteration round is

$$\min_{d \in \mathbb{R}} \quad \left\{ P^0(d) = \xi_1(x_0)d - \xi_2(x_0)d + \frac{1}{2t}\|d\|^2 = x_0 d + \frac{1}{2t}\|d\|^2 \right\}.$$

The solution to this is $d_t = -tx_0$ and with the proximity parameter selection $t = 1$ we obtain a new auxiliary point $y = x_0 + d_t = x_0 - x_0 = 0$. Moreover, in both solvers we use the same descent test (29) and, if the descent parameter m_2 is selected from the interval $(0, 1/2)$, then the objective function decreases enough, that is,

$$f(y) - f(x_0) = -0.5x_0^2 \leq -m_2 x_0^2 = m_2(\Delta_1(d_t) + \Delta_2(d_t)).$$

Thus, we obtain a new iteration point $x_1 = 0$. However, when we continue the execution of the algorithms it is possible that the new subgradients of the DC components at x_1 are $\xi_1(x_1) = \xi_2(x_1) = 0$. In PBDC, this will lead to the fulfillment of the criticality condition and the algorithm is terminated with the final solution $x^* = 0$. However, as we have already seen, this solution is nothing interesting for f. In DBDC, the selection $\xi_1(x_1) = \xi_2(x_1) = 0$ leads to Algorithm 1, where we first calculate utilizing the DC components a subgradient ξ for f using either a direction $d_1 = 1$ or $d_1 = -1$. Regardless of this selection we obtain a subgradient $\xi = 1$ and a new search direction $d_2 = -1$. In addition, the condition (15) is now satisfied, since

$$f'(x_1; d_2) = f'(0; -1) = -1 \leq -m_1 = -m_1\|\xi\|$$

and $m_1 \in (0, 1)$. This proves that we have obtained a descent direction, which provides a better iteration point. Therefore, the solver DBDC bypasses the point $x_1 = 0$ and does not stop at the problematic critical point.

The results of our numerical experiments are presented in Tables 1 and 2 and we have used the following notations:

- *Prob.* is the number of the problem
- n is the number of variables
- n_f is the number of function evaluations for the objective function f

- $n_{\boldsymbol{\xi}}$ is the number of subgradient evaluations for the objective function f
- $n_{\boldsymbol{\xi}_i}$ is the number of subgradient evaluations for the DC component f_i
- *time* is the CPU time in seconds
- f is the obtained value of the objective function when the algorithm stops.

In MPBNGC, we have $n_f = n_{\boldsymbol{\xi}}$. Moreover, in DBDC we have given separately the function and subgradient evaluations used in Algorithm 1 guaranteeing Clarke stationarity and for this algorithm $n_{\boldsymbol{\xi}_1} = n_{\boldsymbol{\xi}_2}$. Otherwise, in the solvers DBDC and PBDC we have reported the subgradient evaluations separately for DC components. Therefore, to obtain somewhat comparable results with MPBNGC we have calculated combined values $n_{\boldsymbol{\xi}_1} + n_{\boldsymbol{\xi}_2}$ and used them in the comparison, even though $n_{\boldsymbol{\xi}_1} + n_{\boldsymbol{\xi}_2}$ overestimates the computational effort when compared to $n_{\boldsymbol{\xi}}$.

All the solvers tested are only local methods and, therefore, we are satisfied with any local minimizer. Nevertheless, from Tables 1 and 2 we first notice that we often find the global minimizer. DBDC is the most successful solver to solve the problems globally and it fails to find a global minimizer only in 4 cases out of 53. Moreover, those four cases seems to be quite difficult ones, since also the other solvers mostly find a local minimizer among them. In addition, both PBDC and MPBNGC are quite reliable to find global minimizers: PBDC provides a local solution only in 8 cases and MPBNGC in 9 cases out of 53. However, Problem 12 seems to be extremely difficult for PBDC whereas Problems 7–10 are challenging for MPBNGC.

The results in Tables 1 and 2 also show that DBDC uses the least evaluations in Problems 12 and 14 and the difference with the other solvers is significant. In Problems 2, 4–6, 11 and 16, the solvers DBDC and PBDC need quite the same amount of evaluations and they are also more efficient than MPBNGC. However, in Problems 4–6 DBDC often requires a little bit more computational effort due to a stronger stopping condition verification procedure, but this does not affect CPU times except for Problem 4 ($n = 250$ and $n = 500$). In the rest of the tests (Problems 1, 7–10, 13 and 15), MPBNGC uses the least evaluations, while the difference between DBDC and PBDC is often quite small but for Problems 13 and 15. It is also worth noting that in most of Problems 7–10 MPBNGC converged to a local minimizer making it hard to say if MPBNGC is really the most efficient solver in those cases.

In terms of CPU time none of the solvers stands out from the others, since for each solver we are able to detect both easy and hard problems. For example, in Problems 1–11 and 16 all the solvers are equally fast, if we leave out of the consideration Problem 4 ($n = 250$), where MPBNGC is faster than the other methods, and Problem 4 ($n = 500$), where PBDC is the fastest one. Moreover, the solver DBDC is fastest to provide a solution in Problem 14, whereas the CPU times of PBDC and MPBNGC have a completely different magnitude when we increase the dimension of this problem. However, MPBNGC beats the other solvers in Problems 12 ($n = 100$) and 15 ($n = 100$).

All in all, the numerical results confirm that DBDC is efficient to solve nonsmooth DC minimization problems. Compared to PBDC the solver DBDC uses sometimes more function and subgradient evaluations, but at the same

Table 1: Summary of numerical results with `DBDC`, `PBDC` and `MPBNGC`

		DBDC							PBDC					MPBNGC		
		Main it.			Clarke alg.											
Prob.	n	n_f	$n_{\boldsymbol{\xi}_1}$	$n_{\boldsymbol{\xi}_2}$	n_f	$n_{\boldsymbol{\xi}_1}, n_{\boldsymbol{\xi}_2}$	*time*	f	n_f	$n_{\boldsymbol{\xi}_1}$	$n_{\boldsymbol{\xi}_2}$	*time*	f	$n_f, n_{\boldsymbol{\xi}}$	*time*	f
1	2	21	16	15	2	3	0.00	2.000000036	22	17	16	0.00	2.000000020	17	0.00	2.000000000
2	2	17	11	11	1	2	0.00	$1.1036 \cdot 10^{-12}$	21	15	15	0.00	$1.1098 \cdot 10^{-12}$	39	0.00	$2.3093 \cdot 10^{-14}$
3	4	22	11	9	4	5	0.00	$2.6234 \cdot 10^{-12}$	25	15	11	0.00	$2.2689 \cdot 10^{-12}$	22	0.00	$1.8308 \cdot 10^{-13}$
4	2	6	3	3	1	2	0.00	$8.4377 \cdot 10^{-15}$	6	3	3	0.00	$8.4377 \cdot 10^{-15}$	7	0.00	$4.4409 \cdot 10^{-16}$
4	5	13	6	5	4	5	0.00	$1.7764 \cdot 10^{-15}$	13	6	5	0.00	0.000000000	30	0.00	$3.5527 \cdot 10^{-15}$
4	10	16	11	10	9	10	0.00	$1.4211 \cdot 10^{-14}$	16	11	9	0.00	$5.6843 \cdot 10^{-14}$	61	0.00	0.000000000
4	100	105	105	32	99	100	2.18	$1.8190 \cdot 10^{-12}$	102	102	30	1.15	$9.0949 \cdot 10^{-13}$	1489	1.89	$1.1731 \cdot 10^{-11}$
4	250	477	477	190	249	250	165.9	$-3.6380 \cdot 10^{-12}$	481	481	194	121.2	$-1.8190 \cdot 10^{-11}$	3619	45.2	$1.0411 \cdot 10^{-10}$
4	500	1492	1492	699	499	500	3952.9	$1.6007 \cdot 10^{-10}$	1443	1443	687	2749.3	$2.1827 \cdot 10^{-10}$	100000	9514.1	$-3.5422 \cdot 10^{-10}$
5	2	10	4	4	1	2	0.00	0.000000000	10	4	4	0.00	0.000000000	5	0.00	$8.8818 \cdot 10^{-16}$
5	10	20	13	10	7	8	0.00	$7.5859 \cdot 10^{-11}$	21	14	11	0.01	$3.5416 \cdot 10^{-13}$	131	0.01	$8.0853 \cdot 10^{-11}$
5	100	42	23	19	11	12	0.05	$2.8562 \cdot 10^{-10}$	47	28	23	0.23	$8.5659 \cdot 10^{-13}$	45	0.01	$1.0043 \cdot 10^{-10}$
5	500	26	19	16	10	11	0.31	$1.1641 \cdot 10^{-8}$	29	22	18	0.24	$2.1682 \cdot 10^{-10}$	52	0.15	$1.8440 \cdot 10^{-12}$
5	1500	23	18	14	10	11	1.06	$9.1006 \cdot 10^{-9}$	24	19	15	0.62	$5.0376 \cdot 10^{-9}$	41	0.35	$4.8965 \cdot 10^{-11}$
6	2	28	21	13	22	1	0.00	-2.499999995	22	15	12	0.00	-2.499999731	53	0.00	-2.500000000
7	2	44	39	30	2	3	0.00	0.500165625	72	63	30	0.00	0.500000004	27	0.00	1.000000000^*
8	3	88	69	46	2	3	0.00	3.500000000	75	56	34	0.00	3.500000158	23	0.00	3.772727273^*
9	4	95	87	57	2	3	0.01	1.833333333	85	75	39	0.00	1.833333432	4	0.00	9.200000000^*
10	2	33	26	20	1	2	0.00	-0.500000000	19	12	6	0.00	-0.499999982	18	0.00	-0.500000000
10	5	36	28	21	1	2	0.00	-2.500000000	20	12	9	0.00	-2.499999876	18	0.00	-2.500000000
10	10	78	65	42	1	2	0.02	-8.500000000	55	42	22	0.01	-8.499999619	27	0.00	-6.500000000^*
10	25	123	106	57	1	2	0.04	-22.500000000	74	57	31	0.02	-22.499980760	99	0.00	-22.500000000
10	50	173	160	64	1	2	0.19	-48.500000000	138	120	53	0.10	-48.499894232	5	0.00	-0.500000000^*
10	100	354	345	110	1	2	1.50	-98.500000000	348	340	92	1.14	-98.499676380	141	0.01	-90.500000000^*
10	150	395	395	114	1	2	2.70	-146.500000000^*	442	442	103	3.56	-126.489324007^*	154	0.02	-134.500000000^*
10	200	431	432	121	3	3	4.94	-150.500000000^*	466	466	124	4.68	$-150.493380460*$	8	0.00	-82.500000000^*

* the obtained value of the objective function f is not globally optimal

Table 2: Summary of numerical results with DBDC, PBDC and MPBNGC (cont.)

		DBDC							PBDC					MPBNGC		
		Main it.			Clarke alg.											
Prob.	n	n_f	n_{ξ_1}	n_{ξ_2}	n_f	n_{ξ_1}, n_{ξ_2}	*time*	f	n_f	n_{ξ_1}	n_{ξ_2}	*time*	f	n_f, n_ξ	*time*	f
11	3	10	8	6	3	4	0.00	116.333333333	10	8	6	0.00	116.333333333	28	0.00	116.333333333
12	2	19	15	12	2	3	0.00	1.618034002*	20	17	12	0.00	1.618033989*	50	0.00	1.618033989*
12	5	79	76	47	5	6	0.01	1.618070039*	58	53	24	0.01	1.618033996*	281	0.00	0.618033989
12	10	195	195	127	10	11	0.03	0.618320571	1415	1410	1103	0.22	0.618034013	380	0.01	0.618033989
12	25	465	465	274	12	13	0.50	0.618787525	100032	100032	97376	35.4	82.884884559*	486	0.04	0.618033989
12	50	344	344	242	12	13	1.39	0.619096167	100012	100012	47289	99.2	430.746179992*	3363	1.09	0.618033989
12	100	408	408	267	13	14	11.51	0.619403704	100042	100042	98998	415.5	422.561066204*	2969	5.06	0.618033990
13	10	102	100	44	0	1	0.01	0.000000000	172960	172956	40239	24.6	0.186079211*	7	0.00	0.000000000
14	2	9	5	5	1	2	0.00	$2.7756 \cdot 10^{-17}$	9	5	5	0.00	$1.3878 \cdot 10^{-16}$	9	0.00	$2.7756 \cdot 10^{-17}$
14	5	131	126	123	4	5	0.01	$4.9281 \cdot 10^{-14}$	178	173	171	0.02	$9.7143 \cdot 10^{-14}$	100000	0.54	$4.7664 \cdot 10^{-8}$
14	10	139	133	125	8	9	0.02	$1.3966 \cdot 10^{-10}$	103565	103559	103501	7.33	$4.3196 \cdot 10^{-11}$	448	0.01	$1.1891 \cdot 10^{-8}$
14	50	194	184	175	10	11	0.35	$1.1724 \cdot 10^{-9}$	100988	100978	100894	74.8	$2.3835 \cdot 10^{-11}$	11252	2.35	$7.5924 \cdot 10^{-10}$
14	100	268	230	226	11	12	1.32	$5.6087 \cdot 10^{-9}$	131661	131623	118710	426.5	$1.4379 \cdot 10^{-10}$	2535	2.61	$4.3027 \cdot 10^{-9}$
14	500	126	121	121	14	15	5.11	$1.1234 \cdot 10^{-5}$	599	594	594	67.6	$1.4623 \cdot 10^{-8}$	7978	260.3	$2.5992 \cdot 10^{-8}$
14	1000	186	180	180	17	18	25.55	$1.8624 \cdot 10^{-5}$	752	746	746	228.9	$5.2580 \cdot 10^{-8}$	7467	998.3	$1.6807 \cdot 10^{-8}$
15	2	1	1	1	0	1	0.00	0.000000000	1	1	1	0.00	0.000000000	1	0.00	0.000000000
15	5	117	82	63	102	103	0.22	$1.7127 \cdot 10^{-8}$	335	265	196	0.03	$7.1054 \cdot 10^{-14}$	31	0.00	$3.9272 \cdot 10^{-11}$
15	10	200	154	113	34	13	0.05	$2.7669 \cdot 10^{-5}$	19245	19143	11990	4.20	$1.4921 \cdot 10^{-12}$	44	0.00	$1.2975 \cdot 10^{-11}$
15	25	1281	1175	492	60	39	1.57	$1.4891 \cdot 10^{-4}$	2382	2339	1218	3.06	$1.9895 \cdot 10^{-13}$	100000	3.28	$2.1333 \cdot 10^{-3}$
15	50	2914	2805	874	265	217	15.49	$6.0652 \cdot 10^{-4}$	630	594	378	4.29	$2.2217 \cdot 10^{-9}$	257	0.22	$9.5838 \cdot 10^{-11}$
15	100	905	867	810	166	148	29.71	$5.1594 \cdot 10^{-5}$	3489	3451	2380	101.9	$6.2007 \cdot 10^{-8}$	513	2.56	$6.7303 \cdot 10^{-11}$
16	2	8	6	6	1	2	0.00	$9.5223 \cdot 10^{-6}$	9	7	7	0.00	$9.0653 \cdot 10^{-11}$	14	0.00	$1.4211 \cdot 10^{-13}$
16	5	17	11	8	1	2	0.00	$2.6263 \cdot 10^{-9}$	14	11	11	0.00	$8.1340 \cdot 10^{-10}$	28	0.00	$8.6975 \cdot 10^{-13}$
16	10	14	11	6	1	2	0.00	$1.0471 \cdot 10^{-10}$	16	13	7	0.00	$1.2543 \cdot 10^{-10}$	42	0.00	$4.4409 \cdot 10^{-16}$
16	50	10	7	6	2	3	0.00	$6.8127 \cdot 10^{-8}$	10	7	6	0.01	$6.8127 \cdot 10^{-8}$	28	0.00	$9.3365 \cdot 10^{-11}$
16	100	16	10	10	2	3	0.00	$5.9585 \cdot 10^{-5}$	17	11	11	0.00	$6.3237 \cdot 10^{-11}$	86	0.00	$3.0973 \cdot 10^{-11}$
16	250	43	23	23	1	2	0.03	$1.1263 \cdot 10^{-4}$	44	24	24	0.03	$1.2957 \cdot 10^{-10}$	183	0.04	$7.8249 \cdot 10^{-12}$

* the obtained value of the objective function f is not globally optimal

time it guarantees Clarke stationarity, which is a stronger stopping condition than criticality used in PBDC. Therefore, a little bit more computational effort cannot be seen as a real disadvantage for DBDC, since at the same time we are able to be more assured about the quality of the solution and also avoid the bad features of critical points. In addition, in some problems DBDC is clearly more efficient hen PBDC. Moreover, DBDC has the best ability to find global minimizer among the methods tested.

7 Conclusions

In this paper, we have presented a new proximal double bundle algorithm (DBDC) for unconstrained nonsmooth DC optimization, which utilizes explicitly the DC decomposition of the objective. The novelty of DBDC is a new stopping procedure guaranteeing Clarke stationarity usingg only the information about the DC components of the objective. This way DBDC can exploit the DC structure through all the algorithm and also avoid the problematic features of criticality, which is the stopping condition typically used in DC optimization algorithms. In addition, the finite convergence of the method DBDC is proved under really mild assumptions.

The numerical results reported also confirm that DBDC is efficient to solve nonsmooth DC programming problems. Moreover, although DBDC is only a local solution method, it nearly always found the global minimizer of a problem. Therefore, we can conclude that DBDC is a good alternative for existing nonconvex bundle methods, when the DC representation of the objective can be formulated.

Acknowledgments This work has been financially supported by the University of Turku Graduate School UTUGS Matti Programme, the University of Turku, the Jenny and Antti Wihuri Foundation, the Magnus Ehrnrooth Foundation, the Academy of Finland (Project No. 289500 and 294002) and the Australian Research Council's Discovery Projects funding scheme (Project No. DP140103213).

References

[1] A. Astorino, A. Fuduli, and M. Gaudioso, *Margin maximization in spherical separation*, Comput. Optim. Appl. 53 (2012), pp. 301–322.

[2] A. M. Bagirov, *A method for minimization of quasidifferentiable functions*, Optim. Methods Softw. 17 (2002), pp. 31–60.

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

[4] A. M. Bagirov, S. Taheri, and J. Ugon, *Nonsmooth dc programming approach to the minimum sum-of-squares clustering problems*, Pattern Recongn. 53 (2016), pp. 12–24.

[5] A. M. Bagirov and J. Ugon, *Nonsmooth dc programming approach to clusterwise linear regression: optimality conditions and algorithms*, submitted.

[6] A. M. Bagirov and J. Ugon, *Codifferential method for minimizing nonsmooth DC functions*, J. Global Optim. 50 (2011), pp. 3–22.

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

[8] V. F. Demyanov and A. M. Rubinov, *Constructive Nonsmooth Analysis (Approximation and Optimization)*, vol. 7, Peter Lang, Frankfurt am Main, Germany, 1995.

[9] A. Ferrer, *Representation of a polynomial function as a difference of convex polynomials with an application*, Lecture Notes in Econom. and Math. Systems, 502 (2001), pp. 189–207.

[10] A. Fuduli, M. Gaudioso, and G. Giallombardo, *Minimizing nonconvex nonsmooth functions via cutting planes and proximity control*, SIAM J. Optim. 14 (2004), pp. 743–756.

[11] A. Fuduli, M. Gaudioso, and E. A. Nurminski, *A splitting bundle approach for non-smooth non-convex minimization*, Optimization, 64 (2015), pp. 1131–1151.

[12] M. Gaudioso and E. Gorgone, *Gradient set splitting in nonconvex nonsmooth numerical optimization*, Optim. Methods Softw. 25 (2010), pp. 59–74.

[13] P. Hartman, *On functions representable as a difference of convex functions*, Pacific J. Math. 9 (1959), pp. 707–713.

[14] J.-B. Hiriart-Urruty, *Generalized differentiability, duality and optimization for problems dealing with differences of convex functions*, Lecture Notes in Econom. and Math. Systems, 256 (1985), pp. 37–70.

[15] K. Holmberg and H. Tuy, *A production-transportation problem with stochastic demand and concave production costs*, Math. Prog. 85 (1999), pp. 157–179.

[16] R. Horst and N. V. Thoai, *DC programming: Overview*, J. Optim. Theory Appl. 103 (1999), pp. 1–43.

[17] K. Joki, A. M. Bagirov, N. Karmitsa, and M. M. Mäkelä, *A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes*, J. Global Optim. (2016), pp. 1–35. DOI 10.1007/s10898-016-0488-3

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

[19] K. C. Kiwiel, *Proximity control in bundle methods for convex nondifferentiable minimization*, Math. Prog. 46 (1990), pp. 105–122.

[20] H. A. Le Thi and T. Pham Dinh, *Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms*, J. Global Optim. 11 (1997), pp. 253–285.

[21] H. A. Le Thi and T. Pham Dinh, *The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems*, Ann. Oper. Res. 133 (2005), pp. 23–46.

[22] H. A. Le Thi, T. Pham Dinh, and H. V. Ngai, *Exact penalty and error bounds in DC programming*, J. Global Optim. 52 (2012), pp. 509–535.

[23] L. Lukšan, *Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minmax approximation*, Kybernetika, 20 (1984), pp. 445–457.

[24] M. M. Mäkelä, *Survey of bundle methods for nonsmooth optimization*, Optim. Methods Softw. 17 (2002), pp. 1–29.

[25] M. M. Mäkelä, *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.

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

[27] R. Mifflin, *An algorithm for constrained optimization with semismooth functions*, Math. Oper. Res. 2 (1977), pp. 191–207.

[28] O. Montonen and K. Joki, *Multiobjective double bundle method for multiobjective DC optimization*, TUCS Technical Report No. 1174, Turku Centre for Computer Science, Turku, (2016).

[29] B. Ordin and A. M. Bagirov, *A heuristic algorithm for solving the minimum sum-of-squares clustering problems*, J. Global Optim. 61 (2015), pp. 341–361.

[30] C. Pey-Chun, P. Hansen, B. Jaumard, and H. Tuy, *Solution of the multisource Weber and conditional Weber problems by d.c. programming*, Oper. Res. 46 (1998), pp. 548–562.

[31] T. Pham Dinh and H. A. Le Thi, *Convex analysis approach to DC programming: Theory, algorithms and applications*, Acta Math. Vietnam. 22 (1997), pp. 289–355.

[32] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, New Jersey, 1970.

[33] H. Schramm and J. Zowe, *A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results*, SIAM J. Optim. 2 (1992), pp. 121–152.

[34] J. C. O. Souza, P. R. Oliveira, and A. Soubeyran, *Global convergence of a proximal linearized algorithm for difference of convex functions*, Optim. Lett. 10 (2016), pp. 1529–1539.

[35] J. F. Toland, *On subdifferential calculus and duality in nonconvex optimization*, Bull. Soc. Math. France, Mémoire, 60 (1979), pp. 177–183.

[36] H. Tuy, *Convex Analysis and Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1st ed., 1998.

Appendix: Test problems

All test problems are unconstrained DC optimization problems and in these problems objective functions are presented as DC functions:

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

Therefore, in the description of all test problems we present only functions f_1 and f_2. The following notations are used to describe test problems:

$\boldsymbol{x}_0 \in \mathbb{R}^n$ – starting point;
$\boldsymbol{x}^* \in \mathbb{R}^n$ – known best solution;
f^* – known best value of the objective function.

Problem 11.
Dimension: $n = 3$,
Component functions: $f_1(x) = 4|x_1| + 2|x_2| + 2|x_3| - 33x_1 + 16x_2 - 24x_3$,
$+ 100 \max\{0, 2|x_2| - 3x_1 - 7\} + 100 \max\{0, |x_3| - 4x_1 - 11\}$,
$f_2(x) = 20\,(-7x_1 + 2|x_2| - |x_3| - 18)$,
Starting point: $x_0 = (10, 10, 10)^T$,
Optimum point: $x^* = (-7/3, 0, 5/3)^T$,
Optimum value: $f^* = 116.33333333$.

Problem 12.
Dimension: $n = 2, 5, 10, 25, 50, 100,$
Component functions: $f_1(x) = \sum_{i=1}^{n} |x_i| + 10 \sum_{i=1}^{n} \max\left\{2(x_i^2 - x_i - 1), 0\right\},$
$f_2(x) = 10 \sum_{i=1}^{n} (x_i^2 - x_i - 1) + \max_{i=1,\ldots,n} \sum_{j=1, j\neq i} |x_i|,$
Starting point: $x_0 = (x_{0,1}, \ldots, x_{0,n})^T,$ *where* $x_{0,i} = 2 \cdot i$ *for* $i = 1, \ldots, n,$
Optimum point: $x^* = (-0.618034, \ldots, -0.618034)^T,$
Optimum value: $f^* = 0.61803.$

Problem 13.
Dimension: $n = 10,$
Component functions: $f_1(x) = \sum_{i=1}^{n-1} |x_i + x_{i+1}| + \sum_{i=1}^{n-2} |x_i + x_{i+2}|$
$+ |x_1 + x_9| + |x_1 + x_{10}| + |x_2 + x_{10}| + |x_1 + x_5| + |x_4 + x_7|,$
$+ 10 \max\{0, \sum_{i=1}^{n} x_i - 1\} + 10 \sum_{i=1}^{n} \max\{0, -x_i\},$
$f_2(x) = \sum_{i=1}^{n-1} (|x_i| + |x_{i+1}|) + \sum_{i=1}^{n-2} (|x_i| + |x_{i+2}|) + 3|x_1| + |x_2|$
$+ |x_4| + |x_5| + |x_7| + |x_9| + 2|x_{10}|,$
Starting point: $x_0 = (10, \ldots, 10)^T,$
Optimum point: $x^* = (x_1^*, \ldots, x_{10}^*)^T,$ *where*
$x_i^* > 0$ *for* $i = 1, \ldots, 10$ *and* $\sum_{i=1}^{n} x_i^* \leq 1$
Optimum value: $f^* = 0.$

Problem 14. *(DC version of L1HILB)*
Dimension: $n = 2, 5, 10, 25, 50, 100, 500, 1000,$
Component functions: $f_1(x) = n \max_{i=1,\ldots,n} \left| \sum_{j=1}^{n} \frac{x_j}{i+j-1} \right|,$
$f_2(x) = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} \frac{x_j}{i+j-1} \right|,$
Starting point: $x_0 = (1, \ldots, 1)^T,$
Optimum point: $x^* = (0, \ldots, 0)^T,$
Optimum value: $f^* = 0.$

Problem 15. *(DC version of CB3 I)*
Dimension: $n = 2, 5, 10, 25, 50, 100,$
Component functions: $f_1(x) = (n-1) \max_{i=1,\ldots,n-1} g_i(x),$
$f_2(x) = \sum_{i=1}^{n-1} g_i(x),$
$g_i(x) = \max\left\{x_i^4 + x_{i+1}^2, (2 - x_i)^2 + (2 - x_{i+1})^2, 2\exp(-x_i + x_{i+1})\right\},$
Starting point: $x_0 = (x_{0,1}, \ldots, x_{0,n})^T,$ *where*
$x_{0,i} = 1$ *for odd* i *and* $x_{0,i} = -1$ *for even* $i,$
Optimum value: $f^* = 0.$

Problem 16. *(Chained Crescent I)*
Dimension: $n = 2, 5, 10, 25, 50, 100, 250,$
Component functions: $f_1(x) = \max\left\{2 \sum_{i=1}^{n-1} \left(x_i^2 + (x_{i+1} - 1)^2 + x_{i+1} - 1\right), 0\right\},$
$f_2(x) = \sum_{i=1}^{n-1} \left(x_i^2 + (x_{i+1} - 1)^2 + x_{i+1} - 1\right)$
Starting point: $x_0 = (x_{0,1}, \ldots, x_{0,n})^T,$ *where*
$x_{0,i} = -1.5$ *for odd* i *and* $x_{0,i} = 2$ *for even* i
Optimum value: $f^* = 0.$

Turku
Centre *for*
Computer
Science

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

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

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

ISBN 978-952-12-3500-9
ISSN 1239-1891