COPYRIGHT NOTICE

SIAM J. OPTIM.
Vol. 28, No. 2, pp. 1892–1919

DOUBLE BUNDLE METHOD FOR FINDING CLARKE STATIONARY POINTS IN NONSMOOTH DC PROGRAMMING[*]

KAISA JOKI[†], ADIL M. BAGIROV[‡], NAPSU KARMITSA[†], MARKO M. MÄKELÄ[†], AND SONA TAHERI[‡]

Abstract. The aim of this paper is to introduce a new proximal double bundle method for unconstrained nonsmooth optimization, where the objective function is presented as a difference of two convex (DC) functions. The novelty in our method is a new escape procedure which enables us to guarantee approximate Clarke stationarity for solutions by utilizing the 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 approximate Clarke stationary, then the escape procedure returns a descent direction. With this escape procedure, we can avoid some shortcomings encountered when criticality is used. The finite termination of the double bundle method to an approximate Clarke stationary point is proved by assuming that the subdifferentials of DC components are polytopes. Finally, some encouraging numerical results are presented.

Key words. nonsmooth optimization, nonconvex optimization, DC functions, bundle methods, cutting plane model, Clarke stationarity

AMS subject classifications. 90C26, 49J52, 65K05

DOI. 10.1137/16M1115733

1. Introduction.

A class of functions represented as a difference of convex (DC) functions constitutes an important subclass of nonconvex functions. These functions preserve, with some modifications, important properties of convex functions. Another advantage is that the class of DC functions maintains the DC structure under simple algebraic operations frequently used 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, 37].

Optimization problems with DC objective and constraint functions are called DC programming problems. Many practical problems can be modelled as a DC programming problem where the explicit DC decompositions of the objective and/or constraint functions are readily available. These problems include production-transportation planning [15], location planning [31], engineering design [22], cluster analysis [4, 30], clusterwise linear regression analysis [6], and supervised data classification [1], to name a few. In general, the calculation of an explicit DC representation is a hard task since these representations appear often in an implicit form. Note that DC decompositions are not unique and each DC function has an infinite number of different DC representations.

[*] Received by the editors February 9, 2017; accepted for publication (in revised form) March 13, 2018; published electronically June 28, 2018.
http://www.siam.org/journals/siopt/28-2/M111573.html

Funding: This work was funded 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 (Projects 289500 and 294002), and the Australian Research Council's Discovery Projects funding scheme (Project DP140103213).

[†] Department of Mathematics and Statistics, University of Turku, FI-20014 Turku, Finland (kjjoki@utu.fi, napsu@karmitsa.fi, makela@utu.fi).

[‡] Faculty of Science and Technology, Federation University Australia, University Drive, Mount Helen, PO Box 663, Ballarat, VIC 3353, Australia (a.bagirov@federation.edu.au, s.taheri@federation.edu.au).

To date, DC programming has been predominantly considered as a part of global optimization, and several algorithms have been designed to solve it globally (see [16, 37] and references therein). However, global optimization techniques such as a cutting plane method for DC programming and the branch and bound method are not efficient for solving large-scale DC programming problems often encountered in real-world applications [23]. The development of local search methods in DC programming has attracted less attention. There exist several methods specifically designed for nonsmooth DC programming problems using their explicit DC representations [2, 5, 17, 35]. 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 the Clarke stationarity typically used in general nonconvex nonsmooth optimization. Unfortunately, it may happen that these algorithms stop at a point which is neither a local minimizer nor a saddle point. This undesirable feature is often a consequence of the selected DC decomposition since it affects the criticality condition tested. However, in general, there is no efficient way to detect the most suitable DC decomposition among the infinite set of possible ones.

In this paper, we introduce a new proximal double bundle method (DBDC) for unconstrained nonsmooth DC minimization problems. The main idea in the DBDC is to combine the proximal bundle method (PBDC) [17] for nonsmooth DC programming with a new escape 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, to preserve all good features obtained from the use of the DC structure in the PBDC method.

In the DBDC method using DC decompositions, the nonconvex cutting plane model is developed to capture both the convex and concave behaviour of a DC function. The escape procedure, in its turn, is able to generate a new search direction whenever a candidate solution (typically a critical point) is not approximate Clarke stationary. Moreover, this procedure is designed in such a way that the approximate Clarke stationarity can be ensured by using only information about the DC components. To prove the finite termination of the escape procedure, we assume that the subdifferentials of DC components are polytopes.

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 drawbacks of critical points. The new escape procedure guaranteeing approximate Clarke stationarity is presented in section 4 and section 5 describes the new minimization algorithm DBDC. Numerical results are reported in section 6 and, finally, in section 7 we give some concluding remarks.

2. Preliminaries.

Consider a *DC minimization problem* of the form

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

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 is called a *DC function* and $f_1 - f_2$ is the *DC decomposition* of f. The convex functions f_1 and f_2 are called *DC components* of f. Note that f is often nonconvex and it is not necessarily differentiable. In addition, when f is nonsmooth, then at least one of its DC components is also nonsmooth.

Next, we present some results from nonsmooth analysis and DC programming. For more details we refer the reader to [3, 7, 13, 22, 28, 32, 33]. 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* of a convex function f at a point $\boldsymbol{x} \in \mathbb{R}^n$ is the set [33]

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

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* of a locally Lipschitz continuous function (LLC) f at a point $\boldsymbol{x} \in \mathbb{R}^n$ is given by [7]

$$\partial f(\boldsymbol{x}) = \operatorname{conv}\left\{ \lim_{i\to\infty} \nabla f(\boldsymbol{x}_i) \;\middle|\; \boldsymbol{x}_i \to \boldsymbol{x} \text{ and } \nabla f(\boldsymbol{x}_i) \text{ exists}\right\},$$

where "conv" denotes the convex hull of a set. A point $\boldsymbol{x}^* \in \mathbb{R}^n$ is called *Clarke stationary* if $\boldsymbol{0} \in \partial f(\boldsymbol{x}^*)$, and this is a necessary condition for local optimality. It is known that for a convex function f defined on $\mathbb{R}^n$ we have $\partial f(\boldsymbol{x}) = \partial_c f(\boldsymbol{x})$ [7].

For $\varepsilon \geq 0$, the *Goldstein ε-subdifferential* of an LLC function f at $\boldsymbol{x} \in \mathbb{R}^n$ is [28]

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

It is obvious that $\partial f(\boldsymbol{x}) \subset \partial_\varepsilon^G f(\boldsymbol{x})$ for all $\varepsilon \geq 0$. In practice, we consider solutions fulfilling the condition $\boldsymbol{0} \in \partial_\varepsilon^G f(\boldsymbol{x})$ since the Goldstein ε-subdifferential approximates $\partial f(\boldsymbol{x})$.

A general LLC function is not necessarily directionally differentiable. However, our objective f is a finite-valued DC function and it is directionally differentiable at any $\boldsymbol{x} \in \mathbb{R}^n$ [3] meaning that the *directional derivative* of f at $\boldsymbol{x}$ exists in every direction $\boldsymbol{d} \in \mathbb{R}^n$ and is defined as

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

In addition, for a DC function f 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.* This means 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 necessary conditions for local optimality in DC programming.

THEOREM 2.1 (see [22, 36]). *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*

$$\partial f_2(\boldsymbol{x}^*) \subseteq \partial f_1(\boldsymbol{x}^*). \tag{1}$$

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

The inf-stationarity condition (1) is not easy to verify since in practice it is not easy to calculate the whole subdifferentials of DC components f_1 and f_2. Therefore, in numerical algorithms a relaxed form of condition (1) is often used requiring that [14, 22, 36]

$$\partial f_1(\boldsymbol{x}^*) \cap \partial f_2(\boldsymbol{x}^*) \neq \emptyset. \tag{2}$$

A point $\boldsymbol{x}^* \in \mathbb{R}^n$ satisfying (2) is called a *critical point*. Due to Theorem 2.1, condition (2) is also a necessary condition for local optimality.

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

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

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 obtain only Clarke stationarity. This is due to the fact that [7]

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

and since $\partial f_2(\boldsymbol{x}^*)$ may contain more than one element it cannot be a subset of $\partial f_1(\boldsymbol{x}^*) = \{\nabla f_1(\boldsymbol{x}^*)\}$. The relationships between different stationarities are shown in Figure 1.

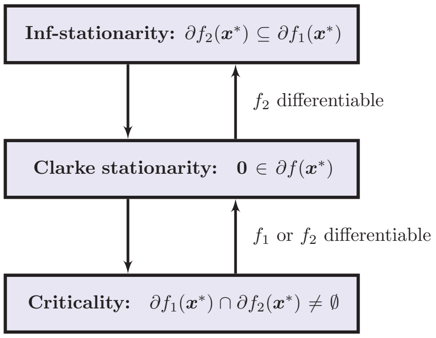
Inf-stationarity: $\partial f_2(\boldsymbol{x}^*) \subseteq \partial f_1(\boldsymbol{x}^*)$
f_2 differentiable
Clarke stationarity: $\boldsymbol{0} \in \partial f(\boldsymbol{x}^*)$
f_1 or f_2 differentiable
Criticality: $\partial f_1(\boldsymbol{x}^*) \cap \partial f_2(\boldsymbol{x}^*) \neq \emptyset$

FIG. 1. *Relationships between different stationary points.*

3. Drawbacks of the criticality condition.

In this section, we present some examples to demonstrate drawbacks of critical points. First, criticality is a weaker condition than Clarke stationarity. This follows from the subdifferential calculus, which only guarantees that for a DC function f we have [7]

$$\partial f(\boldsymbol{x}) \subseteq \partial f_1(\boldsymbol{x}) - \partial f_2(\boldsymbol{x}). \tag{3}$$

Therefore, the difference of arbitrary subgradients of f_1 and f_2 does not need to belong to $\partial f(\boldsymbol{x})$. Nevertheless, there are some exceptions to this as we have already seen in section 2. Second, it is possible that a critical point is neither a local optimum nor a saddle point of the objective f. Thus, a critical point may fail to give us any useful information about f. This means that, in practice, minimization 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 shows the effect of the "bad" selection of the DC decomposition for a linear function, which can happen in a linear part of a more complex function. However, in the second example it is not easy to see if the DC decomposition of f could be selected in 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 $f_1(x) = \max\{-x, 2x\}$ and $f_2(x) = \max\{-2x, x\}$. At a point $x^* = 0$ the DC components f_1 and f_2 are not differentiable and the subdifferentials are $\partial f_1(0) = [-1, 2]$ and $\partial f_2(0) = [-2, 1]$. 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 provides no interesting feature for 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 $f_1(x) = \max\{x^2, x\}$ and $f_2(x) = \max\{0.5x^2, -x\}$. Let us look closer the point $x^* = 0$. Functions f_1 and f_2 are not differentiable at x^* and their subdifferentials are $\partial f_1(0) = [0, 1]$ and $\partial f_2(0) = [-1, 0]$. Since $\partial f_1(0) \cap \partial f_2(0) = \{0\} \neq \emptyset$ the point $x^* = 0$ is a critical point of f. However, f is differentiable at x^* and $\partial f(0) = \{1\}$. Therefore, the critical point x^* is not a local minimizer or even a saddle point. Graphs of the DC components and the objective f are illustrated in Figure 2.

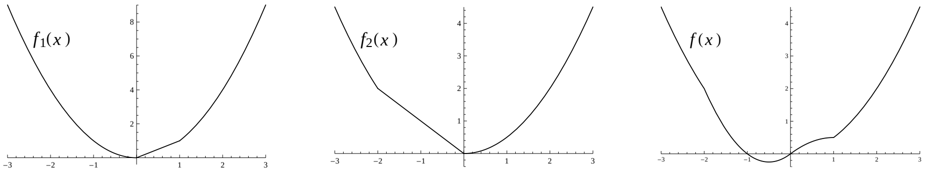

FIG. 2. *The DC components $f_1(x)$ and $f_2(x)$ and the objective function $f(x)$.*

An undesirable behaviour in critical points is sometimes a result of the selected DC decomposition. However, since a DC function has an infinite number of different DC decompositions it is not easy 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 and it depends on the overall structure of the problem. Therefore, there is no efficient way to avoid this poor feature of critical points.

4. Guaranteeing approximate Clarke stationarity. In this section, we describe the new escape procedure, which either guarantees approximate 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 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 section 3, this does not hold in general. Later on, the escape procedure is used in the main iteration of the DBDC method to detect the cases in which a critical point or a promising candidate solution fulfils approximate 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 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 weakly semismooth [29, 34]. This implies that

$$f_i'(\boldsymbol{x}; \boldsymbol{d}) = \lim_{k \to \infty} \boldsymbol{v}_k^T \boldsymbol{d} \tag{4}$$

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 (4) that

$$U_i(\boldsymbol{x}; \boldsymbol{d}) \subseteq G_i(\boldsymbol{x}; \boldsymbol{d}) \quad \text{for } i = 1, 2. \tag{5}$$

The definition of $U_i(\boldsymbol{x}; \boldsymbol{d})$ yields that for any $\varepsilon > 0$ there exists $t_0 > 0$ such that

$$\partial f_i(\boldsymbol{x} + t\boldsymbol{d}) \subset U_i(\boldsymbol{x}; \boldsymbol{d}) + B_\varepsilon(\boldsymbol{0}) \quad \text{for } i = 1, 2 \text{ and } \forall t \in (0, t_0).$$

This together with (5) implies that

$$\partial f_i(\boldsymbol{x} + t\boldsymbol{d}) \subset G_i(\boldsymbol{x}; \boldsymbol{d}) + B_\varepsilon(\boldsymbol{0}) \quad \text{for } i = 1, 2 \text{ and } \forall t \in (0, t_0). \tag{6}$$

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

$$T_{DC}(\boldsymbol{x}) = T_1(\boldsymbol{x}) \cap T_2(\boldsymbol{x}) \subseteq \mathbb{R}^n$$

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

THEOREM 4.1. *Let $f = f_1 - f_2$ be a DC function, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{d} \in T_{DC}(\boldsymbol{x})$, $G_1(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi}_1\}$, 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 (6) 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

$$\|\boldsymbol{v} - \boldsymbol{\xi}_1\| < \varepsilon, \quad \|\boldsymbol{w} - \boldsymbol{\xi}_2\| < \varepsilon \tag{7}$$

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, rule (3) implies that $\partial f(\boldsymbol{x} + t\boldsymbol{d}) \subseteq \partial f_1(\boldsymbol{x} + t\boldsymbol{d}) - \partial f_2(\boldsymbol{x} + t\boldsymbol{d})$ for $t \geq 0$. Therefore, for any $\boldsymbol{\xi}_t \in \partial f(\boldsymbol{x} + t\boldsymbol{d})$ there exist $\boldsymbol{v}_t \in \partial f_1(\boldsymbol{x} + t\boldsymbol{d})$ 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 (7) 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

$$\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x}+t\boldsymbol{d}) + B_{2\varepsilon}(\boldsymbol{0}) \quad \forall t \in (0, t_0). \tag{8}$$

Furthermore, upper semicontinuity of $\partial f(\boldsymbol{x})$ [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 (8) 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 $\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x})$. This completes the proof. □

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

$$\begin{aligned}\partial_{T_{DC}} f(\boldsymbol{x}) = \operatorname{cl}\operatorname{conv}\{\boldsymbol{\xi} \in \mathbb{R}^n \mid \exists \boldsymbol{d} \in T_{DC}(\boldsymbol{x}),\ \boldsymbol{\xi} = \boldsymbol{\xi}_1 - \boldsymbol{\xi}_2,\\ \boldsymbol{\xi}_1 \in G_1(\boldsymbol{x}; \boldsymbol{d}),\ \boldsymbol{\xi}_2 \in G_2(\boldsymbol{x}; \boldsymbol{d})\}.\end{aligned}$$

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 us at a point $\boldsymbol{x}$ for any direction $\boldsymbol{d} \in \mathbb{R}^n$ to find a direction $\bar{\boldsymbol{d}} \in T_{DC}(\boldsymbol{x})$ 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 singletons 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 show how the direction $\bar{\boldsymbol{d}} \in T_{DC}(\boldsymbol{x})$ for any $\boldsymbol{d} \in \mathbb{R}^n$ can be found at a point $\boldsymbol{x} \in \mathbb{R}^n$. For this reason we utilize the so-called R-sets [3] and assume the following.

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

Remark 4.4. Assumption A1 is not very restrictive since in practical applications the subdifferentials of DC components f_1 and f_2 are nearly always polytopes.

Let $A \subset \mathbb{R}^n$ be a polytope, that is, it can be represented as $A = \operatorname{conv} A_0$, where $A_0 = \{\boldsymbol{a}_1, \dots, \boldsymbol{a}_m\}$, $\boldsymbol{a}_i \in \mathbb{R}^n$, and $m \geq 1$. Let

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

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

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

It is clear that

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

Moreover,

$$v_k = w_k \quad \forall \boldsymbol{v}, \boldsymbol{w} \in R_j(\boldsymbol{g}),\ k = 1, \dots, j. \tag{9}$$

LEMMA 4.5. *For any $\boldsymbol{g} \in V$, the set $R_n(\boldsymbol{g})$ is a singleton.*

Proof. It follows from (9) 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 vector $\boldsymbol{e}^n(\alpha) = (\alpha g_1, \alpha^2 g_2, \dots, \alpha^n g_n)$. Denote by σ_A the support function of the polytope A, i.e., $\sigma_A(\boldsymbol{d}) = \max\{\boldsymbol{v}^T\boldsymbol{d} \,|\, \boldsymbol{v} \in A\}$, and for the direction $\boldsymbol{d} \in \mathbb{R}^n$, $\boldsymbol{d} \neq \boldsymbol{0}$, define the set

$$G_A(\boldsymbol{d}) = \left\{\boldsymbol{a} \in A \mid \boldsymbol{a}^T\boldsymbol{d} = \sigma_A(\boldsymbol{d})\right\}.$$

Since the set A is a polytope the set $G_A(\boldsymbol{d})$ is never empty and contains at least one vertex of A. In addition, if $G_A(\boldsymbol{d})$ contains only one point, then this point is a vertex of the set A.

LEMMA 4.6. *Let A be a polytope with a finite vertex set A_0. For a given $\boldsymbol{g} \in V$, there exists $\alpha_0 \in (0, 1]$ such that $G_A(\boldsymbol{e}^n(\alpha))$ 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 4.5, 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 $r \in \{1, \dots, n\}$ such that $\boldsymbol{b} \in R_p(\boldsymbol{g})$ for all $p = 0, \dots, r-1$ 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 (9) we know that for any $\boldsymbol{b} \in A_0 \setminus \{\boldsymbol{a}\}$ we have $a_t = b_t$ for $t = 1, \dots, r-1$ when $r \geq 2$. Thus,

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

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 $\boldsymbol{b} \in A_0 \setminus \{\boldsymbol{a}\}$. If we continue by choosing $\alpha_0 = \min\{1, \frac{\delta}{4Dn}\}$, 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,

$$\boldsymbol{a}^T \boldsymbol{e}^n(\alpha) - \boldsymbol{b}^T \boldsymbol{e}^n(\alpha) > 0 \quad \forall \boldsymbol{b} \in A_0 \setminus \{\boldsymbol{a}\} \tag{10}$$

and since the set A_0 contains all the vertices of the polytope A inequality (10) holds also for all $\boldsymbol{b} \in A \setminus \{\boldsymbol{a}\}$. This shows our claim. □

Lemma 4.6 shows that for any polytope $A \subset \mathbb{R}^n$ and $\boldsymbol{g} \in V$ 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 the polytope.

Lemma 4.7. *Let A be a polytope with a finite vertex set 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. Let us consider a polytope $B(\boldsymbol{d}) = \operatorname{conv} G_A(\boldsymbol{d})$. If we calculate the set $G_{B(\boldsymbol{d})}(\boldsymbol{e}^n(\alpha))$ for a given $\boldsymbol{v} \in V$, then each $\boldsymbol{v}_0 \in G_{B(\boldsymbol{d})}(\boldsymbol{e}^n(\alpha))$ satisfies the inequality

$$\boldsymbol{v}_0^T \boldsymbol{e}^n(\alpha) \geq \boldsymbol{v}^T \boldsymbol{e}^n(\alpha) + \delta_1 \quad \forall \boldsymbol{v} \in \operatorname{conv} G_A(\boldsymbol{d}) \setminus G_{B(\boldsymbol{d})}(\boldsymbol{e}^n(\alpha)), \tag{11}$$

for some $\delta_1 > 0$. It follows from Lemma 4.6 that for the polytope $B(\boldsymbol{d})$ 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 $\alpha \in (0, \alpha_0]$, that is, $G_{B(\boldsymbol{d})}(\boldsymbol{e}^n(\alpha)) = \{\boldsymbol{v}_0\}$. Since $G_{B(\boldsymbol{d})}(\boldsymbol{e}^n(\alpha))$ is a singleton and $B(\boldsymbol{d})$ is a polytope the only point, $\boldsymbol{v}_0 \in G_{B(\boldsymbol{d})}(\boldsymbol{e}^n(\alpha))$, needs to be a vertex of the set $B(\boldsymbol{d})$. Moreover, this vertex has to belong to both sets $G_A(\boldsymbol{d})$ and A_0, that is, $\boldsymbol{v}_0 \in G_A(\boldsymbol{d}) \cap A_0$. Thus,

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

for some $\delta_1 > 0$ since (11) also holds for the smaller set $G_A(\boldsymbol{d}) \subseteq \operatorname{conv} G_A(\boldsymbol{d})$. On the other hand, from the definition of the set $G_A(\boldsymbol{d})$ we deduce that there exists $\delta_2 > 0$ such that

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

since $A_0 \subset A$. 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 write $D = \max\{\|\boldsymbol{a}\| \mid \boldsymbol{a} \in A_0\} < \infty$. For each $\boldsymbol{v}_0, \boldsymbol{v} \in G_A(\boldsymbol{d})$ we know that $\boldsymbol{v}_0^T \boldsymbol{d} = \boldsymbol{v}^T \boldsymbol{d}$. This together with (12) implies that

$$\boldsymbol{v}_0^T \bar{\boldsymbol{d}}(\alpha) \geq \boldsymbol{v}^T \bar{\boldsymbol{d}}(\alpha) + \delta_1 \quad \forall \boldsymbol{v} \in G_A(\boldsymbol{d}) \setminus \{\boldsymbol{v}_0\}. \tag{13}$$

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

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

for all $\boldsymbol{v} \in G_A(\boldsymbol{d})$. By choosing $\alpha_1 = \min\{1, \frac{\delta_2}{4Dn}\}$, 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 (13) gives

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

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

THEOREM 4.8. *Let $\boldsymbol{x} \in \mathbb{R}^n$, let $\boldsymbol{d} \in \mathbb{R}^n$ be any direction such that $\boldsymbol{d} \neq \boldsymbol{0}$, and assume that a DC function $f = f_1 - f_2$ satisfies 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) $\bar{\boldsymbol{d}}(\alpha) = \boldsymbol{d} + \boldsymbol{e}^n(\alpha) \in T_{DC}(\boldsymbol{x})$,

(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. Properties (i) and (ii) follow immediately from Lemma 4.7. To prove case (iii) notice that $f_1'(\boldsymbol{x}; \boldsymbol{d}) = \boldsymbol{\xi}_1^T \boldsymbol{d}$ for $\boldsymbol{\xi}_1 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$, $f_2'(\boldsymbol{x}; \boldsymbol{d}) = \boldsymbol{\xi}_2^T \boldsymbol{d}$ for $\boldsymbol{\xi}_2 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$, and $f'(\boldsymbol{x}; \boldsymbol{d}) = f_1'(\boldsymbol{x}; \boldsymbol{d}) - f_2'(\boldsymbol{x}; \boldsymbol{d})$. Property (iv) is obtained directly from Theorem 4.1 by taking into account (i). □

4.2. Algorithm. In this subsection, we present our new algorithm, which can escape from non-Clarke stationary points and also detect approximate Clarke stationarity. This verification process is designed for a DC function $f = f_1 - f_2$ and it requires that Assumption A1 holds. This assumption is needed to show the termination of the method and also to guarantee that Step 1 in Algorithm 1 is well defined. Moreover, whenever this escape procedure generates a new better iteration point, it belongs to the set

$$\mathcal{F}_{\boldsymbol{x}} = \{\boldsymbol{y} \in \mathbb{R}^n \mid f(\boldsymbol{y}) \leq f(\boldsymbol{x})\}, \tag{14}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the starting point used in the escape procedure since the new algorithm is a descent one. In Algorithm 1, the set C_k estimates $\partial_\varepsilon^G f(\boldsymbol{x})$, which is also an approximation of $\partial f(\boldsymbol{x})$, and we have $C_k \subset \partial_\varepsilon^G f(\boldsymbol{x})$ for all $k \geq 1$. In addition, the smaller the parameter $\varepsilon > 0$ is, the more accurate the approximation of $\partial f(\boldsymbol{x})$ is. Let $S_1 = \{\boldsymbol{d} \in \mathbb{R}^n \mid \|\boldsymbol{d}\| = 1\}$ be a unit sphere in $\mathbb{R}^n$. The final solution in the procedure is denoted by $\boldsymbol{x}^+$.

Remark 4.9. In Step 4 of Algorithm 1, the step length β^* is determined from formula (18). However, in practice we do not need to find the largest β^* since any $\beta \geq \varepsilon$ satisfying the required descent condition gives a strict descent in the objective function. Therefore, we can first test if the step length $\beta = 1$ decreases the value of the objective. If it does, then we can set $\beta^* = 1$. Otherwise, we divide β by 2 and test again if we obtain a descent in the objective. This backtracking procedure is stopped immediately if we find a step length $\beta \geq \varepsilon$ yielding a descent. Otherwise, we continue until β^* fulfilling (18) is determined with sufficient accuracy.

We start by showing the following useful property needed to prove the finite termination of the escape procedure.

LEMMA 4.10. *Let the set $\mathcal{F}_{\boldsymbol{x}}$ be compact. If during iteration k the execution of Algorithm* 1 *is not stopped at Step* 4, *then*

$$f'(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}; \boldsymbol{d}_{k+1}) > -\hat{m} \|\bar{\boldsymbol{u}}_k\| \quad \textit{for all } \hat{m} > m_1.$$

Proof. First of all, the step-length determination formula (18) is well defined due to the assumption that $\mathcal{F}_{\boldsymbol{x}}$ is compact and thus $\beta^* < \infty$. Therefore, if the execution of Algorithm 1 is continued at Step 4, we know that the step length satisfies $\beta^* < \varepsilon$ and

$$f(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}) - f(\boldsymbol{x}) \leq -m_1 \beta^* \|\bar{\boldsymbol{u}}_k\|. \tag{19}$$

Algorithm 1 Escape procedure.

```
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 $\tilde{\boldsymbol{x}} = \boldsymbol{x}$, $C_0 = \emptyset$, and $k = 1$.
    Step 1. New subgradient. Find $\bar{\boldsymbol{d}}_k(\alpha) \in T_{DC}(\tilde{\boldsymbol{x}})$ 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 $\boldsymbol{\xi}_{1,k} \in G_1(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}_k(\alpha))$ and $\boldsymbol{\xi}_{2,k} \in G_2(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}_k(\alpha))$. Set $\boldsymbol{\xi}_k = \boldsymbol{\xi}_{1,k} - \boldsymbol{\xi}_{2,k}$ and $C_k = \operatorname{conv}\{C_{k-1} \cup \{\boldsymbol{\xi}_k\}\}$.
    Step 2. Clarke stationarity. Find $\bar{\boldsymbol{u}}_k$ as the solution to the following problem:
            (15)    $\min_{\boldsymbol{u} \in C_k} \quad \frac{1}{2}\|\boldsymbol{u}\|^2.$
        If
            (16)    $\|\bar{\boldsymbol{u}}_k\| \le \delta,$
        then EXIT with $\boldsymbol{x}^+ = \boldsymbol{x}$ since approximate Clarke stationarity is obtained.
    Step 3. Search direction. Compute the search direction $\boldsymbol{d}_{k+1} = -\bar{\boldsymbol{u}}_k / \|\bar{\boldsymbol{u}}_k\|$. If
            (17)    $f'(\boldsymbol{x}; \boldsymbol{d}_{k+1}) > -m_1 \|\bar{\boldsymbol{u}}_k\|,$
        then set $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ and $k = k + 1$ and go to Step 1.
    Step 4. Step length. Calculate the step length $\beta^* > 0$ from the formula
            (18)    $\beta^* = \arg\max \{\beta > 0 \mid f(\boldsymbol{x} + \beta \boldsymbol{d}_{k+1}) - f(\boldsymbol{x}) \le -m_1 \beta \|\bar{\boldsymbol{u}}_k\|\}.$
        If $\beta^* \ge \varepsilon$, then set $\boldsymbol{x}^+ = \boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$ and EXIT from the algorithm since a better iteration point has been found. Otherwise, set $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$ and $k = k + 1$ and go to Step 1.
```

Moreover, for any $\beta > \beta^*$ the inequality in (18) cannot hold, meaning that

$$f(\boldsymbol{x} + \beta \boldsymbol{d}_{k+1}) - f(\boldsymbol{x}) > -m_1 \beta \|\bar{\boldsymbol{u}}_k\|. \tag{20}$$

Next, we assume contrary to our claim that

$$f'(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}; \boldsymbol{d}_{k+1}) \le -\hat{m} \|\bar{\boldsymbol{u}}_k\| \quad \text{for some } \hat{m} > m_1.$$

Using the definition of the directional derivative of f, we can rewrite the previous inequality in the form

$$\lim_{t \downarrow 0} \frac{f(\boldsymbol{x} + (\beta^* + t)\boldsymbol{d}_{k+1}) - f(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1})}{t} \le -\hat{m} \|\bar{\boldsymbol{u}}_k\|.$$

This means that for each $\rho > 0$ there exists $t^* > 0$ such that

$$\frac{f(\boldsymbol{x} + (\beta^* + t^*)\boldsymbol{d}_{k+1}) - f(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1})}{t^*} \le -\hat{m} \|\bar{\boldsymbol{u}}_k\| + \rho.$$

Since $\hat{m} > m_1$ we can select $\rho = (\hat{m} - m_1)\|\bar{\boldsymbol{u}}_k\| > 0$. Then

$$f(\boldsymbol{x} + (\beta^* + t^*)\boldsymbol{d}_{k+1}) - f(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}) \le -m_1 t^* \|\bar{\boldsymbol{u}}_k\|$$

and combining this inequality with (19) gives

$$f(\boldsymbol{x}+(\beta^*+t^*)\boldsymbol{d}_{k+1})-f(\boldsymbol{x})\le -m_1(\beta^*+t^*)\|\bar{\boldsymbol{u}}_k\|.$$

This shows that the inequality in (18) holds for $\beta=\beta^*+t^*$. However, this contradicts (20) since $\beta^*+t^*>\beta^*$. □

THEOREM 4.11. *Let Assumption* A1 *be valid and assume that the set $\mathcal{F}_{\boldsymbol{x}}$ is compact. Algorithm* 1 *is terminated after at most*

$$N_{max}=\left\lceil \frac{\ln(\delta^2/L^2)}{\ln\left(1-\frac{(1-m_1)^2\delta^2}{8L^2}\right)}\right\rceil+1$$

iterations, where $\lceil\cdot\rceil$ is the ceiling of a number, $m_1\in(0,1)$, and $L>\delta$ is the Lipschitz constant of f at $\boldsymbol{x}\in\mathbb{R}^n$.

Proof. Algorithm 1 terminates if either stopping condition (16) 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 $C_k\subset\partial_\varepsilon^G f(\boldsymbol{x})$. Since $\bar{\boldsymbol{u}}_k$ is the solution to the quadratic programming problem (15) it follows from the necessary and sufficient condition for a minimum that $\bar{\boldsymbol{u}}_k^T\boldsymbol{u}\ge\|\bar{\boldsymbol{u}}_k\|^2$ for all $\boldsymbol{u}\in C_k$, which implies that

$$\boldsymbol{u}^T\boldsymbol{d}_{k+1}\le -\|\bar{\boldsymbol{u}}_k\|\quad\forall\boldsymbol{u}\in C_k. \tag{21}$$

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 condition (17) is satisfied and we obtain

$$f'(\tilde{\boldsymbol{x}};\boldsymbol{d}_{k+1})=f'(\boldsymbol{x};\boldsymbol{d}_{k+1})>-m_1\|\bar{\boldsymbol{u}}_k\|>-\hat{m}\|\bar{\boldsymbol{u}}_k\| \tag{22}$$

for $\hat{m}\in(m_1,1)$. 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 $\partial_\varepsilon^G f(\boldsymbol{x})$. Applying Lemma 4.10 we can guarantee that

$$f'(\tilde{\boldsymbol{x}};\boldsymbol{d}_{k+1})=f'(\boldsymbol{x}+\beta^*\boldsymbol{d}_{k+1};\boldsymbol{d}_{k+1})>-\hat{m}\|\bar{\boldsymbol{u}}_k\| \tag{23}$$

when $\hat{m}\in(m_1,1)$. Properties (iii) and (iv) of Theorem 4.8 mean 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 (22) and (23) implies that

$$\boldsymbol{\xi}_{k+1}^T\boldsymbol{d}_{k+1}>-\hat{m}\|\bar{\boldsymbol{u}}_k\|\quad\text{for all }\hat{m}\in(m_1,1). \tag{24}$$

Since $\hat{m}\in(m_1,1)$, inequalities (21) and (24) yield that $\boldsymbol{\xi}_{k+1}\notin C_k$. 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 (16) will be satisfied after a finite number of iterations if a new iteration point is never found. It is obvious that $t\boldsymbol{\xi}_{k+1}+(1-t)\bar{\boldsymbol{u}}_k\in C_{k+1}$ for any $t\in(0,1)$ and, thus,

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

Since a DC function f is LLC at any point, the Goldstein ε-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 $\boldsymbol{x}$, which is selected in such a way that $L > \delta$. 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 (24) we obtain $\boldsymbol{\xi}_{k+1}^T \bar{\boldsymbol{u}}_k < \hat{m}\|\bar{\boldsymbol{u}}_k\|^2$ since $\boldsymbol{d}_{k+1} = -\bar{\boldsymbol{u}}_k/\|\bar{\boldsymbol{u}}_k\|$ and therefore

$$\|\bar{\boldsymbol{u}}_{k+1}\|^2 \leq \|\bar{\boldsymbol{u}}_k\|^2 + 2t\bar{\boldsymbol{u}}_k^T(\boldsymbol{\xi}_{k+1} - \bar{\boldsymbol{u}}_k) + 4t^2L^2 < \|\bar{\boldsymbol{u}}_k\|^2 - 2t(1-\hat{m})\|\bar{\boldsymbol{u}}_k\|^2 + 4t^2L^2$$

for each $\hat{m} \in (m_1, 1)$. By selecting

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

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

$$\|\bar{\boldsymbol{u}}_{k+1}\|^2 < \|\bar{\boldsymbol{u}}_k\|^2 - \frac{(1-\hat{m})^2\|\bar{\boldsymbol{u}}_k\|^4}{4L^2} \quad \text{for all } \hat{m} \in (m_1, 1).$$

It is possible to select $\hat{m} \in (m_1, 1)$ in such a way that $2(1-\hat{m})^2 > (1-m_1)^2$. Using this selection in the previous inequality we obtain

$$\|\bar{\boldsymbol{u}}_{k+1}\|^2 < \|\bar{\boldsymbol{u}}_k\|^2 - \frac{(1-m_1)^2\|\bar{\boldsymbol{u}}_k\|^4}{8L^2}.$$

If stopping criterion (16) is never met, then $\|\bar{\boldsymbol{u}}_k\| > \delta$ for any $k > 0$, and we get

$$\|\bar{\boldsymbol{u}}_{k+1}\|^2 < \|\bar{\boldsymbol{u}}_k\|^2 \left(1 - \frac{(1-m_1)^2\|\bar{\boldsymbol{u}}_k\|^2}{8L^2}\right) < \|\bar{\boldsymbol{u}}_k\|^2 \left(1 - \frac{(1-m_1)^2\delta^2}{8L^2}\right).$$

It follows from this and $\|\bar{\boldsymbol{u}}_1\| \leq L$ that

$$\|\bar{\boldsymbol{u}}_k\|^2 < \|\bar{\boldsymbol{u}}_1\|^2 \left(1 - \frac{(1-m_1)^2\delta^2}{8L^2}\right)^{k-1} \leq L^2 \left(1 - \frac{(1-m_1)^2\delta^2}{8L^2}\right)^{k-1}.$$

Hence, $\|\bar{\boldsymbol{u}}_k\| \leq \delta$ is satisfied if

$$k \geq \left\lceil \frac{\ln(\delta^2/L^2)}{\ln\left(1 - \frac{(1-m_1)^2\delta^2}{8L^2}\right)} \right\rceil + 1.$$

This completes the proof. ∎

5. Double bundle method for DC functions. In this section, we describe the new proximal double bundle method DBDC for solving unconstrained DC minimization problems. The method DBDC is based on the combination of the PBDC method [17] and the new escape procedure Algorithm 1. The method PBDC utilizes the DC decomposition of the objective in the model construction and, due to this, the cutting plane model well describes the actual behaviour of f. In addition, the PBDC has fast convergence speed, but it guarantees only approximate criticality of the solutions. Therefore, the stopping condition can lead to "arbitrary" points where approximate Clarke stationarity is not satisfied. The aim of our hybridization is to utilize the PBDC method to obtain a promising candidate solution and then to use the escape procedure to guarantee approximate 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 but at the same time improves the quality of the solutions. The use of approximate Clarke stationarity as a stopping condition also enables us to escape from "arbitrary" solutions encountered when criticality is used.

5.1. Model for DC functions. We start by presenting the cutting plane model for nonsmooth DC functions, which is used to determine a search direction in the DBDC method. Since the main idea in the model construction is to utilize information about both DC components separately we assume that 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 $\boldsymbol{\xi}_2 \in \partial f_2(\boldsymbol{x})$ can be evaluated at each $\boldsymbol{x} \in \mathbb{R}^n$.

In order to take into account both the convex and concave behavior of f, we approximate the subdifferentials of both DC components f_1 and f_2. 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 f_i at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$ is denoted by

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

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 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 current iteration point $\boldsymbol{x}_k$. This model is the classical *cutting plane model* used in convex bundle methods (see, e.g., [20, 26, 28, 34]). For the DC component f_i, $i = 1, 2$, it is constructed using

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

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 it supports the epigraph of a convex function from below at every point, and linearization errors are nonnegative.

The approximation for f is obtained by combining the convex cutting plane models of f_1 and f_2. Thus, the *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 piecewise linear cutting plane model can be rewritten as

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

where $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{x}_k$ is the search direction at $\boldsymbol{x}_k$ and

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

are the piecewise affine functions associated with f_1 and f_2. Note that this cutting plane model is nonconvex and takes into account both the convex and concave behavior of f. Thus, we avoid the somewhat arbitrary downward shifting of first order expansions, and we need not use the so-called *subgradient locality measures* [19] commonly used in nonconvex bundle methods.

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

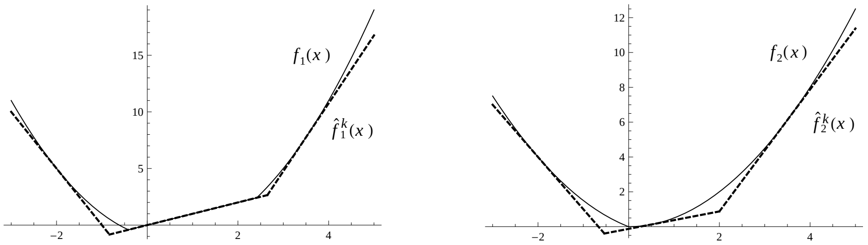

Fig. 3. *The convex cutting plane models $\hat{f}_1^k(x)$ and $\hat{f}_2^k(x)$ of the DC components $f_1(x)$ and $f_2(x)$.*

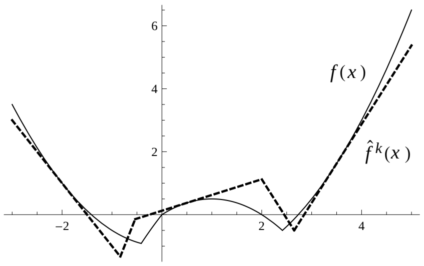

Fig. 4. *The nonconvex cutting plane model $\hat{f}^k(x)$ of the objective function $f(x)$.*

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. Thus, we need to add a quadratic stabilizing term into our model. The search direction $\boldsymbol{d}_t^k$ is obtained by globally solving the nonconvex DC minimization problem

$$(25)\qquad \begin{cases}\text{minimize} & P^k(\boldsymbol{d}) = \Delta_1^k(\boldsymbol{d}) + \Delta_2^k(\boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2\\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n,\end{cases}$$

where $t > 0$ is a proximity parameter used in most bundle methods. The quadratic term keeps our approximation local enough [26] since usually the farther away we are from $\boldsymbol{x}_k$ the more unreliable the cutting plane model becomes.

The term $\Delta_1^k(\boldsymbol{d}_t^k) + \Delta_2^k(\boldsymbol{d}_t^k)$ in problem (25) can be considered as a predicted descent for the actual decrease $f(\boldsymbol{x}_k + \boldsymbol{d}_t^k) - f(\boldsymbol{x}_k)$ in the objective function value. The following Lemma 5.1 supports this interpretation since it demonstrates that $\Delta_1^k(\boldsymbol{d}_t^k) + \Delta_2^k(\boldsymbol{d}_t^k)$ 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 f_1 and $-f_2$, respectively.

Lemma 5.1. *The following properties hold:*

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

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

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

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

In the next lemma, we establish a bound for the norm $\|\boldsymbol{d}_t^k\|$.

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

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

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. See the proof of Lemma 2 in [17]. □

The challenge in problem (25) 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 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 [21, 22, 32]. The main idea in the approach is based on the observation that the objective function P^k can be reformulated as

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

and then problem (25) 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 for each $i \in J_2^k$ separately solve the convex nonsmooth subproblem

$$(26)\qquad \begin{cases}\text{minimize} & P_i^k(\boldsymbol{d}) = \Delta_1^k(\boldsymbol{d}) - (\boldsymbol{\xi}_{2,i})^T\boldsymbol{d} + \alpha_{2,i}^k + \frac{1}{2t}\|\boldsymbol{d}\|^2\\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n,\end{cases}$$

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 (25) is obtained from

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

This means that we select the best solution among all subproblem minimizers. Moreover, for each $i \in J_2^k$ subproblem (26) can be reformulated as a convex quadratic programming problem:

$$(27)\qquad \begin{cases}\text{minimize} & v + \frac{1}{2t}\|\boldsymbol{d}\|^2\\ \text{subject to} & (\boldsymbol{\xi}_{1,j} - \boldsymbol{\xi}_{2,i})^T\boldsymbol{d} - (\alpha_{1,j}^k - \alpha_{2,i}^k) \le v \quad \text{for all } j \in J_1^k,\\ & v \in \mathbb{R},\ \boldsymbol{d} \in \mathbb{R}^n.\end{cases}$$

Alternatively, instead of (27) it is possible to solve its quadratic dual problem [28].

5.3. Algorithm.

We now describe the proximal double bundle algorithm DBDC for unconstrained DC minimization. This method combines the new escape procedure (Algorithm 1) with the proximal bundle method PBDC [17]. The following assumption is required to prove the convergence of the DBDC.

Assumption A2. The set $\mathcal{F}_0 = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \le f(\boldsymbol{x}_0)\}$ is compact for a starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$.

This assumption is often made in bundle methods and it is not restrictive. Moreover, this assumption guarantees that set (14) is always compact whenever Algorithm 1 is executed during the DBDC method.

To make the presentation clear we divide our method into two parts. The first part, Algorithm 2, illustrates mainly the outline of the DBDC while a more significant role is provided by Algorithm 3, which presents the main iteration. Algorithm 3 consists of a sequence of steps where the current iteration point $\boldsymbol{x}_k$ remains unchanged. However, whenever we exit from this algorithm we have either confirmed approximate Clarke stationarity or found a new iteration point decreasing the value of the objective. To guarantee approximate Clarke stationarity, the main iteration utilizes Algorithm 1 whenever a promising candidate solution is found. If in this verification process $\boldsymbol{x}_k$ satisfies the required stopping conditions, then the DBDC method terminates with $\boldsymbol{x}_k$ as the final solution.

Algorithm 2 Double bundle method for unconstrained DC optimization (DBDC).

Data: The starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$, 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.* Compute $f_1(\boldsymbol{x}_0)$ and $f_2(\boldsymbol{x}_0)$. Set $\boldsymbol{y}_1 = \boldsymbol{x}_0$ and $k = 0$. Calculate $\boldsymbol{\xi}_{1,1} = \boldsymbol{\xi}_1(\boldsymbol{x}_0) \in \partial f_1(\boldsymbol{y}_1)$, $\boldsymbol{\xi}_{2,1} = \boldsymbol{\xi}_2(\boldsymbol{x}_0) \in \partial f_2(\boldsymbol{y}_1)$, and set $\alpha_{1,1}^k = \alpha_{2,1}^k = 0$. Initialize $\mathcal{B}_1^k = \{(\boldsymbol{\xi}_{1,1}, \alpha_{1,1}^k)\}$ and $\mathcal{B}_2^k = \{(\boldsymbol{\xi}_{2,1}, \alpha_{2,1}^k)\}$.

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

Step 2. *Bundle update.* Compute $f_i(\boldsymbol{x}_{k+1})$ and $\boldsymbol{\xi}_i(\boldsymbol{x}_{k+1}) \in \partial f_i(\boldsymbol{x}_{k+1})$ for $i = 1, 2$. Select $\mathcal{B}_1^{k+1} \subseteq \mathcal{B}_1^k$ and $\mathcal{B}_2^{k+1} \subseteq \mathcal{B}_2^k$ and update

$$\alpha_{i,j}^{k+1} = \alpha_{i,j}^k + f_i(\boldsymbol{x}_{k+1}) - f_i(\boldsymbol{x}_k) - (\boldsymbol{\xi}_{i,j})^T(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \tag{28}$$

for all $i = 1, 2$ and $j \in J_i^{k+1}$. Insert the element $(\boldsymbol{\xi}_1(\boldsymbol{x}_{k+1}), 0)$ into $\mathcal{B}_1^{k+1}$ and $(\boldsymbol{\xi}_2(\boldsymbol{x}_{k+1}), 0)$ into $\mathcal{B}_2^{k+1}$. Set $k = k + 1$ and go to Step 1.

Remark 5.3. In the bundles $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$, it suffices to store only subgradients $\boldsymbol{\xi}_{i,j} \in \partial f_i(\boldsymbol{y}_j)$ and linearization errors $\alpha_{i,j}^k$. This is due to the fact that the linearization errors can be updated by using formula (28) in Algorithm 2 and, therefore, the new values for the next iteration are easily obtained whenever a new iteration point $\boldsymbol{x}_{k+1}$ is found. Thus, we do not need to store the points $\boldsymbol{y}_j$ or the values $f_i(\boldsymbol{y}_j)$.

Remark 5.4. In Step 2 of Algorithm 2, the bundles $\mathcal{B}_1^{k+1}$ and $\mathcal{B}_2^{k+1}$ can be chosen separately and without any restrictions. Thus, every element stored can also be deleted at this point. Regardless of this, $\mathcal{B}_1^{k+1}$ 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 ensures that the bundles are never empty when we start a new main iteration.

Next, we discuss Algorithm 3 being the core of the DBDC method. To simplify notation, we have omitted the index k except for in $\boldsymbol{x}_k \in \mathbb{R}^n$ 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 $\boldsymbol{\xi}_2(\boldsymbol{x}_k) \in \partial f_2(\boldsymbol{x}_k)$.

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 $j^* = \arg\max_{j \in J_2}\{\|\boldsymbol{\xi}_{2,j}\|\}$, set $\boldsymbol{\xi}_{2,\max} = \boldsymbol{\xi}_{2,j^*}$, and initialize the parameters

$$t_{\min} = r \cdot \frac{\varepsilon_1}{2\big(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|\big)} \tag{29}$$

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

Step 2. *Search direction.* Compute the search direction $\boldsymbol{d}_t$ by solving problem (25). 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.

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

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) \le m_2 \Big(\Delta_1(\boldsymbol{d}_t) + \Delta_2(\boldsymbol{d}_t)\Big), \tag{30}$$

then choose $\boldsymbol{x}_{k+1} = \boldsymbol{y}$ and EXIT.

Step 5. *Bundle update.*

(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 to Step 2.

(b) Otherwise, if

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) \ge -m_2 \Big(\Delta_1(\boldsymbol{d}_t) + \Delta_2(\boldsymbol{d}_t)\Big), \tag{31}$$

then set $t = t - c(t - t_{\min})$. Compute $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{y})$, $\boldsymbol{\xi}_2 \in \partial f_2(\boldsymbol{y})$, and set $\alpha_1 = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}) + \boldsymbol{\xi}_1^T \boldsymbol{d}_t$ and $\alpha_2 = f_2(\boldsymbol{x}_k) - f_2(\boldsymbol{y}) + \boldsymbol{\xi}_2^T \boldsymbol{d}_t$. Insert the element $(\boldsymbol{\xi}_1, \alpha_1)$ into $\mathcal{B}_1$ and, if $\Delta_2(\boldsymbol{d}_t) \ge 0$, then insert the element $(\boldsymbol{\xi}_2, \alpha_2)$ into $\mathcal{B}_2$.

Step 6. *Parameter update.* If $\|\boldsymbol{\xi}_2\| > \|\boldsymbol{\xi}_{2,\max}\|$, then set $\boldsymbol{\xi}_{2,\max} = \boldsymbol{\xi}_2$ and update $t_{\min}$ using formula (29). Go to Step 2.

The search direction problem in Step 2 of Algorithm 3 is the most time-consuming part since during each iteration we need to solve $|J_2|$ convex subproblems. However, after solving all subproblems the global solution of the original nonconvex problem can be easily obtained by choosing the best solution among the subproblem minimizers. In addition, the user can control the number of solved subproblems since the size of the bundle $\mathcal{B}_2$ can be always limited with the maximum number of stored subgradients $J_{\max} \ge 1$. 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 Algorithm 3. 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 in $\mathcal{B}_2$.

Remark 5.5. In Step 5(b) of Algorithm 3, condition (31) is similar to (30) but now instead of the descent we test if the increase 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 5.6. The purpose of Step 5(a) of Algorithm 3 is to guarantee that the points used to constitute the elements inserted into the bundles are on the set $\mathcal{F}_{\varepsilon_1} = \{\boldsymbol{x} \in \mathbb{R}^n \mid d(\boldsymbol{x}, \mathcal{F}_0) \leq \varepsilon_1\}$, where $\varepsilon_1 > 0$ is selected and

$$d(\boldsymbol{x}, \mathcal{F}_0) = \inf\{\|\boldsymbol{x} - \boldsymbol{z}\| \mid \boldsymbol{z} \in \mathcal{F}_0\}.$$

Moreover, all iteration points $\boldsymbol{x}_k$ are on $\mathcal{F}_{\varepsilon_1}$ 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 $\mathcal{F}_{\varepsilon_1}$, respectively. This implies that

$$\|\boldsymbol{\xi}_1\| \leq L_1 \text{ for each element on } \mathcal{B}_1 \text{ and } \|\boldsymbol{\xi}_2\| \leq L_2 \text{ for each element on } \mathcal{B}_2\ . \tag{32}$$

From this we can 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 the condition in Step 0 does not hold, and thus

$$t_{\max} \leq \bar{t}_{\max} = Rr\varepsilon_1/2\delta < \infty. \tag{33}$$

5.4. Convergence. In this section, we prove the convergence of the DBDC method. We especially show that the method terminates after a finite number of steps and the solution obtained is approximate Clarke stationary.

In Theorem 4.11, we have already proved the finite termination of Algorithm 1, and this result holds under Assumptions A1 and A2 during the execution of the DBDC. Next, we show that Algorithm 3 stops after a finite number of iterations. After that, we are ready to present the convergence result for DBDC Algorithm 2.

We begin by stating the following auxiliary lemma.

LEMMA 5.7. *If condition* (30) *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 $\boldsymbol{y} = \boldsymbol{x}_k + \boldsymbol{d}_t$, $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{y})$ *and* $\alpha_1 = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}) + \boldsymbol{\xi}_1^T \boldsymbol{d}_t$.

Proof. The proof is similar to the proof of Lemma 3 in [17]. ☐

Now we prove the termination for Algorithm 3. The proof follows the guidelines of [17].

THEOREM 5.8. *Let 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. Assume the contrary, that is, 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 means that Step 3 is never executed since it would stop the current main iteration and, therefore, for each $i \in \mathcal{I}$ we have $\|\boldsymbol{d}_t^{(i)}\| \geq \delta$.

First, we notice that Step 5(a) cannot occur infinitely many times. Otherwise, the parameter t would be decreased infinitely many times and converge to $t_{\min}$ since $t_{\min}$ is both bounded and monotonically decreasing. Moreover, $t_{\min}$ is selected to be 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 steps t also 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, property (32) and the rule $t \in [t_{\min}, t_{\max}]$. Therefore, there exists a subsequence $\{\boldsymbol{d}_t^{(i)}\}_{i\in\mathcal{I}'\subseteq\mathcal{I}}$ converging to a limit $\hat{\boldsymbol{d}}$. From Remark 5.6 we also know that all the points $\boldsymbol{y}_j$ and $\boldsymbol{x}_k$ belong to the set $\mathcal{F}_{\varepsilon_1}$. Moreover, Assumption A2 implies that $\mathcal{F}_{\varepsilon_1}$ is compact and, thus, there exists $K > 0$ such that $\|\boldsymbol{x}_k - \boldsymbol{y}_j\| \leq K$ for all $\boldsymbol{y}_j$ on $\mathcal{B}_1$. This together with (32) yields

$$|\alpha_{1,j}| \leq |f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}_j)| + \|\boldsymbol{\xi}_{1,j}\|\|\boldsymbol{x}_k - \boldsymbol{y}_j\| \leq L_1\|\boldsymbol{x}_k - \boldsymbol{y}_j\| + L_1K \leq 2L_1K$$

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 for $\boldsymbol{y}_j$ on $\mathcal{B}_2$ and, thus, all subgradients and linearization errors are bounded.

From the boundedness results we know that the corresponding subsequences

$$\{\Delta_1(\boldsymbol{d}_t^{(i)})\}_{i\in\mathcal{I}'\subseteq\mathcal{I}} \quad \text{and} \quad \{\Delta_2(\boldsymbol{d}_t^{(i)})\}_{i\in\mathcal{I}'\subseteq\mathcal{I}}$$

are also 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 property (iii) of Lemma 5.1, we have

$$\Delta_1\big(\boldsymbol{d}_t^{(i)}\big) + \Delta_2\big(\boldsymbol{d}_t^{(i)}\big) \leq -\frac{1}{2t_i}\big\|\boldsymbol{d}_t^{(i)}\big\|^2 \leq -\frac{\delta^2}{2t_i} < 0 \quad \text{for all } i \in \mathcal{I} \tag{34}$$

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

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

where $\hat{t} = \lim_{i\to\infty} t_i$ and $\hat{t} > 0$. Remark 5.6 guarantees that $\hat{t}$ is strictly positive since the sequence $\{t_i\}$ is bounded from below with a positive lower bound. 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

$$\alpha_{1,r} = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{x}_k + \boldsymbol{d}_t^{(r)}) + (\boldsymbol{\xi}_{1,r})^T\boldsymbol{d}_t^{(r)} \quad \text{with } \boldsymbol{\xi}_{1,r} \in \partial f_1(\boldsymbol{x}_k + \boldsymbol{d}_t^{(r)}).$$

From Lemma 5.7 and the definition of $\Delta_1(\boldsymbol{d})$ we obtain

$$(\boldsymbol{\xi}_{1,r})^T\boldsymbol{d}_t^{(r)} - \alpha_{1,r} > m_2\Delta_1\big(\boldsymbol{d}_t^{(r)}\big) + (m_2 - 1)\Delta_2\big(\boldsymbol{d}_t^{(r)}\big) \tag{35}$$

and

$$\Delta_1\big(\boldsymbol{d}_t^{(s)}\big) \geq (\boldsymbol{\xi}_{1,r})^T\boldsymbol{d}_t^{(s)} - \alpha_{1,r}. \tag{36}$$

Finally, combining (35) and (36) gives

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

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 termination for the bundle algorithm DBDC. The proof of this result reveals similar trends to Theorem 6 in [17]. In addition, the DBDC is globally convergent if 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 an approximate Clarke stationary point as a final solution $\boldsymbol{x}^*$ regardless of the starting point used.

THEOREM 5.9. *Let Assumptions* A1 *and* A2 *be valid. For any* $\delta \in (0,1)$ *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*

$$\|\boldsymbol{\xi}^*\| \le \delta \quad \textit{with } \boldsymbol{\xi}^* \in \partial f_\varepsilon^G(\boldsymbol{x}^*).$$

Proof. The termination of the DBDC can happen only if the stopping condition tested at Step 2 of Algorithm 1 is satisfied proving the approximate Clarke stationarity. Next, we prove that if the stopping condition is never reached, then the objective function f is not bounded from below. This leads to a contradiction since f is bounded from below as a consequence of Lipschitz continuity and Assumption A2.

We start by supposing 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 4.11 and 5.8 guarantee that in the main iteration we always find a new iteration point $\boldsymbol{x}_{k+1}$ after a finite number of steps. This point is obtained either from Step 3 or Step 4. If Step 3 provides us with $\boldsymbol{x}_{k+1}$, then from Algorithm 1 we know that the direction of a sufficient descent has been found and the step size $\beta^* \ge \varepsilon$. Thus,

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k-1}) \le -m_1\varepsilon\delta < 0.$$

In the other case, the stopping condition (30) is fulfilled providing

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k-1}) \le m_2\Big(\Delta_1\big(\boldsymbol{d}_t^{(k)}\big) + \Delta_2\big(\boldsymbol{d}_t^{(k)}\big)\Big).$$

From (34) and (33), we notice that

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k-1}) \le -\frac{m_2\delta^2}{2\bar{t}_{\max}} < 0.$$

Therefore, after each round of Algorithm 2 we can guarantee that

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k-1}) \le -\sigma < 0 \quad \text{for all } k \in \mathcal{K},$$

where $\sigma = \min\{m_1\varepsilon\delta, m_2\delta^2/(2\bar{t}_{\max})\} > 0$ and summing up the first k of the above inequalities gives

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_0) \le -k\sigma.$$

Letting $k \to \infty$ we have

$$\lim_{k\to\infty} f(\boldsymbol{x}_k) - f(\boldsymbol{x}_0) \le -\infty,$$

which means that f is not bounded below. This is a contradiction. □

6. Numerical results.

To verify the practical efficiency of the DBDC method, we have applied it to some academic test problems with nonsmooth DC objectives. In order to compare the results, we use two proximal bundle algorithms PBDC [17] and MPBNGC [28] for nonsmooth optimization. The PBDC is the predecessor of the DBDC and it also utilizes the DC decomposition of the objective. The MPBNGC is designed for a general nonsmooth nonconvex problem and does not exploit any specific structure of the objective. This method is also used very often as a benchmark in tests. In [17], the PBDC is also compared to DCA [22, 24], the truncated codifferential method [5], and two other bundle methods [10, 11]. Since the methods PBDC and MPBNGC performed better than other methods in most of the cases, we include only the PBDC and MPBNGC in our comparisons.

The algorithm DBDC requires an unbounded storage for the bundle $\mathcal{B}_1$, but this is not possible in practice. Thus, the implementation of the DBDC slightly differs from Algorithm 2 since we use 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.

The execution of Algorithm 1 is continued if in Step 4 the step size satisfies $\beta^* < \varepsilon$. However, in practice, it might lead to numerical difficulties since the parameter ε is typically selected to be very small. Therefore, in the implementation of the DBDC, the execution of Algorithm 1 is stopped with $\boldsymbol{x}^* = \boldsymbol{x}$ as the final solution if $\beta^* < \varepsilon$.

DBDC is implemented in double precision Fortran 95 and it uses the subroutine PLQDF1 [25] to solve the quadratic programming problems (15) and (27). 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 also use the subroutine PLQDF1 [25] to solve the quadratic direction-finding problems. All codes are compiled using f95, the Fortran 95 compiler, and tests are performed under the Linux Ubuntu system. Fortran source codes of all methods can be downloaded from http://napsu.karmitsa.fi/nsosoftware/.

The codes have been tested on a set of 16 academic test problems. Problems 1–10 are from [17] whereas Problems 11–16 are introduced in [18]. In addition, each problem is nonconvex and its DC decomposition is given. 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

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

the increase parameter $R = 10^7$; and the descent parameters $m_1 = 0.01$ and $m_2 = 0.2$. The size of $\mathcal{B}_1$ is set to $\min\{n+5, 1000\}$ and the size of $\mathcal{B}_2$ is 3. The maximum size of the set C_k in Algorithm 1 is restricted to $2n$. In PBDC, we use the default settings [17].

Furthermore, in MPBNGC we use mostly the default values [27], 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 approximately the same accuracy in solutions.

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

Example 6.1. Consider the function f presented in Example 3.2. If $x_0 \in \mathbb{R}$ is selected from $A = \{x \in \mathbb{R} \,|\, x < -2 \text{ or } x > 1\}$, then the subgradients of DC components are $\xi_1(x_0) = 2x_0$ and $\xi_2(x_0) = x_0$. Thus, $\mathcal{B}_1^0 = \{(\xi_1(x_0), 0)\}$ and $\mathcal{B}_2^0 = \{(\xi_2(x_0), 0)\}$ in both DBDC and PBDC. In addition, both solvers use the same direction-finding problem (25), which during the first iteration 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 problem is $d_t = -tx_0$, and with the selection $t = 1$ we obtain a new point $y = x_0 + d_t = x_0 - x_0 = 0$. Moreover, in both solvers we use the same descent test (30) and, if $m_2 \in (0, 1/2)$, then the objective function decreases enough, that is,

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

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 f_1 and f_2 at x_1 are $\xi_1(x_1) = \xi_2(x_1) = 0$. In PBDC, this leads to the fulfillment of criticality and the algorithm is terminated with the final solution $x^* = 0$. As we have already seen, this solution gives nothing interesting for f. In the DBDC, the selection $\xi_1(x_1) = \xi_2(x_1) = 0$ leads to Algorithm 1, where we first calculate, utilizing DC components, a subgradient ξ for f using either a direction $d_1 = 1$ or $d_1 = -1$. Regardless of this selection, we obtain $\xi = 1$ and $d_2 = -1$. In addition, condition (17) is satisfied since

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

and $m_1 \in (0, 1)$. This proves that d_2 is a descent direction yielding a better iteration point. Thus, the DBDC method bypasses x_1 and does not stop at the problematic critical point.

The results of our numerical experiments are presented in Tables 1 and 2 where we use the following notation:

- "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 best value of the objective function obtained by the algorithm.

In addition, we have $n_f = n_{\boldsymbol{\xi}}$ in MPBNGC. Moreover, in DBDC we separately give the function and subgradient evaluations used in Algorithm 1 guaranteeing approximate Clarke stationarity and for this algorithm $n_{\boldsymbol{\xi}_1} = n_{\boldsymbol{\xi}_2}$. In the solvers DBDC and PBDC, we report the subgradient evaluations separately for DC components. Therefore, to obtain comparable results with MPBNGC we calculate combined values $n_{\boldsymbol{\xi}_1} + n_{\boldsymbol{\xi}_2}$ and

TABLE 1
Summary of numerical results with DBDC, PBDC, *and* MPBNGC.

		DBDC							PBDC					MPBNGC		
		Main it.			Algorithm 1											
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
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 global minimizer of the objective function f is not found.

use 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 in only 5 cases out of 53. These five cases seem to be quite difficult ones since the other solvers also mostly find a local minimizer among them. In addition, both PBDC and MPBNGC are quite reliable for finding global minimizers: PBDC provides a local solution in only 9 cases and MPBNGC in 10 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 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 almost the same amount of evaluations, and they are more efficient than MPBNGC. However, in Problems 4–6 DBDC often needs slightly more computational effort due to a stronger stopping condition verification procedure but this only affects CPU time in 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 rather small except for in Problems 13 and 15. Moreover, 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 can detect both easy and hard problems. For example, in Problems 1–11 and 16 all solvers are equally fast if we leave out of consideration Problem 4 ($n = 250$), where MPBNGC is faster than the other methods and Problem 4 ($n = 500$), where PBDC is the fastest. Moreover, DBDC is fastest for Problem 14, whereas the CPU times of PBDC and MPBNGC have completely different magnitudes when we increase the dimension of this problem. However, MPBNGC outperforms the other solvers in Problems 12 ($n = 100$) and 15 ($n = 100$).

All in all, the numerical results confirm that DBDC is efficient in solving nonsmooth DC minimization problems. Compared to PBDC the solver DBDC sometimes needs more function and subgradient evaluations, but at the same time it guarantees approximate Clarke stationarity, which is a stronger stopping condition than the criticality used in PBDC. Thus, slightly more computational effort cannot be seen as a real disadvantage for DBDC. In addition, in some problems DBDC is clearly more efficient than PBDC. Moreover, DBDC has the best ability to find a 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 explicitly utilizing the DC decomposition of the objective. The novelty of the DBDC is a new escape procedure guaranteeing approximate Clarke stationarity using only the information about the DC components of the objective. This way the DBDC method can exploit the DC structure through the algorithm and avoid the problematic features of criticality, which is the stopping condition typically used in DC optimization algorithms. In addition, the finite termination of the DBDC method is proved under mild assumptions requiring that the subdifferentials of DC components are polytopes.

The numerical results reported confirm that the DBDC is efficient in solving nonsmooth DC programming problems. Moreover, although the DBDC method is only a local solution method, it nearly always found the global minimizers of the

TABLE 2
Summary of numerical results with DBDC, PBDC, *and* MPBNGC *(cont.).*

		DBDC								PBDC					MPBNGC		
		Main it.			Algorithm 1												
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	$1.3956 \cdot 10^{-10}$	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 global minimizer of the objective function f is not found.

tested problems. Therefore, we can conclude that the DBDC is a good alternative for existing nonconvex bundle methods when the DC representation of the objective can be formulated.

Acknowledgments. We thank Outi Montonen for useful discussions. We also thank two anonymous referees for their valuable comments and suggestions, which helped us to improve the paper.

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, Switzerland, 2014.

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

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

[6] A. M. Bagirov and J. Ugon, *Nonsmooth DC programming approach to clusterwise linear regression: Optimality conditions and algorithms*, Optim. Methods Softw., 33 (2018), pp. 194–219.

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

[8] V. F. Demyanov and A. M. Rubinov, *Constructive Nonsmooth Analysis*, Approx. and Optim. 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*, in Generalized Convexity and Generalized Monotonicity, Lecture Notes in Econom. and Math. Systems 502, N. Hadjisavvas, J. E. Martínez-Legaz, and J. P. Penot, eds., Springer, Berlin, Heidelberg, 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*, in Convexity and Duality in Optimization, Lecture Notes in Econom. and Math. Systems 256, J. Ponstein, ed., Springer, Berlin, Heidelberg, 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., 68 (2017), pp. 501–535.

[18] K. Joki, A. M. Bagirov, N. Karmitsa, M. M. Mäkelä, and S. Taheri, *Double Bundle Method for Nonsmooth DC Optimization*, Technical report 1173, Turku Centre for Computer Science (TUCS), Turku, 2017.

[19] K. C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, Lecture Notes in Math. 1133, Springer, Berlin, 1985.

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

[21] 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.

[22] 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.

[23] H. A. Le Thi and T. Pham Dinh, *DC programming and DCA for nonconvex optimization: Theory, algorithms and applications*, in Proceedings of MAMERN 2009: 3rd International Conference on Approximation Methods and Numerical Modelling in Environment and Natural Resources, Pau, France, 2009; available at http://lma.univ-pau.fr/meet/mamern09/en/Lethi-MAMERN09.pdf.

[24] 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.

[25] 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.

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

[27] 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.

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

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

[30] 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.

[31] 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.

[32] 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.

[33] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[34] 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.

[35] 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.

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

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