algorithms

Article

Bundle Enrichment Method for Nonsmooth Difference of Convex Programming Problems

Manlio Gaudioso [1,*], Sona Taheri [2], Adil M. Bagirov [3] and Napsu Karmitsa [4]

[1] DIMES (Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica), Università della Calabria, 87036 Rende, CS, Italy
[2] School of Mathematical Sciences, RMIT University, Melbourne 3000, Australia; sona.taheri@rmit.edu.au
[3] Centre for Smart Analytics, Institute of Innovation, Science and Sustainability, Federation University Australia, Ballarat 3350, Australia; a.bagirov@federation.edu.au
[4] Department of Computing, University of Turku, FI-20014 Turku, Finland; napsu@karmitsa.fi
* Correspondence: manlio.gaudioso@unical.it

Abstract: The Bundle Enrichment Method (BEM-DC) is introduced for solving nonsmooth difference of convex (DC) programming problems. The novelty of the method consists of the dynamic management of the bundle. More specifically, a DC model, being the difference of two convex piecewise affine functions, is formulated. The (global) minimization of the model is tackled by solving a set of convex problems whose cardinality depends on the number of linearizations adopted to approximate the second DC component function. The new bundle management policy distributes the information coming from previous iterations to separately model the DC components of the objective function. Such a distribution is driven by the sign of linearization errors. If the displacement suggested by the model minimization provides no sufficient decrease of the objective function, then the temporary enrichment of the cutting plane approximation of just the first DC component function takes place until either the termination of the algorithm is certified or a sufficient decrease is achieved. The convergence of the BEM-DC method is studied, and computational results on a set of academic test problems with nonsmooth DC objective functions are provided.

Keywords: DC optimization; nonconvex nonsmooth optimization; cutting plane; bundle method

Citation: Gaudioso, M.; Taheri, S.; Bagirov, A.M.; Karmitsa, N. Bundle Enrichment Method for Nonsmooth DC Programming. *Algorithms* **2023**, *16*, 394. https://doi.org/10.3390/a16080394

Academic Editor: Binlin Zhang

Received: 2 August 2023
Revised: 18 August 2023
Accepted: 19 August 2023
Published: 21 August 2023

1. Introduction

Optimization approaches are essential for solving a wide range of practical problems. There are various problems based on these approaches including unconstrained and constrained problems, problems with linear and nonlinear as well as smooth and nonsmooth objective and/or constraint functions, and problems with continuous and integer decision variables [1]. The majority of optimization problems from applications have special structures (for example, convexity) which can be exploited to design efficient and accurate methods for their solutions. Difference of convex (DC) optimization problems are among such problems, where the objective and/or constraint functions are represented as a difference of two convex functions.

In this research work, we consider the unconstrained nonsmooth DC programming problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1}$$

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

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

Here, $f_1 - f_2$ is called a *DC representation (decomposition)* of f while f_1 and f_2 are *DC components* of the function f. The DC components f_1 and f_2 are, in general, nonsmooth [2–5].

Nonsmooth DC programming is an important subclass of DC optimization problems [6], and many practical problems are modeled as a DC programming problem. They include the bridge location problem, the design centering problem [7], the packing problem [8], the production–transportation planning problem [9], the location planning problem [10], the edge detection problem [11], the conic programming problem [12], cluster analysis [13], and regression analysis [14]. Recently, DC optimization problems with uncertain data has become an interesting topic, and the results from robust optimization, in particular those obtained in [15–17], can be extended to robust DC optimization.

DC optimization problems have been studied in the context of both local and global problems, and various methods have been developed for solving these problems globally [7,18]. To the best of our knowledge, the first local search algorithm for solving DC optimization problems is the difference of convex algorithm (DCA) introduced in [19] and further explored, for example, in [8,20]. Since then, the development of local DC optimization methods for solving Problem (1) has attracted remarkable scholarly attention. Next, we provide a short description of such methods and give references for more details. These methods can be classified into three categories:

- The first category consists of the DCA and its modifications. The basic idea of the DCA is to linearize the concave part $-f_2$ around the current iterate by using its subgradient while keeping the convex part f_1 as it is in the minimization process. To improve the convergence of the DCA, various modifications have been developed, for instance, in [21–23]. The boosted DC algorithm (BDCA), proposed in [21,22], accelerates the convergence of the DCA by using an additional line search step. The inertial DCA, introduced in [23], defines trial points whose sequence of functional values is not necessarily monotonically decreasing. This controls the algorithm from converging to a critical point that is not d-stationary. In [24], the BDCA is combined with a simple derivative-free optimization method. This allows one to force the d-stationarity (lack of descent direction) at the obtained point. To avoid the difficulty of solving the DCA's subproblem, in [25], the first DC component is replaced with a convex model, and the second DC component is used without any approximation;
- The methods in the second category, which we refer to as DC-Bundle, are various extensions of the bundle methods for convex problems. The piecewise linear underestimates or subgradients of both DC components are utilized to compute search directions. The methods include the codifferential method [26], the proximal bundle method with the concave affine model [27,28], the proximal bundle method for DC optimization [29,30], the proximal point algorithm [31], the proximal linearized algorithm [32], the double bundle method [33], and the nonlinearly constrained DC bundle method [34];
- The methods in the third category are those that use the convex piecewise linear model of the first DC component and one subgradient of the second DC component to compute search directions at each iteration [35,36]. They differ from those in the first category as at each iteration of these methods, the model of the first DC component is updated and the new subgradient of the second component is calculated. They also differ from those in the second category as they use only one subgradient of the second DC component at each iteration whereas the DC-Bundle methods may use more than one subgradient of this component to build its piecewise linear model. Note that some overlapping is unavoidable when classifying methods into different categories: for instance, the proximal bundle method for DC optimization [30] may be classified into both the second and the third categories. The other methods in the third category include the aggregate subgradient method for DC optimization [35] and the augmented subgradient method [36]. In the former method, the aggregate subgradient of the first DC component and one subgradient of the second component are utilized to compute search directions. In the latter method, augmented subgradients of convex functions are defined and used to model the first DC component. Then, this model

and one subgradient of the second DC component are used for computing search directions.

The proposed BEM-DC method belongs to the DC-bundle family whose main features, with respect to the DCA, are

- developing a model function based on cutting-plane approximations of f_1 and, possibly, f_2;
- adding a regularization term (typically a proximity one) to the model for stabilization purposes.

We assume the familiarity of the reader with the basic notions of bundle methods. We refer to the following books and articles from the vast literature available [37–45]. In our iterative scheme, we adopt two cutting-plane approximations (based, as usual in bundle methods, on information coming from previous iterations) for f_1 and f_2, respectively. Consequently, we have a model which is still DC, being the difference of two convex piecewise affine functions. Similarly to [29,33], we tackle the (global) minimization of the model by solving a set of convex problems whose cardinality depends on the number of linearizations adopted to approximate the function f_2. The novelty of the approach consists of the dynamic management of the bundle. This enables us to achieve a parsimonious use of the information available, in view of reducing the computational burden to minimize the model function at each iteration. The adopted strategies are based on the following:

- The information coming from the previous iterates contributes to the approximation of exactly one of the DC components f_1 and f_2, depending on the sign of the linearization error relative to the current iterate. During the iterative process, every time a new point is generated, subgradients of both component functions f_1 and f_2 are calculated, but only one of them will enter the calculation of the search direction: the choice being driven, at each of the successive iterations, by the sign of the linearization error with respect to the current point. Such sign may change at each iteration. Therefore, we define a *dynamic* bundling strategy, which implies a consistent reduction (approximately one-half) of the size of the auxiliary problems to be solved;
- If the displacement suggested by the model minimization provides no sufficient decrease of the objective function (the null step in bundle parlance), the temporary enrichment exclusively of the cutting-plane approximation of f_1 takes place.

Note that a significant difference between the proposed approach with those introduced in [29,33] is in the reduction of the size of the bundle of the function f_2. This has a strong impact on the solution of the auxiliary problem to be solved at each iteration.

The structure of the paper is organized as follows. Section 2 provides necessary notations and some preliminaries. Section 3 presents the new model function formulation. Section 4 describes the new method, and its convergence is discussed in Section 5. Section 6 reports the results of numerical experiments. Section 7 provides some concluding remarks.

2. Notations and Background

First, we provide some notations and definitions that we will use throughout the paper. The inner product in $\mathbb{R}^n$ is $\langle u, v\rangle = \sum_{i=1}^n u_i v_i$, and $\|\cdot\|$ is the associated norm. For $x \in \mathbb{R}^n$ and $\varepsilon > 0$, $B(x;\varepsilon)$ is an open ball of the radius $\varepsilon > 0$ centered at x. The function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous on $\mathbb{R}^n$ if for every $x \in \mathbb{R}^n$, there exists a Lipschitz constant $L > 0$ and $\varepsilon > 0$ such that $|f(y) - f(z)| \leq L\|y - z\|$ for all $y, z \in B(x;\varepsilon)$.

For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, its subdifferential at a point $x \in \mathbb{R}^n$ is [38,41]

$$\partial f(x) = \left\{ \xi \in \mathbb{R}^n : \ f(y) - f(x) \geq \langle \xi, y - x\rangle \quad \text{for all } y \in \mathbb{R}^n \right\},$$

and for $\varepsilon > 0$, its ε-subdifferential is

$$\partial_\varepsilon f(x) = \left\{ \xi_\varepsilon \in \mathbb{R}^n : \ f(y) - f(x) \geq \langle \xi_\varepsilon, y - x\rangle - \varepsilon \quad \text{for all } y \in \mathbb{R}^n \right\}.$$

Each vector $\xi \in \partial f(x)$ $\big(\xi_\varepsilon \in \partial_\varepsilon f(x)\big)$ is called a subgradient (ε-subgradient) of f at x.

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

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

and a point $\bar{x}^* \in \mathbb{R}^n$ is said to be an ε-critical point if [31]

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

Next, we recall the basic idea of the standard cutting-plane model for any convex nonsmooth function f. Let $x_k \in \mathbb{R}^n$ be the current iteration point, $x_j \in \mathbb{R}^n$ be some auxiliary points (from past iterations), $\xi_j \in \partial f(x_j)$ be the subgradients of the function f computed at the point $x_j \in \mathbb{R}^n$ for $j \in J^k$, and J^k is a nonempty subset of $\{1, \ldots, k\}$. The cutting-plane model for the function f can be given by

$$\hat{f}^k(x) = \max_{j \in J^k} \bar{f}_j(x),$$

where $\bar{f}_j(x) = f(x_j) + \langle \xi_j, x - x_j \rangle$. The linearization error

$$\alpha_j^k = f(x_k) - \bar{f}_j(x_k) \quad \text{for all } j \in J^k$$

defines how well $\bar{f}_j$ approximates the function f at the current iteration point x_k.

In this paper, we assume that f is the DC function, and due to the lack of its convexity, the straightforward application of the cutting-plane approach is meaningless. Nevertheless, it can be separately applied to model the DC components $f_i,\ i = 1, 2$. Thus, we have

$$\hat{f}_i^k(x) = \max_{j \in J^k} \Big\{ f_i(x_j) + \langle \xi_{i,j}, x - x_j \rangle \Big\} \quad \text{for } x \in \mathbb{R}^n.$$

Here, $\xi_{i,j} \in \partial f_i(x_j),\ i = 1, 2$ are the subgradients of the DC components f_i computed at the auxiliary point $x_j \in \mathbb{R}^n$ for $j \in J^k$. This approximation can be rewritten as

$$\hat{f}_i^k(x) = \max_{j \in J^k} \Big\{ f_i(x_k) + \langle \xi_{i,j}, x - x_k \rangle - \alpha_{i,j}^k \Big\}, \tag{2}$$

where $x_k \in \mathbb{R}^n$ is the current iteration point, and $\alpha_{i,j}^k,\ i = 1, 2$ are the linearization errors associated with the j-th first order expansion of $f_i,\ i = 1, 2$ rooted at the point x_j, given by

$$\alpha_{i,j}^k = f_i(x_k) - f_i(x_j) - \langle \xi_{i,j}, x_k - x_j \rangle \quad \text{for all } j \in J^k.$$

Suppose that information coming from some auxiliary points $x_j \in \mathbb{R}^n$, $j \in J^k$, along with $\big(x_k,\ \xi_{1,k} \in \partial f_1(x_k),\ \xi_{2,k} \in \partial f_2(x_k)\big)$, is available. We condense all such information into a bundle set B^k defined as a set of tuples, one for each point x_j. That is,

$$b(x_j) = \Big(x_j, f_1(x_j), f_2(x_j), \xi_{1,j} \in \partial f_1(x_j), \xi_{2,j} \in \partial f_2(x_j), \alpha_{1,j}^k, \alpha_{2,j}^k \Big).$$

In general, we assume that for some appropriate index j, the tuple

$$b(x_k) = \Big(x_k, f_1(x_k), f_2(x_k), \xi_{1,k} \in \partial f_1(x_k), \xi_{2,k} \in \partial f_2(x_k), 0, 0 \Big) \tag{3}$$

associated with x_k is in the bundle too.

3. The New Model Function

In this section, we formulate our new model function. First, we distribute the bundle index set J^k into the subsets

$$J_1^k = \left\{ j : j \in J^k,\ \alpha_j^k \le 0 \right\} \quad \text{and} \quad J_2^k = \left\{ j : j \in J^k,\ \alpha_j^k \ge 0 \right\}, \tag{4}$$

where $J^k = J_1^k \cup J_2^k$. It is worth noting that this does not properly define a partition of J^k as the indexes corresponding to $\alpha_j^k = 0$ are in both subsets J_1^k and J_2^k. In addition, $\alpha_j^k = \alpha_{1,j}^k - \alpha_{2,j}^k$ can take any sign due to the nonconvexity of the function f. Based on the sign of α_j^k, we extract elements from the set J^k and modify the definition of the cutting-plane models, given in (2), accordingly. It is clear that $\alpha_j^k \le 0$ corresponds to $\alpha_{1,j}^k \le \alpha_{2,j}^k$. That is, the linearization error at x_k associated with the linearization of the function f_1 rooted at x_j is not bigger than that associated with the function f_2. This means that the information provided by x_j is more suited to approximate f_1 than f_2 around x_k. The reverse holds for the case $\alpha_j^k \ge 0$.

Remark 1. *The structures of the subsets J_i^k, $i = 1, 2$ depend on the point x_k. Thus, they should be updated every time a new iterate x_{k+1} is calculated.*

Consider the subsets J_i^k, $i = 1, 2$. Aiming at a parsimonious use of the available information, we restrict the definition of the cutting-plane functions, given in (2), as

$$\hat{f}_i^k(x) = \max_{j \in J_i^k} \left\{ f_i(x_k) + \langle \xi_{i,j}, x - x_k \rangle - \alpha_{i,j}^k \right\},\ i = 1, 2,$$

which reduces the number of affine pieces defining the convex approximations of the functions f_i, $i = 1, 2$.

Next, we introduce the variable $d = x - x_k$ and define the function $h^k(d)$ (the model of the difference function $f(x_k + d) - f(x_k)$) as

$$h^k(d) = h_1^k(d) - h_2^k(d) = \max_{j \in J_1^k} \left(\langle \xi_{1,j}, d \rangle - \alpha_{1,j}^k \right) - \max_{j \in J_2^k} \left(\langle \xi_{2,j}, d \rangle - \alpha_{2,j}^k \right),$$

and calculate

$$w^k = \min_d w^k(d) = \min_d h^k(d) + \frac{1}{2}\delta_k \|d\|^2. \tag{5}$$

Here, δ_k is the proximity parameter used in most bundle methods, and $\frac{1}{2}\delta_k\|d\|^2$ is a stabilizing term used to guarantee the existence of the solution. Letting d_k be the solution to (5), we take $x_{k+1} = x_k + d_k$ as the (tentative) new iterate point. Then, we check this point for a possible sufficient decrease in the objective function f as follows:

$$f(x_k + d_k) \le f(x_k) + \mu v^k, \tag{6}$$

where $\mu \in (0, 1)$ is a given parameter, and v^k is the predicted reduction of the function f provided by the model function $h^k(d)$. That is, $v^k = v_1^k - v_2^k$, where

$$\begin{aligned} v_1^k &= h_1^k(d_k) = \max_{j \in J_1^k} \langle \xi_{1,j}, d_k \rangle - \alpha_{1,j}^k, \quad \text{and} \\ v_2^k &= h_2^k(d_k) = \max_{j \in J_2^k} \langle \xi_{2,j}, d_k \rangle - \alpha_{2,j}^k. \end{aligned}$$

Note that we have $v^k \leq 0$. It is sufficient to observe that both functions $h_1^k(d)$ and $h_2^k(d)$ are non-positive at $d = 0$. If (6) is satisfied (serious step in bundle method parlance), then the point $x_{k+1} = x_k + d_k$ becomes the new estimate of a minimum. Thus, the related information to this point is appended to the bundle, and the bundle index set is updated. Next, the elements of the new index set J^{k+1} are distributed between J_1^{k+1} and J_2^{k+1} according to (4) (see Remark 1), and the procedure is iterated.

If the condition (6) is not satisfied, that is $f(x_k + d_k) > f(x_k) + \mu v^k$, then there is no sufficient decrease (null step). Taking into account $\mu \in (0,1)$ and $v^k \leq 0$, it follows that

$$\begin{aligned} f_1(x_k + d_k) - f_2(x_k + d_k) &> f_1(x_k) - f_2(x_k) + \mu(v_1^k - v_2^k) \\ &> f_1(x_k) - f_2(x_k) + v_1^k - v_2^k, \end{aligned}$$

which in turn implies that

$$\Big(f_1(x_k + d_k) - f_1(x_k)\Big) - v_1^k > \Big(f_2(x_k + d_k) - f_2(x_k)\Big) - v_2^k. \tag{7}$$

This means that whenever a sufficient decrease does not occur, then the gap between the actual and the predicted reduction for the function f_1 is bigger than that of f_2. Such an observation is at the basis of our bundle enlargement strategy in the case of the null step. More precisely, whenever a null step occurs, we invest in improving the approximation of f_1 more than in that of f_2.

The following proposition ensures that in the case of the null step, by inserting the point $x_{k+1} = x_k + d_k$ into the bundle, any couple (ξ_1^+, α^+) generates a substantial cut of the epigraph of h_1^k and, thus, an improved model of the function f_1. Here, $\xi_1^+ \in \partial f_1(x_k + d_k)$ and

$$\alpha^+ = f_1(x_k) - f_1(x_k + d_k) + \langle \xi_1^+, d_k \rangle.$$

Proposition 1. *Assume that $v^k < -\eta$ for some $\eta > 0$, and the sufficient decrease condition (6) is not fulfilled at the point $x_k + d_k$. Then, for any $\xi_1^+ \in \partial f_1(x_k + d_k)$, we have*

$$\langle \xi_1^+, d_k \rangle - \alpha^+ > v_1^k + \rho\eta,$$

where $\rho = 1 - \mu$.

Proof. From the definition of α^+, it follows

$$\langle \xi_1^+, d_k \rangle - \alpha^+ = f_1(x_k + d_k) - f_1(x_k).$$

Further, from (7), considering the definition of v_2^k and $v^k = v_1^k - v_2^k \leq 0$, we have

$$\begin{aligned} \langle \xi_1^+, d_k \rangle - \alpha^+ &= f_1(x_k + d_k) - f_1(x_k) \\ &> f_2(x_k + d_k) - f_2(x_k) + \mu(v_1^k - v_2^k) \\ &\geq v_2^k + \mu(v_1^k - v_2^k) = v_1^k + \rho(v_2^k - v_1^k) \\ &= v_1^k - \rho v^k \geq v_1^k + \rho\eta. \end{aligned}$$

This completes the proof. □

4. The Proposed BEM-DC Algorithm

In this section, we describe the BEM-DC algorithm for solving Problem (1) and give its step-by-step format. The algorithm has both inner and outer iterations. The inner iteration contains the evaluation, in terms of the decrease in the objective function, of tentative displacements from the current point. Null steps might occur within the inner iteration whenever a sufficient decrease is not achieved. Once, instead, such reduction is obtained (serious step), the current estimate of the minimum is updated and a new outer iteration takes place. Based on Proposition 1, a subgradient accumulation process takes place any time a null step occurs within the inner iteration. The specific feature of the BEM-DC algorithm is such that a process involves only information about the function f_1 which is stored in a temporary bundle of tuples, thus exclusively enriching the bundle J_1^k.

Let us denote the outer iteration counter by k. We use l to count for the l-th inner iteration within the k-th outer iteration. Denote the current temporary bundle by TB_l and its corresponding index set by TJ_l. Then, the displacement finding subproblem (5) at the inner iteration l within the k-th iteration takes the following form:

$$w_l^k = \min_d w_l^k(d).$$

Define

$$\begin{aligned} d_l^k &= \arg\min_d w_l^k(d) = \arg\min_d h_l^k(d) + \frac{1}{2}\delta_k \|d\|^2, \quad \text{and} \\ v_l^k &= h_l^k(d_l^k). \end{aligned} \tag{8}$$

Then, considering the presence of two distinct bundles for the function f, we have

$$h_l^k(d) = h_{1,l}^k(d) - h_2^k(d),$$

where

$$\begin{aligned} h_{1,l}^k(d) &= \max\left\{ \max_{j\in J_1^k} \left(\langle \xi_{1,j}^k, d\rangle - \alpha_{1,j}^k\right), \max_{j\in TJ_l} \left(\langle \xi_{1,j}^k, d\rangle - \alpha_{1,j}^k\right) \right\} \\ &= \max_{j\in J_1^k \cup TJ_l} \left(\langle \xi_{1,j}^k, d\rangle - \alpha_{1,j}^k\right), \quad \text{and} \\ h_2^k(d) &= \max_{j\in J_2^k} \left(\langle \xi_{2,j}^k, d\rangle - \alpha_{2,j}^k\right). \end{aligned}$$

The sets TB_l and TJ_l are updated at each inner iteration, while they are reset to the empty sets at the beginning of each outer iteration. Note that the set J_2^k remains unchanged during the inner iteration process.

Let us focus on Problem (8). Note that it is a DC programming problem whose global optimal solution can be found by solving $|J_2^k|$ convex problems (see [29,33]). Consider the following (convex) problem $P_{l,j}^k$, $j \in J_2^k$:

$$\min_d w_{l,j}^k(d)$$

with

$$w_{l,j}^k(d) = h_{1,l}^k(d) - \left(\langle \xi_{2,j}^k, d\rangle - \alpha_{2,j}^k\right) + \frac{1}{2}\delta_k\|d\|^2.$$

Let $d_{l,j}^k = \arg\min_d w_{l,j}^k(d)$. It is clear that, defining

$$j^* = \arg\min_{j\in J_2^k} w_{l,j}^k(d_{l,j}^k),$$

the global optimal solution of Problem (8) is

$$\begin{aligned} d_l^k &= d_{l,j^*}^k = \arg\min_d w_{l,j^*}^k(d), \quad \text{and} \\ v_l^k &= h_l^k(d_l^k) = h_{1,l}^k(d_l^k) - \left(\langle \xi_{2,j^*}^k, d_l^k\rangle - \alpha_{2,j^*}^k\right). \end{aligned}$$

Summing up, d_l^k is a global optimal solution of Problem (8), and the couple (d_l^k, v_l^k) is the unique optimal solution of the convex problem P_{l,j^*}^k

$$\begin{aligned} \min_{d \in \mathbb{R}^n, v \in \mathbb{R}} \quad & v + \frac{1}{2}\delta_k \|d\|^2 \\ & v \geq \langle \xi_{1,j}^k - \xi_{2,j^*}^k, d \rangle - \left(\alpha_{1,j}^k - \alpha_{2,j^*}^k\right), \quad j \in J_1^k \cup TJ_l. \end{aligned} \tag{9}$$

Then, applying the standard duality arguments to Problem (9), we have the following primal-dual relations:

$$d_l^k = -\frac{1}{\delta_k}\Big(\sum_{j \in J_1^k \cup TJ_l} \lambda_j \xi_{1,j}^k - \xi_{2,j^*}^k \Big), \tag{10}$$

$$v_l^k = -\frac{1}{\delta_k}\Big\| \sum_{j \in J_1^k \cup TJ_l} \lambda_j \xi_{1,j}^k - \xi_{2,j^*}^k \Big\|^2 - \sum_{j \in J_1^k \cup TJ_l} \lambda_j \alpha_{1,j}^k + \alpha_{2,j^*}^k, \tag{11}$$

$$w_l^k = -\frac{1}{2\delta_k}\Big\| \sum_{j \in J_1^k \cup TJ_l} \lambda_j \xi_{1,j}^k - \xi_{2,j^*}^k \Big\|^2 - \sum_{j \in J_1^k \cup TJ_l} \lambda_j \alpha_{1,j}^k + \alpha_{2,j^*}^k. \tag{12}$$

Here, $\lambda_j \geq 0$, $j \in J_1^k \cup TJ_l$, with $\sum_{j \in J_1^k \cup TJ_l} \lambda_j = 1$, are the optimal variables of the dual of Problem (9). In addition, the definition of the set of problems $P_{l,j}^k$, $j \in J_2^k$ implies that

$$w_l^k = \min_d w_l^k(d) = w_{l,j^*}^k(d_{l,j^*}^k) \leq w_{l,j}^k(d_{l,j}^k), \quad j \in J_2^k.$$

Since in the bundle index set J_2^k, there exists an index, say j_0, associated with the tuple $b(x_k)$ (see (3)) for which $\xi_{2,j_0}^k \in \partial f_2(x_k)$ and $\alpha_{2,j_0}^k = 0$, we have

$$w_l^k \leq w_{l,j_0}^k = w_{l,j_0}^k(d_{l,j_0}^k) = -\frac{1}{2\delta_k}\Big\| \sum_{j \in J_1^k \cup TJ_l} \lambda_j^0 \xi_{1,j}^k - \xi_{2,j_0}^k \Big\|^2 - \sum_{j \in J_1^k \cup TJ_l} \lambda_j^0 \alpha_{1,j}^k, \tag{13}$$

for $\lambda_j^0 \geq 0$, $j \in J_1^k \cup TJ_l$, with $\sum_{j \in J_1^k \cup TJ_l} \lambda_j^0 = 1$. This together with (12) suggests a possible termination criterion for the proposed algorithm. In fact, whenever $w_l^k \geq -\eta$, we have

$$\frac{1}{2\delta_k}\Big(\Big\| \sum_{j \in J_1^k \cup TJ_l} \lambda_j^0 \xi_{1,j}^k - \xi_{2,j_0}^k \Big\|^2 \Big) + \sum_{j \in J_1^k \cup TJ_l} \lambda_j^0 \alpha_{1,j}^k \leq \eta,$$

which indicates that the subgradient $\xi_{2,j_0}^k \in \partial f_2(x_k)$ is at a distance not bigger than $\sqrt{2\delta_k \eta}$ from the ε-subdifferential of f_1 at x_k, where

$$\varepsilon = \sum_{j \in J_1^k \cup TJ_l} \lambda_j^0 \alpha_{1,j}^k \leq \eta.$$

This property can be interpreted as the approximate satisfaction at the point x_k of criticality. Nevertheless, we implement the termination test of the proposed algorithm in the more common form of $v_l^k \geq -\eta$. In fact, from (10) and (11), $v_l^k \geq -\eta$ implies $w_l^k \geq -\eta$. Further, the fulfillment of the condition $w_{l,j_0}^k \geq -\eta$ provides an alternative termination criterion as

$$v_{l,j_0}^k = -\frac{1}{\delta_k}\Big(\Big\| \sum_{j \in J_1^k \cup TJ_l} \lambda_j^0 \xi_{1,j}^k - \xi_{2,j_0}^k \Big\|^2 \Big) - \sum_{j \in J_1^k \cup TJ_l} \lambda_j^0 \alpha_{1,j}^k \geq -\eta. \tag{14}$$

Remark 2. *In the above case, we embed the switching direction technique. That is, we use d_l^k as a tentative displacement, and if the descent failure occurs, then we implement an Armijo-type line search along the direction d_{l,j_0}^k. Only in the case of failure of the latter is a null step declared.*

Next, we present the proposed BEM-DC algorithm 1 in the step-by-step format. We denote by $i_{\mathrm{ds}} = 1$ when the switching direction technique is embedded.

Algorithm 1: BEM-DC.

```
Require: The stopping tolerance parameter $\eta > 0$, the null step parameter $\theta > 0$, the proximity threshold
    $\delta_{min} > 0$, the sufficient descent parameter $\mu \in (0,1)$, and the step size reduction parameters $\sigma_1, \sigma_2 \in (0,1)$.
Ensure: An approximate critical points of Problem (1).
  Select a starting point $x_1 \in R^n$. Compute $\xi_{1,1} \in \partial f_1(x_1)$ and $\xi_{2,1} \in \partial f_2(x_1)$.
  Set $B^1 = \Big\{(x_1, f_1(x_1), f_2(x_1), \xi_{1,1}, \xi_{2,1}, 0, 0)\Big\}$, $J_1^1 = \{1\}$, $J_2^1 = \{1\}$, and $k = 1$.
  Outer iteration
    Set $TB_1 = TJ_1 = \emptyset$, and $i_{\mathrm{ds}} = 0$. Compute the proximity parameter $\delta_k$, and set $l = 1$.
    Inner iteration
      Step 1. (Calculation of $d_l^k$ and $v_l^k$) If $i_{\mathrm{ds}} = 0$, then compute $d_l^k$ and $v_l^k$ using (8). Otherwise,
          compute $v_{l,j_0}^k$ from (14) and $d_{l,j_0}^k$ according to Remark 2. Set $d_l^k = d_{l,j_0}^k$ and $v_l^k = v_{l,j_0}^k$.
      Step 2. (Stopping test). If $v_l^k \geq -\eta$, then STOP.
      Step 3. Set $t = 1$.
      Step 4. (Descent test) If
                  $f(x_k + td_l^k) - f(x_k) < t\mu v_l^k,$        (15)
          then EXIT the inner iteration and go to Step 8.
      Step 5. (Step size update). If $i_{\mathrm{ds}} = 0$, then set $t = \sigma_1 t$, else $t = \sigma_2 t$. If $t\|d_l^k\| > \theta$, then go to Step 4.
          If $t\|d_l^k\| \leq \theta$ and $i_{\mathrm{ds}} = 0$, then set $i_{\mathrm{ds}} = 1$ and go to Step 1. Otherwise, go to the next step.
      Step 6. (Null step). Set $y = x_k + td_l^k$. Compute $\xi_{1,l}^k \in \partial f_1(y)$. Construct
                  $b(y) = (y, f_1(y), \xi_{1,l}^k, \alpha_{1,l}^k)$, where $\alpha_{1,l}^k = f_1(x_k) - f_1(y) + t\langle \xi_{1,l}^k, d_l^k \rangle$.
      Step 7. (Bundle enrichment). Update $TB_l = TB_l \cup \{b(y)\}$ and $TJ_l = TJ_l \cup \{l\}$. Set $i_{\mathrm{ds}} = 0$,
          $l = l + 1$ and go to Step 1.
    Step 8. (Serious step). Set $x_{k+1} = x_k + td_l^k$. Compute $\xi_{1,k+1} \in \partial f_1(x_{k+1})$ and $\xi_{2,k+1} \in \partial f_2(x_{k+1})$.
        Construct the bundle tuple
                  $b(x_{k+1}) = \Big(x_{k+1}, f_1(x_{k+1}), f_2(x_{k+1}), \xi_{1,k+1}, \xi_{2,k+1}, 0, 0\Big)$.
    Step 9. (Bundle and index sets update). Set $B^{k+1} = B^k \cup \{b(x_{k+1})\}$ and $J^{k+1} = J^k \cup \{k+1\}$. For each
        bundle point, re-calculate the linearization error with respect to $x_{k+1}$, update the sets $J_1^{k+1}$ and
        $J_2^{k+1}$ according to (4). Set $k = k + 1$ and go to the next outer iteration.
```

Remark 3. *We update the proximity parameter δ_k using the following formula given in [46]:*

$$\bar{\delta}_{k+1} = \delta_k \Big(1 - \frac{f(x_k + d_k) - f(x_k)}{v_k}\Big),$$
$$\delta_{k+1} = \max\Big\{\bar{\delta}_{k+1}, \delta_k/10, \delta_{min}\Big\}. \quad (16)$$

Remark 4. *An outline of the main differences between BEM-DC and PBDC described in [29] is in order.*

- *In* BEM-DC, *only one of the two subgradients $\xi_{1,j}$ and $\xi_{2,j}$, gathered at any iteration j, enters into the calculation of the tentative displacement at each of the successive iterations; the choice is driven by the sign of the linearization error at the current iterate. This is not the case in PBDC;*
- *In* BEM-DC, *there exists a temporary bundle whose "birth and death" takes place within the procedure for escaping from the null step and does not increase the bundle size once a descent*

is achieved, while PBDC adds more information to the bundles at each step and, thus, increases the bundle sizes at every iteration;

- *In* BEM-DC, *a second direction is checked for the descent before the null step is declared (see Remark 2), whereas there is only one possible direction used in each step of PBDC.*

5. Termination Property of Algorithm BEM-DC

In this section, we provide the proof of the Algorithm BEM-DC, taking any starting point $x_1 \in \mathbb{R}^n$ as an input, that returns an approximate critical point x^*. Assume that the set

$$\mathcal{F}_1 \triangleq \Big\{x \in \mathbb{R}^n : f(x) \le f(x_1)\Big\}$$

is bounded and the numbers L_1 and L_2 are the Lipschitz constants of f_1 and f_2, respectively, on the set $\mathcal{F}_1$. Note that, whenever the tuple $b(y)$ is inserted into the temporary bundle TB_l associated with the function f_1 at the trial point y (see *Step 6*), then $t\|d_l^k\| \le \theta$ implies $\xi_1^+(y) \in \partial_\varepsilon f_1(x_k)$ for $\varepsilon \le 2\theta L_{1\theta}$, where $L_{1\theta}$ is the Lipschitz constant of f_1 on the set

$$\mathcal{F}_\theta \triangleq \Big\{x \in \mathbb{R}^n : dist(x, \mathcal{F}_1) \le \theta\Big\}.$$

Lemma 1. *Let $L_{1\theta}$ and L_2 be the Lipschitz constants of f_1 and f_2, respectively, on the set $\mathcal{F}_\theta$. Then the following bound holds:*

$$\|d_l^k\| \le \frac{L_{1\theta} + L_2}{\delta_{min}} = D. \tag{17}$$

Proof. Throughout the algorithm, we have $\delta_k \ge \delta_{min}$, and thus, the inequality follows from (10). □

Remark 5. *The bound given in (17) is also valid for $\|d_{l,j_0}^*\|$.*

Next, we prove that the number of inner iterations of algorithm BEM-DC is finite.

Lemma 2. *At any given k-th outer iteration, the inner iteration terminates, either fulfilling the sufficient decrease condition (Step 4) or satisfying the stopping condition (Step 2).*

Proof. Suppose by contradiction that the inner iteration does not terminate. Since (see Remark 5) $\|d_{l,j_0}^*\|$ is bounded and t becomes arbitrarily small, the algorithm cannot loop infinitely many times between Steps 4 and 5. Thus, it loops infinitely many times between Steps 1 and 7, giving rise to an infinite number of null steps.

Observe now that every time a null step occurs and the tuple $b(y)$ is generated at Step 6, we have

$$\begin{aligned} f(x_k + t_l d_l^k) - f(x_k) &= f_1(x_k + t_l d_l^k) - f_2(x_k + t_l d_l^k) - \Big(f_1(x_k) - f_2(x_k)\Big) \\ &\ge t_l \mu v_l^k, \quad \text{for some} \quad 0 < t_l < 1. \end{aligned}$$

Considering

$$\alpha_{1,l+1}^k = f_1(x_k) - f_1(x_k + t_l d_l^k) + t_l \langle \xi_{1,l+1}^k, d_l^k \rangle,$$

the convexity of f_2, and $\xi_{2,j_0}^k \in \partial f_2(x_k)$, it follows

$$\langle \xi_{1,l+1}^k - \xi_{2,j_0}^k, d_l^k \rangle - \alpha_{1,l+1}^k \ge \Big(\frac{1}{t_l} - 1\Big)\alpha_{1,l+1}^k + \mu v_l^k > \mu v_l^k. \tag{18}$$

Furthermore, since the sequence $\{w_l^k\}_{l\to\infty}$, is monotonically non-decreasing and bounded from above by zero, it is convergent. Note that the sequence $\{w_{l,j_0}^k\}_{l\to\infty}$ (see (13)) is convergent as well. Consequently, from boundedness of $\|d_{l,j_0}^k\|$, there exists a convergent subsequence $\{d^k_{l,j_0}\}_{l\in\mathcal{L}} \to \bar{d}^k_0$, and thus, the subsequence $\{v^k_{l,j_0}\}_{l\in\mathcal{L}}$ also converges to a limit, say $\bar{v}^k_0 \le 0$, for some subset of indices $\mathcal{L} \subset \{1,2,\dots\}$.

Next, consider two successive indices $p, q \in \mathcal{L}$. From (18) and the definition of Problem P^k_{l,j_0}, we obtain

$$\langle \xi^k_{1,p+1} - \xi^k_{2,j_0}, d^k_p\rangle - \alpha^k_{1,p+1} > \mu v^k_{p,j_0}, \quad \text{and}$$
$$\langle \xi^k_{1,p+1} - \xi^k_{2,j_0}, d^k_q\rangle - \alpha^k_{1,p+1} \le v^k_{q,j_0}.$$

Therefore, we obtain

$$v^k_{q,j_0} - \mu v^k_{q,j_0} > \langle \xi^k_{1,p+1} - \xi^k_{2,j_0}, d^k_q - d^k_p\rangle.$$

Passing this to the limit, we have $(1-\mu)\bar{v}^k_0 \ge 0$, which, taking into account $\bar{v}^k_0 \le 0$, implies $\bar{v}^k_0 = 0$. This contradicts that the stopping condition at Step 2 is not satisfied infinitely many times. □

The next theorem proves the termination of Algorithm BEM-DC.

Theorem 1. *For any starting point $x_1 \in \mathbb{R}^n$, the algorithm terminates after finitely many iterations at a point satisfying the stopping criterion at Step* 2.

Proof. Assume the contrary. That is, the stopping criterion at Step 2 is never fulfilled. This implies, taking into account Lemma 2, that an infinite sequence of serious steps takes place, giving rise to infinitely many outer iterations.

Note that every time a serious step is achieved, considering the failed stopping test at Step 2, we have $v^k_l \le -\eta < 0$. Furthermore, it holds that either $t = 1$ or $t\|d^k\| > \theta$. Consequently, it follows from (15) that in the case of $t = 1$, we obtain

$$f(x_k + td^k) - f(x_k) < -\mu\eta, \tag{19}$$

or in the second case, considering (17), we have

$$f(x_k + td^k) - f(x_k) < -\frac{\mu\eta\theta}{D}.$$

This together with (19) implies that the decrease of the objective function value in both cases is bounded away from zero every time a serious step occurs. This is a contradiction under the assumption that $\mathcal{F}_1$ is bounded. □

Remark 6. *The "escape procedure" (Algorithm 1, introduced in [33]) can escape from critical points which are not Clarke stationary. Algorithm BEM-DC can be combined with this procedure to design an algorithm for finding Clarke stationary points of Problem* (1)*. More precisely, once an approximate critical point is obtained by* BEM-DC*, the escape procedure could be applied to approximate the Clarke subdifferential at this point [42,43]. Then, it verifies whether this approximation contains the origin with respect to some tolerance. During this approximation, the procedure generates directions that are used to find elements of the Clarke subdifferential. It is proved that this procedure is finitely convergent. That is, after a finite number of steps, it either confirms that the current critical point is also Clarke stationary or the descent direction is found to escape this point (see [33] for more details about different stationary points and their relationships).*

6. Numerical Experiments

To evaluate the performance of the BEM-DC algorithm and to compare it with some existing nonsmooth DC programming algorithms, we carry out numerical experiments using different known academic test problems designed for DC programming. Problems 1–9 are from [29], Problems 11–14 are described in [33], and Problems 15–17 are designed in [36]. We exclude Problem 10 from [29] in our experiments as it has many local minimizers and its usage does not provide an unbiased picture of the performance of local search solvers. For the formulations, optimal values, and initial points of these test problems, we refer to references [29,33,36].

6.1. Solvers and Parameters

We consider the following nonsmooth DC optimization solvers in our comparison:

- Augmented subgradient method for nonsmooth DC optimization (ASM-DC) [36];
- Aggregate subgradient method for nonsmooth DC optimization (AggSub) [35];
- Proximal bundle method for nonsmooth DC optimization (PBDC) [29];
- DC Algorithm (DCA) [8,20];
- Proximal bundle method for nonsmooth DC programming (PBMDC) [30];
- Sequential DC programming with cutting-plane models (SDCA-LP) [25].

The parameters in algorithm BEM-DC are chosen as follows: $\eta = 10^{-7}$, $\delta_{min} = 10^{-5}$, $\theta = 0.5$, $\mu = 0.2$, $\sigma_1 = 0.2$, and $\sigma_2 = 0.4$. The same set of parameters is used with all the test problems. The initial value of the proximity parameter is computed according to the recommendation given in [46]; that is, $\delta_1 = \|\xi_{1,1} - \xi_{2,1}\|$. We select the parameters of other algorithms considering the recommendations given in their references.

The algorithms BEM-DC, ASM-DC, AggSub, PBDC, and DCA are implemented in Fortran 95 and compiled using the gfortran compiler. For PBMDC and SDCA-LP, we use their MATLAB implementations, available at http://www.oliveira.mat.br/solvers (accessed on 1 August 2023). Since the SDCA-LP algorithm requires the feasible set to be compact, to apply this method to unconstrained problems, we define a large n-dimensional box around the starting point and consider only those points generated by the SDCA-LP that belong to this box.

We set a time limit for all algorithms. For solvers in MATLAB, we consider the limit to be three hours, and for those in Fortran, the limit is half an hour. The source code of algorithm BEM-DC is freely available on GitHub: https://github.com/SnTa2019/Nonsmooth-Optimization (accessed on 1 August 2023) and at http://napsu.karmitsa.fi/bemdc (accessed on 1 August 2023). We carry out our numerical experiments on a computer with Intel(R) Core (TM), i7-9750H, CPU @ 2.60 GHz, 32GB (RAM), under Windows 10, 64Bits. Since the solvers PBMDC and SDCA-LP are implemented in MATLAB, we do not report their CPU time in our comparison.

6.2. Evaluation Measures and Notations

We report the numbers of function and subgradient evaluations and the final value of the objective function found by algorithms. We also provide the computational time (in seconds) required by the algorithms implemented in the same platform (Fortran). The following notations are used:

- Prob. is the label of the problem;
- n is the number of variables;
- N_f is the number of function evaluations for the objective function f;
- N_ξ is the number of subgradient evaluations defined as the average of subgradient evaluations for the DC components f_1 and f_2;
- CPU is the computational time in seconds;
- f_A^* is the best value of the objective function obtained by algorithm A.

All methods used in this paper are local search methods. In this case, a method is successful if it can find a stationary point of a problem with the given accuracy. For each test problem, a subset of stationary points is known. This subset contains, in particular, stationary points found in methods used in this paper. We say that an algorithm “A” finds a solution with respect to a tolerance β_1 if

$$0 \le \frac{f_A^* - f_{opt}}{|f_{opt}| + 1} \le \beta_1, \tag{20}$$

where f_{opt} is the value of the objective function at one of the known stationary points. Otherwise, we say that the solver fails. We set $\beta_1 = 10^{-4}$. In addition, we apply performance profiles, introduced in [47], to analyze the results.

6.3. Results

In this subsection, we report and discuss the results of the numerical experiments. We present the results obtained by algorithms for Problems 1–9 from [29] and Problems 11–14 from [33]. We consider two cases by using two types of starting points. In the first case, for each problem, we use one starting point, given in [29,33]. In this case, using tables, we report the values of the objective functions and the numbers of function and subgradient evaluations. In the second case, we use 20 randomly generated starting points for each problem and present the results by applying performance profiles. We also report the average CPU time required by the BEM-DC algorithm on these problems for a different number of variables using both 1 and 20 randomly generated starting points. Further, we show the results for three large-scale test problems from [36] by reporting the values of objective functions and the numbers of function and subgradient evaluations. Tables 1–5 report the results, and the sign "–" is used to show the failure of a method in finding the correct solution.

6.3.1. Results with One Starting Point

Table 1 presents the best value of the objective function obtained by solvers using one starting point. We can see that the BEM-DC algorithm is sufficiently accurate in finding local solutions. Except for Problem 9, it finds global minimizers of DC optimization problems in all other cases. Other solvers fail in one or several problems. More specifically, ASM-DC, AggSub, and DCA fail to find solutions in Problem 4 with $n = 200$. Furthermore, AggSub fails in Problems 5 and 14 with $n = 5, 10, 50, 100, 200$; PBDC fails in Problem 12 with $n = 50, 100, 200$ and in Problem 13; DCA fails in Problem 8; PBMDC fails in Problems 7, 12, 13 and also fails in Problem 14 with $n = 100, 200$; and finally, SDCA-LP fails in Problems 1, 8, 9, 13 and also fails in Problem 12 with $n = 10, 50, 100, 200$.

In Table 2, we report the number of function and subgradient evaluations required by the algorithms. These numbers are computed as an average of the number of function and subgradient evaluations of DC components. Note that in SDCA-LP, the number of function and subgradient evaluations are the same. The results show that, in general, PBMDC requires the least number of function and subgradient evaluations among all algorithms. These numbers are similar for PBDC and SDCA-LP. We can see that the computational effort required by the BEM-DC is reasonable and in some instances it is similar to that of by PBMDC, PBDC, and SDCA-LP. Here, Problem 4 with $n = 50, 100, 200$ is an exception. Three other solvers, ASM-DC, AggSub, and DCA also require a reasonable computational effort in most cases; however, in general, they use (in some cases significantly) more function and subgradient evaluations than BEM-DC PBMDC, PBDC, and SDCA-LP.

Table 1. Best values of the objective functions obtained by solvers (one starting point).

Prob.	n	BEM-DC	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP
1	2	2.00000	2.00000	2.00000	2.00000	2.00000	2.00000	–
2	2	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000
3	4	0.00000	0.00001	0.00001	0.00000	0.00000	0.00000	2.00000
4	2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	10	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	50	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	100	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table 1. *Cont.*

Prob.	*n*	BEM-DC	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP
4	200	0.00000	–	–	0.00000	–	0.00000	0.00000
5	2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
5	5	0.00000	0.00000	–	0.00000	0.00000	0.00000	0.00000
5	10	0.00001	0.00000	–	0.00000	0.00000	0.00000	0.00001
5	50	0.00004	0.00001	–	0.00000	0.00011	0.00000	0.00002
5	100	0.00000	0.00001	–	0.00000	0.00031	0.00000	0.00007
5	200	0.00000	0.00001	–	0.00000	0.00003	0.00000	0.00008
6	2	−2.50000	−2.50000	−2.50000	−2.50000	−2.50000	−2.50000	−2.50000
7	2	0.50000	0.50000	0.50000	0.50000	1.00000	–	0.50000
8	3	3.50000	3.50000	3.50000	3.50000	–	3.50000	–
9	4	9.20000	9.20000	9.20000	1.83333	9.20000	1.83333	–
11	3	116.33333	116.33333	116.33377	116.33333	116.33333	116.33330	116.33330
12	2	0.61804	0.61804	0.61804	1.61803	0.61803	–	1.61809
12	5	0.61803	0.61804	0.61804	1.61803	0.61803	–	0.61803
12	10	0.61803	0.61804	0.61804	0.61803	0.61803	–	–
12	50	0.61803	0.61804	0.61804	–	0.61804	–	–
12	100	0.61803	0.61803	0.61804	–	0.61804	–	–
12	200	0.61803	0.61803	0.61804	–	0.61804	–	–
13	10	0.00000	0.00000	0.00000	–	0.00000	–	–
14	2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
14	5	0.00001	0.00000	–	0.00000	0.00000	0.00001	0.00001
14	10	0.00000	0.00000	–	0.00000	0.00000	0.00001	0.00003
14	50	0.00002	0.00000	–	0.00000	0.00000	0.00002	0.00005
14	100	0.00007	0.00001	–	0.00000	0.00000	–	0.00006
14	200	0.00003	0.00003	–	0.00000	0.00000	–	0.00003

Table 2. Number of function and subgradient evaluations required by solvers (one starting point).

Prob.	*n*	BEM-DC		ASM-DC		AggSub		PBDC		DCA		PBMDC		SDCA-LP
		N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f, N_ξ
1	2	105	25	189	52	162	75	22	17	60	41	10	16	–
2	2	175	25	109	41	255	92	18	13	72	54	3	5	7
3	4	87	17	149	53	391	174	23	11	222	181	5	8	16
4	2	75	17	43	19	64	31	6	3	52	28	2	2	6
4	5	40	9	126	44	235	120	13	6	165	124	3	4	13
4	10	70	18	256	88	545	273	16	10	361	309	5	8	26
4	50	1865	874	3430	1124	3206	1597	52	32	2915	2807	25	38	103
4	100	9049	4379	10,878	3539	6824	3405	102	66	6312	6232	50	75	204
4	200	40,713	19,626	–	–	–	–	475	546	–	–	101	151	403
5	2	8	3	231	50	75	31	18	4	37	24	2	3	8
5	5	64	26	118	44	–	–	15	7	474	297	6	8	9
5	10	87	42	335	111	–	–	23	15	54,116	53,287	12	24	34
5	50	526	235	1413	457	–	–	135	124	247,813	246,256	17	24	55
5	100	141	71	1337	438	–	–	47	25	471,198	469,845	20	29	63
5	200	129	65	1885	594	–	–	108	52	46,2821	461,267	18	28	66
6	2	39	10	135	43	105	55	22	14	41	33	17	24	29
7	2	303	48	388	111	285	107	72	47	3368	2471	–	–	49
8	3	125	49	271	75	176	88	60	32	–	–	–	–	–
9	4	9	3	199	63	169	80	86	72	63	56	17	25	–
11	3	146	28	174	61	258	126	10	7	18	17	5	8	14
12	2	130	26	290	79	127	64	20	15	61	60	–	–	37
12	5	202	52	457	122	213	103	58	39	67	65	–	–	696
12	10	393	105	831	257	359	170	1415	1257	76	74	–	–	–

Table 2. *Cont.*

Prob.	n	**BEM-DC**		**ASM-DC**		**AggSub**		**PBDC**		**DCA**		**PBMDC**		**SDCA-LP**
		N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f, N_ξ
12	50	739	156	3015	771	1411	694	–	–	99	97	–	–	–
12	100	1032	173	6998	2519	1239	907	–	–	117	115	–	–	–
12	200	2757	174	15,302	5090	2850	1397	–	–	338	337	–	–	–
13	10	145	21	26	13	70	38	–	–	20	19	–	–	–
14	2	19	4	50	23	71	34	9	5	56	54	11	16	29
14	5	145	26	206	73	226	100	109	105	63	61	22	31	58
14	10	249	59	357	107	322	155	6169	6151	61	60	34	50	107
14	50	1168	439	672	211	1653	790	991	966	16	14	59	89	280
14	100	2273	708	1779	549	–	–	780	740	18	16	–	–	552
14	200	2591	724	1746	541	–	–	983	961	28	26	–	–	619

6.3.2. Results with 20 Starting Points

Performance profiles using the number of function evaluations, the number of subgradient evaluations, and the computational time (CPU time) required by algorithms are depicted in Figures 1–3. Here, we use results obtained by solving Problems 1–9 and 11–14 with 20 random starting points. We report the pairwise comparison of performance profiles as per the note by [48]. The comparison with other methods shows that the BEM-DC is more efficient and robust than the AggSub, DCA, PBMDC, and SDCA-LP methods. Compared with the ASM-DC method, we can see that the BEM-DC uses significantly fewer subgradient evaluations, and the performance of these two methods are similar concerning two other measures. The performance of the BEM-DC and PBDC methods are similar for the number of function and subgradient evaluations; however, the BEM-DC is more efficient than PBDC if one uses computational time. One reason for this can be the fact that the BEM-DC better manages the bundle of the second DC component than PBDC and decreases the number of the solving of the quadratic programming subproblems required to find search directions.

For a given number n of variables, we also report the average CPU time (in seconds) required by the BEM-DC for Problems 1–9 and 11–14 (29 problems considering different variables) with 1 starting point and 20 random starting points. The results given in Table 3 indicate that the CPU time required by this algorithm is very small for problems with $n \leq 10$ number of variables.

Table 3. CPU time (in seconds) required by BEM-DC.

n	2	3	4	5	10	50	100
1 starting point	0.000	0.000	0.001	0.012	0.009	0.672	16.328
20 starting points	0.000	0.000	0.002	0.018	0.616	14.613	42.461

6.3.3. Results for Large-Scale Problems

Tables 4 and 5 provide the results obtained by different algorithms for three large-scale problems from [36]: P_{L1}, P_{L2}, and P_{L3}. More specifically, Table 4 contains the best objective function values found by each solver, and Table 5 displays the number of function and subgradient evaluations. As we mentioned above, the number of function and subgradient evaluations are the same for SDCA-LP. Only two algorithms, BEM-DC and ASM-DC, can find approximate solutions to all three problems. AggSub fails in Problem P_{L3}, PBDC in Problem P_{L1}, DCA in Problems P_{L1} and P_{L3}, PBMDC in Problem P_{L1}, and SDCA-LP in Problems P_{L1} and P_{L2}. Summarizing results from Table 4, we can conclude that the BEM-DC is the most accurate among all algorithms for solving large-scale problems used in numerical experiments.

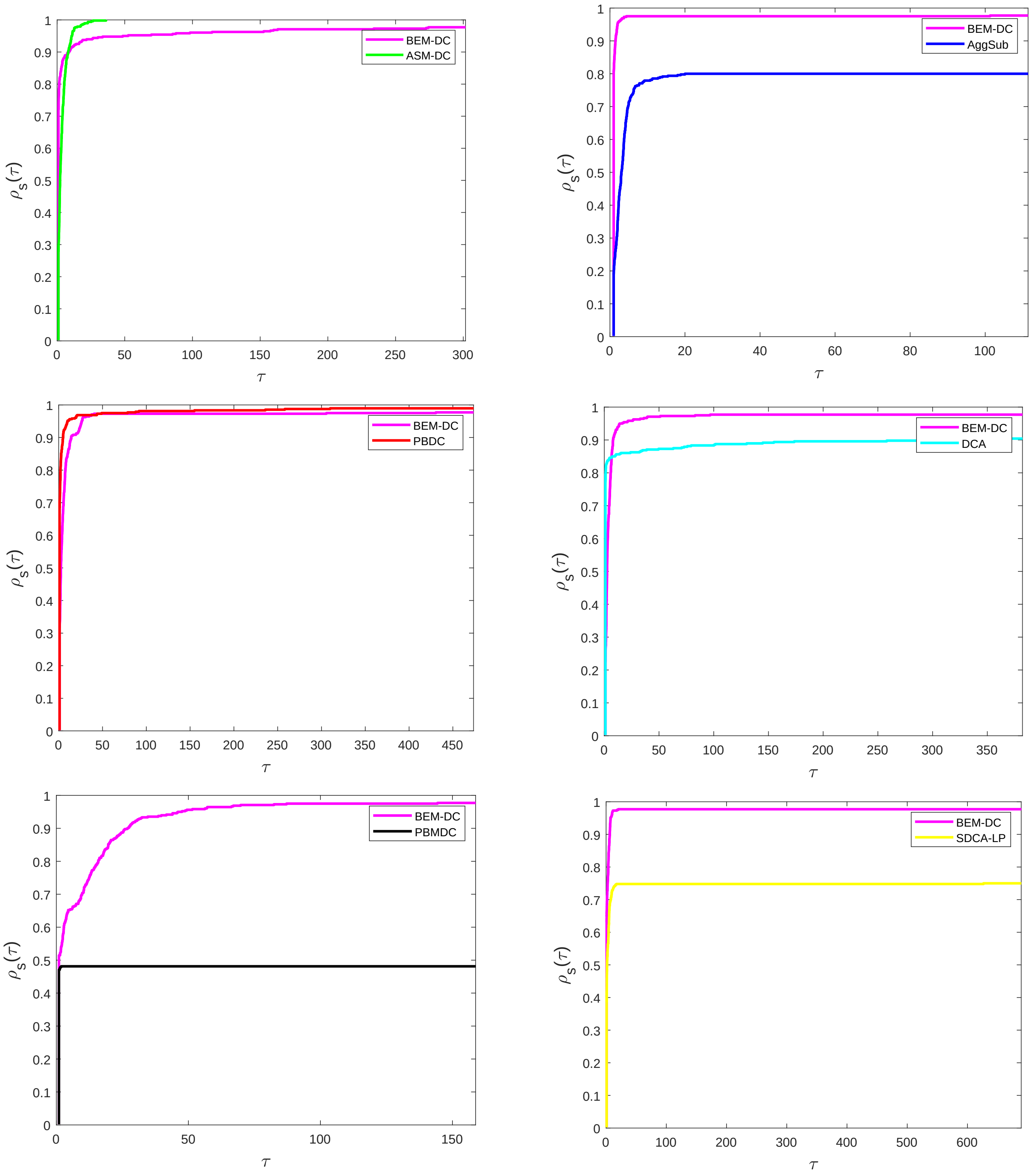

Figure 1. Performance profiles using the number of function evaluations (20 starting points).

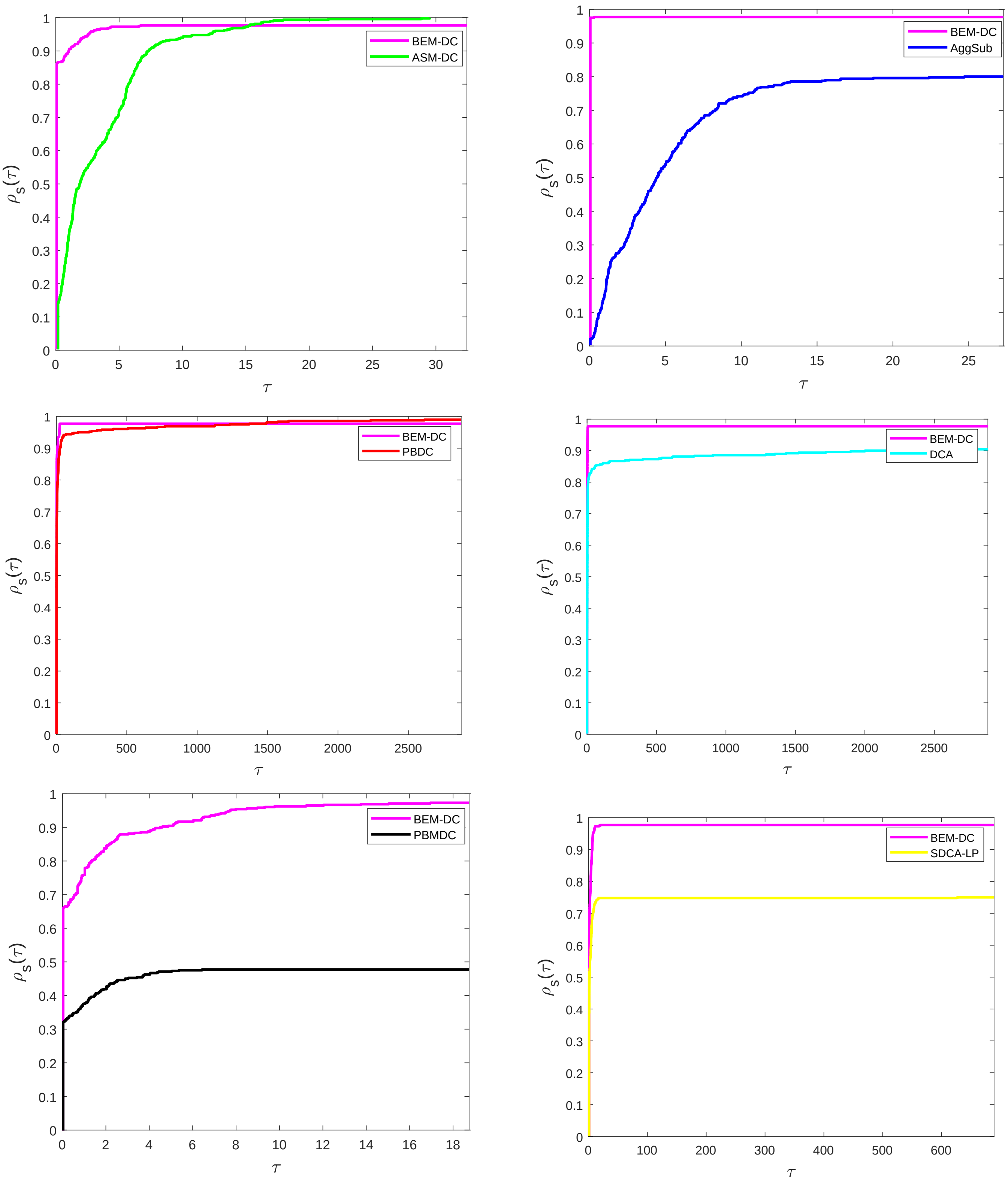

Figure 2. Performance profiles using the number of subgradient evaluations (20 starting points).

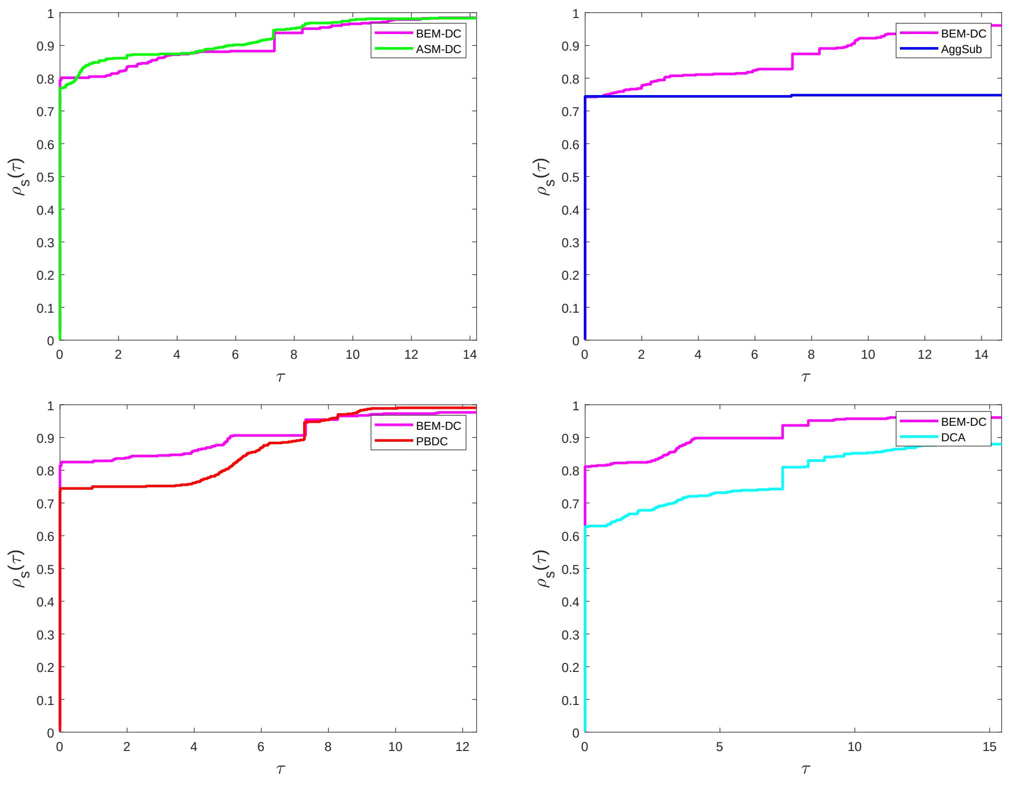

Figure 3. Performance profiles using CPU time (20 starting points).

Table 4. Best values of objective functions obtained by solvers for large-scale problems.

P	n	BEM-DC	ASM-DC	AggSub	PBDC	DCA	PBMDC	SDCA-LP
P_{L1}	200	0.00000	0.00006	0.00007	98.56721	153.29050	–	–
P_{L1}	500	0.00000	0.00009	0.00007	249.00000	539.99295	–	–
P_{L1}	1000	0.00000	0.00032	0.00032	499.00000	510.84581	–	–
P_{L1}	1500	0.00000	0.00072	0.00051	749.00000	1.90755×10^3	–	–
P_{L1}	2000	0.00000	0.00247	0.00083	999.00000	1.00229×10^3	–	–
P_{L2}	200	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	3.59539×10^3
P_{L2}	500	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.08151×10^4
P_{L2}	1000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001	2.43987×10^4
P_{L2}	1500	0.00000	0.00000	0.00000	0.00000	0.00000	0.00002	3.90289×10^4
P_{L2}	2000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00003	5.43386×10^4
P_{L3}	200	0.00004	0.00000	4.66526	0.00000	0.16124	0.00001	0.00005
P_{L3}	500	0.00001	0.00001	7.91275	0.00000	0.08352	0.00002	0.00015
P_{L3}	1000	0.00006	0.00037	17.32160	0.00000	0.05060	0.00001	0.00004
P_{L3}	1500	0.00002	0.00157	10.50161	0.00000	2.78735	0.00001	0.00011
P_{L3}	2000	0.00008	0.00056	12.63033	0.00000	26.17493	0.00002	0.00003

Results presented in Table 5 show that among all algorithms, BEM-DC uses the least number of function and subgradient evaluations for solving Problem P_{L1}, and PBMDC requires the least computational effort for solving Problems P_{L2} and P_{L3}. Although the BEM-DC can find accurate solutions for Problems P_{L2} and P_{L3}, it nevertheless requires more—and in the case of Problem P_{L3}, significantly more—function and subgradient evaluations than other solvers. All other algorithms also require reasonable computational effort in problems where they succeeded to find solutions.

Table 5. Number of function and subgradient evaluations for large-scale problems.

Prob.	***n***	**BEM-DC**		**ASM-DC**		**AggSub**		**PBDC**		**DCA**		**PBMDC**		**SDCA-LP**
		N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f	N_ξ	N_f, N_ξ
P_{L1}	200	49	3	2328	628	1075	277	60	48	2508	2502	–	–	–
P_{L1}	500	94	3	2083	456	1400	364	146	115	2006	2002	–	–	–
P_{L1}	1000	158	3	2165	497	2127	607	50	37	3511	3503	–	–	–
P_{L1}	1500	218	4	2691	664	1734	479	102	80	1003	1001	–	–	–
P_{L1}	2000	244	4	3123	793	1299	631	49	34	4526	4515	–	–	–
P_{L2}	200	863	248	1835	593	1604	787	974	780	5010	5005	108	177	4509
P_{L2}	500	1430	424	2161	705	2342	1157	1355	1057	5010	5005	144	23	2264
P_{L2}	1000	3589	1057	3153	1041	2277	1125	1664	1416	5904	5896	219	360	1655
P_{L2}	1500	4293	1329	3966	1310	2722	1344	2044	1740	4158	4153	295	482	1509
P_{L2}	2000	6433	1817	4433	1466	2936	1459	2244	1914	5511	5505	299	497	955
P_{L3}	200	1989	536	1044	333	14639	7151	1928	1845	3006	3003	55	97	97
P_{L3}	500	4598	729	1444	469	12869	6252	2539	2513	3507	3503	58	99	3555
P_{L3}	1000	13,844	3006	3445	1107	21,456	10,555	2980	2958	3006	3003	73	126	370
P_{L3}	1500	12,384	2291	2408	730	41,096	20,239	2763	2747	1503	1501	69	121	1130
P_{L3}	2000	17,520	3156	4017	1283	34,758	17,149	3766	3741	1503	1501	73	127	1090

7. Conclusions

In this paper, a new method, named the bundle enrichment method (BEM-DC), is introduced for solving nonsmooth unconstrained difference of convex (DC) optimization problems. This method belongs to the family of bundle-type methods. It exploits cutting plane models of DC components to build the model of the DC objective function. The main difference between the proposed method and other bundle methods for DC optimization is in the dynamic management of the bundle. In the implementation of most bundle methods, the size of the bundle for the cutting plane models of the second DC component is given by the user, and it is restricted to reduce the number of subproblems for finding search directions. However, in the BEM-DC, this size is self-determined by the method. This allows for avoiding solving subproblems that do not provide descent directions.

We prove that the BEM-DC computes approximate critical points of the unconstrained DC optimization problems in a finite number of iterations. The performance of this method is evaluated using two groups of nonsmooth DC optimization test problems: small- and medium-sized sized test problems and test problems with a large number of variables. We consider two types of starting points for problems from the first group: the single starting point available in the literature and 20 randomly generated starting points. The use of randomly generated starting points allows us to investigate the robustness of the proposed method. In addition, we provide a comparison of the BEM-DC with six other DC optimization methods.

Results of numerical experiments show that the BEM-DC is able to find accurate solutions using reasonable computational effort in most test problems. Nevertheless, in some large-scale problems, the number of function and subgradient evaluations required by the BEM-DC algorithm may increase significantly as the number of variables increases. This means that the BEM-DC algorithm may not be applicable to some nonsmooth DC optimization problems with a very large number of variables ($n \geq 5000$). The extension of this method for solving such problems will be the subject of future research.

The remarkable feature of the BEM-DC is that it is able to dynamically manage the number of subgradients of the second DC components and significantly decrease the number of quadratic programming subproblems for finding search directions. Results obtained using many randomly generated starting points allow us to conclude that the BEM-DC is efficient and is among the most robust methods used in numerical experiments. Further, the BEM-DC algorithm is able to efficiently solve relatively large nonsmooth DC optimization problems.

Author Contributions: All authors contribute equally. All authors have read and agreed to the published version of the manuscript.

Funding: A.M. Bagirov's research is funded by the Australian Government through the Australian Research Council's Discovery Projects (Project No. DP190100580), and the research by N. Karmitsa is supported by the Research Council of Finland (Projects No. 345804 and No. 345805)

Data Availability Statement: The source code of BEM-DC is freely available on GitHub: https://github.com/SnTa2019/Nonsmooth-Optimization (accessed on 1 August 2023) and at http://napsu.karmitsa.fi/bemdc (accessed on 1 August 2023).

Acknowledgments: A.M. Bagirov's research is funded by the Australian Government through the Australian Research Council's Discovery Projects (Project No. DP190100580), and the research by N. Karmitsa is supported by the Academy of Finland (Projects No. 345804 and No. 345805).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bertsekas, D.P. Nonlinear programming. In *Theoretical Solutions Manual*, 3rd ed.; Athena Scientific: Nashua, NH, USA, 2016.
2. Hiriart-Urruty, J.B. Generalized differentiability/ duality and optimization for problems dealing with differences of convex functions. In *Lecture Notes in Economics and Mathematical Systems*; Springer: Berlin/Heidelberg, Germany, 1986; Volume 256, pp. 37–70.
3. Strekalovsky, A.S. Global optimality conditions for nonconvex optimization. *J. Glob. Optim.* **1998**, *12*, 415–434. [CrossRef]
4. Strekalovsky, A.S. On a Global Search in D.C. Optimization Problems. In *Optimization and Applications*; Springer: Cham, Swizerland, 2020; pp. 222–236.
5. Strekalovsky, A.S. Local Search for Nonsmooth DC Optimization with DC Equality and Inequality Constraints. In *Numerical Nonsmooth Optimization*; Springer: Cham, Swizerland, 2020; pp. 229–261.
6. de Oliveira, W. The ABC of DC programming. *Set-Valued Var. Anal.* **2020**, *28*, 679–706. [CrossRef]
7. Horst, R.; Thoai, N.V. DC programming: Overview. *J. Optim. Theory Appl.* **1999**, *103*, 1–43. [CrossRef]
8. An, L.T.H.; Tao, P.D. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Ann. Oper. Res.* **2005**, *133*, 23–46. [CrossRef]
9. Holmberg, K.; Tuy, H. A production-transportation problem with stochastic demand and concave production costs. *Math. Program.* **1999**, *85*, 157–179. [CrossRef]
10. Pey-Chun, C.; Hansen, P.; Jaumard, B.; Tuy, H. Solution of the multisource Weber and conditional Weber problems by DC programming. *Oper. Res.* **1998**, *46*, 548–562. [CrossRef]
11. Khalaf, W.; Astorino, A.; D'Alessandro, P.; Gaudioso, M. A DC optimization-based clustering technique for edge detection. *Optim. Lett.* **2017**, *11*, 627–640. [CrossRef]
12. Sun, X.K. Regularity conditions characterizing Fenchel–Lagrange duality and Farkas-type results in DC infinite programming. *J. Math. Anal. Appl.* **2014**, *414*, 590–611. [CrossRef]
13. Bagirov, A.M.; Karmitsa, N.; Taheri, S. *Partitional Clustering via Nonsmooth Optimization*; Springer: Berlin/Heidelberg, Germany, 2020.
14. Bagirov, A.M.; Taheri, S.; Cimen, E. Incremental DC Optimization Algorithm for Large-Scale Clusterwise Linear Regression. *J. Comput. Appl. Math.* **2021**, *389*, 113323. [CrossRef]
15. Sun, X.; Teo, K.L.; Zeng, J.; Liu, L. Robust approximate optimal solutions for nonlinear semi-infinite programming with uncertainty. *Optimization* **2020**, *69*, 2109–2129. [CrossRef]
16. Sun, X.; Tan, W.; and Teo, K.L. Characterizing a Class of Robust Vector Polynomial Optimization via Sum of Squares Conditions. *J. Optim. Theory Appl.* **2023**, *197*, 737–764. [CrossRef]
17. Sun, X.; Teo, K.L.; Long, X.J. Some characterizations of approximate solutions for robust semi-infinite optimization problems. *J. Optim. Theory Appl.* **2021**, *191*, 281–310. [CrossRef]
18. Tuy, H. *Convex Analysis and Global Optimization*; Kluwer: Dordrecht, The Netherlands, 1998.
19. Tao, P.D.; Souad, E.B. Algorithms for solving a class of nonconvex optimization problems: Methods of subgradient. *North-Holl. Math. Stud.* **1986**, *129*, 249–271.
20. An, L.T.H.; Tao, P.D.; Ngai, H.V. Exact penalty and error bounds in DC programming. *J. Glob. Optim.* **2012**, *52*, 509–535.
21. Artacho, F.J.A.; Fleming, R.M.T.; Vuong, P.T. Accelerating the DC algorithm for smooth functions. *Math. Program.* **2018**, *169*, 95–118. [CrossRef]
22. Artacho, F.J.A.; Vuong, P.T. The Boosted Difference of Convex Functions Algorithm for Nonsmooth Functions. *Siam J. Optim.* **2020**, *30*, 980–1006. [CrossRef]
23. de Oliveira, W.; Tcheou, M.P. An inertial algorithm for DC programming. *Set-Valued Var. Anal.* **2019**, *27*, 895–919. [CrossRef]
24. Artacho, F.J.A.; Campoy, R.; Vuong, P.T. Using positive spanning sets to achieve d-stationarity with the boosted DC algorithm. *Vietnam. J. Math.* **2020**, *48*, 363–376. [CrossRef]
25. de Oliveira, W. Sequential difference-of-convex programming. *J. Optim. Theory Appl.* **2020**, *186*, 936–959. [CrossRef]

26. Dolgopolik, M.V. A convergence analysis of the method of codifferential descent. *Comput. Optim. Appl.* **2018**, *71*, 879–913. [CrossRef]
27. Gaudioso, M.; Giallombardo, G.; Miglionico, G. Minimizing piecewise-concave functions over polytopes. *Math. Oper. Res.* **2018**, *43*, 580–597. [CrossRef]
28. Gaudioso, M.; Giallombardo, G.; Miglionico, G.; Bagirov, A.M. Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. *J. Glob. Optim.* **2018**, *71*, 37–55. [CrossRef]
29. Joki, K.; Bagirov, A.M.; Karmitsa, N.; Mäkelä, M.M. A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. *J. Glob. Optim.* **2017**, *68*, 501–535. [CrossRef]
30. de Oliveira, W. Proximal bundle methods for nonsmooth DC programming. *J. Glob. Optim.* **2019**, *75*, 523–563. [CrossRef]
31. Sun, W.Y.; Sampaio, R.J.B.; Candido, M.A.B. Proximal point algorithm for minimization of DC functions. *J. Comput. Math.* **2003**, *21*, 451–462.
32. Souza, J.C.O.; Oliveira, P.R.; Soubeyran, A. Global convergence of a proximal linearized algorithm for difference of convex functions. *Optim. Lett.* **2016**, *10*, 1529–1539. [CrossRef]
33. Joki, K.; Bagirov, A.M.; Karmitsa, N.; Mäkelä, M.M.; Taheri, S. Double bundle method for finding Clarke stationary points in nonsmooth DC programming. *Siam J. Optim.* **2018**, *28*, 1892–1919. [CrossRef]
34. Ackooij, W.; Demassey, S.; Javal, P.; Morais, H.; de Oliveira, W.; Swaminathan, B. A bundle method for nonsmooth dc programming with application to chance–constrained problems. *Comput. Optim. Appl.* **2021**, *78*, 451–490. [CrossRef]
35. Bagirov, A.M.; Taheri, S.; Joki, K.; Karmitsa, N.; Mäkelä, M.M. Aggregate subgradient method for nonsmooth DC optimization. *Optim. Lett.* **2020**, *15*, 83–96. [CrossRef]
36. Bagirov, A.M.; Hoseini, M.N.; Taheri, S. An augmented subgradient method for minimizing nonsmooth DC functions. *Comput. Optim. Appl.* **2021**, *80*, 411–438. [CrossRef]
37. Astorino, A.; Frangioni, A.; Gaudioso, M.; Gorgone,E. Piecewise quadratic approximations in convex numerical optimization. *SIAM J. Optim.* **2011**, *21*, 1418–1438. [CrossRef]
38. Bagirov, A.M.; Gaudioso, M.; Karmitsa, N.; Mäkelä, M.M.; Taheri, S. (Eds.) *Numerical Nonsmooth Optimization, State of the Art Algorithms*; Springer: Berlin/Heidelberg, Germany, 2020.
39. Gaudioso, M.; Monaco, M.F. Variants to the cutting plane approach for convex nondifferentiable optimization. *Optimization* **1992**, *25*, 65–75. [CrossRef]
40. Hiriart–Urruty, J.B.; Lemaréchal, C. *Convex Analysis and Minimization Algorithms*; Springer: Berlin, Germany, 1993; Volume I and II.
41. Bagirov, A.M.; Karmitsa, N.; Mäkelä, M.M. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*; Springer: New York, NY, USA, 2014.
42. Mäkelä, M.M.; Neittaanmäki, P. *Nonsmooth Optimization*; World Scientific: Singapore, 1992.
43. Clarke, F.H. *Optimization and Nonsmooth Analysis*; John Wiley & Sons: New York, NY, USA, 1983.
44. Demyanov, V.F.; Vasilev, L.V. *Nondifferentiable Optimization*; Springer: New York, NY, USA, 1985.
45. Polyak, B.T. Minimization of unsmooth functionals. *Ussr Comput. Math. Math. Phys.* **1969**, *9*, 14–29. [CrossRef]
46. Kiwiel, K.C. Proximity control in bundle methods for convex nondifferentiable minimization. *Math. Program.* **1990**, *46*, 105–122. [CrossRef]
47. Dolan, E.D.; More, J.J. Benchmarking optimization software with performance profiles. *Math. Program.* **2002**, *91*, 201–213. [CrossRef]
48. Gould, N.; Scott, J. A note of performance profiles for benchmarking software. *Acm Trans. Math. Softw.* **2016**, *43*, 1–5. [CrossRef]