A Bundle-type Method for Nonsmooth DC Programs

Christian Kanzow* Tanja Neder*

February 11, 2022

A bundle method for minimizing the difference of convex (DC) and possible nonsmooth functions is developed. The method may be viewed as an inexact version of the DC algorithm, where each subproblem is solved only approximately by a bundle method. We always terminate the inner bundle method after the first serious step. This yields a descent direction for the original objective function, and it is shown that at least a full step is accepted in this way. Using a line search, even larger stepsizes are possible. The overall method is shown to be globally convergent to critical points of DC programs. The new algorithm is tested and compared to some other solution methods on several examples and realistic applications.

Keywords— DC optimization, Bundle method, Global convergence, Critical points

1 Introduction

The problem under consideration, called a *DC program*, is the minimization problem

$$\min \; f(x) := g(x) - h(x), \quad x \in \mathbb{R}^n, \tag{1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is the difference of two convex and possibly nonsmooth functions $g, h : \mathbb{R}^n \to \mathbb{R}$. Therefore, f is referred to as a *DC function*, g and h are the corresponding *DC components* of f (these DC components are, of course, not unique).

These DC programs occur frequently in a variety of applications, among them are the detection of edges in digital images [14], techniques utilized in data mining [4] like the Minimum Sum-of-Squares Clustering [22] or the Multidimensional Scaling problem [16], and the modeling of biochemical reaction networks [3], to name only a few.

The cornerstones for numerically tackling DC programs have already been placed some time ago, see [17] for an extensive treatment of the history of DC programs. The classical approach for solving (1) is the DC Algorithm (DCA), see, e.g., [2], which can be applied to DC programs with both DC components being nonsmooth. Provided that the first DC component

*University of Würzburg, Institute of Mathematics, Emil-Fischer-Straße 30, 97074 Würzburg, Germany; email: kanzow@mathematik.uni-wuerzburg.de, tanja.neder@mathematik.uni-wuerzburg.de

g is continuously differentiable, the convergence of DCA can often be accelerated by applying a boosted version of the classical DC Algorithm, the so called Boosted DCA (BDCA). The crucial point thereby is that the iterates computed by DCA can be used to derive descent directions for the objective function which allows to add a line search to the algorithm, see [4].

The idea of using bundle methods to solve DC programs is also not new. In [13], gathering the subgradient information for each of the DC components in two separate bundles, leads to a non-convex cutting plane model of the objective function which incorporates both the convex and the concave behavior of the DC function. The resulting proximal bundle method (PBDCA) keeps in the bundles only information related to points close to the current iterate. Another algorithm presented in [8] is also based on a cutting plane approach, but this time just the bundle with respect to the first DC component is restricted to local information, whereas the one with respect to the second DC component keeps information related to distant points. Once again, a non-convex DC piecewise-affine model is derived, which gives rise to the name DC Piecewise-Concave algorithm (DCPCA) of the resulting method. Moreover, bundle methods for the minimization of DC functions subject to some constraints have also been developed during the last few years, although these algorithms usually are restricted to a certain structure of the constraints, see, e.g., [1, 21].

In contrast to these bundle methods, our approach utilizes the standard subproblem from the classical DC algorithm. This results in a convex subproblem which is then solved inexactly by a simple bundle method. We terminate this inner bundle method after its first serious step, i.e., we do not require to solve these subproblems exactly or almost exactly, not even close to a solution. Nevertheless, the resulting inexact solution is shown to yield a descent direction, hence a line search can be applied to globalize the overall method. This line search is shown to accept at least the full step, and it even allows to take larger stepsizes, though it is known from the boosted DCA that, for both DC components being nonsmooth, it may not share the improved descent property of the boosted DCA. On the other hand, the new bundle-type DC method is shown to have nice global convergence properties for both functions g and h being nondifferentiable, whereas the boosted DCA requires g to be smooth.

The paper is organized as follows. We first recall some basic concepts and definitions as well as our basic bundle method in Section 2. We then present the new bundle-type DC algorithm in Section 3, together with a convergence theory and an additional discussion of some descent properties. The results of an extensive numerical testing are provided in Section 4. We close with some final remarks in Section 5.

2 Preliminaries

This section first recalls some basic definitions and results from nonsmooth and convex analysis, cf. [11, 23]. We then provide some details concerning a basic bundle method from, e.g., [9, 15, 18].

2.1 Tools from Nonsmooth and Convex Analysis

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *convex* if

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \quad \forall x, y \in \mathbb{R}^n,\ \forall \lambda \in (0,1).$$

It is called *uniformly convex* with modulus $\mu > 0$ if $f - \frac{\mu}{2}\|\cdot\|^2$ is convex, where $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$. Recall that uniformly convex functions always attain a unique global minimum. The (one-sided) *directional derivative* of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $x \in \mathbb{R}^n$ in the direction $d \in \mathbb{R}^n$ is defined as

$$f'(x,d) := \lim_{t \downarrow 0} \frac{f(x+td) - f(x)}{t}$$

provided that the limit on the right-hand side exists. The latter holds, in particular, for convex functions f.

Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$ and a parameter $\epsilon \geq 0$, the *ϵ-subdifferential* at a point $x \in \mathbb{R}^n$ is the set

$$\partial_\epsilon f(x) := \left\{ s \in \mathbb{R}^n \mid f(y) \geq f(x) + s^T(y-x) - \epsilon \quad \forall y \in \mathbb{R}^n \right\},$$

the special case $\partial f(x) := \partial_0 f(x)$ is known as the *(convex) subdifferential* of f at x. Each element of the latter set is called a *subgradient* of f at x. For arbitrary $\epsilon \geq 0$, the ϵ-subdifferential $\partial_\epsilon f(x)$ is a non-empty, convex and compact set for every $x \in \mathbb{R}^n$. The directional derivative of a convex function f can be calculated using its subdifferential via the formula

$$f'(x,d) = \max_{s \in \partial f(x)} s^T d. \tag{2}$$

For a locally Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}$ (recall that every convex function is locally Lipschitz), we denote the Clarke subdifferential of f in x by $\partial_C f(x)$. For a precise definition and some basic properties of the Clarke subdifferential, we refer to [6]. Note that both subdifferentials coincide for convex functions f and that $0 \in \partial_C f(x^*)$ is a necessary optimality condition for some point x^* to be a local minimum of f. Application of this optimality condition to the DC-function $f := g - h$ and using some calculus rules of the Clarke subdifferential leads to the stationarity condition

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

Each point $x^* \in \mathbb{R}^n$ satisfying (3) is called a *critical point* of the DC-function f, see also [10] for characterizations of a minimizer of DC functions. For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, the optimality condition $0 \in \partial f(x^*)$ becomes even sufficient for having a (global) minimum in $x^* \in \mathbb{R}^n$.

Given a directionally differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, we say that some vector $d \in \mathbb{R}^n$ is a *descent direction* of f at x if there exists some $t^* > 0$ such that $f(x+td) < f(x)$ for all $t \in (0, t^*]$. Note that the descent property $f'(x,d) < 0$ is a sufficient criterion for d being a descent direction.

Finally, we define the (Euclidean) distance between two sets $A,\ B \subseteq \mathbb{R}^n$ by

$$dist(A,B) := \inf \{ \|a-b\| \mid a \in A,\ b \in B \}.$$

In particular, the *projection* of a point $y \in \mathbb{R}^n$ onto a non-empty, closed, and convex set $X \subseteq \mathbb{R}^n$ is determined as the (unique) point having the least distance towards y, and we write

$$P_X(y) := \operatorname*{argmin}_{x \in X} \|y - x\|$$

for this projection.

2.2 A Bundle Method for Convex Optimization

This section gives a short introduction to a (simple) bundle method for minimizing a convex function, mainly following the references [9, 15, 18]. Note that there exist more involved bundle schemes, but the aim is to keep the presentation as simple as possible within this section. The bundle method and its convergence theory will later be used to solve the (convex, but nonsmooth) subproblems resulting in our algorithm for solving DC programs.

Therefore, consider the minimization problem

$$\min_{x\in\mathbb{R}^n} f(x) \tag{4}$$

for a convex function $f : \mathbb{R}^n \to \mathbb{R}$. Similar to the classical steepest descent method, a first idea is to compute a descent direction by solving the subproblem

$$\min f'(x^k, d) \quad \text{s.t.} \quad \|d\| \leq 1.$$

Using the relation (2), it is not difficult to see that

$$d^k = -\frac{g^k}{\|g^k\|} \quad \text{with} \quad g^k := P_{\partial f(x^k)}(0)$$

is the unique solution of this subproblem. This suggests to choose $d^k = -g^k$ as a search direction. Although d^k can indeed be verified to be a descent direction of f at x^k, simple examples show that the resulting method may not converge to a minimum, and that a successful version should include information of some neighboring subgradients. This idea leads to the search direction

$$d^k = -g^k \quad \text{with} \quad g^k = P_{\partial_\epsilon f(x^k)}(0).$$

Unfortunately, the ϵ-subdifferential is difficult to compute and projections onto this set might not be easy to calculate. The idea is then to replace the ϵ-subdifferential by an inner approximation G_ϵ^k which has a simpler structure (like being polyhedral) and which therefore allows the calculation of the projection

$$g^k := P_{G_\epsilon^k}(0) \tag{5}$$

with a significantly reduced effort. A suitable approximation G_ϵ^k can be obtained by using previously computed subgradients $s^j \in \partial f(x^j),\ j \in \{0, 1, ..., k\}$. More precisely, denoting the respective (nonnegative) *linearization errors* of f by

$$\alpha_j^k := f(x^k) - f(x^j) - (s^j)^T(x^k - x^j) \quad \forall j = 0, 1, ..., k, \tag{6}$$

the set G_ϵ^k is defined by

$$G_\epsilon^k := \left\{ \sum_{j=0}^k \lambda_j s^j \;\middle|\; \sum_{j=0}^k \lambda_j \alpha_j^k \leq \epsilon,\ \sum_{j=0}^k \lambda_j = 1,\ \lambda_j \geq 0\ \forall j = 0, 1, ..., k \right\}.$$

One can verify that this set has indeed the property that $G_\epsilon^k \subseteq \partial_\epsilon f(x^k)$ holds, which justifies

to call it an *inner approximation*. Since G_ϵ^k is a non-empty, convex, and compact set, the projections (5) do exist. Numerically, these projections require the solution of the quadratic program

$$\min \ \frac{1}{2}\Big\| \sum_{j=0}^{k} \lambda_j s^j \Big\|^2 \quad \text{s.t.} \quad \sum_{j=0}^{k} \lambda_j \alpha_j^k \le \epsilon, \quad \sum_{j=0}^{k} \lambda_j = 1, \quad \lambda_j \ge 0 \quad \forall j = 0, 1, .., k. \tag{7}$$

If $\lambda^k := (\lambda_0^k, \lambda_1^k, ..., \lambda_k^k)$ denotes a solution of this quadratic program, then the projection $g^k := P_{G_\epsilon^k}(0)$ is given by

$$g^k = \sum_{j=0}^{k} \lambda_j^k s^j.$$

In practice, the coefficients gathered in λ^k are often computed by considering the closely related quadratic program

$$\min \ \frac{1}{2}\Big\| \sum_{j=0}^{k} \lambda_j s^j \Big\|^2 + \sum_{j=0}^{k} \lambda_j \alpha_j^k \quad \text{s.t.} \quad \sum_{j=0}^{k} \lambda_j = 1, \lambda_j \ge 0 \quad \forall j = 0, 1, .., k. \tag{8}$$

Altogether, this (almost) motivates the following algorithm.

Algorithm 2.1. *(Bundle method)*

(S.0). Choose $x^1 \in \mathbb{R}^n$, $s^1 \in \partial f(x^1)$, $m \in (0,1)$, *set* $k := 1$, $y^1 := x^1$, $\alpha_1^1 := 0$, $J_1 := \{1\}$.

(S.1). Compute λ_j^k, $j \in J_k$, *as a solution of the quadratic program*

$$\min \ \frac{1}{2}\Big\| \sum_{j\in J_k} \lambda_j s^j \Big\|^2 + \sum_{j\in J_k} \lambda_j \alpha_j^k \quad \textit{s.t.} \quad \sum_{j\in J_k} \lambda_j = 1, \lambda_j \ge 0 \qquad \forall j \in J_k.$$

(S.2). Set

$$g^k := \sum_{j\in J_k} \lambda_j^k s^j, \quad \epsilon_k := \sum_{j\in J_k} \lambda_j^k \alpha_j^k, \quad d^k := -g^k, \quad \zeta_k := -\|g^k\|^2 - \epsilon_k.$$

(S.3). If $\zeta_k = 0$*: STOP.*

(S.4). Set $y^{k+1} = x^k + d^k$, *choose* $s^{k+1} \in \partial f(y^{k+1})$.
If

$$f(x^k + d^k) \le f(x^k) + m\zeta_k,$$

set ("serious step")

$$t_k := 1, \quad x^{k+1} := x^k + d^k,$$

otherwise set ("null step")

$$t_k := 0, \quad x^{k+1} := x^k.$$

(S.5). Set

$$J_k^p := \{j \in J_k \mid \lambda_j^k > 0\}, \quad J_{k+1} := J_k^p \cup \{k+1\},$$
$$\alpha_j^{k+1} := f(x^{k+1}) - f(y^j) - (s^j)^T(x^{k+1} - y^j) \quad \forall j \in J_{k+1}.$$

(S.6). Set $k \leftarrow k+1$, and go to (S.1).

In order to restrict the number of constraints in (8) and to limit the amount of subgradients and linearization errors to be stored, the index set in the algorithm is reduced to a suitable subset $J_k \subseteq \{1, ..., k\}$. Moreover, the linearization errors in (S.5) are slightly modified in comparison to (6), since the intermediate points y^j, $j \in J_k$, are also taken into account. The underlying principle is very simple: If the search direction d^k provides a sufficient decrease in the function value, one proceeds in this direction (with stepsize $t_k = 1$), otherwise one sticks with the current iterate, but adds some further information to the bundle in order to get a better search direction during the next iteration. Furthermore, the termination criterion in (S.3) gets motivated by the subsequent observation.

Lemma 2.2. *If $\zeta_k = 0$ holds for some $k \in \mathbb{N}$, then the corresponding iterate x^k is already a minimizer of the objective function f.*

This last assertion comes from the elementary observation that

$$g^k \in \partial_{\epsilon_k} f(x^k) \quad \forall k \geq 1. \tag{9}$$

In case a solution of the convex optimization problem (4) exists, one gets the following global convergence result for Algorithm 2.1.

Theorem 2.3. *Assume that the solution set $\mathcal{S} := \{x^* \in \mathbb{R}^n \mid f(x^*) = \inf_{x \in \mathbb{R}^n} f(x)\}$ is nonempty. Then every sequence $\{x^k\}$ generated by Algorithm* 2.1 *converges towards a minimizer $x^* \in \mathcal{S}$ of the objective function f.*

3 A Bundle Method for DC Optimization

This section introduces the new algorithm for DC optimization using the approach of the classical DC algorithm in combination with the previously presented bundle method. The precise statement together with a convergence theory are given in Section 3.1, whereas some additional descent properties are discussed in Section 3.2.

3.1 Algorithm and Convergence Properties

The aim of our approach is to develop an algorithm which, similar to the Boosted DCA (BDCA), computes descent directions of the objective function using the approximations arising in the classical DC Algorithm (DCA), see [2, 4]. This allows a line search to determine the subsequent iterate. In contrast to BDCA, however, the new algorithm should be applicable to DC functions with both components being nonsmooth. To gain a suitable descent direction from the convex subproblems, we allow an inexact solution of these subproblems by the previous bundle technique.

Having a DC function f as defined in (1) to be minimized, the classical DCA approach replaces, in each step $l \in \mathbb{N}_0$, the second DC component h by some linear minorization

$$h_l(x) := h(x^l) + (s^l)^T (x - x^l)$$

with a certain subgradient $s^l \in \partial h(x^l)$ to obtain a convex majorization of the objective function f. Minimizing this model function is then equivalent to minimizing

$$\phi_l(x) := g(x) - (s^l)^T x. \tag{10}$$

To guarantee the existence of a minimizer one usually assumes g to be uniformly convex, which can be done without loss of generality by adding a uniformly convex term, e.g. $\frac{\rho}{2}\|\cdot\|^2$ with $\rho > 0$, to each convex component function, if necessary. In contrast to the BDCA, the new algorithm does not require the exact minimization of the convex function ϕ_l in order to obtain a descent direction of the objective function. Instead, the bundle method from Section 2.2 is applied until a serious step is carried out. It turns out that the search direction of this step is a descent direction of f at the current iterate x^l, cf. Proposition 3.8. A subsequent line search is then used to compute the next iteration. The convergence theory shows that this line search always accepts the full step, even larger stepsizes are possible.

Algorithm 3.1. *(DCBA – DC Bundle Algorithm)*

(S.0). Choose $x^0 \in \mathbb{R}^n$, $\beta \in (0,1)$, $m \in (0,1)$, $\gamma \in (0,m]$, *set* $l := 0$.

(S.1). Choose $s^l \in \partial h(x^l)$, *and define* ϕ_l *as in* (10).

(S.2). Apply the bundle method from Algorithm 2.1 to minimize $\phi_l(x)$ *until a serious step is carried out or until it terminates. Retain* $(d^l, \epsilon_l, \zeta_l)$ *from the corresponding quantities of the serious step or the termination step, respectively.*
In case of termination of the bundle method: STOP.

(S.3). Choose $\bar{\tau}_l \geq 1$, *compute* $\tau_l = \max\left(\left\{\bar{\tau}_l\beta^j \;\middle|\; j \in \mathbb{N}_0\right\} \cup \{1\}\right)$ *such that*

$$f(x^l + \tau_l d^l) \leq f(x^l) + \gamma\tau_l^2\zeta_l.$$

(S.4). Set $x^{l+1} := x^l + \tau_l d^l$, $l \leftarrow l+1$, *and go to (S.1).*

To guarantee that (S.2) always terminates, we have to make sure that the function ϕ_l attains a minimum, cf. Theorem 2.3. Recall that this automatically holds if g is uniformly convex, hence we state this assumption explicitly in the following, which is implicitly supposed to hold throughout our convergence analysis. We stress once more, however, that this assumption is not at all restrictive since we can always add and subtract a uniformly continuous function to the DC decomposition of f.

Assumption 3.2. *The DC component* g *is a uniformly convex function.*

Some comments are in order regarding Algorithm 3.1. First note that l denotes the iteration counter for the (outer) DC-type method, whereas we will use the letter k to denote the iterations of the inner (bundle) method. Hence, $J_{k,l}$ denotes the index set that occurs in iteration k of the bundle method, called in iteration l of Algorithm 3.1. The notation $d^{k,l}$ is defined similarly.

Note that the inner bundle method executes null steps only except possibly in the last iteration. This, in particular, allows a simplified calculation of the linearization errors. Since the iterate x^l does not change in such a situation, only the computation of the linearization error corresponding to the new intermediate point is required, but not the computation for each

index $j \in J_{k,l}$. Hence, in the kth sub-iteration with search direction $d^{k,l}$ and corresponding subgradient $v^{k+1,l} \in \partial\phi_l(x^l + d^{k,l})$, the required linearization error can be obtained by

$$\alpha_{k+1,l} := \phi_l(x^l) - \phi_l(x^l + d^{k,l}) + (v^{k+1,l})^T d^{k,l}, \tag{11}$$

where $d^{0,l} := 0$ for all $l \in \mathbb{N}_0$. Furthermore, a subgradient $v^{k+1,l} \in \partial\phi_l(x^l + d^{k,l})$ can be computed by selecting some element $t^{k+1,l} \in \partial g(x^l + d^{k,l})$ and setting $v^{k+1,l} := t^{k+1,l} - s^l$.

The line search is an Armijo-type one with the slight modification of looking for a decrease in the function value of at least $\gamma\tau_l^2(-\zeta_l)$. At a first glance, one might expect $\|d^l\|^2$ instead of the enlarged value $-\zeta_l = \|d^l\|^2 + \epsilon_l$. This adjustment is motivated by Lemma 3.4 below. In addition, the initial stepsize $\bar{\tau}_l$ can be determined as a self-adaptive trial stepsize like the one suggested in [4]. One only needs to ensure $\bar{\tau}_l \geq 1$.

Before proving a global convergence result for Algorithm 3.1, we begin with some preliminary observations. To this end, we first justify the termination criterion in (S.2).

Lemma 3.3. *Suppose $\zeta_l = 0$ holds for some l. Then the current iterate x^l is a critical point of the objective function f.*

Proof. As the iterate does not change during the bundle process, having $\zeta_l = 0$ for some l, Lemma 2.2 implies that x^l minimizes ϕ_l. Hence, we have $0 \in \partial\phi_l(x^l) = \partial g(x^l) - s^l$. On the other hand, $s^l \in \partial h(x^l)$ by our choice, consequently $\partial g(x^l) \cap \partial h(x^l) \neq \emptyset$ follows, showing that x^l is indeed a critical point of the DC function f. □

Motivated by Lemma 3.3, we assume, from now on, that $\zeta_l < 0$ holds for all l, i.e., Algorithm 3.1 does not stop after finitely many iterations. The following result then shows that the Armijo-type line search is always well-defined (together with Assumption 3.2 this implies that the entire Algorithm 3.1 is well-defined), and that the full step satisfies the line search criterion.

Lemma 3.4. *At each iteration l, there exists a stepsize τ_l, $\tau_l = 1$ or $\tau_l = \bar{\tau}_l\beta^j \geq 1$ with some $j \in \mathbb{N}_0$, such that*

$$f(x^l + \tau_l d^l) \leq f(x^l) + \gamma\tau_l^2\zeta_l \tag{12}$$

holds.

Proof. Let l be arbitrarily chosen. Then

$$\begin{aligned} f(x^l + d^l) - f(x^l) &= g(x^l + d^l) - g(x^l) - h(x^l + d^l) + h(x^l) \\ &\leq g(x^l + d^l) - g(x^l) - (s^l)^T d^l \\ &= \phi_l(x^l + d^l) - \phi_l(x^l) \\ &\leq m\zeta_l \leq \gamma\zeta_l, \end{aligned}$$

where the first inequality exploits the fact that $s^l \in \partial h(x^l)$, the penultimate inequality comes from the serious step termination of the inner bundle method, and the last estimate uses $\gamma \in (0, m]$ as well as $\zeta_l < 0$ (see the previous discussion). This shows that at least $\tau_l = 1$ has the desired property. Taking into account the construction of the stepsize yields the desired claim. □

Finally, we need the following auxiliary result for proving global convergence of Algorithm 3.1.

Lemma 3.5. *Assume that Algorithm* 3.1 *generates an infinite sequence* $\{x^l\}$. *Then the sequence* $\{f(x^l)\}$ *is monotonically decreasing. If, in addition, there exists a lower bound* $f^* \in \mathbb{R}$ *such that* $f(x^l) \geq f^*$ *holds for all* l, *then the estimate*

$$\sum_{l=1}^{\infty} \left(\|d^l\|^2 + \epsilon_l \right) \leq \frac{f(x^0) - f^*}{\gamma}$$

holds. In particular, we then have $d^l \to 0$ *and* $\epsilon_l \to 0$ *for* $l \to \infty$.

Proof. The monotonicity of the function values follows directly from ζ_l being negative and the line search in (S.3).

To establish the second assertion, note that (S.3) can be written as $-\gamma \tau_l^2 \zeta_l \leq f(x^l) - f(x^{l+1})$ for all l. Taking the sum over $l = 0, ..., j-1$, we get

$$\gamma \sum_{l=0}^{j-1} \tau_l^2 (-\zeta_l) \leq f(x^0) - f(x^j) \leq f(x^0) - f^* \qquad \forall j \in \mathbb{N}$$

by the boundedness assumption. Letting $j \to \infty$ therefore gives

$$\sum_{l=0}^{\infty} \tau_l^2 (-\zeta_l) \leq \frac{f(x^0) - f^*}{\gamma}.$$

Using $\tau_l \geq 1$ and inserting the definition of ζ_l gives the desired inequality. □

The following is the main global convergence result for Algorithm 3.1.

Theorem 3.6. *Every accumulation point of a sequence* $\{x^l\}$ *generated by Algorithm* 3.1 *is a critical point of the objective function* f.

Proof. Let x^* be an accumulation point of the sequence $\{x^l\}$ and $\{x^l\}_L$ be a corresponding subsequence converging to x^*. Since $s^l \in \partial h(x^l)$ for all l, the convergence of $\{x^l\}_L$ implies the boundedness of the sequence $\{s^l\}_L$. Hence, without loss of generality, we may assume that $\{s^l\}_L$ converges to some limit s^*. Due to the closedness property of the convex subdifferential, it follows that $s^* \in \partial h(x^*)$.

By the continuity of f, we have $f(x^l) \to_L f(x^*)$. Hence, the monotonicity of the entire sequence $\{f(x^l)\}$ yields convergence of the entire sequence $\{f(x^l)\}$ to $f(x^*)$. The monotonicity also implies $f(x^l) \geq f(x^*)$ for all l. Thus, the previous lemma can be applied to obtain $d^l \to 0$ and $\epsilon_l \to 0$ for $l \to \infty$.

Furthermore, since we have $-d^l \in \partial_{\epsilon_l} \phi_l(x^l)$ in view of (9), we get

$$\phi_l(x) \geq \phi_l(x^l) - (d^l)^T (x - x^l) - \epsilon_l \qquad \forall x \in \mathbb{R}^n \ \forall l \in \mathbb{N}.$$

Using the definition of ϕ_l, this can be rewritten as

$$g(x) - (s^l)^T x \geq g(x^l) - (s^l)^T x^l - (d^l)^T (x - x^l) - \epsilon_l \qquad \forall x \in \mathbb{R}^n \ \forall l \in \mathbb{N}.$$

Taking $l \to_L \infty$ and exploiting the continuity of g therefore yields

$$g(x) - (s^*)^T x \geq g(x^*) - (s^*)^T x^* \qquad \forall x \in \mathbb{R}^n$$

or, equivalently,

$$g(x) \geq g(x^*) + (s^*)^T(x - x^*) \qquad \forall x \in \mathbb{R}^n.$$

Consequently, we have $s^* \in \partial g(x^*)$. Together with $s^* \in \partial h(x^*)$, this shows that $\partial g(x^*) \cap \partial h(x^*) \neq \emptyset$, hence x^* is a critical point of the DC function f. □

Recall that we terminate our method(s), for our theoretical considerations, only if $\zeta_l = 0$. Numerically, one should replace this condition in (S.3) of the bundle method 2.1 by a more practical condition like

$$|\zeta_k| < \delta \quad \text{or, equivalently,} \quad \zeta_k > -\delta \tag{13}$$

with some given tolerance $\delta > 0$. The following result then shows that Algorithm 3.1 terminates after finitely many iterations in a point which approximately satisfies the condition of being a critical point of the DC function f.

Theorem 3.7. *Assume that f is bounded from below. Then Algorithm* 3.1*, with the modified termination criterion* (13)*, terminates after finitely many iterations in a point x^L satisfying*

$$dist\left(\partial_{\epsilon_L} g(x^L), \partial h(x^L)\right) < \sqrt{\delta} \quad \textit{with} \quad \epsilon_L < \delta. \tag{14}$$

Proof. First recall that, in each outer iteration l, due to Assumption 3.2, the bundle step (S.2) terminates after a finite number of inner iterations either meeting the termination criterion or detecting a descent direction d^l. This is based on the fact that carrying out only null steps within the bundle iteration leads to $\{\zeta_k\}$ tending to zero, see, e.g., [9], hence the condition (13) eventually holds.

We next show that Algorithm 3.1 terminates after finitely many iterations. Assume, by contradiction, that an infinite sequence $\{x^l\}$ is generated. Then each call of the bundle method ends with a serious step and hence the computation of a descent direction d^l. The subsequent line search then yields

$$f(x^{l+1}) \leq f(x^l) + \gamma \tau_l^2 \zeta_l \leq f(x^l) + \gamma \zeta_l \qquad \forall l \in \mathbb{N},$$

where the final inequality results from $\tau_l \geq 1$ and ζ_l being negative. Summation over $l = 0, ..., j-1$ gives

$$f(x^j) - f(x^0) \leq \gamma \sum_{l=0}^{j-1} \zeta_l \leq -\gamma \delta j \qquad \forall j \in \mathbb{N},$$

since $\zeta_l \leq -\delta$ for all l by assumption (the inexact termination criterion never holds). Letting $j \to \infty$, the right-hand side tends to $-\infty$, whereas the left-hand side is bounded from below by assumption. This contradiction shows that Algorithm 3.1 terminates within a finite number of iterations.

Let x^L denote the point of termination. It remains to show that x^L satisfies the properties from (14). To this end, we first note that a simple calculation shows that $\partial_\epsilon \phi_l(x) = \partial_\epsilon g(x) - s^l$ holds for all $l \in \mathbb{N}$, $\epsilon \geq 0$, and $x \in \mathbb{R}^n$. Since $-d^l \in \partial_{\epsilon_l} \phi_l(x^l)$ by (9), we therefore obtain the existence of an element $\tilde{t}^l \in \partial_{\epsilon_l} g(x^l)$ such that $-d^l = \tilde{t}^l - s^l$. Together with the fact that $s^l \in \partial h(x^l)$, we get

$$dist\big(\partial_{\epsilon_l} g(x^l), \partial h(x^l)\big) = \inf\left\{\|t - s\| \;\middle|\; t \in \partial_{\epsilon_l} g(x^l),\ s \in \partial h(x^l)\right\} \leq \left\|\tilde{t}^l - s^l\right\| = \|d^l\| \leq \sqrt{|\zeta_l|}, \tag{15}$$

where the last inequality just exploits the definition of ζ_l. This definition also implies $\epsilon_l \leq |\zeta_l|$. Since, upon termination, we have $|\zeta_L| < \delta$, the two estimates (14) follow. □

Note that (14) can be seen as an approximation of

$$dist\,(\partial g(x^*), \partial h(x^*)) = 0. \tag{16}$$

Due to the closedness of the subdifferential, (16) is equivalent to $\partial g(x^*) \cap \partial h(x^*) \neq \emptyset$, which means that x^* is a critical point of the DC function f. We may therefore view the point of termination as an approximate critical point of the objective function.

3.2 Descent Properties of Search Directions

The subject of this section is to discuss some additional properties of the search direction d^l. We will see that d^l is indeed a descent direction of the objective function f at the current iterate x^l, but, in general, not in the point $x^l + d^l$. Note that this latter property holds for the boosted DCA method from [4] if g is smooth. We also discuss a modified version for determining a suitable stepsize in (S.3).

Note that the convergence theory in Section 3.1 is completely independent of any descent properties of d^l. In particular, the line search in (S.3) makes no explicit use of this feature (see also Lemma 3.4). Nevertheless, it is interesting to see that d^l is indeed a descent direction of f in x^l.

Proposition 3.8. *The directional derivative of the objective function satisfies*

$$f'(x^l, d^l) \leq \phi_l'(x^l, d^l) < 0 \quad \forall l \in \mathbb{N}.$$

Hence, d^l is a descent direction of f in x^l.

Proof. Let l be fixed. We then obtain

$$f'(x^l, d^l) = g'(x^l, d^l) - h'(x^l, d^l) = g'(x^l, d^l) - \max_{s \in \partial h(x^l)} s^T d^l \leq g'(x^l, d^l) - (s^l)^T d^l = \phi_l'(x^l, d^l),$$

where the relation (2) together with $s^l \in \partial h(x^l)$ was exploited. By construction, d^l results from a serious step of the bundle method, hence $\phi_l(x^l + d^l) < \phi_l(x^l)$ holds. A standard characterization of the directional derivative therefore yields

$$\phi_l'(x^l, d^l) = \inf_{t>0} \frac{\phi_l(x^l + td^l) - \phi_l(x^l)}{t} \leq \frac{\phi_l(x^l + d^l) - \phi_l(x^l)}{1} < 0.$$

Putting both estimates together completes the proof. □

Recall that Lemma 3.4 shows that a full step in the direction d^l is always accepted by the line search criterion (S.3). Using a minor modification in the bundle procedure ensures that DCBA even allows to take a stepsize τ_l strictly larger than one. The details are given in the subsequent proposition.

Proposition 3.9. *Assume that we replace the inequality in (S.4) of the Bundle Algorithm 2.1 by a strict one. Then there exists a stepsize $\tau_l > 1$ such that* (12) *holds.*

Proof. For arbitrary l, one can follow the proof of Lemma 3.4 to see that the sharp estimate

$$f(x^l + d^l) < f(x^l) + \gamma\zeta_l$$

holds. Consequently, (12) is satisfied for $\tau_l = 1$, but with a strict inequality. Since the mapping $\tau \mapsto f(x^l + \tau d^l) - f(x^l) - \gamma\tau^2\zeta_l$ is continuous, (12) holds on an interval $[1, \tau_*)$ for some $\tau_* > 1$ (depending on l). This completes the proof. □

The previous result motivates to search for a suitable stepsize by using a strategy like

$$\tau_l = \max\left\{1 + \bar{\tau}_l\beta^j \;\middle|\; j \in \mathbb{N}_0\right\} \quad \text{such that} \quad f(x^l + \tau_l d^l) \leq f(x^l) + \gamma\tau_l^2\zeta_l$$

for some $\beta \in (0, 1)$, where $\bar{\tau}_l$ can be determined by the self-adaptive trial stepsize strategy introduced in [4]. In this way, our approach shares some properties of the boosted version of DCA, but for general nonsmooth functions g and h.

At a first glance, this modified stepsize seems to be rather promising, and is more likely to yield a stepsize larger than one than the original stepsize rule from (S.3). However, numerical tests indicate that the overall results are often better for the original stepsize rule, at least for most applications considered in this work. The computation of stepsizes larger than one by the modified rule sometimes leads to the situation where the method crosses a valley with a one-dimensional minimizer, ending up in a region of ascent again. Therefore, the progress in the function value of the objective is often less than accepting a stepsize of one. As this behavior accumulates, one keeps jumping from one side of the valley to the other, sometimes even on a straight line. We illustrate this behavior in Figure 1 where Example 3.10 (see below) is used, with the standard choice of parameters given in Section 4.1 and initial point $(x^0, y^0)^T := (2.5, 1)^T$. It is remarkable that the vector provided by the bundle method often seems to be a pretty good choice for updating the iterate without any scaling.

Recall that Lemma 3.4 and Proposition 3.9 show that the full step in the direction d^l is always accepted by our line search rule(s). Similar to [4], one may therefore ask whether d^l is also a direction of descent at the point $x^l + d^l$. Note, however, that this cannot be expected for nonsmooth functions g since it was already shown in [4] that this property dos not hold, in general, even if the DC subproblems are solved exactly. It is therefore not surprising to see that this descent property is also violated for our inexact solution d^l of the DC subproblem. This is shown by the following counterexample taken from [4].

Example 3.10. (Failure of a boosted version of DCBA)
Consider the function

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x, y) := -\frac{5}{2}x + \frac{1}{2}(x^2 + y^2) + |x| + |y|$$

with uniformly convex DC-component functions $g, h : \mathbb{R}^2 \to \mathbb{R}$ chosen as

$$g(x, y) := -\frac{5}{2}x + x^2 + y^2 + |x| + |y|, \quad h(x, y) := \frac{1}{2}(x^2 + y^2).$$

Taking $(x^0, y^0)^T := (0.5, 0.1)^T$ as a starting point, the bundle method applied to $\min_{x\in\mathbb{R}^2} \phi_0(x, y)$ with $m = 0.1$ stops after $k = 2$ iterations with the detection of the descent direction

$$d^0 \approx (0.61091, -0.28290)^T.$$

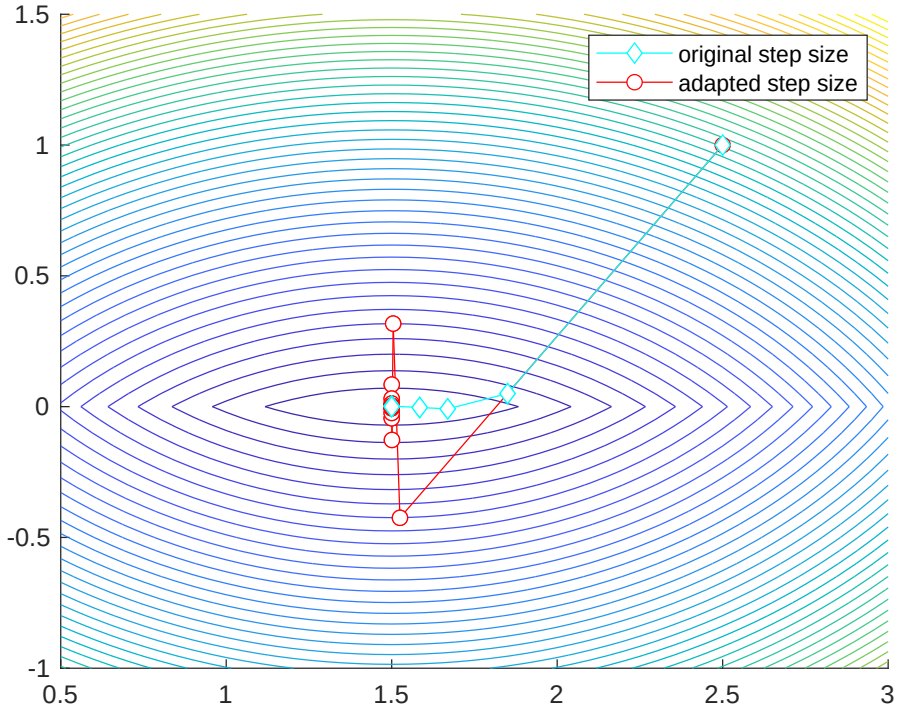


Figure 1: Variants of stepsize strategies

Let us note, for the sake of completeness, that selecting m in a different way would not open the door for gaining a descent direction in the first iteration, as the first search direction $(d_1^{1,0}, d_2^{1,0})^T$ satisfies $\phi_0(x^0 + d_1^{1,0}, y^0 + d_2^{1,0}) - \phi_0(x^0, y^0) = 2 > 0$.

Before verifying analytically that d^0 is indeed not a descent direction of f at $(x^0, y^0)^T + d^0$, let us have a look at Figure 2, which contains a contour plot of the function f, revealing the basic situation. Starting at $(x^0, y^0)^T = (0.5, 0.1)^T$ heading in direction d^0, one initially achieves a decrease in the function value, as it was expected because of d^0 being a descent direction of f at $(x^0, y^0)^T$. But proceeding further (below the line $y = 0$ to be more precise), one leaves the region of descent, entering a region of ascent. Accepting a full step $\tau_0 = 1$, one touches the region of ascent. Therefore, moving further in direction d^0 from $(x^0, y^0)^T + d^0$ would result in a continuing increase of the function value, demonstrating that d^0 is not a descent direction of f at the point $(x^0, y^0)^T + d^0$. Analytically this claim is confirmed by the corresponding scalar product

$$\nabla f(x^0 + d_1^0, y^0 + d_2^0)^T d^0 \approx 0.09695$$

being positive (note that one can indeed consider the gradient of the objective, as the considered point is located in a region where f is differentiable). In addition, this instance shows that the search direction d^0 is not running tangential towards the contour line through $(x^0 + d_0^1, y^0 + d_2^0)^T$, which is marked pink in Figure 2, although it might seem to be the case at first glance.

Recall that [4] shows, for continuously differentiable functions g, that the exact solution d^l of the convex DC subproblem yields a descent property of f at the new point $x^l + d^l$, leading itself to the boosted version BDCA of the DCA method. This property does not necessarily hold for our approach where d^l is only computed as an inexact solution of the DC subproblem.

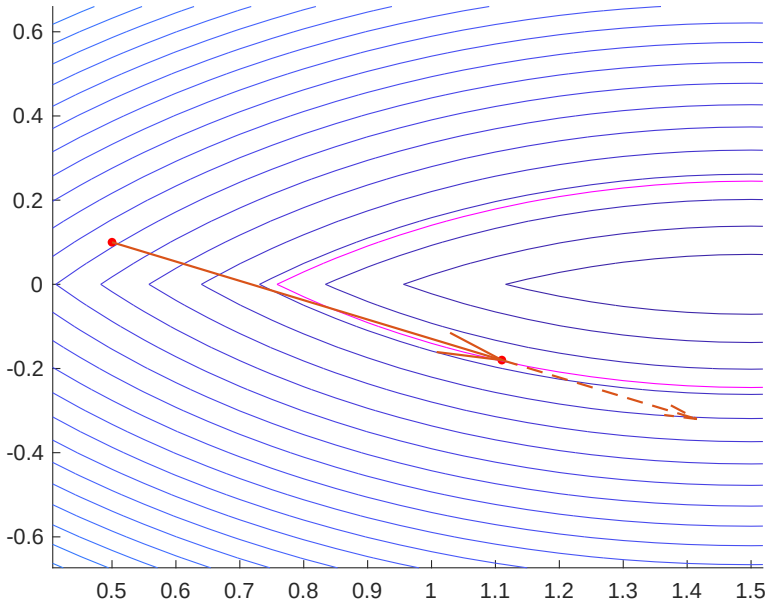


Figure 2: Change of the function value in direction d^0 starting at $(0.5, 0.1)^T$

This is illustrated in the following example which results from the previous one by replacing the absolute value function using a scaled and shifted version of Huber's loss, see [12].

Example 3.11. (Failure of a boosted version of DCBA in case of smooth DC components) Consider the modified function

$$\tilde{f} : \mathbb{R}^2 \to \mathbb{R}, \quad \tilde{f}(x, y) := -\frac{5}{2}x + \frac{1}{2}(x^2 + y^2) + \psi_\epsilon(x) + \psi_\epsilon(y)$$

with

$$\psi_\epsilon : \mathbb{R} \to \mathbb{R}, \quad \psi_\epsilon(x) := \begin{cases} |x| & \text{if } |x| \geq \epsilon, \\ \frac{1}{2\epsilon}x^2 + \frac{\epsilon}{2} & \text{if } |x| < \epsilon \end{cases}$$

being the described smooth adaption of the absolute value function for sufficiently small $\epsilon > 0$, e.g., $\epsilon = 10^{-3}$. Similarly to the previous example the uniformly convex DC-component functions $\tilde{g}, \tilde{h} : \mathbb{R}^2 \to \mathbb{R}$ are chosen as

$$\tilde{g}(x, y) := -\frac{5}{2}x + x^2 + y^2 + \psi_\epsilon(x) + \psi_\epsilon(y), \quad \tilde{h}(x, y) := \frac{1}{2}(x^2 + y^2).$$

Note that the smooth modification $\tilde{f}$ coincides with the function f from the previous example outside the interval $I_\epsilon := (-\epsilon, \epsilon)$. Therefore, taking $\epsilon > 0$ sufficiently small, and starting again in $(x^0, y^0)^T = (0.5, 0.1)^T$, the approximation function $\tilde{\phi}_0$ matches the corresponding one from Example 3.10 at least on the relevant part of the domain of definition, namely outside the interval I_ϵ. Thus, all calculations from Example 3.10 remain valid, as we stay outside the critical interval I_ϵ during the whole computation. Consequently, also in this case the accepted search direction turns out not to be a descent direction of the objective $\tilde{f}$ at the intermediate point $(x^0 + d_1^0, y^0 + d_2^0)^T$.

The previous example, of course, shows that our approach has a drawback compared to the boosted DCA. Recall, however, that our primary aim was to develop a method for the solution of DC programs where both component functions g and h are nonsmooth, and in this situation, the descent property of d^l cannot be expected at the point $x^l + d^l$ even for the exact solution of the DC subproblem.

4 Numerical Experiments and Applications

This section presents some numerical experiments of the new algorithm DCBA and gives a comparison with some existing solvers for DC programs. In particular, our method is compared with the solvers DCA [2, 4], BDCA [4], PBDCA [13], and DCPCA [8], which are briefly reviewed in Section 4.1, together with some details of our implementation. The numerical experiments are then carried out using an academic test problem [4] as well as some examples arising from applications, namely Minimum Sum-of-Squares Clustering [4, 22], Multidimensional Scaling [4, 16], and edge detection by means of a clustering technique [14].

4.1 Methods and Implementation

This section first provides some details of the DC solvers that are used in our numerical studies. The standard method for solving DC problems is the DCA [2, 4], which can be accelerated to a boosted version, namely the BDCA [4], in suitable cases. In addition, we use two bundle methods, PBDCA [13] and DCPCA [8]. A brief overview of these algorithms is given in Table 1.

As already noted, DCA derives, in each iteration, a convex majorization of the objective function by approximating the second DC component by some linear minorization. The minimizer of this model function yields the next iteration, i.e., DCA solves the subproblems exactly and uses no line search globalization, see [2, 4].

BDCA was introduced in [4] and is an accelerated version of DCA, which is based on the observation that, in case of a smooth first DC component, the solution of the DC subproblem gives rise to a descent direction at the point that is accepted by DCA as the next iterate. The latter detection allows to add a line search for speeding up the convergence.

PBDCA is described in [13]. This bundle method constructs two separate cutting plane models, one for each DC component. Combining both leads to a piecewise linear, non-convex model of the objective function which incorporates the convex behavior of the DC function as well as its concave one. The computation of the search direction uses a stabilizing term which includes a proximity parameter, thus making a line search superfluous. The termination criterion directly refers to the definition of a critical point of the DC function (see (3)) and estimates the distance between the respective two subdifferentials.

DCPCA originates from [8]. Similar to PBDCA, it develops two separate cutting plane models, one for each component, initially leading to a non-convex piecewise linear approximation. The two DC components, however, are not treated equally. The bundle related to the first component is restricted to local information only, whereas the bundle concerning the second component is not. The resulting model for the computation of a search direction, a pointwise maximum of concave functions, is then approximated by a local quadratic program. In case no satisfactory solution can be found, the method switches to an auxiliary (also quadratic) program. Having found an appropriate search direction, a line search follows.

abbreviation	denomination	reference
DCA	Difference of Convex Algorithm	[2, 4]
BDCA	Boosted DCA	[4]
DCBA	DC Bundle Algorithm	Algorithm 3.1
PBDCA	Proximal Bundle DCA	[13]
DCPCA	DC Piecewise-Concave Algorithm	[8]

Table 1: Summary of the methods used in our numerical experiments

In order to achieve a better comparability of the numerical results, the termination criteria of the different algorithms were adapted to some extent. While BDCA stops whenever the computed descent direction d^l is (close to) zero, for DCBA the sum of the squared norm of the search direction plus the ϵ-tolerance of the current approximation of the subdifferential is checked to be (close to) zero. This motivates to split the termination criterion for DCBA into two parts, namely

$$\|d^l\| < \varepsilon_1 \quad \text{and} \quad \epsilon_l < \varepsilon_2$$

with some given tolerances $\varepsilon_1,\ \varepsilon_2 > 0$. Note that suppressing the square of the norm in the first condition is not essential. Accordingly, the termination criterion for BDCA is inherited as $\|d^l\| < \varepsilon_1$. As the stopping condition for DCA and BDCA coincide, it is clear how to choose the respective one for DCA.

Comparing the convergence theorems for PBDCA, see Theorem 6 in [13], which ensures that the approximate criticality condition $dist\left(\partial_\epsilon f_1(x^*), \partial_\epsilon f_2(x^*)\right) \leq \delta$ is satisfied in the point of termination x^*, with the corresponding Theorem 3.7 for DCBA, which results in the final estimate $dist\left(\partial_{\epsilon_L} g(x^L), \partial h(x^L)\right) < \varepsilon_1$ with $\epsilon_L < \varepsilon_2$, motivates to keep

$$\|\xi_1^* - \xi_2^*\| < \varepsilon_1$$

with the termination tolerance $\delta = \varepsilon_1$ as a stopping test for PBDCA. Furthermore, this parallelism suggests to take $\varepsilon_2 = \epsilon$ as well.

Having in mind Remark 2 of [8], one gets a direct connection of the termination criterion for DCPCA to the one for DCBA, which gives rise to adapt the stopping condition of DCPCA in the same manner as the one for DCBA towards

$$\|\bar{d}\| < \varepsilon_1 \quad \text{and} \quad -\|\bar{d}\|^2 - \bar{v} = \sum_{i \in I} \bar{\lambda}_i \alpha_i^{(1)} < \varepsilon_2$$

(the notation is taken from [8]). Note that both, $\alpha_i^{(1)}$ in terms of DCPCA and $\alpha_{k+1,l}$ in terms of DCBA, denote linearization errors corresponding to the first DC component. Indeed the latter one was defined as a linearization error of the approximation ϕ_l in (11), but, as this function differs from the first component g only by a linear function, $\alpha_{k+1,l}$ in fact yields the linearization error with respect to g. This last modification ensures that for DCPCA in the point of termination x^* the estimate $dist\left(\partial_{\varepsilon_2} f_1(x^*), f_2(x^*)\right) \leq \varepsilon_1$ similar to the previous ones for PBDCA and DCBA holds true.

Note that even though choosing $\varepsilon_1 \ll \varepsilon_2$ in the presented examples, in case of DCBA as well as DCPCA, the second termination quantity related to the linearization errors undershot

even the lower critical tolerance ε_1 in the point of termination, with the only exception of reproducing the map of Bavaria by means of Multidimensional Scaling in Section 4.4. For this special instance, both algorithms ended with a precision of 0.07, while having a comparatively large $\varepsilon_1 = 10^{-2}$.

The codes were implemented in GNU Octave 5.1.0 and run on a Radeon Vega Mobile Gfx 2.00 GHz computer with AMD Ryzen 5 2500U CPU and 8.00 GB RAM under Windows 10 (64-bit). The only exception is the reproduction of the map of Bavaria by Multidimensional Scaling, where the corresponding tests were implemented in Matlab version 2020b and executed on a 8xIntel®Core$^{\text{TM}}$i7-7700 CPU @ 3.60 GHz computer with 31.1 GiB RAM under an open SUSE Leap 15.3 (64-bit) system.

In all numerical experiments, the initial stepsize of every line search procedure contained in BDCA as well as DCBA gets computed by means of a self-adaptive trial stepsize strategy introduced in [4]. Furthermore, the proximity parameter $t \in [t_{min}, t_{max}]$ for PBDCA is chosen as the arithmetic mean of the limits of the feasible domain. The quadratic programs occurring in the bundle methods DCBA, PBDCA and DCPCA are solved using the `quadprog` command, with the exception of the reproduction of the Bavaria map by means of Multidimensional Scaling, where the spectral gradient method is used in the context of DCBA, cf. [5]. The solution methods applied to the convex subproblems of DCA and BDCA differ depending on the application. Hence, they will be given at the respective sections.

For most of the numerical tests, a similar parameter setting was used. Unless said otherwise, the termination tolerances were set to $\varepsilon_1 = 10^{-3}$ and $\varepsilon_2 = 10^{-1}$, and referring once again to the notation in the respective papers given at the beginning of this chapter, the remaining parameters were chosen as $\alpha = 0.1,\ \beta = 0.5, \gamma = 4$, and $\bar{\lambda}_1 = 4$ for BDCA. The last two parameters concerning the self-adaptive trial stepsize strategy are also used for DCBA. In addition, the remaining parameters for this algorithm were set to $\beta = 0.5,\ m = 0.5$ and $\gamma = 0.1$. The missing parameters for PBDCA were chosen as $m = 0.5,\ R = 10^7,\ L_1 = L_2 = 1000$ as well as the maximum size of the second bundle as 3 and $r = 0.75$ whenever the spacial dimension n is less than 10, $r = 0.99$ in case of $n \geq 300$, and $r = \lfloor \frac{n}{n+5} \cdot 100 \rfloor / 100$ else. This last bunch of selections was taken from [13]. Moreover, for DCPCA, the still missing parameters were taken as $\eta = 0.7,\ m = 0.5,\ \sigma = 0.5$ and $\rho = 0.95$, these choices are motivated by [8].

4.2 An Academic Test Problem

The essential aim of examining the subsequent academic test problem is to investigate how often the algorithms under consideration converge to the known minimum of the objective function and not just to a critical point. This test setting is inspired by Example 3.1 in [4].

For the objective function $f : \mathbb{R}^2 \to \mathbb{R}$ with

$$f(x, y) := x^2 + y^2 + x + y - |x| - |y| \qquad x, y \in \mathbb{R},$$

the DC-composition $f := g - h$ to be examined is chosen as

$$g(x, y) := \frac{3}{2}(x^2 + y^2) + x + y, \qquad h(x, y) := |x| + |y| + \frac{1}{2}(x^2 + y^2) \qquad x, y \in \mathbb{R},$$

so that the component functions g, h are uniformly convex. The global minimum is at $(-1, -1)$, but there exist three additional critical and non-optimal points $(-1, 0),\ (0, -1)$, and $(0, 0)$. To investigate the ability of the different solvers to find the optimal point, 10,000 test runs for

minimizing f were considered. Thereby, the initial points were chosen quasi-randomly from the rectangle $[-1.5, 1.5]^2$, and the sequences converging to each of the stationary points were counted. For solving the convex subproblems arising in DCA and BDCA, a gradient method was applied. The result is shown in Table 2. DCBA is the only method finding the minimizer in every single instance. Also PBDCA and BDCA were successful in 99.9% of the test cases and DCPCA in 97.7%. On the other hand, DCA converges to each of the four critical points more or less the same number of times, leading to a success rate in determining the optimum of 24.9%. Although, at this point, no detailed information regarding further test quantities is provided, let us note that DCBA requires, on average, only two and at most three iterations until termination, which is less than half the number needed by all other algorithms.

	$(-1,-1)$	$(-1,0)$	$(0,-1)$	$(0,0)$
DCA	2,487	2,424	2,523	2,566
BDCA	9,989	5	6	0
DCBA	10,000	0	0	0
PBDCA	9,992	7	0	1
DCPCA	9,772	97	131	0

Table 2: Absolute frequency of sequences converging to the respective critical point by the different DC algorithms

4.3 The Minimum Sum-of-Squares Clustering Problem

We first provide a short introduction of the Minimum Sum-of-Squares Clustering method and then present two test settings that are examined afterwards, one related to randomly generated data and another one referring to real data in the form of geographic coordinates of Bavarian cities. The reformulation of the underlying optimization problem in DC form can be found in [4, 22]. The first experiment gets evaluated with the help of performance profiles, see [7].

Clustering describes the separation of a data set into disjoint subsets, so called clusters, by gathering points of similarity. The method is used in data mining for the analysis of huge data sets to get a better (or condensed) overview of the information actually contained in the given data. Thereby, the measure of similarity may differ depending on the application. In the following, each cluster gets characterized by its centroid and the classification gets done by considering the (minimal) squared Euclidean distance of each data point towards these centroids. Hence, denoting with $A := \{a^1, ..., a^n\}$ the set of points $a^i \in \mathbb{R}^m,\ i = 1, ..., n$, to be partitioned, and letting k be the desired number of clusters, the aim is to determine k centroids $x^j \in \mathbb{R}^m,\ j = 1, ..., k$, such that the (averaged) sum over the squared distance of each data point towards the corresponding centroid gets minimal. Thus, using the notation $X := (x^1, ..., x^k) \in \mathbb{R}^{m \times k}$, the problem under consideration is

$$\min_{X \in \mathbb{R}^{m \times k}} f(X) := \frac{1}{n} \sum_{i=1}^{n} \min_{j=1,...,k} \left\| a^i - x^j \right\|^2.$$

In [22], a DC-composition of f is derived. Adding a quadratic term to each component function

$g, h : \mathbb{R}^{m \times k} \to \mathbb{R}$ like suggested in [4], one obtains

$$g(X) := \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \left\| a^i - x^j \right\|^2 + \frac{\rho}{2} \|X\|^2,$$

$$h(X) := \frac{1}{n} \sum_{i=1}^{n} \max_{j=1,\ldots,k} \sum_{\substack{t=1,\ldots,k \\ t \neq j}} \left\| a^i - x^t \right\|^2 + \frac{\rho}{2} \|X\|^2,$$

where a modulus $\rho > 0$ ensures uniform convexity and $\|X\|$ denotes the Frobenius norm of the matrix X. The following tests were carried out using $\rho = 0.1$.

In a first numerical experiment for a varying combination of parameters, n quasi-randomly generated vectors in $\mathbb{R}^m$ with normally distributed entries having a mean of zero and a standard deviation of ten got to be clustered in k groups. Thereby, $n \in \{500, 750, 1000\}$, $m \in \{2, 5, 10\}$, and $k \in \{5, 7, 10\}$ were considered. For each triple of parameters, ten test runs with all algorithms listed at the beginning of this section were passed, starting with k vectors in $\mathbb{R}^m$ as centroid candidates generated according to the same rules as the points to be clustered. Once again, the gradient method was applied for solving the convex subproblems in DCA and BDCA. In the process, methods got rated successful whenever reaching the smallest function value in comparison. In addition, they were considered as failed in case the running time exceeded two hours for one single test run. This last incidence only occurred three times while applying DCA. Throughout this experiment, the number of iterations as well as the running time was reported and evaluated by means of the performance profiles introduced in [7]. The resulting diagrams are shown in Figure 3. Note that differences between the profiles regarding time and iterations arise from the fact that the iterations of the respective methods are not directly comparable in view of computational effort. Nevertheless, it is obvious that BDCA, which was specifically designed for DC problems with smooth first component, outperforms all other algorithms, both in terms of the quality of the function value and in terms of computation time. Among the remaining four algorithms, the bundle methods DCBA and PBDCA perform best, while DCPCA as well as DCA already require, on average, significantly more iterations and computation time.

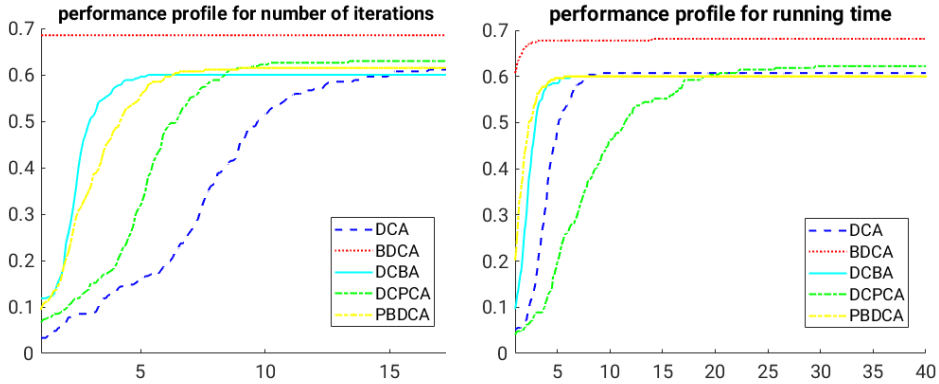


Figure 3: Performance profiles of the Minimum Sum-of-Squares Clustering problem with random data

In a second illustrative example, we investigate how the administrative districts of Bavaria would look like if their cities got grouped by the Minimum Sum-of-Squares Clustering method. To this end, a data set of geographic coordinates of $n = 2{,}073$ Bavarian cities and towns was considered[1], which consequently got to be separated in $k = 7$ clusters. As initial guess, seven vectors with quasi-random entries in the range $[9.06, 13.80] \times [47.41, 50.52]$ of the geographic coordinates of the considered data set were selected. All five algorithms determine (approximately) the same seven centroids, but differ in convergence speed. In Figure 4, the resulting clusters are shown with their centroids marked as pentagrams next to the first ten iterations of each method beside the ones of DCBA. Moreover, in Figure 5 the evolution of the function value with proceeding iterations for each algorithm is plotted. Essentially, the differences in convergence speed are similar to the experiment with random data, with the exception that, in this special instance, PBDCA is the most promising solution method. BDCA is still ahead of DCBA which, in turn, beats DCPCA as well as DCA.

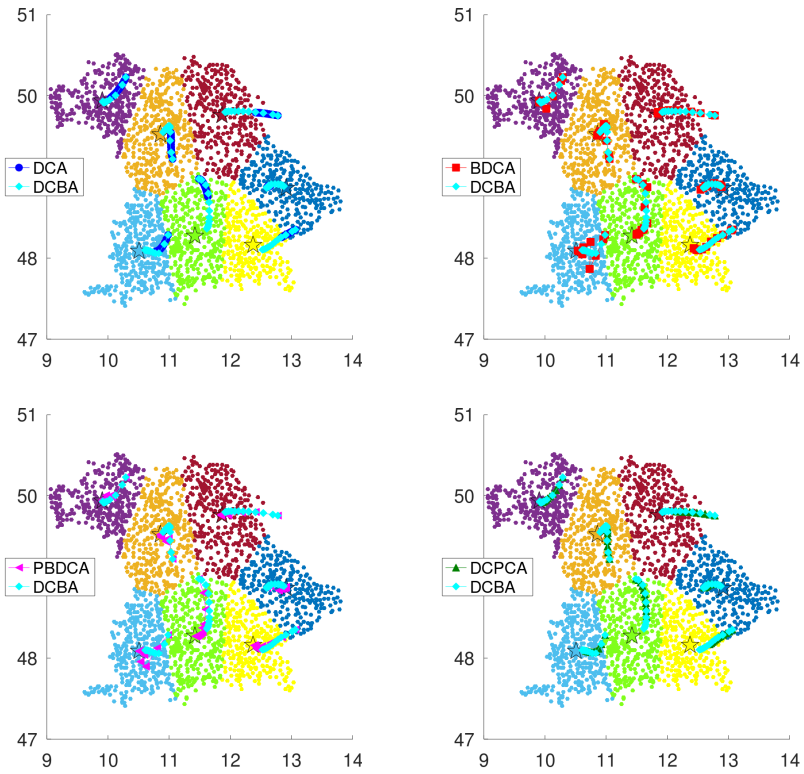


Figure 4: Division of Bavarian cities into administrative districts by means of Minimum Sum-of-Squares Clustering

4.4 The Multidimensional Scaling Problem

Similar to the organization of the previous section, we first give a brief introduction of the problem under consideration, see [4, 16], and then use two different test settings, one based

[1]source: http://www.fa-technik.adfc.de/code/opengeodb/?C=D;O=A, called on April 15, 2021; determination of membership towards Bavaria by means of postal code

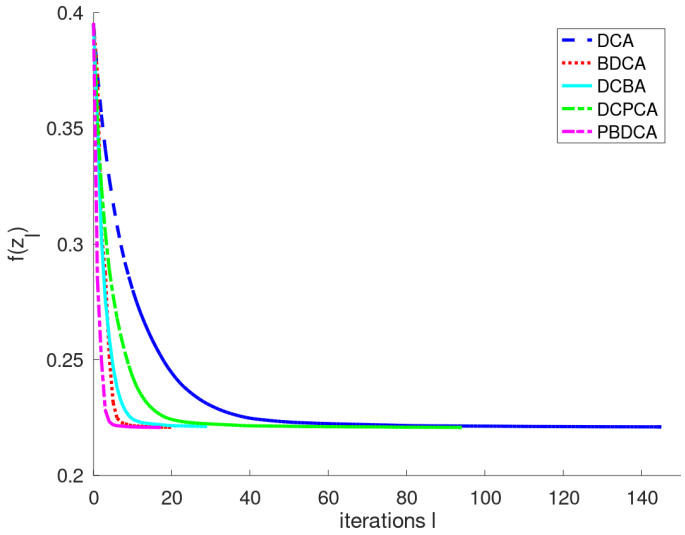


Figure 5: Evolution of the function value with proceeding iterations while applying Minimum Sum-of-Squares Clustering towards Bavarian cities

on random data and the other one with real data based on the geographic problem from the previous section.

Multidimensional Scaling is also a method from data mining. This time, the preprocessing of large data sets for further analysis happens by summarizing data through reduction. To be more precise, having a data set consisting of n points, each of dimension q, the aim is to replace them by the same number of points having now a dimension of $p \leq q$. Of course, one tries to keep the relevant information in the best possible way. To this end, the differences within the data set are considered by means of the dissimilarity matrix $\delta \in \mathbb{R}^{n \times n}$ with entries

$$\delta_{ij} := d_{ij}(\tilde{X}) := \left\| \tilde{x}^i - \tilde{x}^j \right\| \qquad i, j = 1, .., n,$$

where $\tilde{X} := (\tilde{x}^1, \tilde{x}^2, ..., \tilde{x}^n) \in \mathbb{R}^{q \times n}$ contains the data points to be analyzed. Now, the goal is to find a matrix in $\mathbb{R}^{p \times n}$ whose dissimilarity matrix approximates the one of the original data δ in an optimal way, and hence reflects the existing differences in the data. Therefore, the underlying optimization problem is

$$\min_{X \in \mathbb{R}^{p \times n}} \tilde{f}(X) := \sum_{i<j} \omega_{ij} \left(d_{ij}(X) - \delta_{ij} \right)^2,$$

where $\omega = (\omega_{ij})_{i,j=1,...,n}$ is a symmetric matrix consisting of non-negative weights with zeros on its diagonal. In the following experiments, it was taken as

$$\omega_{ij} = \begin{cases} 1 & \text{für } i \neq j, \\ 0 & \text{für } i = j. \end{cases}$$

Obviously, the case $p = q$ does not yield a reduction in the data set, but leads towards the question whether the original data set can be reproduced by its dissimilarity matrix.

Consequently, the optimal function value for this special instance is known to be zero.

Neglecting constant terms, the primary optimization problem can be rewritten as a DC problem with the adapted objective function $f : \mathbb{R}^{p\times n} \to \mathbb{R}$ given by its components $g,\ h : \mathbb{R}^{p\times n} \to \mathbb{R}$,

$$g(X) := \frac{1}{2}\sum_{i<j} \omega_{ij} d_{ij}^2(X) + \frac{\rho}{2}\|X\|^2, \quad h(X) := \sum_{i<j} \omega_{ij}\delta_{ij} d_{ij}(X) + \frac{\rho}{2}\|X\|^2,$$

where a modulus $\rho > 0$ ensures the uniform convexity of g and h and $\|X\|$ denotes the Frobenius norm of the matrix X. In the subsequent experiments, $\rho = \frac{1}{np}$ is chosen depending on the size of the data set and the dimension of the destination space.

The first numerical experiment uses quasi-randomly generated data $\tilde{X} \in \mathbb{R}^{q\times n}$ consisting of normally distributed entries, having a mean of zero and a standard deviation of ten. Reproducing the data as well as reducing each data point to half its size was tested. Thereby, the parameters were taken as $n \in \{25, 50, 75, 100, 125, 150\}$ and $p \in \{2, 3\}$ (and, consequently, $q \in \{p, 2p\}$). For each parameter combination, ten test runs with all algorithms from the beginning of this section were executed. To construct a suitable initial guess $X^0 \in \mathbb{R}^{p\times n}$, following the suggestion in [4], we first create a matrix $\tilde{X}^0$ of the same size with quasi-random entries drawn from the same normal distribution as the data set itself. Afterwards,

$$X^0 := \tilde{X^0}\Big(\mathbf{I}_{n\times n} - \frac{1}{n}\mathbf{1}_{n\times n}\Big) \tag{17}$$

was set, with $\mathbf{I}_{n\times n}$ denoting the identity matrix and $\mathbf{1}_{n\times n}$ the matrix consisting of ones only. This time, some parameters had been adopted, namely in case of BDCA $\alpha = 0.05,\ \beta = 0.1$ as well as the quantities related to the self-adaptive trial stepsize strategy, which also got used for DCBA, $\gamma = 10$ and $\bar{\lambda}_1 = 10$. For the latter algorithm, also $\beta = 0.1,\ m = 0.2$, and $\gamma = 0.05$ were changed. Comparably, $m = 0.2$ was also set for the bundle methods PBDCA and DCPCA, together with $\sigma = 0.1$ for the latter method. The subproblems arising in DCA and BDCA were solved by means of `fminunc`. During the overall process, in the case of $p = q$, a method gets rated successful whenever approaching the (a priori known) optimal function value, whereas in case of $p < q$ yielding the in comparison smallest function value gets decisive. Additionally, an algorithm got considered to be failed if the running time exceeded four hours or the maximum number of 10,000 iterations was reached. A failure of PBDCA was often traced back towards the last criterion. In particular, the number of inner iterations went beyond the critical bound. Furthermore, DCA failed in five instances, and DCPCA in six ones. Reporting for each test run the respective running time as well as the number of iterations led to the performance profiles shown in extracts in Figure 6. Within a factor of more than 60, DCPCA finally approaches the level of BDCA, so that, in the end, PBDCA is the only method with a significantly smaller final success rate. But the discrepancies of DCPCA in evolution, in relation to most of the other algorithms, diminish visibly when considering the performance profiles regarding running time. However, this time a factor of far more than 2,000 is necessary until hardly any changes in the overall profile occur. It is remarkable that purely taking the running time into account, DCBA is the method outperforming all the other algorithms. But, altogether, next to DCBA, especially the established BDCA seem to be the most promising method for this application.

The second example deals, once again, with the problem of geographic coordinates of Bavarian cities. Taking the identical set of data as in Section 4.3, the task was to reproduce the geographic coordinates on the basis of the dissimilarity matrix related to the raw data.

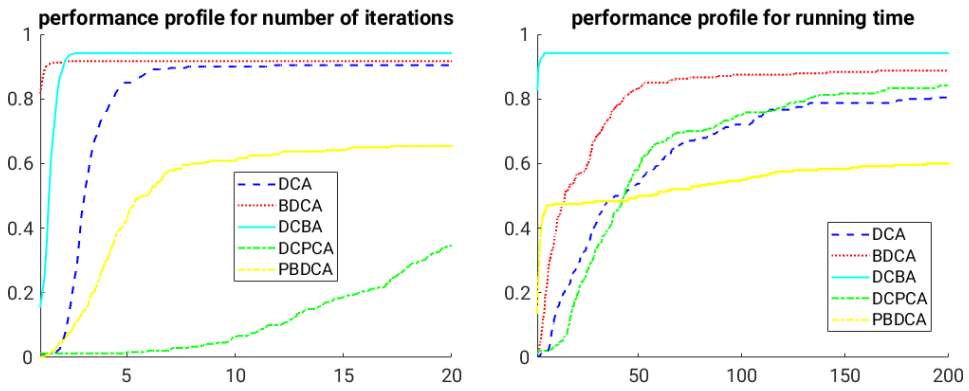


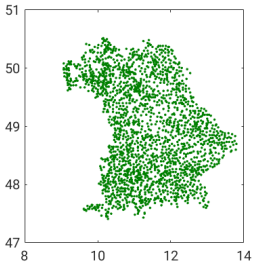
Figure 6: Performance profiles (extracts) of the Multidimensional Scaling problem with random data

Thereby, the initial guess was constructed similarly as in the previous example. For this purpose, the columns of the auxiliary matrix $\tilde{X}^0 \in \mathbb{R}^{2\times 2,073}$ took quasi-random values ranging in $[0, 4.74] \times [0, 3.12]$, motivated by the east-west as well as the north-south extension of Bavaria, determined on the basis of the underlying data set. To the resulting X^0 from (17), the geographic mean of Bavaria $(11.43, 48.93)^T$ (once again calculated with respect to the cities under consideration) was added. We use the standard parameter settings from the beginning of this section, with the only exception that $\varepsilon_1 = 10^{-2}$. Furthermore, the subproblems occurring in DCA and BDCA were solved by means of a Limited-Memory BFGS method, see [19, 20]. All algorithms managed to restore the map of Bavarian cities, but with differences in the speed of convergence. This distinction gets already foreshadowed in Figure 7, in which next to the map to be reproduced and the initial guess also, for each method, the current standing at iteration 25 as well as the final result is pictured (note that the slight rotation is due to the formulation of the problem, estimating only the distances between cities). Further plots reveal that the major progress happens during the first 60% of iterations, whereas the changes following afterwards can hardly be seen in corresponding images. All methods yield satisfying results although BDCA and PBDCA came out on top in terms of iterations.

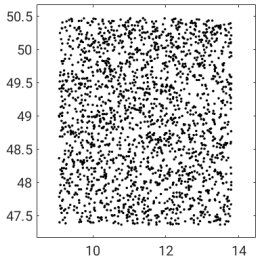
4.5 Edge Detection by Means of a DC Optimization Based Clustering Technique

Edge detection is a well known method in image segmentation for carving out certain objects in an image. It is based on the idea of determining contours of objects marked by discontinuities and erratic changes in the brightness values of the grey scale image. The technique presented in the following, using a DC optimization based clustering approach, was developed in [14]. To each pixel, a vector representing the differences in the grey scale values with respect to nearby pixels gets assigned. Subsequently the norms of these vectors are split into two groups yielding a differentiation into pixels belonging to an edge and the ones which does not.

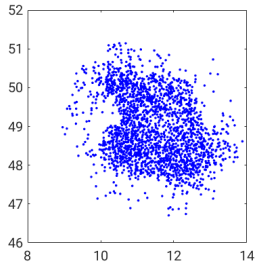
Having a grey scale image consisting of $(n+2) \times (m+2)$ pixels with coordinates $(x, y) \in [1, n+2] \times [1, m+2]$ to each pixel of the interior of the image, i.e., with coordinates $(x, y) \in [2, n+1] \times [2, m+1]$, a vector $v_i \in \mathbb{R}^4, i = 1, ..., N$ with $N = nm$, containing the differences in the brightness values of the pixel under consideration and the four vertical and horizontal



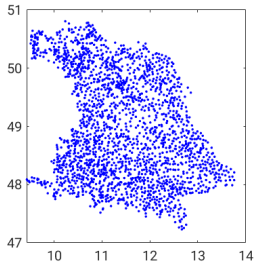
Map of Bavaria to be reproduced



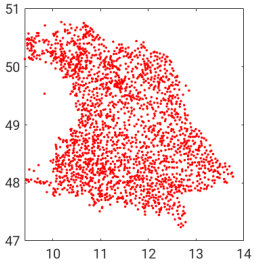
Initial guess



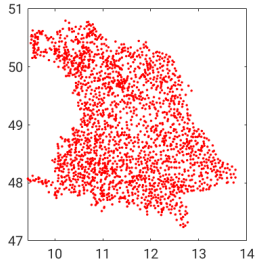
DCA: iteration 25



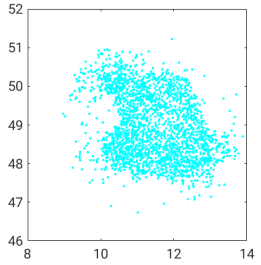
DCA: iteration 79



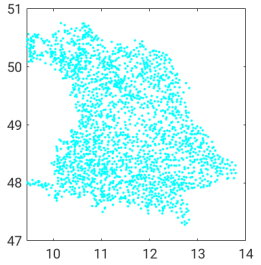
BDCA: iteration 25



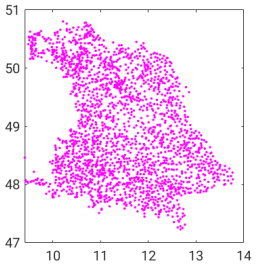
BDCA: iteration 39



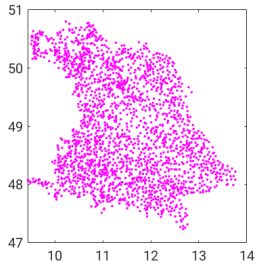
DCBA: iteration 25



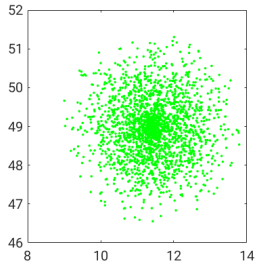
DCBA: iteration 65



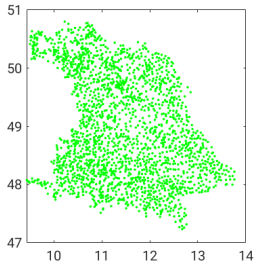
PBDCA: iteration 25



PBDCA: iteration 38



DCPCA: iteration 25



DCPCA: iteration 214

Figure 7: Reproducing the map of Bavaria by means of Multidimensional Scaling

immediately adjacent pixels gets assigned. For obvious reasons, marginal pixels are neglected and also will not get classified in the progress. Taking the norm $a_i := \|v_i\|, \ i = 1, ..., N$, of each such vector, one attains a measure of change in the grey scale values in relation to the neighborhood of the central pixel. A high value indicates an affiliation towards an edge, whereas a low value does not. This motivates to separate the set $\{a_i\}_{i=1,...,N}$ into two groups. To this end, the well known K-means clustering method gets applied with $K = 2$. Denoting with $z_1, z_2 \in \mathbb{R}$ the variables for determining the centroids, the resulting optimization problem is given by

$$\min_{z_1,z_2\in\mathbb{R}} f(z_1, z_2) := \sum_{i=1}^{N} \min\{|a_i - z_1|, |a_i - z_2|\}.$$

Similar to the clustering method introduced in Section 4.3, the objective function f can be written as a DC function with (uniformly) convex components $g, h : \mathbb{R}^2 \to \mathbb{R}$,

$$g(z_1, z_2) := \sum_{i=1}^{N} (|a_i - z_1| + |a_i - z_2|) + \frac{\rho}{2} \left\| (z_1, z_2)^T \right\|^2,$$

$$h(z_1, z_2) := \sum_{i=1}^{N} \max\{|a_i - z_1|, |a_i - z_2|\} + \frac{\rho}{2} \left\| (z_1, z_2)^T \right\|^2$$

with modulus $\rho \geq 0$. In the following, ρ was chosen to be 0.1. As initial value

$$\begin{pmatrix} z_1^0 \\ z_2^0 \end{pmatrix} := \kappa (a_{max} - a_{min}) \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

with $a_{max} = \max_{i=1,...,N} a_i$ and $a_{min} = \min_{i=1,...,N} a_i$ was selected on the basis of [14]. However, κ varies in our experiments with the image under consideration, as especially DCBA turned out to be pretty sensitive regarding the initial choice. Moreover, in contrast to many other applications, the clustering method has not to be carried out with a high precision in order to yield a satisfactory classification of edges. Instead, for our subsequent test runs, a maximum of five iterations proved to be sufficient. In addition, the termination tolerance ε_1 was reduced to 0.1 for the numerical experiments with some classical test images for edge detection. Besides, the parameter m for all three bundle methods was adapted to 0.2. Note that, this time, BDCA can not be applied for solving the optimization problem as the corresponding first DC component g is nonsmooth. For the solution of the subproblems arising in DCA, `fminsearch` was used. Three classical test images for edge detection were considered. For the cameraman, having a size of 256×256 pixels, $\kappa = \frac{1}{3}$ was set as parameter for the initial point, for the house, covering the same number of pixels, $\kappa = 0.3$, and for the moon, spanning 537×358 pixels, $\kappa = \frac{1}{2}$ was chosen. The results are shown in Figure 8, in which also the input images are displayed in the first column. PBDCA exceeded the maximum number of 10,000 inner iterations in case of the cameraman and the house, which is why no output has been produced. In the images, differences only get visible while considering the house. Here, DCBA recognizes the top end of the chimney a bit better than the two remaining algorithms, and also stronger indicates some less pronounced edges by dotted lines. Let us note that for this test image the sensitivity of DCBA regarding the initial value got particularly clear, especially when adapting the parameter m at the same time. Only slight modifications led from recognizing hardly any edges over a result comparable to the ones from the other methods to detection of every brick

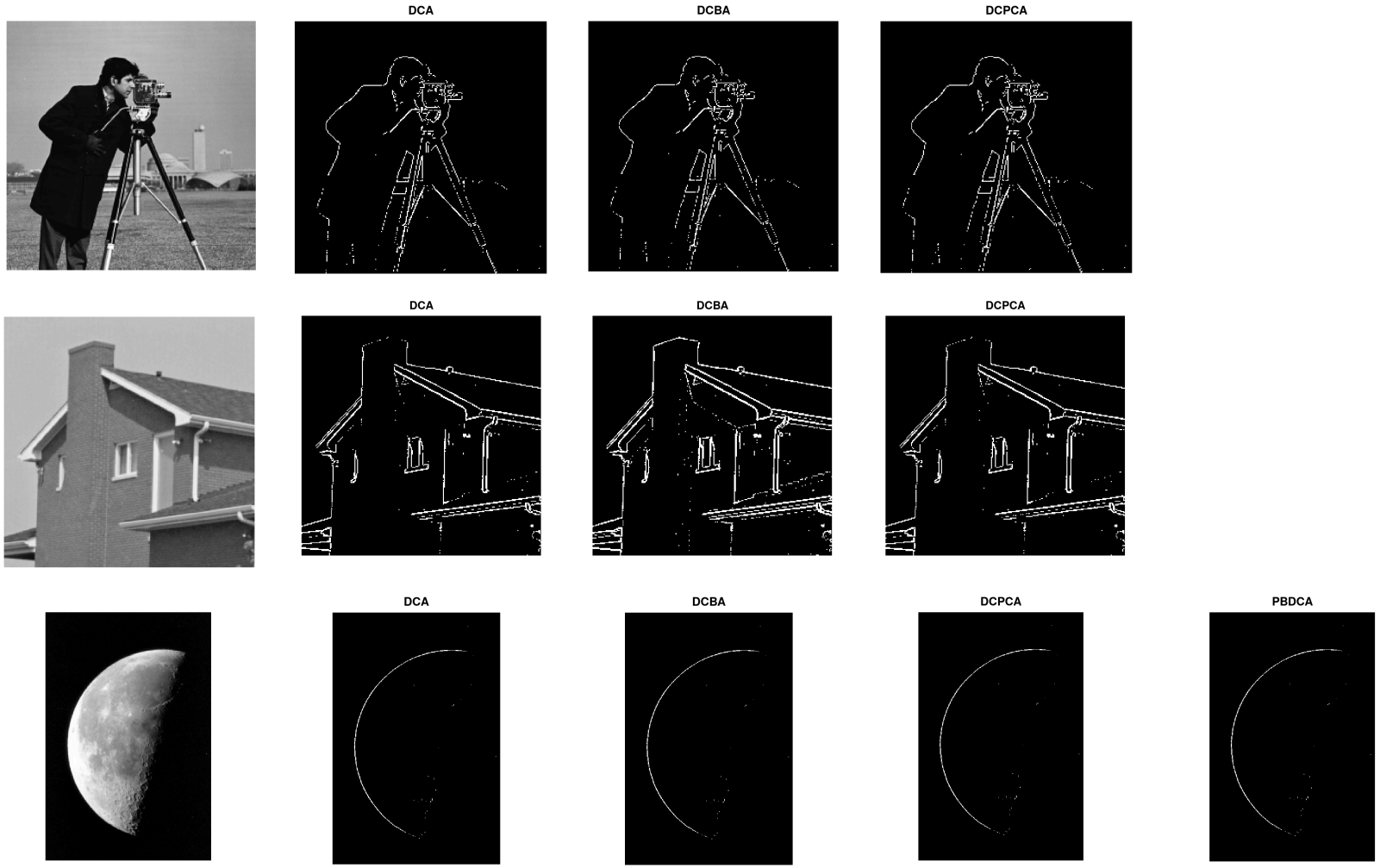


Figure 8: Edge detection by means of a DC optimization based clustering technique

stone (see Figure 9 for which $m = 0.5$ and $\kappa = \frac{1}{4}$ was chosen), whereas the output of the remaining algorithms kept pretty stable. Although the number of iterations is not reported here in detail, it is worth mentioning that DCA reaches the desired accuracy in all three cases within only two iterations. Altogether, with the exception of PBDCA, all remaining methods yield pretty similar results for this application, detecting sharp edges quite reliably.

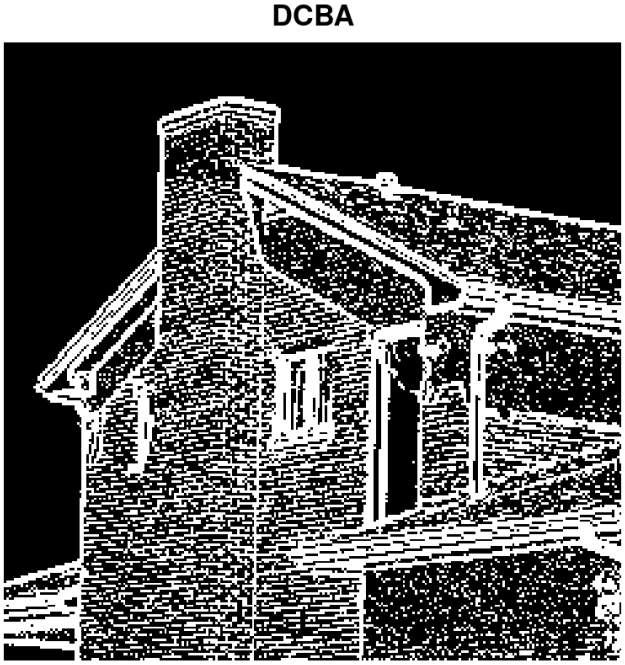


Figure 9: Detecting brick stones with DCBA

5 Concluding Remarks

In this article, a bundle method for the solution of unconstrained DC optimization problems (DCBA) was introduced. In contrast to various existing bundle methods designed for this topic, the bundles are not directly constructed with respect to the DC components of the objective function, instead a convex approximation of the function to be minimized, which is already known from the classical DC Algorithm, gets constructed first. Applying the bundle method towards the model function yields a descent direction for the objective function which allows to add a line search afterwards. Contrary to BDCA, the bundle method is not carried out until a minimizer of the approximation is found, but only until a serious step is executed. An advantage of DCBA against BDCA is that the new method can be applied to DC problems with both components being nonsmooth.

The algorithm was shown to be well-defined for functions being bounded from below, and being globally convergent in the sense that every accumulation point is a critical point of the objective function. Moreover, termination of the algorithm (with numerically implementable termination criterion) occurs within a finite number of iterations in a point satisfying an approximate criticality measure.

So far, this method is applicable to unconstrained DC programs only. Our future work will concentrate on suitable extensions to constrained problems, first to DC programs with convex constraints, and then also to DC programs with general DC-type constraints.

References

[1] Ackooij, W. van, Demassey, S., Javal, P., Morais, H., Oliveira, W. de, et al. "A bundle method for nonsmooth DC programming with application to chance-constrained problems". In: *Comput. Optim. Appl.* 78.2 (2021), pages 451–490. URL: https://doi.org/10.1007/s10589-020-00241-8.

[2] An, L. T. H., Tao, P. D., and Muu, L. D. "Numerical solution for optimization over the efficient set by d.c. optimization algorithms". In: *Oper. Res. Lett.* 19.3 (1996), pages 117–128. URL: https://doi.org/10.1016/0167-6377(96)00022-3.

[3] Aragón Artacho, F. J., Fleming, R. M. T., and Vuong, P. T. "Accelerating the DC algorithm for smooth functions". In: *Math. Program.* 169.1 (2018), pages 95–118. URL: https://doi.org/10.1007/s10107-017-1180-1.

[4] Aragón Artacho, F. J. and Vuong, P. T. "The boosted difference of convex functions algorithm for nonsmooth functions". In: *SIAM J. Optim.* 30.1 (2020), pages 980–1006. URL: https://doi.org/10.1137/18M123339X.

[5] Birgin, E., Martínez, J. M., and Raydan, M. "Nonmonotone spectral projected gradient methods on convex sets". In: *SIAM J. Optim.* 10 (2000), pages 1196–1211. URL: https://doi.org/10.1137/S1052623497330963.

[6] Clarke, F. H. *Optimization and Nonsmooth Analysis*. Canadian Mathematical Society series of monographs and advanced texts. Wiley New York, 1983.

[7] Dolan, E. D. and Moré, J. J. "Benchmarking optimization software with performance profiles". In: *Math. Program.* 91.2 (2002), pages 201–213. URL: https://doi.org/10.1007/s101070100263.

[8] Gaudioso, M., Giallombardo, G., Miglionico, G., and Bagirov, A. M. "Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations". In: *J. Glob. Optim.* 71.1 (2018), pages 37–55. URL: https://doi.org/10.1007/s10898-017-0568-z.

[9] Geiger, C. and Kanzow, C. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer-Lehrbuch Masterclass. Springer Berlin Heidelberg, 2002.

[10] Hiriart-Urruty, J.-B. "Generalized differentiability / duality and optimization for problems dealing with differences of convex functions". In: *Convexity and Duality in Optimization*. Edited by J. Ponstein. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pages 37–70.

[11] Hiriart-Urruty, J.-B. and Lemaréchal, C. *Fundamentals of Convex Analysis*. Grundlehren Text Editions. Springer Berlin, 2001.

[12] Huber, P. J. "Robust estimation of a location parameter". In: *Ann. Math. Stat.* 35.1 (1964), pages 73 –101. URL: https://doi.org/10.1214/aoms/1177703732.

[13] Joki, K., Bagirov, A. M., Karmitsa, N., and Mäkelä, M. M. "A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes". In: *J. Glob. Optim.* 68 (2017), pages 501–535. URL: https://doi.org/10.1007/s10898-016-0488-3.

[14] Khalaf, W., Astorino, A., D'Alessandro, P., and Gaudioso, M. "A DC optimization-based clustering technique for edge detection". In: *Optim. Lett.* 11.3 (2017), pages 627–640. URL: https://doi.org/10.1007/s11590-016-1031-7.

[15] Kiwiel, K. C. "An aggregate subgradient method for nonsmooth convex minimization". In: *Math.Program.* 27 (1983), pages 320 –341. URL: https://doi.org/10.1007/BF02591907.

[16] Le Thi, H. A. and Pham Dinh, T. "D.C. programming approach to the multidimensional scaling problem". In: *From Local to Global Optimization*. Edited by A. Migdalas, P. M. Pardalos, and P. Värbrand. Boston, MA: Springer US, 2001, pages 231–276. URL: https://doi.org/10.1007/978-1-4757-5284-7_11.

[17] Le Thi, H. A. and Pham Dinh, T. "DC programming and DCA: thirty years of developments". In: *Math. Program.* 169.1 (2018), pages 5–68. URL: https://doi.org/10.1007/s10107-018-1235-y.

[18] Lemaréchal, C. and Mifflin, R. *Nonsmooth Optimization: Proceedings of a IIASA Workshop, March 28 - April 8, 1977*. Elsevier Science, 2014.

[19] Liu, D. C. and Nocedal, J. "On the limited memory BFGS method for large scale optimization". In: *Math. Program.* 45 (1989), pages 503 –528. URL: https://doi.org/10.1007/BF01589116.

[20] Nocedal, J. and Wright, S. J. *Numerical Optimization*. 2nd edition. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[21] Oliveira, W. de. "Proximal bundle methods for nonsmooth DC programming". In: *J. Glob. Optim.* 75.2 (2019), pages 523–563. URL: https://doi.org/10.1007/s10898-019-00755-4.

[22] Ordin, B. and Bagirov, A. M. "A heuristic algorithm for solving the minimum sum-of-squares clustering problems". In: *J. Glob. Optim.* 61 (2015), pages 341–361. URL: https://doi.org/10.1007/s10898-014-0171-5.

[23] Rockafellar, R. T. *Convex Analysis*. Volume 28. Princeton Mathematical Series. Princeston University Press, 1970.