Set-Valued and Variational Analysis (2019) 27:895–919
https://doi.org/10.1007/s11228-018-0497-0

An Inertial Algorithm for DC Programming

Welington de Oliveira[1] · **Michel P. Tcheou**[2]

Received: 12 January 2018 / Accepted: 1 October 2018 / Published online: 25 October 2018

Abstract
We consider nonsmooth optimization problems whose objective function is defined by the *Difference of Convex* (DC) functions. With the aim of computing critical points that are also d(irectional)-stationary for such a class of nonconvex programs we propose an algorithmic scheme equipped with an inertial-force procedure. In contrast to the classical DC algorithm of P. D. Tao and L. T. H. An, the proposed inertial DC algorithm defines trial points whose sequence of functional values is not necessary monotonically decreasing, a property that proves useful to prevent the algorithm from converging to a critical point that is not d-stationary. Moreover, our method can handle inexactness in the solution of convex subproblems yielding trial points. This is another property of practical interest that substantially reduces the computational burden to compute d-stationary/critical points of DC programs. Convergence analysis of the proposed algorithm yields global convergence to critical points, and convergence rate is established for the considered class of problems. Numerical experiments on large-scale (nonconvex and nonsmooth) image denoising models show that the proposed algorithm outperforms the classic one in this particular application, specifically in the case of piecewise constant images with neat edges such as QR codes.

Keywords DC programming · Nonsmooth optimization · Variational analysis

Mathematics Subject Classification (2010) 49J52 · 49J53 · 49K99 · 90C26

1 Introduction

In this work we consider nonconvex nonsmooth optimization problems of the form

$$\min_{x\in\mathbb{R}^n} f(x), \qquad \text{with} \quad f(x) := f_1(x) - f_2(x)\,, \tag{1}$$

✉ Welington de Oliveira
welington.oliveira@mines-paristech.fr

Michel P. Tcheou
mtcheou@uerj.br

[1] MINES ParisTech, PSL – Research University, CMA – Centre de Mathématiques Appliquées, Sophia Antipolis, France

[2] Rio de Janeiro State University (UERJ), Rio de Janeiro, Brazil

where $f_1,\ f_2 : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are convex and possibly nonsmooth functions. Problems of this type are known in the literature as DC programs, with "DC" standing for *difference of convex* functions, [19]. We assume throughout this manuscript that f_1 is a closed function and that $\mathcal{D}\text{om}(f_1) \subset \Omega \subset \mathcal{D}\text{om}(f_2)$, where Ω is an open and convex set in $\mathbb{R}^n$. This assumption allows us to encompass convex constrained DC programs in formulation (1). Indeed, notice that the first component function f_1 can be, for example, the sum of a convex function $\varphi : \mathbb{R}^n \to \mathbb{R}$ with the indicator function i_X of a convex set $X \subset \Omega$, i.e., $f_1(x) = \varphi(x) + i_X(x)$ with $i_X(x) = 0$ if $x \in X$ and $i_X(x) = +\infty$ otherwise.

DC programming forms an important sub-field of nonconvex programming and has been receiving much attention from the mathematical programming community, [16, 17, 19, 23, 26, 35, 39, 41].

More general DC programs with DC constraints are investigated in [27, 32, 36–38, 42]. Some applications include production-transportation planning problems [21], location planning problems [41, Chapter 5], physical layer based security in a digital communication systems [32], chance-constrained problems [11], cluster analysis [5, 24], engineering design problems [13, 41], energy management problems [42] and others. We refer to [41, Part II] for a comprehensive presentation of several algorithms designed for DC optimization problems. Solving globally nonsmooth programs as (1) is a challenging task, especially in the large-scale setting. We will, therefore, deal with this class of problems by employing local-solution approaches.

A well-known method for dealing with the optimization problem (1) is the *DC Algorithm – DCA –* of [39] (see also [4, 26, 40]). The classical DCA handles problem (1) by defining a sequence of trial points according to the following rule, for a given starting point $x^0 \in \mathcal{D}\text{om}(f_1)$:

$$\text{for all } k = 0, 1, 2, \ldots, \text{ compute } g_2^k \in \partial f_2(x^k) \text{ and } x^{k+1} \in \arg\min_{x \in \mathbb{R}^n} f_1(x) - \langle g_2^k, x\rangle, \quad (2)$$

where $\partial f_2(x^k)$ is the subdifferential of the convex function f_2 at point x^k (see definition in Section 2 below). It can be shown [39, Theorem 3] that every cluster point $\bar{x}$ (if any) of the sequence $\{x^k\}$ generated by rule (2) is a critical point of problem (1), i.e., $\bar{x}$ satisfies

$$\partial f_1(\bar{x}) \cap \partial f_2(\bar{x}) \neq \emptyset\,. \quad (3)$$

It follows from convexity of the component functions f_1 and f_2 that rule (2) yields a monotone sequence of function values, i.e., $f(x^{k+1}) \leq f(x^k)$ for all $k = 0, 1, \ldots$ (see [39, Theorem 3(i)] for more details). While monotonicity might be seen as a quality of the method, demanding monotonically decreasing function values might not be an ideal scheme in nonconvex optimization: depending on starting points, iterates are attracted by poor-quality critical points that prevent the (monotone) algorithm from computing a critical point of better quality. In the context of DC programming, we mean by a "critical point of better quality" a critical point $\bar{x}$ that is d(irectional)-stationary, i.e., $\bar{x}$ satisfies

$$\partial f_2(\bar{x}) \subset \partial f_1(\bar{x})\,. \quad (4)$$

As shown in [19, 32], d-stationarity is the sharpest stationary definition for nonconvex problems of type (1); see also additional comments in Section 2. It is clear from its definition that computing d-stationary points for (1) is not a trivial task in general. Few exceptions are the situations in which either f_1 or f_2 are differentiable, [14, Section 2], and the case when f_2 is the pointwise maximum of finitely many convex and differentiable functions. The latter case is investigated in [32], where the authors propose a proximal linearized method that solves several convex programs per iteration, and thus has a high computational burden. A less computational demanding method is a variant of the proximal bundle algorithm proposed in [11], but may require solving many quadratic programs per iteration; see [11, Algorithm 2].

In this work, we are concerned with algorithms of low computational costs to compute critical points. Moreover, we do not assume that f_2 is the pointwise maximum of finitely many convex and differentiable functions. In order to try to transpose critical points that are not d-stationary, we furnish the DCA algorithm represented by rule (2) with an inertial scheme that can be seen as a version of the heavy-ball method of Polyak [33].

In a differentiable and convex framework, the heavy-ball method is a two-step gradient algorithm that can be interpreted as an explicit finite differences discretization of the so-called *heavy-ball with friction dynamical system* [31]. In summary, the algorithm incorporates an inertial force in the iterative process of gradient methods by appending to the negative-gradient direction the inertial term $\gamma(x^k - x^{k-1})$, where $\gamma \geq 0$ is a given parameter. The heavy-ball method has been generalized in several manners: the authors of [44] consider differentiable but nonconvex optimization problems, and in [2, 3, 29] the heavy-ball method was extended to handle maximal monotone operators. The paper [31] deals with nonsmooth nonconvex optimization problems of the form $\min_{x\in\mathbb{R}^n} f_1(x) + \xi(x)$, where f_1 is as above and $\xi : \mathbb{R}^n \to \mathbb{R}$ is a smooth nonconvex function with Lipschitz continuous gradient. The authors of [31] assume $f_1 + \xi$ to be coercive and propose a linearized proximal method with inertial force to compute stationary points.

In the DC context, $\xi = -f_2$ is a concave function. However, in this work we do not assume that either f_2 is smooth nor $f = f_1 - f_2$ is coercive. Our proposal follows the general lines of the DCA and replaces the subgradient g_2^k of the second component function f_2 in rule (2) by $g_2^k + \gamma(x^k - x^{k-1})$, with $\gamma \geq 0$. This leads to the following iterative scheme, for a given point $x^0 \in \mathcal{D}\mathrm{om}(f_1)$:

$$\text{for all } k=0,1,\ldots,\text{compute } g_2^k \in \partial f_2(x^k) \text{ and } x^{k+1} \in \arg\min_{x\in\mathbb{R}^n} f_1(x) - \langle g_2^k + \gamma(x^k - x^{k-1}), x\rangle. \tag{5}$$

As we will see in Section 4, the sequence of function values issued by the above scheme is not necessarily monotone due to the inertia imposed by the term $\gamma(x^k - x^{k-1})$. As already said, this property can be beneficial for the quest of computing critical points that are also d-stationary for (1). Nevertheless, it is not assured that our *Inertial DC Algorithm – InDCA* – illustrated by rule (5) will always compute d-stationary points, but critical ones. Numerical experiments reported in Section 6 below show that InDCA is more robust than DCA, meaning that for the same starting points InDCA computes very often better critical points than DCA does.

As for (2), rule (5) is not practical when solving the resulting convex subproblem

$$\min_{x\in\mathbb{R}^n} f_1(x) - \langle g_2^k + \gamma(x^k - x^{k-1}), x\rangle \tag{6}$$

is too time consuming. For this reason, the given algorithm only demands x^{k+1} to be an inexact solution with a vanishing approximation error, an idea already explored in [35, 38, 42] by employing different techniques. This is a second property of practical interest of our algorithm as it substantially reduces the computational burden to compute a d-stationary/critical point of DC programs.

The remainder of this work is organized as follows: Section 2 provides some notation and preliminary results that will be employed throughout the text. In addition, the section contains two examples illustrating the benefits of incorporating an inertial force to the DC algorithm. Section 3 presents our inertial DC algorithmic pattern as well as some practical issues concerning the implementation of some of its variants. Convergence analysis and convergence rate are considered in Section 4, and the application of interest is discussed in

Section 5: nonconvex image denoising models. Section 6 reports some preliminary numerical results comparing InDCA against DCA and the nonconvex algorithm iPiano of [31]. Finally, Section 7 closes the paper with some concluding remarks.

2 Notation, Main Definitions and Illustrative Examples

For any points $x, y \in \mathbb{R}^n$, $\langle x, y\rangle$ stands for the Euclidean inner product, and $\|\cdot\|$ for the associated norm, i.e., $\|x\| = \sqrt{\langle x, x\rangle}$. For a set $X \subset \mathbb{R}^n$, we denote by i_X its indicator function, i.e., $i_X(x) = 0$ if $x \in X$ and $i_X(x) = +\infty$ otherwise. For a convex set X its normal cone at the point x is denoted by $N_X(x)$, which is the set $\{y : \langle y, z - x\rangle \leqslant 0 \ \forall z \in X\}$ if $x \in X$ and the empty set otherwise.

As convex functions are directionally differentiable in the interior of their domains [34, Theorems 23.1 and 23.4], the limit

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

for f_i, $i = 1, 2$, is well defined for all x in the interior of $\mathcal{D}\text{om}(f_i)$ and all $d \in \mathbb{R}^n$. It is well known that $f_i'(x; d) = \max_{g \in \partial f_i(x)} \langle g, d\rangle$, where

$$\partial f_i(x) := \{g \in \mathbb{R}^n : \ f_i(y) \geq f_i(x) + \langle g, y - x\rangle \quad \forall\, y \in \mathbb{R}^n\}$$

is the subdiferential of f_i at point x. For $\epsilon \geq 0$ the ϵ-subdiferential is denoted by

$$\partial_\epsilon f_i(x) := \{g \in \mathbb{R}^n : \ f_i(y) \geq f_i(x) + \langle g, y - x\rangle - \epsilon \quad \forall\, y \in \mathbb{R}^n\}.$$

Since DC functions are also locally Lipschitz continuous (because their components f_i are so), their directional derivatives are well defined for all x in the interior of $\mathcal{D}\text{om}(f_i)$, $i = 1, 2$:

$$f'(x; d) = f_1'(x; d) - f_2'(x; d)\,.$$

A point $\bar{x} \in \mathbb{R}^n$ is a d(irectional)-stationary point of problem (1) if $f'(\bar{x}; (x - \bar{x})) \geq 0$ for all $x \in \mathbb{R}^n$, which can be shown to be equivalent to the inclusion (4), [19]. Notice that verifying (4) computationally is impractical in many cases of interest. Hence, one generally employs a weaker notion of stationarity: a point $\bar{x} \in \mathbb{R}^n$ is called a *critical point* of problem (1) if $\bar{x}$ satisfies (3). In summary, all local minimizers of problem (1) are d-stationary points, which in turn are critical points of (1). The reverse implications are, in general, not true as illustrated in [32, Example 2] (see also Example 1 below).

Throughout this work we assume the following condition, which is a mild hypothesis in the DC setting:

Assumption A1. *Function f_2 is strongly convex on Ω with a known parameter $\rho > 0$, that is, for every $g_2 \in \partial f_2(x)$ one has*

$$f_2(y) \geq f_2(x) + \langle g_2, y - x\rangle + \frac{\rho}{2}\|y - x\|^2, \quad \forall\, x, y \in \Omega. \tag{7}$$

We care to mention that A1, also present in [42], is not a restrictive assumption at all. In fact, if A1 does not hold for a certain DC function $f = \varphi - \psi$ we can obtain another DC decomposition of f satisfying A1 by adding an arbitrary strongly convex function $\omega : \Omega \to \mathbb{R}$ to the component functions: note that $f = f_1 - f_2$ with $f_1 = \varphi + \omega$ and $f_2 = \psi + \omega$. Since one can always take $\omega(\cdot) = \|\cdot\|^2$, then ρ can be assumed known in A1 without loss of generality (just take $\rho = 2$ in this case).

Under A1 and the hypothesis that the inertial parameter γ in (5) satisfies $0 \leq \gamma < \rho/2$ we illustrate the behavior of the sequences generated by rules (2) and (5) in the following two examples.

Example 1 (Transposing critical points that are not d-stationary.) Consider the bi-dimensional DC function $f(x) = f_1(x) - f_2(x)$, with $f_1(x) = \|x\|^2$ and $f_2(x) = \max(-x_1, 0) + \max(-x_2, 0) + 0.5\|x\|^2$. Its curve is plotted in Fig. 1, as well as the behavior of some sequences of points generated by defining the next iterate x^{k+1} as in (2) (DCA, Fig. 1b) and sequences generated by the new rule (5) (InDCA, Fig. 1c) with inertial factor $\gamma = 0.49$, which was chosen to be less than $\rho/2 = 0.5$.

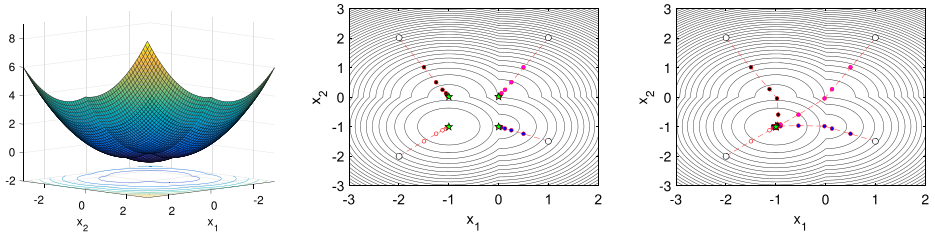
For the four different starting points, DCA determined four different critical points, all presented in Table 1. However, the global solution $\bar{x} = (-1, -1)^\top$ is the only d-stationary point of the problem of minimizing f over $\mathbb{R}^2$.

While the critical points computed by the classical DC algorithm depend strongly on the starting points, the inertial DC algorithm is able (in this example) to compute the d-stationary point (in this case a global solution) regardless the initial point. This is thanks to the inertial factor $\gamma(x^k - x^{k-1})$ that prevents the iterative process from stopping at critical points that are not d-stationary. In some situations it is also possible to overcome local solutions, as illustrated by the following example.

Example 2 (Trasposing local minimizers: a two-variable nonconvex 1D denoising model.) We consider the following nonconvex denoising optimization problem for 1D signals:

$$\min_{x \in \mathbb{R}^n} \frac{\mu}{2}\|x - b\|^2 + \sum_{i=1}^{n-1} \phi(|x_{i+1} - x_i|)\,.$$

The concave function $\phi(r) := \log(1 + 2r)/2$ is employed to induce sparsity of the one-lag-difference of the reconstructed signal $\bar{x}$: one wishes to reconstruct piecewise constant signals. As it will be shown later (see Proposition 2) the above nonconvex objective is indeed a DC function $f = f_1 - f_2$, with possible DC components given by $f_1(x) := \frac{\mu}{2}\|x - b\|^2 + \sum_{i=1}^{n-1} |x_{i+1} - x_i| + \|x\|^2$ and $f_2(x) := \sum_{i=1}^{n-1} |x_{i+1} - x_i| - \sum_{i=1}^{n-1} \phi(|x_{i+1} - x_i|) + \|x\|^2$ (hence the parameter of strongly convexity of f_2 is $\rho \geq 2$). In order to analyze the iterative process yielded by rules (2) and (5) applied to this problem, we consider dimension $n = 2$ and parameters $\mu = 0.6$, $b = (0.1, 3)^\top$. Notice that differently from Example 1, subproblems (2) and (5) do not have explicit solutions. We therefore compute iterates by



(a) DC function (b) DCA (c) InDCA with $\gamma = 0.49$

Fig. 1 Iterative process and level curves of function $f(x) = f_1(x) - f_2(x)$, with $f_1(x) = \|x\|^2$ and $f_2(x) = \max(-x_1, 0) + \max(-x_2, 0) + 0.5\|x\|^2$. Comparison between the classical DCA and the proposed inertial DC algorithm

Table 1 Critical points of function $f(x) = \|x\|^2 - [\max(-x_1, 0) + \max(-x_2, 0) + 0.5\|x\|^2]$ determined by rule (2)

$\bar{x}$	$f(\bar{x})$	$\partial f_1(\bar{x})$	$\partial f_2(\bar{x})$
$(-1, -1)^\top$	-1	$(-2, -2)^\top$	$(-2, -2)^\top$
$(0, -1)^\top$	-0.5	$(0, -2)^\top$	$(s, -2)^\top$ with $s \in [-1, 0]$
$(-1, 0)^\top$	-0.5	$(-2, 0)^\top$	$(-2, s)^\top$ with $s \in [-1, 0]$
$(0, 0)^\top$	0	$(0, 0)^\top$	$(s_1, s_2)^\top$ with $s_1, s_2 \in [-1, 0]$

solving these subproblems numerically up to a given tolerance. The iterative processes with five different initial points are presented in Fig. 2, with $0 \leq \gamma < \rho/2$. In three of the five initial points, sequences generated by DCA could not converge to the global minimum $\bar{x} = (0.3970,\ 2.7030)^\top$, but to the critical point $(31/20,\ 31/20)^\top$ that is a local solution (thus a d-stationary point) of the problem. Instead, rule (5) was more successful due to inertial factor γ: for $\gamma = 0.7$, only one sequence converged to the critical point that is not the global minimum. Yet, for $\gamma = 0.9$ all five sequences converged to the global solution.

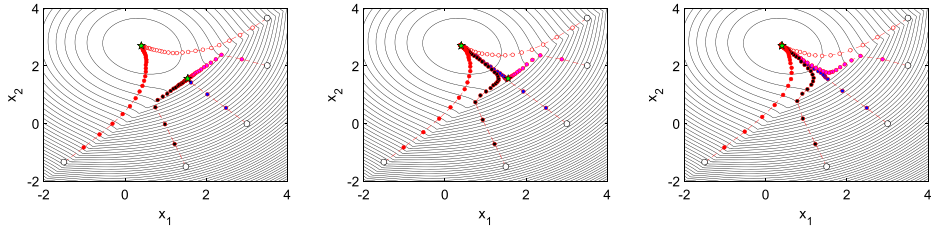
Example 2 suggests to consider a larger factor of inertia $\gamma \geq 0$. However, our analysis given in Section 3 shows that γ cannot be arbitrarily large: the inertial parameter can vary along the interval $[0,\ \rho/2)$, where $\rho \geq 0$ is the constant of strongly convexity of the second component function f_2.

3 An Inertial DC Algorithmic Pattern

In this section, we formalize our inertial DC algorithm (InDCA) represented by rule (5). We care to mention that if subproblem (6) is difficult to solve (e.g. when f_1 is assessed via simulation, optimization, numerical multidimensional integration etc.) then defining trial points by rule (5) (as well as (2)) can be too time consuming. To overcome this difficulty we follow the lead of [35, 38] and allow trial points to be inexact solutions of subproblem (6). In [38] trial points are defined as ϵ^{k+1}-solutions of the convex subproblems, where $\epsilon^{k+1} \to 0$. This idea is also explored in the context of linearized proximal methods in [35].

Differently from [35, 38] we define the trial point x^{k+1} in such a manner that the ϵ^{k+1}-subdifferential of f_1 at x^{k+1} intersects the set $\partial f_2(x^k) + \gamma(x^k - x^{k-1})$:

$$\partial_{\epsilon^{k+1}} f_1(x^{k+1}) \cap \partial f_2(x^k) + \gamma(x^k - x^{k-1}) \neq \emptyset\,.$$



(a) DCA (b) InDCA $\gamma = 0.7$ (c) InDCA $\gamma = 0.9$

Fig. 2 Iterative process and level curves of function $f(x) = \frac{\mu}{2}\|x - b\|^2 + \sum_{i=1}^{n-1} \phi(|x_{i+1} - x_i|)$, with $\mu = 0.6$, $b = (0.1,\ 3)^\top$ and $\phi(r) = \log(1 + 2r)/2$. Global solution is $\bar{x} = (0.3970,\ 2.7030)^\top$ and the optimal value is $f(\bar{x}) \approx 0.91538$. The set of critical points of f that does not contain the global solution is $C = \{(r, r) : \max\{b_1, b_2\} - 1/\mu \leq r \leq \min\{b_1, b_2\} + 1/\mu\}$, i.e, $C \approx \{(r, r) :\ 1.333 \leq r \leq 1.7667\}$

The motivation for such a strategy lies in the fact that if the sequence $\{x^k\}$ converges to a point $\bar{x}$ and $\{\epsilon^k\}$ vanishes, then the above condition eventually implies criticality (3) of $\bar{x}$. In fact, as it will be shown in Section 4, convergence of the whole sequence $\{x^k\}$ is not required: any cluster point of $\{x^k\}$ can be shown to be a critical point of (1). Furthermore, the inexactness ϵ^{k+1} involved in the iterative process is automatically controlled by our algorithm in such a manner that errors vanish as the iterative process progresses. The InDCA is presented in the following algorithmic pattern.

Algorithm 1 Inertial DC algorithmic pattern

1: Let $x^0 \in \mathcal{D}\text{om}(f_1)$, $\texttt{Tol} \geq 0$, $\lambda \in [0, 1)$ and $\gamma \in [0, (1-\lambda)\rho/2)$ be given. Set $x^{-1} = x^0$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Set $d^k = \gamma(x^k - x^{k-1})$ and find $x^{k+1} \in \mathbb{R}^n$ such that

$$\partial_{\epsilon^{k+1}} f_1(x^{k+1}) \cap \partial f_2(x^k) + d^k \neq \emptyset \quad \text{with} \quad 0 \leq \epsilon^{k+1} \leq \lambda \frac{\rho}{2} \|x^{k+1} - x^k\|^2 \tag{8}$$

4: **if** $\|x^{k+1} - x^k\| \leq \texttt{Tol}$ and $\|d^k\| \leq \texttt{Tol}$ **then**
5: Stop and return $(x^k, f(x^k))$
6: **end if**
7: **end for**

The following is an alternative to condition (8) that is suitable when the first component function f_1 is smooth: compute $x^{k+1} \in \mathbb{R}^n$ such that

$$\|g_1^{k+1} - (g_2^k + d^k)\| \leq \lambda \frac{\rho}{2} \|x^{k+1} - x^k\| \text{ for some } g_1^{k+1} \in \partial f_1(x^{k+1}) \text{ and } g_2^k \in \partial f_2(x^k)\,. \tag{9}$$

The Algorithmic pattern 1 boils down to specific optimization algorithms upon the choice of its parameters. We start by addressing some particular cases issued by the choice $\lambda = 0$. The choice $\lambda > 0$ means that subproblem (5) can be inexactly solved. Practical details on how to implement (8) and (9) for $\lambda > 0$ are given in Section 3.2.

3.1 Some Specific Settings for the Algorithmic Pattern with $\lambda = 0$

3.1.1 The DC Algorithm with/without Inertial Force

Consider the Algorithmic pattern 1 with $\lambda = \gamma = 0$. With this choice of parameters condition (8) is equivalent to (9), which reads as

$$\partial f_1(x^{k+1}) \cap \partial f_2(x^k) \neq \emptyset\,.$$

Such condition is ensured, for instance, if $g_2^k \in \partial f_2(x^k)$ and x^{k+1} solves $\min_{x \in \mathbb{R}^n} f_1(x) - \langle g_2^k, x\rangle$, the subproblem of the classic DC algorithm of [39]: optimality of x^{k+1} implies $g_2^k \in \partial f_1(x^{k+1})$, and therefore $g_2^k \in \partial f_1(x^{k+1}) \cap \partial f_2(x^k)$.

If $\lambda = 0$ but $\gamma > 0$ then either (8) or (9) is equivalent to define x^{k+1} by solving (6): its optimality condition is $g_2^k + \gamma(x^k - x^k) \in \partial f_1(x^{k+1})$. We use this property in the sequel.

3.1.2 The Linearized Proximal Method with/without Inertial Force

Consider $\lambda = 0$ in the Algorithmic pattern 1. Suppose that $\omega : \mathbb{R}^n \to \mathbb{R}_+$ is a strongly convex and continuously differentiable function, and that $f_1(x) = \varphi(x) + \omega(x)$ and $f_2(x) = \psi(x) + \omega(x)$, i.e., ω is a regularizing function. Under this assumption, $g_2^k \in \partial f_2(x^k)$ is

given by $g_2^k = g_\psi^k + \nabla\omega(x^k)$, with $g_\psi^k \in \partial\psi(x^k)$. Accordingly, subproblem (6) (yielding (8) when $\lambda = 0$) becomes

$$\min_{x\in\mathbb{R}^n} \varphi(x) + \omega(x) - \langle g_\psi^k + \nabla\omega(x^k) + \gamma(x^k - x^{k-1}), x\rangle\,.$$

By adding the constant term $-\omega(x^k) + \langle g_\psi^k + \nabla\omega(x^k) + \gamma(x^k - x^{k-1}), x^k\rangle$ to the above subproblem we get

$$\begin{aligned}&\min_{x\in\mathbb{R}^n} \varphi(x) + \omega(x) - \omega(x^k) - \langle g_\psi^k + \nabla\omega(x^k) + \gamma(x^k - x^{k-1}), x - x^k\rangle\\ =\ &\min_{x\in\mathbb{R}^n} \varphi(x) - \langle g_\psi^k + \gamma(x^k - x^{k-1}), x - x^k\rangle + D(x, x^k),\end{aligned} \tag{10}$$

with $D(x, x^k) := \omega(x) - \omega(x^k) - \langle\nabla\omega(x^k), x - x^k\rangle$ the Bregman function induced by ω. Hence, Algorithm 1 becomes an *Inertial Linearized Proximal Method* for DC programming, a new variant of proximal methods.

In particular, suppose that $\omega(x) = \frac{\rho}{2}\|x\|^2$. Then the solution of subproblem (10) also solves

$$\min_{x\in\mathbb{R}^n} \varphi(x) - \langle g_\psi^k, x\rangle + \frac{\rho}{2}\left\|x - \left[x^k + \frac{\gamma}{\rho}(x^k - x^{k-1})\right]\right\|^2. \tag{11}$$

We care to mention that depending on the structure of φ, subproblem (11) can be efficiently solved by specialized algorithms. This is the case of nonconvex image denoising models, the application considered in Sections 5 and 6.

If one chooses $\gamma = 0$, then Algorithm 1 satisfying (8) with $\lambda = 0$ by solving (11) is just the linearized proximal method applied to problem (1) [32, 35]. On the other hand, if $\gamma > 0$ then Algorithm 1 can be seen as an extension of the *iPiano algorithm* of [31] to deal with nonsmooth DC programs (the original algorithm of [31] requires the nonconvex function -ψ to be differentiable and its gradient to be Lipschitz continuous, an assumption that is not required here).

3.1.3 Convex Setting: Proximal Method and Proximal Subgradient Splitting Method

Once again, suppose that f in (1) is given by $f_1(x) = \varphi(x) + \frac{\rho}{2}\|x\|^2$ and $f_2(x) = \psi(x) + \frac{\rho}{2}\|x\|^2$. As seen above, subproblem (6) becomes (11). Furthermore, suppose that ψ is not convex but a concave function. Then Algorithm 1 satisfying (8) with $\lambda = 0$ by solving (11) becomes a *proximal subgradient splitting method* [7] applied to the (now convex) problem $\min_{x\in\mathbb{R}^n} \varphi(x) - \psi(x)$. Differently from the proximal subgradient splitting methods found in the literature employing (11) with $\gamma = 0$, the one resulting from Algorithm 1 (under the above assumptions) is of the inertial type because it allows $\gamma \neq 0$. Besides, if $\psi := 0$ then subproblem (11) yields an inertial iteration of a *proximal method* applied to (1), which is (in this particular case) simply $\min_{x\in\mathbb{R}^n} \varphi(x)$.

3.2 Some Specific Settings for the Algorithmic Pattern with $\lambda > 0$

We start by showing that if $\lambda > 0$, then Algorithm 1 relates to the local-search method of [38].

3.2.1 A Local-search Like Method with/without Inertial Force

Note that (8) implies the following inclusion, where g_2^k is a subgradient of f_2 at point x^k,

$$g_2^k + \gamma(x^k - x^{k-1}) \in \partial_{\epsilon^{k+1}} f_1(x^{k+1}).$$

The definition of the ϵ^{k+1}-subdifferential $\partial_{\epsilon^{k+1}} f_1(x^{k+1})$ provides the inequality

$$f_1(x) \geq f_1(x^{k+1}) + \langle g_2^k + \gamma(x^k - x^{k-1}), x - x^{k+1}\rangle - \epsilon^{k+1} \quad \text{for all } x \in \mathbb{R}^n,$$

which in turn gives

$$f_1(x) - \langle g_2^k + \gamma(x^k - x^{k-1}), x\rangle \geq f_1(x^{k+1}) - \langle g_2^k + \gamma(x^k - x^{k-1}), x^{k+1}\rangle - \epsilon^{k+1} \quad \text{for all } x \in \mathbb{R}^n.$$

In particular, x^{k+1} is an ϵ^{k+1}-solution of (6). This procedure is akin to the local-search method of [38], but with the following differences: in [38] $\gamma = 0$ and the error ϵ^{k+1} must be chosen to form a summable series. We care to mention that the local-search of [38] is general enough to handle DC constraints, which is not the case of Algorithm 1. An extension of our algorithm to handle DC constraints is left for future investigation.

3.2.2 Bundle Like Algorithm with/without Inertial Force

Suppose that $\omega : \mathbb{R}^n \to \mathbb{R}_+$ is a strongly convex and continuously differentiable function, $f_1(x) = \varphi(x) + i_X(x) + \omega(x)$, $f_2(x) = \psi(x) + \omega(x)$ and X is a convex set. In order to compute a point x^{k+1} satisfying (8) we consider subproblem (6), which reads as

$$\min_{x \in X} \varphi(x) + \omega(x) - \langle g_2^k + \gamma(x^k - x^{k-1}), x\rangle\,, \tag{12}$$

and an "inner" iterative process $\nu = 0, 1, 2, \ldots$ generating a sequence of iterates $\{z^\nu\}$ whose clusters points solve (12). Instead of defining x^{k+1} as one of the cluster points of $\{z^\nu\}$, we break the inner iterative process and define $x^{k+1} := z^{\nu+1}$ as soon as $z^{\nu+1}$ is an $\epsilon^{\nu+1}$-solution of (12), with $\epsilon^{\nu+1}$ satisfying the right-side of (8). To this end, we replace the convex function φ in (12) with a cutting-plane model $\check{\varphi}^\nu$ defined by

$$\check{\varphi}^\nu(x) := \max_{j \leq \nu}\{\varphi(z^j) + \langle g_\varphi^j, x - z^j\rangle\} \quad \text{with} \quad g_\varphi^j \in \partial\varphi(z^j),\ j = 0, 1, \ldots, \nu\,. \tag{13}$$

Convexity of φ ensures that φ^ν approximates φ from below: $\varphi^\nu(x) \leq \varphi(x)$ for all x. These are the main ingredients of the following implementable scheme for computing a trial point x^{k+1} satisfying (8).

Algorithm 2 An implementable scheme for satisfying (8)

1: Given $\lambda > 0$, $0 \leq \gamma < \rho/2$, x^k and x^{k-1}, set $d^k = \gamma(x^k - x^{k-1})$ and $z^0 = x^k$
2: Compute $(\varphi(z^0),\ g_\varphi^0 \in \partial\varphi(z^0))$
3: **for** $\nu = 0, 1, 2, \ldots$ **do**
4: Define $\check{\varphi}^\nu$ as in (13) and let $z^{\nu+1}$ be a solution of

$$\min_{x \in X} \check{\varphi}^\nu(x) + \omega(x) - \langle g_2^k + d^k, x\rangle \tag{14}$$

5: Compute $(\varphi(z^{\nu+1}),\ g_\varphi^{\nu+1} \in \partial\varphi(z^{\nu+1}))$ and set $\epsilon^{\nu+1} := \varphi(z^{\nu+1}) - \check{\varphi}^\nu(z^{\nu+1})$
6: **if** $\epsilon^{\nu+1} \leq \lambda\frac{\rho}{2}\|z^{\nu+1} - x^k\|^2$ **then**
7: Stop and exit with $x^{k+1} := z^{\nu+1}$
8: **end if**
9: **end for**

Proposition 1 *Let k be fixed and suppose that x^k does not solve (12). Then Algorithm 2 stops after finitely many steps ν with a point $x^{k+1} := z^{\nu+1}$ satisfying (8).*

Proof Optimality condition of subproblem (14) yields

$$g_2^k + d^k \in \partial\check{\varphi}^\nu(z^{\nu+1}) + \nabla\omega(z^{\nu+1}) + N_X(z^{\nu+1})\,. \tag{15}$$

Since $z^{\nu+1} \in X$, we have that $N_X(z^{\nu+1}) = \partial i_X(z^{\nu+1})$ [20]. Then

$$\begin{aligned}
f_1(x) &= \varphi(x) + \omega(x) + i_X(x) && \text{(definition of } f_1\text{)}\\
&\geq \check{\varphi}^\nu(x) + \omega(x) + i_X(x) && \text{(by (13))}\\
&\geq \check{\varphi}^\nu(z^{\nu+1}) + \omega(z^{\nu+1}) + i_X(z^{\nu+1}) + \langle g_2^k + d^k, x - z^{\nu+1}\rangle && \text{(by (15))}\\
&= \varphi(z^{\nu+1}) + \omega(z^{\nu+1}) + \langle g_2^k + d^k, x - z^{\nu+1}\rangle\\
&\quad -[\varphi(z^{\nu+1}) - \check{\varphi}^\nu(z^{\nu+1})], && \text{(because } z^{\nu+1} \in X\text{)}
\end{aligned}$$

showing that $g_2^k + d^k$ belongs to $\partial_{\epsilon^{\nu+1}} f_1(z^{\nu+1})$. It remains to show that the algorithm satisfies $\epsilon^{\nu+1} = \varphi(z^{\nu+1}) - \check{\varphi}^\nu(z^{\nu+1}) \leq \lambda\frac{\rho}{2}\|z^{\nu+1} - x^k\|^2$ after finitely many steps. To this end, we recall that Algorithm 2 can be understood as a prox-form of bundle methods [10, Algorithm 4.1]. Since (12) is a strictly convex program, it thus follows from the analysis of[1] [10] that the limit point of $\{z^{\nu+1}\}$ is a solution of (12) and, moreover,

$$\lim_{\nu\to\infty}[\varphi(z^{\nu+1}) - \check{\varphi}^\nu(z^{\nu+1})] = 0, \quad \text{implying that} \quad \lim_{\nu\to\infty}\epsilon^{\nu+1} = 0\,.$$

As by assumption x^k is not a solution of (12), then the inequality $\epsilon^{\nu+1} \leq \lambda\frac{\rho}{2}\|z^{\nu+1} - x^k\|^2$ must be satisfied after finitely many steps ν. □

If x^k is the unique solution of (12), then Algorithm 2 ensures (under the analysis of [10]) that $\{z^\nu\}$ converges to x^k. Therefore, to guarantee that Algorithm 2 will always halt after finitely many steps one may consider the following additional test ensuring that $z^{\nu+1}$ is a `Tol`-solution of (12): if $\varphi(z^{\nu+1}) - \varphi^\nu(z^{\nu+1}) \leq$ `Tol`, then exit $x^{k+1} := z^{\nu+1}$.

3.2.3 Computing a point satisfying condition (9)

We now show how to compute x^{k+1} satisfying (9) for a particular class of problem (1) whose first component f_1 is of class C^1 and the domains of both f_1 and f_2 is the whole space $\mathbb{R}^n$. Given an arbitrary vector $g_2^k \in \partial f_2(x^k)$, let $y(x^k) \in \mathbb{R}^n$ be a solution of subproblem (5). Under the given assumptions, it follows that $\nabla f_1(y(x^k)) - (g_2^k + d^k) = 0$. Let $\{z^\nu\}$ be a sequence of points generated by a convergent algorithm applied to (5) (e.g. a Newtonian method, [22]) such that $\lim_{\nu\in N'} z^{\nu+1} = y(x^k)$. Then, by continuity of ∇f_1 we conclude that

$$\lim_{\nu\in N'}\nabla f_1(z^{\nu+1}) = \nabla f_1(y(x^k)) = g_2^k + d^k.$$

If there is no index $\nu \in N'$ such that $\|\nabla f_1(z^{\nu+1}) - (g_2^k + d^k)\| \leq \lambda\frac{\rho}{2}\|z^{\nu+1} - x^k\|$, then $\{z^{\nu+1}\}_{N'}$ would converge to x^k faster than $\{\nabla f_1(z^{\nu+1})\}_{N'}$ converges to $g_2^k + d^k$. This yields $y(x^k) = x^k$ and $\nabla f_1(x^k) = g_2^k + d^k$, proving that x^k is a critical point of (1) if $d^k = 0$. Otherwise, Algorithm 1 sets $x^{k+1} = x^k$ (resulting $d^{k+1} = 0$) and proceeds to next iteration. (If $d^k = 0$ and x^k is not a critical point, the condition of (9) can be satisfied after finitely many steps by some inner iterate $z^{\nu+1}$.)

[1] See Proposition 4.3 in [10] for the particular choice $\omega(x) = \frac{\rho}{2}\|x\|^2$.

3.3 An Alternative Stopping Test

The stopping test given in the Algorithmic pattern 1 is a reliable and straightforward one. However, it may not scale well the underlying optimization problem when the function is "flat" around a critical point and/or when the dimension of x is very large.

A more practical stopping test depending on the function values has been investigated in [38, Remark 4]. We thus rely on [38] and Lemma 2 below to propose the following alternative stopping test for Algorithm 1:

An alternative stopping test for Algorithm 1

1: **if** $\left| f(x^{k+1}) + \frac{(1-\lambda)\rho-\gamma}{2}\|x^{k+1}-x^k\|^2 - \left[f(x^k) + \frac{(1-\lambda)\rho-\gamma}{2}\|x^k - x^{k-1}\|^2\right]\right| \leq \texttt{Tol}$ **then**

2: Stop and return $(x^k, f(x^k))$

3: **end if**

As it will be seen in Lemma 2, the sequence $\{f(x^{k+1}) + \frac{(1-\lambda)\rho-\gamma}{2}\|x^{k+1}-x^k\|^2\}$ is monotonically decreasing. The reasoning of the above test is to stop the algorithm when the decrease issued by such a sequence is small enough.

4 Convergence Analysis

As point x^{k+1} is chosen to satisfy either (8) or (9) we conclude that $\partial f_1(x^{k+1}) \neq \emptyset$ and therefore $x^{k+1} \in \mathcal{D}\text{om}(f_1)$, implying that $\{x^k\} \subset \mathcal{D}\text{om}(f_1)$. By assumption, there exists an open set in $\mathbb{R}^n$ satisfying $\mathcal{D}\text{om}(f_1) \subset \Omega \subset \mathcal{D}\text{om}(f_2)$. Hence, the sequence of points generated by Algorithm 1 is well defined. Furthermore, since f_2 is a convex function, its subdifferential ∂f_2 is locally bounded. As a result, any sequence $\{g^k\}$ with $g^k \in \partial_{\epsilon^{k+1}} f_1(x^{k+1}) \cap \partial f_2(x^k) + d^k$ is bounded as long as $\{x^k\}$ is bounded as well. With this in mind, we start the convergence analysis of Algorithm 1 with the following simple lemma. Throughout this subsection we consider Algorithm 1 with $\texttt{Tol} = 0$.

Lemma 1 *Suppose Algorithm 1 terminates at iteration k. Then x^k is a critical point of problem (1).*

Proof Under these assumptions, the algorithm stopping test yields $x^{k+1} = x^k$ and $d^k = 0$. Either (8) or (9) provides $\partial f_1(x^k) \cap \partial f_2(x^k) \neq \emptyset$, i.e., x^k is a critical point of problem (1). □

The following lemma plays an important role in the convergence analysis of Algorithm 1.

Lemma 2 *Let $\{x^k\}$ be the sequence generated by Algorithm 1. If assumption A1 holds, $\lambda \in [0, 1)$ and $\gamma \in [0, (1-\lambda)\rho/2)$, then $\frac{(1-\lambda)\rho-2\gamma}{2} > 0$ and the sequence $\{f(x^k) + \frac{(1-\lambda)\rho-\gamma}{2}\|x^k - x^{k-1}\|^2\}$ is monotonically decreasing, that is*

$$f(x^{k+1}) + \frac{(1-\lambda)\rho-\gamma}{2}\|x^{k+1}-x^k\|^2 \leq f(x^k) + \frac{(1-\lambda)\rho-\gamma}{2}\|x^k - x^{k-1}\|^2 - \frac{(1-\lambda)\rho-2\gamma}{2}\|x^k - x^{k-1}\|^2 \quad \text{for all } k = 0, 1, 2, \ldots$$

Proof Suppose that x^{k+1} satisfies (8), and let $g_2^k + d^k \in \partial_{\epsilon^{k+1}} f_1(x^{k+1})$ with $g_2^k \in \partial f_2(x^k)$ and $d^k = \gamma(x^k - x^{k-1})$. Then $f_1(x) \geq f_1(x^{k+1}) + \langle g_2^k + d^k, x - x^{k+1}\rangle - \epsilon^{k+1}$ for all $x \in \mathbb{R}^n$. In particular, by setting $x = x^k$ and recalling that $\epsilon^{k+1} \leq \lambda\frac{\rho}{2}\|x^{k+1} - x^k\|^2$ by (8) we get

$$f_1(x^k) \geq f_1(x^{k+1}) + \langle g_2^k + d^k, x^k - x^{k+1}\rangle - \lambda\frac{\rho}{2}\|x^{k+1} - x^k\|^2. \tag{16}$$

Suppose now that x^{k+1} satisfies (9). Then

$$\begin{aligned} f_1(x^k) &\geq f_1(x^{k+1}) + \langle g_1^{k+1}, x^k - x^{k+1}\rangle \\ &= f_1(x^{k+1}) + \langle g_2^k + d^k, x^k - x^{k+1}\rangle - \langle g_2^k + d^k - g_1^{k+1}, x^k - x^{k+1}\rangle \\ &\geq f_1(x^{k+1}) + \langle g_2^k + d^k, x^k - x^{k+1}\rangle - \|g_1^{k+1} - (g_2^k + d^k)\|\|x^{k+1} - x^k\|, \end{aligned}$$

where the last step is due to the Cauchy-Schwartz inequality. Since condition (9) ensures that $\|g_1^{k+1} - (g_2^k + d^k)\| \leq \lambda\frac{\rho}{2}\|x^{k+1} - x^k\|$, we get once again inequality (16).

Assumption A1 gives the inequality $f_2(x^{k+1}) \geq f_2(x^k) + \langle g_2^k, x^{k+1} - x^k\rangle + \frac{\rho}{2}\|x^{k+1} - x^k\|^2$, which summed to (16) yields $f_1(x^k) - f_2(x^k) \geq f_1(x^{k+1}) - f_2(x^{k+1}) + \langle d^k, x^k - x^{k+1}\rangle + (1-\lambda)\frac{\rho}{2}\|x^{k+1} - x^k\|^2$. Therefore,

$$f(x^{k+1}) + (1-\lambda)\frac{\rho}{2}\|x^{k+1} - x^k\|^2 \leq f(x^k) + \langle d^k, x^{k+1} - x^k\rangle.$$

Since $\langle d^k, x^{k+1} - x^k\rangle = \gamma\langle x^k - x^{k-1}, x^{k+1} - x^k\rangle \leq \frac{\gamma}{2}\|x^k - x^{k-1}\|^2 + \frac{\gamma}{2}\|x^{k+1} - x^k\|^2$ it follows that

$$f(x^{k+1}) + \frac{(1-\lambda)\rho - \gamma}{2}\|x^{k+1} - x^k\|^2 \leq f(x^k) + \frac{\gamma}{2}\|x^k - x^{k-1}\|^2.$$

The stated result thus follows from the identity $\frac{\gamma}{2} = \frac{(1-\lambda)\rho-\gamma}{2} - \frac{(1-\lambda)\rho-2\gamma}{2}$. □

The property that the sequence $\{f(x^k) + \frac{(1-\lambda)\rho-\gamma}{2}\|x^k - x^{k-1}\|^2\}$ is monotonically decreasing is enough to prove convergence of Algorithm 1. We care to mention that the sequence of functional value $\{f(x^k)\}$ is not necessary monotone, in contrast to all other DC algorithms found in the literature (see for instance, [11, 14, 23, 32, 35, 39]). Remind that the non-monotonicity of the function values was crucial for InDCA to escape from the local solution in Example 2.

Theorem 1 *Consider Algorithm 1, assume A1,* $\lambda \in [0, 1)$, $\gamma \in [0, (1-\lambda)\rho/2)$, *and* $x^0 \in \mathcal{D}\text{om}(f_1)$. *Assume also that the level set* $L_f(x^0) := \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$ *is bounded. Then any cluster point* $\bar{x}$ *of the sequence* $\{x^k\}$ *generated by the algorithm is a critical point of problem* (1).

Proof If the algorithm terminates at a certain iteration k, Lemma 1 provides the result. Let us assume the algorithm does not stop. Lemma 2 ensures that the sequence $\{f(x^k) + \frac{(1-\lambda)\rho-\gamma}{2}\|x^k - x^{k-1}\|^2\}$ is monotonically decreasing. Since

$$\begin{aligned} f(x^k) \leq f(x^k) + \frac{(1-\lambda)\rho - \gamma}{2}\|x^k - x^{k-1}\|^2 &\leq f(x^0) + \frac{(1-\lambda)\rho - \gamma}{2}\|x^0 - x^{-1}\|^2 \\ &= f(x^0), \end{aligned}$$

the assumption of bounded level set ensures that $\{x^k\}$ is a bounded sequence. By summing up the inequality in Lemma 2 we obtain

$$\begin{aligned} 0 &\leq \frac{(1-\lambda)\rho - 2\gamma}{2} \sum_{k=0}^{\infty} \|x^k - x^{k+1}\|^2 \\ &\leq \sum_{k=0}^{\infty} \left[f(x^k) + \frac{(1-\lambda)\rho - \gamma}{2} \|x^k - x^{k-1}\|^2 \right. \\ &\qquad \left. - \left(f(x^{k+1}) + \frac{(1-\lambda)\rho - \gamma}{2} \|x^{k+1} - x^k\|^2 \right) \right] \\ &= f(x^0) - \lim_k \left(f(x^{k+1}) + \frac{(1-\lambda)\rho - \gamma}{2} \|x^{k+1} - x^k\|^2 \right) \\ &\leq f(x^0) - \inf_{x \in L_f(x^0)} f(x) < \infty \,. \end{aligned} \tag{17}$$

As a result, $\lim_k \|x^k - x^{k+1}\| = 0$ and thus, $\lim_k \epsilon^{k+1} = 0$ (Eq. (8)) and $\lim_k d^k = 0$.

Let $K \subset \{0, 1, \ldots\}$ be an index set such that $\lim_{k \in K} x^{k+1} = \bar{x}$. Since the bounded subsequence $\{x^k\}_K$ belongs to the open set $\Omega \subset \mathcal{D}\mathrm{om}(f_2)$ and f_2 is convex, then by [18, Theorem 3.1.2] the sequence $\{g_2^k\}_K$ is bounded as well and, therefore, it has a cluster point $\bar{g}_2$. Let $K' \subset K$ be an index set such that $\lim_{k \in K'} g_2^k = \bar{g}_2$. By recalling that $g_2^k + d^k \in \partial_{\epsilon^{k+1}} f_1(x^{k+1})$ if (8) holds and that the subdifferentials of convex functions are closed sets, we conclude that $\bar{g}_2 \in \partial f_1(\bar{x})$ regardless the rule (8) or (9) (because $\lim_{k \in K'} x^{k+1} = \bar{x}$ and $\lim_{k \in K'} g_2^k = \bar{g}_2$). It remains to show that $\bar{g}_2 \in \partial f_2(\bar{x})$. But this follows from the fact that $g_2^k \in \partial f_2(x^k)$ and $0 \leq \lim_{k \in K'} \|x^k - \bar{x}\| \leq \lim_{k \in K'} \|x^k - x^{k+1}\| + \lim_{k \in K'} \|x^{k+1} - \bar{x}\| = 0$ (where we have used the triangular inequality on $\|x^k - x^{k+1} - (\bar{x} - x^{k+1})\|$ and recalled the limits established above). Therefore, the closeness of the subdifferential of f_2 yields $\bar{g}_2 \in \partial f_2(\bar{x})$ and the proof is complete. □

Once the convergence analysis of Algorithm 1 has been established, we now turn our attention to its speed of convergence. To this end, we consider a particular instance of Algorithm 1.

4.1 Rate of Convergence

Throughout this section, we assume that $f_1(x) = \varphi(x) + \|x\|^2/2$ and $f_2(x) = \psi(x) + \|x\|^2/2$, yielding $\rho \geq 1$ in A1 and $\gamma \in [0, 1/2)$ in Algorithm 1. We moreover assume that $\lambda = 0$ in condition (8) and let g_ψ^k be an arbitrary subgradient of ψ at point x^k. In this manner, determining x^{k+1} satisfying either (8) or (9) results in defining (for $g_2^k = g_\psi^k + x^k$)

$$\begin{aligned} x^{k+1} &= \arg\min_{x \in \mathbb{R}^n} \varphi(x) + \|x\|^2/2 - \langle g_\psi^k + x^k + \gamma(x^k - x^{k-1}), x \rangle \\ &= \arg\min_{x \in \mathbb{R}^n} \varphi(x) + \frac{1}{2} \|x - (x^k + g_\psi^k + \gamma(x^k - x^{k-1}))\|^2 \\ &:= (I + \partial\varphi)^{-1}(x^k + g_\psi^k + \gamma(x^k - x^{k-1})) \,. \end{aligned}$$

As a result, this choice of parameter makes Algorithm 1 a linearized proximal method with inertia force. This allows us to rely on the analysis of [31, § 4.6] to establish the rate of

convergence of this method applied to the DC program (1). To this end, we denote by $r(x)$ the following residuum

$$r(x) := x - (I + \partial\varphi)^{-1}(x + g_\psi), \quad \text{with } g_\psi \in \partial\psi(x).$$

Notice that if $r(x^k) = 0$, then x^k solves $\min_{x\in\mathbb{R}^n} \varphi(x) + \frac{1}{2}\|x - (x^k + g_\psi^k)\|^2$. Its optimality condition yields $g_\psi^k \in \partial\varphi(x^k)$, showing that x^k is a critical point for problem (1) (which reads for this particular setting as $\min_{x\in\mathbb{R}^n} \varphi(x) - \psi(x)$). The following result shows that the rate of convergence of both sequences $\{\|r(x^k)\|^2\}$ and $\{\|x^{k+1} - x^k\|^2\}$ is $\mathcal{O}\left(\frac{1}{k}\right)$.

Theorem 2 *Suppose that the level set $L_f(x^0) := \{x \in \mathbb{R}^n : f(x) \le f(x^0)\}$ is bounded, $\gamma \in [0, 1/2)$ and let $\bar{f} := \min_{L_f(x^0)} f(x)$ and $k > 0$. Then*

$$(a) \qquad \min_{i\in\{0,\dots,k\}} \|x^{i+1} - x^i\|^2 \le \left(\frac{2}{1-2\gamma}\right)\frac{f(x^0) - \bar{f}}{k+1}$$

and

$$(b) \qquad \min_{i\in\{0,\dots,k\}} \|r(x^i)\|^2 \le \left(\frac{16}{1-2\gamma}\right)\frac{f(x^0) - \bar{f}}{k+1}.$$

Proof Item (a) follows directly from the development below

$$(k+1)\frac{1-2\gamma}{2}\min_{i\in\{0,\dots,k\}} \|x^{i+1} - x^i\|^2 \le \frac{1-2\gamma}{2}\sum_{i=0}^{k} \|x^i - x^{i+1}\|^2 \le f(x^0) - \bar{f},$$

where the last inequality is due to (17) by recalling that $\lambda = 0$ and $\rho \ge 1$. We now proceed to show item (b). By using the non-expansiveness of the operator $(I + \partial\varphi)^{-1}$ we have that

$$\begin{aligned}
\|x^i - x^{i-1}\| \ge \gamma\|x^i - x^{i-1}\| &= \|x^i + g_\psi^i + \gamma(x^i - x^{i-1}) - (x^i + g_\psi^i)\| \\
&\ge \|(I+\partial\varphi)^{-1}(x^i + g_\psi^i + \gamma(x^i - x^{i-1})) - (I+\partial\varphi)^{-1}(x^i + g_\psi^i)\| \\
&= \|x^{i+1} - (I+\partial\varphi)^{-1}(x^i + g_\psi^i)\|. \qquad (18)
\end{aligned}$$

As a result,

$$\begin{aligned}
\|x^{i+1} - x^i\| &\ge \|x^{i+1} - x^i\| + \|x^{i+1} - (I + \partial\varphi)^{-1}(x^i + g_\psi^i)\| - \|x^i - x^{i-1}\| \\
&\ge \|x^i - (I + \partial\varphi)^{-1}(x^i + g_\psi^i)\| - \|x^i - x^{i-1}\| \\
&= \|r(x^i)\| - \|x^i - x^{i-1}\|
\end{aligned}$$

where the first inequality follows from (18) and the second one by the triangular inequality. Therefore, $\|r(x^i)\| \le 2\max\{\|x^{i+1} - x^i\|, \|x^i - x^{i-1}\|\}$. Item (b) follows from the following development:

$$\begin{aligned}
(k+1)\min_{i\in\{0,\dots,k\}} \|r(x^i)\|^2 &\le \sum_{i=0}^{k} \|r(x^i)\|^2 \le 4\sum_{i=0}^{k} \max\{\|x^{i+1} - x^i\|^2, \|x^i - x^{i-1}\|^2\} \\
&\le 4\left[\sum_{i=0}^{k} \|x^{i+1} - x^i\|^2 + \sum_{i=0}^{k} \|x^i - x^{i-1}\|^2\right] \\
&\le 8\sum_{i=0}^{k} \|x^{i+1} - x^i\|^2 \le 8\frac{2}{1-2\gamma}[f(x^0) - \bar{f}],
\end{aligned}$$

where the second inequality holds because the arguments in the max-function are positive, and the last one is again due to (17) with $\lambda = 0$ and $\rho \geq 1$. □

5 Application of Interest: Nonconvex Image Denoising

Image reconstruction techniques have become important tools in computer vision systems and many other applications that require sharp images obtained from noisy/corrupted ones. The convex total variation (TV) formulations have proven to provide a good mathematical basis for several basic operations in image reconstruction [8]. In order to present such formulations, let $b \in \mathbb{R}^n$ be the vectorization of a corrupted $n_1 \times n_2$ *grayscale* image B (in this case, $n = n_1 \cdot n_2$). A TV formulation, with penalizing function $\phi : \mathbb{R}_+ \to \mathbb{R}$, consists in solving the following minimization problem

$$\min_{x \in \mathbb{R}^n} \frac{\mu}{2}\|x - b\|^2 + TV_\phi(x), \quad \text{with} \quad TV_\phi(x) := \sum_{i=1}^{n} \phi(\|(\nabla x)_i\|), \tag{19}$$

where $\mu > 0$ is a fidelity parameter and $(\nabla x)_i \in \mathbb{R}^2$ denotes the discretization of the gradient of image x at pixel i, that is, $(\nabla x)_i$ represents finite difference approximations of first-order horizontal and vertical partial derivatives. In a matrix representation $X \in \mathbb{R}^{n_1 \times n_2}$ of $x \in \mathbb{R}^n$ we have that

$$(\nabla x)_i := \begin{pmatrix} X_{l+1,j} - X_{l,j} \\ X_{l,j+1} - X_{l,j} \end{pmatrix}, \quad \text{with } i^{th} \text{ the coordinate of } x \text{ where the pixel } X_{l,j} \text{ is stored.}$$

Thus $\|(\nabla x)_i\| = \sqrt{(X_{l+1,j} - X_{l,j})^2 + (X_{l,j+1} - X_{l,j})^2}$.

If the penalizing function is chosen to be $\phi(r) = r$, then problem (19) consists in a convex nonsmooth optimization problem that can be efficiently solved by several specialized algorithms such the ones proposed in [1, 6, 9]. This is the main benefit of using a convex formulation for image denoising. Nevertheless, nonconvex regularizations have remarkable advantages over convex ones for restoring images, in particular, high-quality piecewise constant images with neat edges [30]. In order to preserve edges in the restoration process, some authors [25, 28] employ a nonconvex penalizing function ϕ to induce sparse image gradients $(\nabla x)_i$. This makes (19) a nonconvex and nonsmooth problem, that has been recently dealt with by local (strongly) convex approximations in [25].

In what follows we show that for a wide class of penalizing functions $\phi : \mathbb{R}_+ \to \mathbb{R}$, problem (19) is indeed a DC programming problem with available DC decompositions.

5.1 DC Decomposition of Nonconvex Denoising Models

Assume that $\phi : \mathbb{R}_+ \to \mathbb{R}$ is a concave and non-decreasing function. As a result, its right derivative is well defined for all $r \geq 0$, that is, the limit $\phi'_+(r) := \lim_{h \downarrow 0} \frac{\phi(r+h) - \phi(r)}{h}$ exists for all $r \geq 0$. Our goal is to prove that under these assumptions, the composite function $TV_\phi(x) = \sum_{i=1}^{n} \phi(\|(\nabla x)_i\|)$ can be written as a difference of two convex functions. To this end, we will need the following useful result, whose proof can be obtained by combining some developments presented in [41, § 4]. For the sake of completeness, we provide below a short proof.

Lemma 3 *Let $c : \mathbb{R}^n \to \mathbb{R}_+$ be a convex function. If $\phi : \mathbb{R}_+ \to \mathbb{R}$ is a concave and non-decreasing function such that $\phi'_+(0) < \infty$, then $\tau\, c(x) - \phi(c(x))$ is convex for all $\tau \geq \phi'_+(0)$.*

Proof The hypotheses on ϕ imply that $\phi'_+(r) \geq 0$ for all $r \geq 0$ and $\phi'_+(r_0) \geq \phi'_+(r_1)$ for all $0 \leq r_0 \leq r_1$. Let $\tau \geq \phi'_+(0) \geq 0$ be fixed and define the convex function $w(r) = \tau\, r - \phi(r)$ for all $r \geq 0$. It follows that

$$w'_+(r) = \tau - \phi'_+(r) \geq \phi'_+(0) - \phi'_+(r) \geq 0 \quad \forall r \geq 0\,,$$

showing that w is non-decreasing on $\mathbb{R}_+$. As a result, the composite function $w(c(x))$ is convex as well (because c is). Since $w(c(x)) = \tau\, c(x) - \phi(c(x))$, the result follows. □

Proposition 2 *Let $\phi : \mathbb{R}_+ \to \mathbb{R}$ be as in Lemma 3, and the TV function given by $TV(x) := \sum_{i=1}^n \|(\nabla x)_i\|$. Then $\tau\, TV(x) - TV_\phi(x)$ is a convex function for all $\tau \geq \phi'_+(0) \geq 0$.*

Proof Fix a pixel i. Since $c(x) = \|(\nabla x)_i\|$ is convex and non-negative, Lemma 3 ensures that $\tau\, \|(\nabla x)_i\| - \phi(\|(\nabla x)_i\|)$ is a convex function. The result thus follows from the fact that the sum of convex functions is convex and definitions of TV and TV_ϕ. □

Since the requirements on the penalizing function ϕ are mild, Proposition 2 is quite general. For instance, all the functions ϕ considered in [25] and reported in Table 2 satisfy the assumptions of Proposition 2 with $\tau \geq 1$. (This ensures that function f_2 in Example 2 is indeed convex.)

5.2 Specific DC Models

We now focus on subproblem (6) resulting from the considered nonconvex image denoising model. Without loss of generality, we assume in the remaining of this paper that the penalizing function $\phi : \mathbb{R}_+ \to \mathbb{R}$ satisfies $\phi'_+(0) = 1$ (this is the case of the functions presented in Table 2). In this setting, the DC function reads as

$$f(x) = \frac{\mu}{2}\|x-b\|^2 + TV_\phi(x) = \underbrace{\frac{\mu+\rho}{2}\|x-b\|^2 + TV(x)}_{f_1(x)} - \underbrace{\left[\frac{\rho}{2}\|x-b\|^2 + TV(x) - TV_\phi(x)\right]}_{f_2(x)} \tag{20}$$

In the above definition, $\rho > 0$ is an arbitrarily chosen parameter to satisfy Assumption A1. In our numerical experiments we set $\rho = 1$. Since f is nonnegative and centered around b, f is coercive and thus has bounded-level sets, satisfying thus the assumption of Theorem 1.

With this formulation, subproblem (6) reads as

$$\min_{x\in\mathbb{R}^n} \frac{\mu+\rho}{2}\|x - \zeta^k\|^2 + TV(x), \quad \text{with } \zeta^k := b + (g_2^k + \gamma(x^k - x^{k-1}))/(\mu+\rho)\,, \tag{21}$$

Table 2 Some examples of concave, differentiable and non-decreasing penalizing functions $\phi_a : \mathbb{R}_+ \to \mathbb{R}_+$ parameterized by $a > 0$

	$\phi_{\log}$	$\phi_{\mathtt{rat}}$	$\phi_{\mathtt{atan}}$	$\phi_{\exp}$
$\phi_a(r)$	$\frac{\log(1+ar)}{a}$	$\frac{r}{1+ar/2}$	$\frac{\mathtt{atan}((1+ar)/\sqrt{3})-\pi/6}{a\sqrt{3}/2}$	$\frac{1-\exp(-ar)}{a}$
$\phi'_a(r)$	$\frac{1}{1+ar}$	$\frac{1}{(1+ar/2)^2}$	$\frac{1}{1+ar+a^2r^2}$	$\frac{1}{\exp(ar)}$

i.e., a convex denoising problem with corrupted image b perturbed by $(g_2^k + \gamma(x^k - x^{k-1})/(\mu + \rho)$. As already mentioned this subproblem can be efficiently solved by several specialized methods [1, 6, 9].

6 Numerical Results

In this section, we consider the nonconvex image denoising model (19) with penalizing function $\phi_{\texttt{atan}}$ given by $\phi(r) := \frac{\texttt{atan}((1+ar)/\sqrt{3})-\pi/6}{a\sqrt{3}/2}$ with $a = 4$. Given that similar results can be achieved with the remaining functions of Table 2, adequately tuned, we only provide detailed results for this penalizing function. Since $\phi'_+(0) = 1$, Proposition 2 ensures that the objective function of (19) has the DC decomposition $f_1 - f_2$, with f_1 and f_2 given in (20).

In our numerical experiments we set $\rho = 1$ in (20) and $\lambda = 0$ in Algorithm 1. As a result, condition (8) is equivalent (for $\lambda = 0$) to define the next iterate x^{k+1} as a solution of the convex image denoising problem (21) with corrupted image perturbed by an inertial force. This task is accomplished at every iteration of our algorithm by employing the convex nonsmooth denoising method known as FISTA (Fast Iterative Shrinkage/Thresholding Algorithm) [6]. We have also tested our algorithm with $\lambda > 0$ employing the bundle-method's idea described in Algorithm 2. However, for this class of problems, the resulting DC bundle algorithm was not competitive with its exact counterpart employing FISTA. Hence, we do not report results on inexact variants of Algorithm 1. In what follows we examine the numerical performance of the following solvers, all of them coded in MATLAB version R2015b:

- `DCA` - Algorithm 1 with $\gamma = \lambda = 0$. The trial point x^{k+1} is defined by solving the convex subproblem (21) with a MATLAB implementation of FISTA.[2]
- `InDCA` - The same as `DCA`, but with inertial factor $\gamma = 0.499$ instead.
- $\texttt{InDCA}_{\gamma_k}$ - The same as `DCA`, but varying the inertial factor $\gamma > 0$ along the iterative process. We initialize the solver with $\gamma = 2.5$ and set

$$\gamma \leftarrow \max\{0.499,\ \gamma/2\} \quad \text{whenever} \quad f(x^{k+1}) + \frac{1-\gamma}{2}\|x^{k+1} - x^k\|^2 > f(x^k) + \frac{\gamma}{2}\|x^k - x^{k-1}\|^2.$$

  The reasoning behind this rule is given by Lemma 2, which ensures $f(x^{k+1}) + \frac{1-\gamma}{2}\|x^{k+1} - x^k\|^2 \leq f(x^k) + \frac{\gamma}{2}\|x^k - x^{k-1}\|^2$ for all $\gamma \in [0, 1/2)$ (because we set $\rho = 1$). However, due to the nature of function f_2, Assumption A1 may hold for a larger ρ (this is why we start with $\gamma = 2.5$ instead of $\gamma < 1/2$). If the inequality in the above rule holds and $\gamma \geq 1/2$, then γ is found to be too large to ensure convergence of the algorithm. We thus must reduce γ until becoming lower than the threshold $\rho/2 = 1/2$.
- `iPiano` - *Inertial proximal algorithm for nonconvex optimization*. This is an implementation of Algorithm 2 given in [31], with the inertial parameter therein fixed to[3] 0.8, and constants L, α given by 100 and 0.003, respectively. This choice of parameters

[2] Available at https://web.iem.technion.ac.il/images/user-files/becka/papers/tv_fista.zip

[3] For this algorithm, the inertial parameter needs to be less than one.

was made upon some tuning. This solver requires function $-f_2$ to be differentiable and Lipschitz continuous, which is not the case of f_2 given in (20). Therefore, for `iPiano` only, we replaced f_2 with the convex and differentiable function $f_2^{\texttt{smooth}}(x) := \frac{\mu}{2}\|x-b\|^2 + \sum_{i=1}^{n}[s(\|(\nabla x)_i\|) - \phi(s(\|(\nabla x)_i\|))]$, where $s(t) = \sqrt{t^2+c^2} - c$ is the pseudo-Huber function with parameter $c = 10^{-3}$. Once an oracle provides the gradient g^k of $\sum_{i=1}^{n}[s(\|(\nabla x)_i\|) - \phi(s(\|(\nabla x)_i\|))]$ at $x = x^k$, the next iterate is computed in a closed form: $x^{k+1} := [\alpha\mu b + (x^k - g^k + \gamma(x^k - x^{k-1}))]/(1+\alpha\mu)$. Therefore, when compared to the DC solvers above, `iPiano` possesses a much lower computational burden per iteration. Although we considered $f_2^{\texttt{smooth}}$ along the optimization process of `iPiano`, the function values provided in the tables below correspond to the function $f = f_1 - f_2$ (without smoothing).

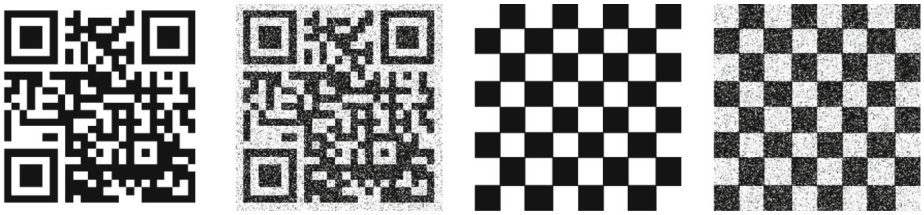
All these solvers employ the same black-box for f_2 and the same stopping-test: the iterative process terminates when the inequality

$$\max\{\|x^{k+1} - x^k\|,\ \gamma\|x^k - x^{k-1}\|\} \leq 5 \times 10^{-4}\,(1 + \|x^{k-1}\|) \quad \text{is satisfied.}$$

We set the maximum number of iterations of the DC solvers to 100. Since `iPiano` has a low computational burden per iteration, its maximum number of iterations was fixed to 1000.

The numerical performances of these four nonconvex solvers (with three of them exploiting the DC decomposition (20) of (19)) are assessed on two piecewise-constant images corrupted by a Gaussian noise with mean 0 and variance 0.1. Figure 3a and c present the original (non-corrupted) images whereas Fig. 3b and d show the corrupted ones. Each image has dimension 200×200, which yields large-scale nonsmooth DC optimization problems of dimension $n = 40\,000$. Numerical experiments were performed on a computer with Intel(R) Core(TM), i3-3110M CPU 2.40, 4G (RAM), under Windows 10, 64Bits.

In order to check the quality of the restored images we employ two well-known measures in the community of computational vision: the *peak signal-to-noise ratio* (PSNR) and the *structural similarity* (SSIM). Detailed descriptions of these measures can be found in [15, 43]. For our purposes, it is enough to keep in mind that the larger the PSNR is the better is the restoration. The same indication can be yielded when the SSIM is closer to one. Nevertheless, the focus here is on CPU time and function values provided by considered solvers: we aim at illustrating the performance of the new proposal, rather than investigating technical matters of Digital Images/Computational Vision.



(a) QR code Original (b) QR code Corrupted (c) Checkerboard Original (d) Checkerboard Corrupted

Fig. 3 Piecewise constant images. The corrupted images were obtained by the original ones by adding a Gaussian noise. All images have dimension 200×200

We start by presenting in Table 3 the results obtained by applying the four considered solvers to the nonconvex image denoising model (19) issued by the corrupted QR-code image of Fig. 3b. In our experiments, the fidelity parameter μ ranges from 0.55 to 1.25.

The results show that the inertial DC algorithms provide lower function values in less CPU time/subgradient evaluations. Concerning these features, the performance of `InDCA`

Table 3 Restoration of the corrupted QR-code image

Solver	μ	CPU(s)	# g_2	$f(x^{\texttt{best}})$	PSNR	SSIM
`iPiano`	0.55	469	1000	9322.314	21.687	0.882
`DCA`	0.55	122	58	9084.983	21.931	0.882
`InDCA`	0.55	111	53	9084.814	21.855	0.881
$\texttt{InDCA}_{\gamma_k}$	0.55	62	26	9045.975	19.253	0.839
`iPiano`	0.65	477	1000	9543.678	21.719	0.890
`DCA`	0.65	223	100	9302.109	22.033	0.889
`InDCA`	0.65	223	100	9300.447	21.997	0.888
$\texttt{InDCA}_{\gamma_k}$	0.65	140	60	9263.575	21.332	0.874
`iPiano`	0.75	473	1000	9763.515	21.591	0.891
`DCA`	0.75	234	100	9528.582	21.951	0.892
`InDCA`	0.75	189	80	9526.251	21.935	0.892
$\texttt{InDCA}_{\gamma_k}$	0.75	242	100	9476.832	21.817	0.897
`iPiano`	0.85	479	1000	9989.072	21.355	0.888
`DCA`	0.85	213	88	9762.382	21.687	0.893
`InDCA`	0.85	190	79	9755.709	21.731	0.894
$\texttt{InDCA}_{\gamma_k}$	0.85	109	45	9686.490	21.835	0.902
`iPiano`	0.95	472	1000	10209.231	21.107	0.884
`DCA`	0.95	174	72	9995.409	21.323	0.891
`InDCA`	0.95	137	57	9993.745	21.390	0.890
$\texttt{InDCA}_{\gamma_k}$	0.95	95	39	9900.572	21.505	0.876
`iPiano`	1.05	470	1000	10434.161	20.810	0.876
`DCA`	1.05	192	80	10221.634	21.103	0.888
`InDCA`	1.05	134	56	10220.961	21.055	0.887
$\texttt{InDCA}_{\gamma_k}$	1.05	84	35	10118.874	21.453	0.879
`iPiano`	1.15	472	1000	10672.028	20.545	0.864
`DCA`	1.15	163	67	10461.936	20.738	0.879
`InDCA`	1.15	162	67	10452.538	20.785	0.880
$\texttt{InDCA}_{\gamma_k}$	1.15	82	34	10338.301	21.263	0.879
`iPiano`	1.25	487	1000	10909.249	20.226	0.856
`DCA`	1.25	162	68	10706.295	20.433	0.872
`InDCA`	1.25	158	66	10694.461	20.479	0.875
$\texttt{InDCA}_{\gamma_k}$	1.25	102	42	10547.789	20.932	0.858

CPU times are given in seconds. The notation # g_2 stands for the number of subgradient evaluations of function f_2 (or $f_2^{\texttt{smooth}}$). This coincides with the number of iterations performed by the algorithms. The maximum number of subgradient evaluations was set to 1000 for solver `iPiano` and to 100 for the other solvers

was better than the one of DCA. Moreover, solver $\texttt{InDCA}_{\gamma_k}$ (that has a larger inertial factor) significantly outperformed DCA in this application. For instance, for the case $\mu = 1.15$ solver $\texttt{InDCA}_{\gamma_k}$ found a critical point with function value around 1.18% lower than the function value provided by DCA. Furthermore, in this case, solver $\texttt{InDCA}_{\gamma_k}$ was almost 50% faster than DCA.

In Fig. 4 we present the restored images obtained by the solvers applied to the instance with $\mu = 0.85$ (the one with highest values of PSNR and SSIM). In addition to the nonconvex models, we present in Fig. 4a the image restored by employing a convex model (resulting from setting $\phi(r) := r$ in (19), i.e., $TV(x)$ instead of $TV_\phi(x)$). The convex denoising model was solved by algorithm FISTA [6]. Among all the considered values of the fidelity parameter μ in (19), the best results were obtained with $\mu = 3.75$ for the convex model. The quality-measure values in Fig. 4a indicates that convex models are not effective to preserve edges in the restoration process of piecewise-constant images, corroborating thus with the conclusion drawn in [30]. Moreover, the quality of the restored image of Fig. 4a is visibly worse than the one restored by solver $\texttt{InDCA}_{\gamma_k}$. In fact, Fig. 4e contains less noise than the images restored by solvers iPiano, DCA and InDCA.

In what follows we examine the corrupted checkerboard image of Fig. 3d. Table 4 contains some results obtained by applying the four considered solvers to this image by varying the fidelity parameter μ from 0.8 to 1.6. Once again, the inertial DC solver $\texttt{InDCA}_{\gamma_k}$ provided better results than the other considered solvers. The inertial solvers required fewer iterations than DCA to terminate with a critical point of better quality (except for the instance $\mu = 0.9$, at which solver InDCA performed more iterations than DCA).

In Fig. 5 we present the restored images obtained by the solvers applied to the instance with $\mu = 1$. Once again, the convex denoising model was solved by algorithm FISTA.

6.1 Assessing Numerical Performance on Several Instances

In order to assess the numerical performances of the considered DC solvers we examine 72 instances of the nonconvex image denoising model (19), obtained by varying μ as in Table 4 and by considering four different penalizing functions ϕ as in Table 2, and the two corrupted images of Fig. 3.

We present the performance profiles [12] of DCA, InDCA and $\texttt{InDCA}_{\gamma_k}$ on these 72 instances. As an example, let the criterion of analysis be CPU time. For each solver, we plot

(a) Convex model
$PSNR = 19.453$
$SSIM = 0.813$
$f(x^k) = 10334.721$

(b) iPiano
$PSNR = 21.355$
$SSIM = 0.888$
$f(x^k) = 9989.072$

(c) DCA
$PSNR = 21.687$
$SSIM = 0.893$
$f(x^k) = 9762.382$

(d) InDCA
$PSNR = 21.731$
$SSIM = 0.894$
$f(x^k) = 9755.709$

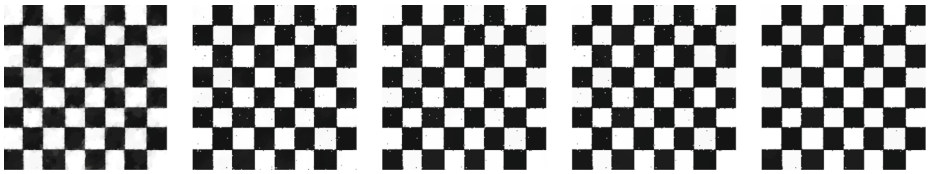
(e) $\texttt{InDCA}_{\gamma_k}$
$PSNR = 21.835$
$SSIM = 0.902$
$f(x^k) = 9686.490$

Fig. 4 Restored QR-code images. Convex model with fidelity parameter $\mu = 3.75$ and nonconvex model with $\mu = 0.85$. For comparison reasons, the final value $f(x^k)$ in (**a**) was computed with $\mu = 0.85$

Table 4 Restoration of the corrupted checkerboard image

Solver	μ	CPU(s)	# g_2	$f(x^{\texttt{best}})$	PSNR	SSIM
iPiano	0.80	474	1000	9026.884	23.879	0.820
DCA	0.80	105	56	8778.145	24.049	0.826
InDCA	0.80	81	44	8779.285	24.160	0.826
InDCA$_{\gamma_k}$	0.80	110	58	8716.393	24.239	0.772
iPiano	0.90	479	1000	9251.406	23.481	0.812
DCA	0.90	81	44	9014.267	23.670	0.822
InDCA	0.90	86	47	9006.346	23.702	0.824
InDCA$_{\gamma_k}$	0.90	60	32	8921.761	23.862	0.734
iPiano	1.00	474	1000	9475.528	23.166	0.801
DCA	1.00	112	61	9237.498	23.337	0.811
InDCA	1.00	106	57	9232.004	23.365	0.812
InDCA$_{\gamma_k}$	1.00	56	30	9138.142	24.230	0.849
iPiano	1.10	470	1000	9708.161	22.730	0.786
DCA	1.10	114	61	9481.109	22.952	0.799
InDCA	1.10	105	56	9474.022	22.968	0.801
InDCA$_{\gamma_k}$	1.10	52	28	9326.864	24.059	0.773
iPiano	1.20	472	1000	9949.721	22.205	0.761
DCA	1.20	109	58	9727.282	22.451	0.782
InDCA	1.20	92	49	9719.249	22.511	0.783
InDCA$_{\gamma_k}$	1.20	63	34	9536.481	23.969	0.803
iPiano	1.30	476	1000	10187.130	21.754	0.739
DCA	1.30	140	73	9968.101	21.992	0.765
InDCA	1.30	134	70	9959.879	22.045	0.768
InDCA$_{\gamma_k}$	1.30	77	42	9757.747	23.434	0.794
iPiano	1.40	475	1000	10424.813	21.317	0.714
DCA	1.40	149	75	10215.140	21.553	0.742
InDCA	1.40	116	59	10208.485	21.593	0.744
InDCA$_{\gamma_k}$	1.40	55	29	10035.034	22.944	0.822
iPiano	1.50	476	1000	10673.945	20.855	0.684
DCA	1.50	165	85	10473.202	21.082	0.713
InDCA	1.50	127	64	10464.077	21.091	0.716
InDCA$_{\gamma_k}$	1.50	61	31	10277.361	22.424	0.808
iPiano	1.60	476	1000	10921.842	20.390	0.661
DCA	1.60	155	80	10731.337	20.626	0.683
InDCA	1.60	136	70	10724.553	20.635	0.687
InDCA$_{\gamma_k}$	1.60	54	28	10524.680	21.858	0.786

The notation # g_2 stands for the number of subgradient evaluations of function f_2 (or $f_2^{\texttt{smooth}}$). This coincides with the number of iterations performed by the algorithms. The maximum number of subgradient evaluations was set to 1000 for solver iPiano and to 100 for the other solvers



(a) Convex model
$PSNR = 22.101$
$SSIM = 0.725$
$f(x^k) = 9513.226$

(b) `iPiano`
$PSNR = 23.166$
$SSIM = 0.801$
$f(x^k) = 9475.528$

(c) `DCA`
$PSNR = 23.337$
$SSIM = 0.811$
$f(x^k) = 9237.498$

(d) `InDCA`
$PSNR = 23.365$
$SSIM = 0.812$
$f(x^k) = 9232.004$

(e) $\texttt{InDCA}_{\gamma_k}$
$PSNR = 24.230$
$SSIM = 0.849$
$f(x^k) = 9138.142$

Fig. 5 Restored checkerboard images. Convex model with fidelity parameter $\mu = 3.4$ and nonconvex model with $\mu = 1$. For comparison reasons, the final value $f(x^k)$ in (**a**) was computed with $\mu = 1$

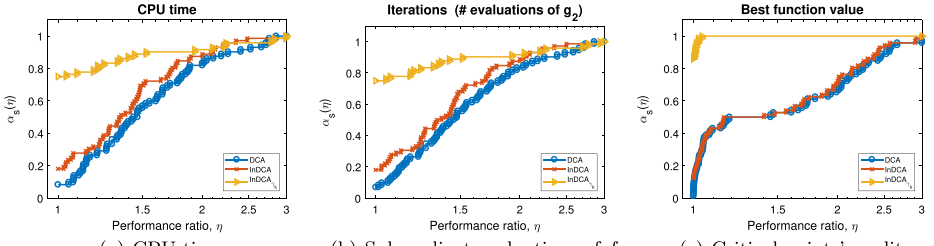
the proportion of instances that is solved within a factor η of the time required by the best algorithm. More specifically, denoting by $t_s(i)$ the time spent by solver s to solve instance i and by $t^*(i)$ the best time for the same instance among all the solvers, the proportion of instances solved by s within a factor η is

$$\alpha_s(\eta) := \frac{\text{number of instances } i \text{ such that } t_s(i) \leq \eta\, t^*(i)}{\text{total number of instances}}.$$

Therefore, the value $\alpha_s(1)$ gives the probability of the solver s to be the best by a given criterion. Furthermore, unless $t_s(i) = \infty$ (which means that solver s failed to solve instance i), it follows that $\lim_{\eta\to\infty} \alpha_s(\eta) = 1$. Thus, the higher is the line, the better is the solver (by this criterion).

Figure 6a presents the performance profile of the solvers with respect to CPU time. Solver $\texttt{InDCA}_{\gamma_k}$ was the fastest one in 74% of the considered instances, followed by `InDCA` (18%). A similar conclusion can be drawn concerning the number of subgradient evaluations of f_2 (that coincides with the number of iterations): Figure 6b shows that $\texttt{InDCA}_{\gamma_k}$ was the solver that required less subgradient evaluations in 75% of the instances.

We recall that the optimal values of the considered instances of problem (19) are unknown. To assess the quality of the solutions computed by the solvers we proceed as follows. Let $\bar{f}_i^s$ be the function value of instance i computed by solver s, and let $\bar{f}_i^{best} := \min_s \bar{f}_i^s$ be the best function value computed by the three solvers. In Fig. 6c we plot the



(a) CPU time (b) Subgradient evaluations of f_2 (c) Critical points' quality

Fig. 6 Performance profile over 72 instances of the nonconvex image denoising model (19)

performance profile of the solvers with respect to the criterion[4] $\frac{\bar{f}_i^s - f_i^{best}+1}{f_i^{best}+1}$. Solver $\texttt{InDCA}_{\gamma_k}$ computed the best function value in 86% of the cases, followed by InDCA that was the most effective one in 12.5% of the instances.

These results show that furnishing the DC algorithm with an inertial force pays off: for the considered application we observed that the quality of the computed critical points improves whereas the CPU time decreases. Finally, we comment that the total CPU time to solve these 72 instances by DCA was 137 minutes, by InDCA 127 minutes, and by $\texttt{InDCA}_{\gamma_k}$ 108 minutes. The latter provided a CPU time reduction of around 21% concerning DCA.

7 Concluding Remarks

With the purpose of computing critical points of better quality in (unconstrained or convex-constrained) DC programs we have equipped the classical DC algorithm with an inertial force. Convergence analysis and rate of convergence of the new proposal have been established. Moreover, we have investigated less demanding procedures to compute trial points and have shown that the given algorithmic pattern covers and extends some well-known optimization methods found in the literature, such as the DCA, proximal linearized method, and DC bundle methods.

The numerical performance of two variants of the given algorithmic pattern was assessed on nonconvex and nonsmooth image denoising models yielding optimization problems of dimension 40 000. In this application, every iteration of our inertial DC algorithm amounts to solving a convex image denoising model with corrupted image perturbed by an inertial force. As presented in the numerical section, such a perturbed model can be efficiently solved by specialized approaches such as the FISTA algorithm. At least for this application, our numerical experiments indicate that the proposed algorithm outperforms the classic one in terms of CPU time, number of subgradient evaluations (of the second-component function) and, mainly, in terms of quality of the computed critical points.

Acknowledgments The authors are grateful to the Reviewers and the Associate Editor for their remarks and constructive suggestions that considerably improved the original version of this article.

References

1. Afonso, M.V., Bioucas-Dias, J.M., Figueiredo, M.A.T.: Fast image recovery using variable splitting and constrained optimization. IEEE Trans. Image Process. **19**(9), 2345–2356 (2010)
2. Alvarez, F.: Weak convergence of a relaxed and inertial hybrid projection-proximal point algorithm for maximal monotone operators in hilbert space. SIAM J. Optim. **14**(3), 773–782 (2004)
3. Alvarez, F., Attouch, H.: An inertial proximal method for maximal monotone operators via discretization of a nonlinear oscillator with damping. Set-Val. Anal. **9**(1), 3–11 (2001)
4. 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. **133**(1), 23–46 (2005)
5. Bagirov, A.M., Yearwood, J.: A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. Eur. J. Oper. Res. **170**(2), 578–596 (2006)
6. Beck, A., Teboulle, M.: Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. IEEE Trans. Image Process. **18**(11), 2419–2434 (2009)

[4] We added "1" to prevent this measure being zero.

7. Bello Cruz, J.Y.: On proximal subgradient splitting method for minimizing the sum of two nonsmooth convex functions. Set-Val. Var. Anal. **25**(2), 245–263 (2017)
8. Caselles, V., Chambolle, A., Cremers, D., Novaga, M., Pock, T.: An introduction to total variation for image analysis. Theoretical Foundations and Numerical Methods for Sparse Recovery, De Gruyter, Radon Series. Comp. Appl. Math. **9**, 263–340 (2010)
9. Condat, L.: A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms. J. Optim. Theory Appl. **158**(2), 460–479 (2013)
10. Correa, R., Lemaréchal, C.: Convergence of some algorithms for convex minimization. Math. Program. **62**(2), 261–275 (1993)
11. de Oliveira, W.: Proximal Bundle Methods for Nonsmooth DC Programming. Technical report. Available at www.oliveira.mat.br (2017)
12. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
13. Gaudioso, M., Giallombardo, G., Miglionico, G.: Minimizing piecewise-concave functions over polyhedra. Math. Oper. Res. **43**(2), 580–597 (2018)
14. Gaudioso, M., Giallombardo, G., Miglionico, G., Bagirov, A.M.: Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. J. Glob. Optim. **71**, 37–55 (2018)
15. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice-Hall, Inc., Upper Saddle River (2006)
16. Gruzdeva, T., Strekalovsky, A.: A D.C. Programming Approach to Fractional Problems, pp. 331–337. Springer International Publishing, Cham (2017)
17. Hiriart-Urruty, J.: Optimisation et analyse convexe. Presses Universitaires de France, France (1998)
18. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms I, 2nd edn. No. 305 in Grundlehren der mathematischen Wissenschaften. Springer, Berlin (1996)
19. Hiriart-Urruty, J.B.: Generalized Differentiability/Duality and Optimization for Problems Dealing with Differences of Convex Functions, pp. 37–70. Springer, Berlin (1985)
20. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. No. 305-306 in Grund. der math. Wiss. Springer, Berlin (1993). (two volumes)
21. Holmberg, K., Tuy, H.: A production-transportation problem with stochastic demand and concave production costs. Math. Program. **85**(1), 157–179 (1999)
22. Izmailov, A.F., Solodov, M.V.: Newton-type Methods for Optimization and Variational Problems, 1st edn. Springer Series in Operations Research and Financial Engineering Springer International Publishing (2014)
23. 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. **68**(3), 501–535 (2017)
24. Khalaf, W., Astorino, A., d'Alessandro, P., Gaudioso, M.: A DC optimization-based clustering technique for edge detection. Optim. Lett. **11**(3), 627–640 (2017)
25. Lanza, A., Morigi, S., Sgallari, F.: Convex image denoising via non-convex regularization with parameter selection. J. Math. Imaging Vis. **56**(2), 195–220 (2016)
26. Le Thi, H.A., Pham Dinh, T.: Dc programming in communication systems: challenging problems and methods. Vietnam J. Comput. Sci. **1**(1), 15–28 (2014)
27. Le Thi, H.A., Pham Dinh, T., Ngai, H.V.: Exact penalty and error bounds in dc programming. J. Glob. Optim. **52**(3), 509–535 (2012)
28. w. Lu, C.: Image restoration and decomposition using nonconvex non-smooth regularisation and negative hilbert-sobolev norm. IET Image Process. **6**(6), 706–716 (2012)
29. Moudafi, A., Oliny, M.: Convergence of a splitting inertial proximal method for monotone operators. J. Comput. Appl. Math. **155**(2), 447–454 (2003)
30. Nikolova, M., Ng, M.K., Tam, C.P.: Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction. IEEE Trans. Image Process. **19**(12), 3073–3088 (2010)
31. Ochs, P., Chen, Y., Brox, T., Pock, T.: iPiano: Inertial proximal algorithm for nonconvex optimization. SIAM J. Imaging Sci. **7**(2), 1388–1419 (2014)
32. Pang, J.S., Razaviyayn, M., Alvarado, A.: Computing B-stationary points of nonsmooth DC programs. Math. Oper. Res. **42**(1), 95–118 (2017)
33. Polyak, B.: Some methods of speeding up the convergence of iteration methods. USSR Comput. Math. Math. Phys. **4**(5), 1–17 (1964)
34. Rockafellar, R.: Convex Analysis, 1st edn. Princeton University Press, Princeton (1970)
35. Souza, J.C.O., Oliveira, P.R., Soubeyran, A.: Global convergence of a proximal linearized algorithm for difference of convex functions. Optim. Lett. **10**(7), 1529–1539 (2016)
36. Strekalovskiy, A.: An exact penalty method for d.c. optimization. AIP Conf. Proc. **1776**(1), 060,003 (2016)

37. Strekalovsky, A., Minarchenko, I.: On Local Search in d.c. Optimization. In: 2017 Constructive Nonsmooth Analysis and Related Topics (Dedicated to the Memory of V.F. Demyanov) (CNSA), pp. 1–4 (2017)
38. Strekalovsky, A.S.: On local search in d.c. optimization problems. Appl. Math. Comput. **255**(1), 73–83 (2015)
39. Tao, P.D., An, L.T.H.: Convex analysis approach to DC programming: theory, algorithms and applications. Acta Math. Vietnamica **22**(1), 289–355 (1997)
40. Tao, P.D., Souad, E.B.: Algorithms for solving a class of nonconvex optimization problems. methods of subgradients. North-Holland Math. Stud. **129**, 249–271 (1986). Fermat Days 85: Mathematics for Optimization
41. Tuy, H.: Convex Analysis and Global Optimization, 2nd edn. Springer Optimization and Its Applications. Springer (2016)
42. van Ackooij, W., de Oliveira, W.: DC Programming Techniques with Inexact Subproblems' Solution for General DC Programs. Technical Report, CMA. Available at www.oliveira.mat.br (2017)
43. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
44. Zavriev, S.K., Kostyuk, F.V.: Heavy-ball method in nonconvex optimization problems. Comput. Math. Model. **4**(4), 336–341 (1993)