

Recovering sparse signals with a certain family of non-convex penalties and DC programming

Gilles Gasso*, Alain Rakotomamonjy, Stéphane Canu

LITIS, EA 4108 - INSA / Université de Rouen

Avenue de l'Université - 76801 Saint-Etienne du Rouvray Cedex

{gilles.gasso, alain.rakotomamonjy, stephane.canu}@insa-rouen.fr

Abstract— This paper considers the problem of recovering a sparse signal representation according to a signal dictionary. This problem could be formalized as a penalized least-squares problem in which sparsity is usually induced by a ℓ_1 -norm penalty on the coefficients. Such an approach known as the *Lasso* or *Basis Pursuit Denoising* has been shown to perform reasonably well in some situations. However, it was also proved that non-convex penalties like the pseudo ℓ_q -norm with $q < 1$ or SCAD penalty are able to recover sparsity in a more efficient way than the Lasso. Several algorithms have been proposed for solving the resulting non-convex least-squares problem. This paper proposes a generic algorithm to address such a sparsity recovery problem for some class of non-convex penalties. Our main contribution is that the proposed methodology is based on an iterative algorithm which solves at each iteration a *convex* weighted Lasso problem. It relies on the family of non-convex penalties which can be decomposed as a difference of convex functions. This allows us to apply difference of convex functions programming which is a generic and principled way for solving non-smooth and non-convex optimization problem. We also show that several algorithms in the literature dealing with non-convex penalties are particular instances of our algorithm. Experimental results demonstrate the effectiveness of the proposed generic framework compared to existing algorithms, including iterative reweighted least-squares methods.

EDICS: DSP-TFSR, MLR-LEAR

I. INTRODUCTION

“Entia non sunt multiplicanda praeter necessitatem” is a statement attributed to the 14th century philosopher, William of Occam. It is known as the Occam’s razor principle. Roughly translated, this principle becomes “entities should not be multiplied beyond necessity” and it is usually interpreted as a preference for simple models rather than complex ones for explaining a given phenomenon. This quest for simple models, where simple is understood as sparse, is still pursued by many researchers in various domains where true models are known to be sparse. Indeed, recovering sparse representation is of great interest in the signal processing and the machine learning community due to the proliferation of communication technologies and the huge data streams that are now available.

For instance, in several signal processing applications, one looks for a sparse signal representation according to a dictionary of elementary signals (wavelet, Fourier, ...). Such a problem of sparsity recovery arises for instance in compressed sensing problems [7], [16].

Similarly, supervised machine learning problems are of increasing size both in terms of number of examples and in

dimensionality. In such a context, variable selection becomes a core issue for knowledge discovery and for building predictive model in high-dimensional data space [3].

A. Problem formulation

Let consider a set of n samples gathered in the vector $\mathbf{y} \in \mathbb{R}^n$ and the $n \times d$ matrix \mathbf{X} . Within a sparse signal approximation context, \mathbf{X} would be a matrix which columns are the elements of a dictionary and \mathbf{y} the target signal, while for a machine learning problem, \mathbf{X} would be observations-variables matrix and \mathbf{y} the target output. We assume the data are described by the model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\xi}$ parameterized by $\boldsymbol{\beta} \in \mathbb{R}^d$ and where $\boldsymbol{\xi}$ is a vector of noises corrupting the data. Sparse approximation problems or variable selection problems aim to yield a model with a good accuracy evaluated by the loss $\mathcal{L}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$ along with a good sparsity measured by $\Omega(\boldsymbol{\beta})$. One formal mathematical statement of this problem can take the form of a constrained minimization problem

$$\mathcal{P}_1(\delta) \quad \min_{\boldsymbol{\beta} \in \mathbb{R}^d} \Omega(\boldsymbol{\beta}) \quad \text{s.t.} \quad \mathcal{L}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \leq \delta$$

where δ is the desired level of the discrepancy between the data and the prediction $\mathbf{X}\boldsymbol{\beta}$. In this framework, the noise free situation is handled by considering the linear constraint $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$ rather than the inequality constraint and have led to numerous papers, especially those related to compressed sensing [7], [16].

Another point of view of the problem could be

$$\mathcal{P}_2(\gamma) \quad \min_{\boldsymbol{\beta} \in \mathbb{R}^d} \mathcal{L}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \quad \text{s.t.} \quad \Omega(\boldsymbol{\beta}) \leq \gamma$$

which focuses on the minimization of the modeling error subject to a restriction on the sparsity measure. Finally a third flavor can be raised as the unconstrained minimization problem

$$\mathcal{P}_3(\lambda) \quad \min_{\boldsymbol{\beta} \in \mathbb{R}^d} \mathcal{L}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \Omega(\boldsymbol{\beta})$$

with $\lambda \in \mathbb{R}^+$, a trade-off parameter that balances the two antagonist terms. This latter formulation has the nice appeal to be related to the Bayesian framework via the the maximum a posteriori (MAP) estimation. Indeed, if the noise $\boldsymbol{\xi}$ follows an exponential type probability density function $p_{\boldsymbol{\xi}} \propto \exp(-\sigma_{\boldsymbol{\xi}} \mathcal{L}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}))$ and some priors are considered on the parameters such that $p_{\boldsymbol{\beta}} \propto \exp(-\sigma_{\boldsymbol{\beta}} \Omega(\boldsymbol{\beta}))$, the MAP framework boils down to problem $\mathcal{P}_3(\lambda)$ with $\lambda \propto \sigma_{\boldsymbol{\xi}}/\sigma_{\boldsymbol{\beta}}$ (in this formulation $\sigma_{\boldsymbol{\xi}}$ and $\sigma_{\boldsymbol{\beta}}$ represent parameters of the density

functions.). Hereafter, we will focus on the third formulation $\mathcal{P}_3(\lambda)$. We will also consider a least-squares loss function.

The obvious way to measure the sparsity is simply by counting the non-zeros elements of β . This leads to the mathematical programming problem

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \|\beta\|_0. \quad (1)$$

This problem can be understood as a penalized least-squares where the complexity of the model is related to the number of variables involved in the model. However, since the $\|\cdot\|_0$ penalty function is discrete and non-convex, solving problem (1) is NP-hard and hardly tractable when d is large. Hence in order to overcome such an issue, several works [40], [33], [8], [42] have proposed to relax the problem and instead to consider the following convex optimization problem:

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \|\beta\|_1. \quad (2)$$

This latter problem, known as the *Lasso*, has been introduced in the nineties by Tibshirani [40] and it has also been independently proposed by Chen et al. [13] as the *Basis Pursuit Denoising* problem. Although the original Lasso algorithm was proposed in the form $\mathcal{P}_2(\gamma)$ with $\Omega(\beta) = \|\beta\|_1$, it is worth mentioning that due to convexity arguments there is an equivalence between problems (2) and $\mathcal{P}_3(\lambda)$ when the samples size n is greater than the number of variables d . Even in the case this assumption does not hold, for simplicity sake, we will refer to (2) as Lasso problem. Notice also that the Lasso problem can be related to a Gaussian realization of the noise with a Laplacian prior on the parameters.

Since its introduction, the Lasso problem has been of increasing popularity. This success story comes from the fact that the problem can be easily solved either by quadratic programming approach [40], [43], homotopy approaches [34], coordinate wise optimization [22], or gradient projection method [19]. Furthermore, some theoretical arguments support the Lasso and state that under certain conditions variable selection with the Lasso can be consistent [49], [31], [44]. Other works such as those proposed by Donoho and Elad [14], [15] and Tropp [42] have proved that in certain situations the ℓ_1 penalization is able to recover the sparsity profile of the true coefficient β^* .

However, Fan and Li [18] provided some arguments against the Lasso, since they have shown that the ℓ_1 penalty associated to the Lasso tends to produce biased estimates for large coefficients. They thus suggested to replace the ℓ_1 penalty with other penalty functions that lead to sparse and unbiased models. For these purposes, they advocate that penalty functions should be singular at the origin for achieving sparsity (the same argument was also developed in [2], [32]) and should be such that their derivatives vanish for large values. These two conditions lead then to the following (more general) optimization problem

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \sum_{j=1}^d g_\lambda(|\beta_j|) \quad (3)$$

which involves a non-smooth and non-convex penalty function $g_\lambda(\cdot)$ instead of the ℓ_1 regularization term. For some reasons

TABLE I
SOME EXAMPLES OF USUAL PENALTY FUNCTIONS

Penalty	Formula
Ridge	$g_\lambda(\beta_j) = \lambda \beta_j ^2$
Lasso	$g_\lambda(\beta_j) = \lambda \beta_j $
SCAD	$g_\lambda(\beta_j) = \begin{cases} \lambda \beta_j & \beta_j \leq \lambda \\ \frac{- \beta_j ^2 + 2a\lambda \beta_j - \lambda^2}{2(a-1)} & \lambda < \beta_j \leq a\lambda \\ \frac{(a+1)\lambda^2}{2} & \beta_j > a\lambda \end{cases}$
ℓ_q	$g_\lambda(\beta_j) = \lambda \beta_j ^q, \quad 0 < q < 1$
Log	$g_\lambda(\beta_j) = \lambda \log(\beta_j + \varepsilon) - \lambda \log(\varepsilon)$
Zhang	$g_\lambda(\beta_j) = \begin{cases} \lambda \beta_j & \text{if } \beta_j < \eta \\ \lambda \eta & \text{otherwise} \end{cases}$

that will be made clear on the sequel, instead of problem (3), we will consider the following equivalent problem, obtained by splitting β_j into the difference of two positive terms $\beta_j = \beta_j^+ - \beta_j^-$

$$\min_{\beta^+, \beta^- \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{y} - \mathbf{X}(\beta^+ - \beta^-)\|^2 + \sum_{j=1}^d g_\lambda(\beta_j^+ + \beta_j^-) \quad (4)$$

s.t. $\beta_j^+ \geq 0, \quad \beta_j^- \geq 0, \quad \forall j = 1, \dots, d$

where the vectors β^+ and β^- are respectively composed of the β_j^+ and β_j^- .

B. Usual non-convex penalties

In this subsection, we focus on usual non-convex penalties proposed in the literature for recovering sparsity from equation (3). Table I and Figure 1 give an overview of the below-mentioned penalties (as well as other classical convex penalties).

One of the non-convex penalties that will have our attention throughout the paper is the Smoothly Clipped Absolute Deviation (SCAD) penalty (see Table I) promoted by Fan and Li. Indeed, after having highlighted the drawbacks of the ℓ_1 penalty, Fan and Li [18] proposed such a penalty to circumvent Lasso weak points. They then proved that the resulting estimate has interesting theoretical properties such as unbiasedness.

In the same vein, we can consider the penalty function named herein Zhang's penalty. This penalty can be viewed as a linear approximation of the SCAD penalty. It corresponds to an interpretation of a two-stage reweighted ℓ_1 penalized optimization problem. The first stage corresponds to the genuine Lasso whereas the second stage solves a Lasso problem where large parameters are not penalized anymore [47] leading thus to an unbiased model.

Among all other penalty functions which lead to sparsity, a popular one is the so-called Bridge penalty function also

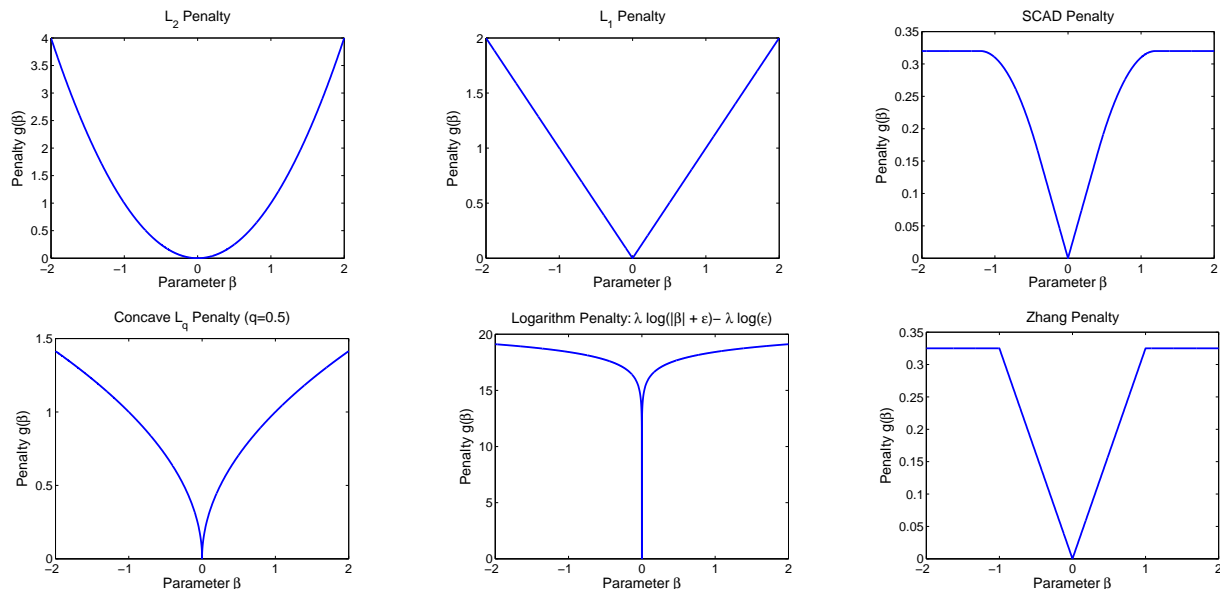


Fig. 1. Illustration of some common convex and non-convex penalty functions described in Table I.

known as the ℓ_q pseudo-norm when $0 < q < 1$ (see table I). This type of penalty has been used in ad hoc manner in different applications fields (see for instance [29] for earlier use) and popularized further in statistical community by Frank and Friedman [21] and Fu [23]. The main interest of this penalty resides in its quasi-smooth approximation of the ℓ_0 sparsity measure as q tends toward the null value. It has been shown to provide sparser solutions than the Lasso. For instance, Knight and Fu [28] provided a theoretical result which justifies the use of such a penalty for variable selection. Due to recent interests for sparse approximations and compressed sensing, other works have brought back the attention on ℓ_q penalty [10], [36]. Despite the difficulties raised by the optimization problem, several works have provided some theoretical guarantees on the use of the ℓ_q penalty [28], [24], [26], [11], [20].

Another popular non-convex penalty is the logarithm penalty (named in the sequel log penalty). This penalty was used as an approximation of the $\|\beta\|_0$ pseudo-norm by Weston et al. in a context of variable selection [45]. For sparse signal approximations, Candès et al. [9] have also investigated the use of this log penalty and empirically prove the nice capability of the resulting estimator for recovering sparsity. As depicted in Table I, for this log penalty, we shift the coefficients by a small quantity $0 < \epsilon \ll 1$ to avoid infinite value when the parameter vanishes. A Bayesian interpretation of this penalty corresponds to parameters with priors following a t-Student type distribution $p_\beta(\beta) \propto \frac{1}{|\beta|}$ [41].

C. Brief review of existing algorithms to solve least-squares problems with non-convex penalties

There exist theoretical results [49], [44], [6] which state the conditions (related to the dependence between atoms of the dictionary) under which the solution provided by the Lasso could fail to recover the sparsity pattern of the compressible

signal one intends to approximate. It turns out that a remedy to this situation is the use of at least two-stage approaches (e.g. adaptive Lasso by Zou [50] or the procedure proposed by Zhang [47]) leading to a non-convex problem. The same consequence holds if the main concern is the simultaneous model estimation and variable selection using some non-convex approximation of the ℓ_0 sparsity measure. As a result in both cases, a more challenging optimization problem has to be considered. In this subsection, we briefly review some of the algorithms available in the literature for solving (3) when $g_\lambda(\cdot)$ is non-convex.

When dealing with the SCAD penalty, Fan and Li [18] initialized their method with the least-squares solution. Then they locally approximated the non-convex penalty function with a quadratic function and a single Newton step is used for optimizing the resulting objective function. In the same flavor, Zou and Li [51] suggested to replace the local quadratic with a local linear approximation (LLA) of the penalty function leading to a one-step linear local approximation estimator. Following a discussion on the one-step SCAD estimate of Zou and Li [51], Bühlmann and Meier [5] have suggested that multi-step LLA can be used for improving sparsity of the solution. Indeed, they introduced a weighted multi-step Lasso, for which weights depend on both current solution and on user-defined regularization parameters.

When considering the ℓ_q with $q < 1$, the function $g_\lambda(\beta_j)$ is not differentiable as soon as its argument vanishes. To deal with this issue, Huang et al. [26] have proposed a parameterized differentiable approximation of the ℓ_q penalty. The approximation has been built so that it converges towards the ℓ_q function as the parameter goes to 0. For solving the resulting optimization problem, the authors use a gradient descent technique. In the context of compressed sensing, Chartrand et al. [12], [36] use an iteratively reweighted least-squares algorithm (IRLS). As a side contribution, they investigate as

well the properties of a reweighted ℓ_1 algorithm.

A reweighting procedure is also the main algorithm proposed in the literature for solving problem (3) with a log penalty. Indeed, Candès et al. [9] have used a linear approximation of the log penalty and then have iteratively solved a weighted ℓ_1 -penalized optimization problem. At each step, the weights depend on the log penalty derivative at the current solution.

Note that solving the above-presented difficult non-convex problems can be done only at the expense of higher computational complexity.

D. Our contribution

Most of the aforementioned algorithms are actually based on the same idea: using an iterative reweighted ℓ_1 or ℓ_2 algorithms for solving the non-convex problem (3). Some authors propose to use only few iterations (one or two steps) while other researchers suggest to iterate until a stopping criterion is met. While it is clear that a single iteration is computationally cheap, the optimality of such a scheme is questionable since convergence to a local or global minimum of the optimization problem is not guaranteed. Bühlmann and Meier [5] have proposed a full iterative scheme but they do not fit their algorithm into any optimization problem canvas so it is not clear which kind of non-convex penalty they are using in problem (3). Furthermore, as stated by Candès et al. [9] in their concluding remarks, one of the main drawback of a reweighted ℓ_1 scheme is its lack of convergence guarantee.

In this paper, we propose a general algorithmic framework for solving the non-convex problem (3) or (4) resulting from the use of a family of non-convex penalties. The approach relies on Difference of Convex (DC) functions programming [25]. Such a principled method for addressing non-convexity consists in decomposing a non-convex objective function into the difference of convex functions, and then in solving the resulting problem through a primal-dual approach.

Owing to such a framework, our main purposes in this paper are then to:

- develop a generic algorithm for solving some non-convex Lasso-type problems through an iterative convex scheme based on reweighted Lasso,
- show that by fitting a reweighted algorithm into a proper framework, we can get some theoretical guarantees about its convergence and give then a positive answer to an open issue raised by Candès et al. [9],
- give empirical evidences that using our DC programming approach leads to better performance in terms of sparsity recovery compared to the one-step algorithms like the one of Zou and Li [51],
- show that many of the above-described algorithms are actually particular cases of the methodology we propose.

The paper is organized as follows: Section II presents our algorithm for solving such a non-convex Lasso-like problem. After having introduced the DC programming and its convergence properties, its application to the non-convex problem (4) yielding a reweighted Lasso algorithm is detailed. The

algorithms used to solve each iteration of the DC programming are also presented. The links of our algorithm with existing procedures are highlighted in Section III. In section IV, empirical results are reported with comparison to the previously mentioned algorithms. Conclusions and perspectives of this work are discussed in Section V. The software related to this work and used for producing all the figures will be released on the authors' website.

II. DC PROGRAMMING FOR LASSO-TYPE PROBLEMS

Due to the non-convexity of function $g_\lambda(\cdot)$, solving Problem (4) is not an easy task. We address this difficulty by using an iterative procedure known as DC (Difference of Convex functions) programming [25]. This section introduces our algorithm used for solving problem (4). Before delving into the details, we start by reviewing some important notions about DC programming. Some important remarks are also added to the general DC framework in order to fit DC programming to our optimization problem.

A. Difference of Convex functions Algorithm

Let us consider the general minimization problem

$$\min_{\beta \in \mathbb{R}^d} J(\beta) \quad (5)$$

where $J(\cdot)$ is a non-convex (possibly non-smooth) criterion. The main idea of DC programming [25] is to decompose $J(\cdot)$ as:

$$J(\beta) = J_1(\beta) - J_2(\beta) \quad (6)$$

where $J_1(\cdot)$ and $J_2(\cdot)$ are lower semi-continuous, proper convex functions on \mathbb{R}^d . Then, using Fenchel conjugate notion, the dual of the minimization problem is given by [38, Section 3]

$$\min_{\alpha \in \mathbb{R}^d} J_2^*(\alpha) - J_1^*(\alpha) \quad (7)$$

where J_1^* and J_2^* are respectively the conjugate function of J_1 and J_2 and they are defined as

$$J_k^*(\alpha) = \sup_{\beta \in \mathbb{R}^d} \{\langle \beta, \alpha \rangle - J_k(\beta)\} \quad k = \{1, 2\}.$$

The DC approach is a robust algorithm based on an iterative chain-rule consisting in iteratively optimizing the primal (5)-(6) and dual (7) problems. The algorithm we use is a simplified version of the complete DC algorithm [25] and it can be summarized as follows.

From an initial estimation β^0 , the algorithm consists in building two sequences $\{\beta^t\}_{t \in \mathbb{N}}$ and $\{\alpha^t\}_{t \in \mathbb{N}}$ as illustrated in Algorithm 1. The first step of the iteration solves an approximation of the dual problem (7). Indeed, $\partial J_2(\beta)$ is the subdifferential of J_2 defined as

$$\partial J_2(\beta) = \{\alpha \in \mathbb{R}^d : J_2(\mathbf{w}) \geq J_2(\beta) + \langle \mathbf{w} - \beta, \alpha \rangle, \forall \mathbf{w} \in \mathbb{R}^d\}.$$

For differentiable criterion, the subdifferential is a singleton, hence $\partial J_2(\beta) = \{\nabla J_2(\beta)\}$. Using standard results on convex

Algorithm 1 DC Algorithm

Set $t = 0$ and an initial estimation $\beta^0 \in \text{dom } J_1$
with $\text{dom } J_1 = \{\beta \in \mathbb{R}^d : J_1(\beta) < \infty\}$
repeat
 (D_t) Determine $\alpha^t \in \partial J_2(\beta^t)$
 (P_t) Determine $\beta^{t+1} \in \partial J_1^*(\alpha^t)$
 $t = t + 1$
until convergence

optimization [35, Theorem 23.5], the following equations also hold for a given β' and α'

$$\begin{aligned} \partial J_2(\beta') &= \operatorname{argmin}_{\alpha \in \mathbb{R}^d} \{J_2^*(\alpha) - \langle \alpha, \beta' \rangle\} \\ \partial J_1^*(\alpha') &= \operatorname{argmin}_{\beta \in \mathbb{R}^d} \{J_1(\beta) - \langle \beta, \alpha' \rangle\}. \end{aligned}$$

From these definitions, we can see that by replacing $J_1^*(\alpha)$ in (7) by its affine minorization $J_1^*(\alpha^t) + \langle \alpha - \alpha^t, \beta^t \rangle$ at the point α^t , the dual problem (7) becomes equivalent to find $\partial J_2(\beta^t)$. Hence, choosing an $\alpha^t \in \partial J_2(\beta^t)$ is equivalent to solve an approximation of problem (7). Similarly, by replacing the convex function $J_2(\beta)$ in (6) by its affine minorization $J_2(\beta^t) + \langle \beta - \beta^t, \alpha^t \rangle$ at the neighborhood of β^t leads to the primal problem (P_t) . According to all these points, the iterative primal-dual resolution of problem (5) leads to Algorithm 1.

Before explaining how we have adapted this algorithm to our non-convex problem (4), we state that the local convergence of the algorithm is guaranteed and depends on the adopted DC decomposition and the initial point [38]. These convergence properties rely on the following theorem.

Theorem 1: Assume a DC decomposition of the function J as $J = J_1 - J_2$ with $J_1, J_2 : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{\infty\}$ lower semi-continuous, proper convex functions such as $\text{dom } J_1 \subset \text{dom } J_2$ and $\text{dom } J_2^* \subset \text{dom } J_1^*$ (The finiteness of the minimum of the function J implies these conditions). It holds that

- (i) The sequence $\{\beta^t\}_{t \in \mathbb{N}}$ (respectively $\{\alpha^t\}_{t \in \mathbb{N}}$) is well defined or equivalently $\text{dom } \partial J_1 \subset \text{dom } \partial J_2$ (respectively $\text{dom } \partial J_2^* \subset \text{dom } \partial J_1^*$).
- (ii) The primal objective value sequence $\{J_1(\beta^t) - J_2(\beta^t)\}_{t \in \mathbb{N}}$ (respectively dual objective value sequence $\{J_2^*(\beta^t) - J_1^*(\beta^t)\}_{t \in \mathbb{N}}$) is monotonically decreasing (resp. increasing).
- (iii) If the minimum of J is finite and the sequence $\{\beta^t\}_{t \in \mathbb{N}}$ (respectively $\{\alpha^t\}_{t \in \mathbb{N}}$) is bounded, every limit point $\hat{\beta}^*$ (respectively $\hat{\alpha}^*$) is a critical point of $J_1 - J_2$ (respectively $J_2^* - J_1^*$), that is this point satisfies the local optimality condition.

The theorem ensures a decrease of J from an iteration to the next and therefore a convergence to a stationary point of the objective function. The formal proof of the convergence properties is beyond the scope of this paper and we refer the reader to [38, Theorem 3.7 and Appendix] for details.

Remarks

- (i) If the objective function $J(\cdot)$ is defined over a convex set $\mathcal{X} \subset \mathbb{R}^d$, the DC framework is still valid by applying the

previous analysis to the unconstrained minimization problem over the objective value function $J(\beta) + \Gamma_{\mathcal{X}}(\beta)$ where $\Gamma_{\mathcal{X}}(\beta)$ is the indicator function defined as $\Gamma_{\mathcal{X}}(\beta) = 0$ if $\beta \in \mathcal{X}$ and $\Gamma_{\mathcal{X}}(\beta) = +\infty$ otherwise [39]. This remark makes the DC framework suitable for our problem, since the feasible domain of our optimization problem given in Equation (4) is \mathbb{R}_+^{2d} .

(ii) The general DC framework presented above can also be derived by considering functions on \mathbb{R}_+^{2d} . Indeed, since the dual cone of \mathbb{R}_+^{2d} is still \mathbb{R}_+^{2d} , all the derivations would have been the same, leading then to the same DC algorithm.

(iii) Notice that Yuille and Rangarajan [46] have independently proposed a similar approach based on a concave-convex decomposition of the objective function. The DC algorithm is also highly related to the surrogate maximization (or minorization-maximization) algorithms which extend the spirit of the EM algorithm. The basic idea is to consider a two-step algorithm : a surrogate step consisting in bounding the objective function as a surrogate at the current solution and a maximization step which yields the next estimation of the parameters by optimizing the surrogate function. The key point of this approach is the construction of the surrogate function. A recent paper of Zhang and al. [48] nicely reviews how to build such a surrogate function in different machine learning problems. Unlike DC programming, these related approaches do not suit to non-smooth objective function.

B. Formulating Lasso-type problems as a DC program

With these elements on DC programming in hands, let us turn back to the original optimization problem (4). The first step for applying DC programming is to decompose the penalty $g_{\lambda}(\cdot)$ defined on \mathbb{R}_+ as the difference of two convex functions on \mathbb{R}_+ :

$$g_{\lambda}(\cdot) = g_{\text{vex}}(\cdot) - h(\cdot). \quad (8)$$

Then according to this decomposition, we define

$$\begin{aligned} J_1(\beta^+, \beta^-) &= \frac{1}{2} \|\mathbf{y} - \mathbf{X}(\beta^+ - \beta^-)\|^2 + \sum_{j=1}^d g_{\text{vex}}(\beta_j^+ + \beta_j^-) \\ J_2(\beta^+, \beta^-) &= \sum_{j=1}^d h(\beta_j^+ + \beta_j^-). \end{aligned}$$

As we will show in the sequel, for the penalty functions $g_{\lambda}(\cdot)$ we use, the function h is explicitly known. Hence, at each iteration t , we are able to analytically determine an element of the subdifferential α^t of $J_2(\beta^+, \beta^-)$. Thus, if $h(\cdot)$ is differentiable, we have $\alpha_j^t = h'(\beta_j^{+t} + \beta_j^{-t})$, $j = 1, \dots, d$ whereas if $h(\cdot)$ is not differentiable, the procedure still holds by picking arbitrarily any element of $\partial h(\beta_j^{+t} + \beta_j^{-t})$ at the points where h is not differentiable as in [1], [30].

Now, according to the (P_t) step of Algorithm 1 and to Equation (8), once α^t is known, the next value of β can be found by minimizing with respect to β^+ and β^- the following objective function

$$\frac{1}{2} \|\mathbf{y} - \mathbf{X}(\beta^+ - \beta^-)\|^2 + \sum_{j=1}^d g_{\text{vex}}(\beta_j^+ + \beta_j^-) - \langle \beta_j^+ + \beta_j^-, \alpha_j^t \rangle \quad (9)$$

Algorithm 2 Iterative optimization based on DC programming

Set an initial estimation β^0 , $t = 0$

repeat

Determine $\beta^{+,t+1}$ and $\beta^{-,t+1}$ by minimizing

$$\frac{1}{2} \|\mathbf{y} - \mathbf{X}(\beta^+ - \beta^-)\|^2 + \sum_{j=1}^d \mathbf{g}_{\text{vex}}(\beta_j^+ + \beta_j^-)$$

$$- \sum_{j=1}^d \alpha_j^t (\beta_j^+ + \beta_j^-)$$

$$\text{s.t. } \beta_j^+ \geq 0, \beta_j^- \geq 0, \quad \forall j = 1, \dots, d$$

$$\text{with } \alpha_j^t = h'(\beta_j^{+,t} + \beta_j^{-,t})$$

$$t = t + 1$$

until convergence of β

We then get the iterative scheme summarized by Algorithm 2.

C. DC decomposition of non-convex penalty functions

Now let us detail how we have decomposed the non-convex penalties into a difference of convex functions. It is easy to show that for any function $\mathbf{g}_\lambda(\cdot)$ such a decomposition is not unique and that its choice may be crucial in order to make the problem (9) tractable.

In this work, we have chosen a decomposition so that problem (9) is a Lasso type problem. By doing so, we are then able to adapt or re-use efficient algorithms developed for the Lasso. The decomposition is as following

$$\mathbf{g}_{\text{vex}}(\cdot) = \lambda |\cdot| \quad (10)$$

$$h(\cdot) = \lambda |\cdot| - \mathbf{g}_\lambda(\cdot) \quad (11)$$

where $h(\cdot)$ is a convex function. Due to the non-negativity of $\mathbf{g}_\lambda(\cdot)$, we necessarily have $h(\beta) \leq \lambda |\beta|$. From Equations (10-11), we can derive the expressions of $h(\cdot)$ for the non-convex penalties of Table I. These expressions are gathered in Table II. Graphical illustrations of these DC decompositions are depicted in Figure 2.

TABLE II

ASSUMING $\mathbf{g}_{\text{vex}}(\beta_j) = \lambda |\beta_j|$ OF THE DC DECOMPOSITION, THIS TABLE PRESENTS THE CORRESPONDING FUNCTION h OF THE NON-CONVEX PENALTIES OF TABLE I.

Penalty	Expression of h
SCAD	$h_{\text{Scad}}(\beta_j) = \begin{cases} 0 & \text{if } \beta_j \leq \lambda \\ \frac{ \beta_j ^2 - 2\lambda \beta_j + \lambda^2}{2(\alpha-1)}, & \lambda < \beta_j \leq a\lambda \\ \lambda \beta_j - \frac{(\alpha+1)\lambda^2}{2}, & \beta_j > a\lambda \end{cases}$
ℓ_q	$h_{\ell_q}(\beta_j) = \lambda (\beta_j - \beta_j ^q)$
Log	$h_{\text{log}}(\beta_j) = \lambda (\beta_j - \log(\beta_j + \varepsilon) + \log(\varepsilon))$
Zhang	$h_{\text{Zhang}}(\beta_j) = \begin{cases} 0 & \text{if } \beta_j < \eta \\ \lambda \beta_j - \lambda\eta & \text{otherwise} \end{cases}$

From these figures, we can note that the functions $h(\cdot)$ for the log and ℓ_q penalties are actually not convex on \mathbb{R} . However,

TABLE III
EXPRESSION OF THE SUBDIFFERENTIAL OF THE FUNCTIONS h PRESENTED IN TABLE II.

Penalty	Expression of ∂h
SCAD	$h'_{\text{Scad}}(\beta_j) = \begin{cases} 0, & \beta_j \leq \lambda \\ \frac{2\beta_j - (2\alpha-1)\lambda \text{sign}(\beta_j)}{2(\alpha-1)}, & \lambda < \beta_j \leq a\lambda \\ \lambda \text{sign}(\beta_j), & \beta_j > a\lambda \end{cases}$
ℓ_q	$h'_{\ell_q}(\beta_j) = \lambda \text{sign}(\beta_j) \left(1 - \frac{q}{ \beta_j ^{1-q+\varepsilon}}\right)$
Log	$h'_{\text{log}}(\beta_j) = \lambda \text{sign}(\beta_j) \left(1 - \frac{1}{ \beta_j +\varepsilon}\right)$
Zhang	$h'_{\text{Zhang}}(\beta_j) = \begin{cases} 0 & \text{if } \beta_j < \eta \\ \lambda \text{sign}(\beta_j) & \text{otherwise} \end{cases}$

their restrictions on the interval \mathbb{R}_+ are convex and since in our problem β_j^+ and β_j^- are positive, our difference of convex functions decomposition still holds for these two penalties and the DC algorithm still applies as stated in the above remark. This is the main reason why we consider problem (4) instead of problem (3).

D. Solving each DC iteration

According to the proposed decomposition and the problem formulation, we now detail how each DC iteration can be solved. At each iteration t , the minimization problem is

$$\min_{\beta^+, \beta^-} \frac{1}{2} \|\mathbf{y} - \mathbf{X}(\beta^+ - \beta^-)\|^2 + \sum_{j=1}^d \lambda (\beta_j^+ + \beta_j^-) \quad (12)$$

$$- \sum_{j=1}^d \alpha_j^t (\beta_j^+ + \beta_j^-)$$

$$\text{s.t. } \beta_j^+ \geq 0, \beta_j^- \geq 0, \quad \forall j = 1, \dots, d$$

where $\alpha_j^t \in \partial h(\beta_j^{+,t} + \beta_j^{-,t})$. This supposes that at each iteration, we are able to compute the subdifferential of h for any β_j .

For differentiable penalty like SCAD, the subdifferential is equivalent to the derivative. For other penalties like the Zhang's one, at any point where the subdifferential is not reduced to a singleton due to the non-differentiability, we can set α as any element of the subdifferential. Using such a trick does not harm the convergence guarantee. For penalties function that are not differentiable at zero, we have used the classical trick (used for instance, by Candès et al. [9] and Chartrand et al. [36]) which consists in adding a ε term to the coefficient β_j for avoiding a division by zero. All the subgradients we use for the different penalties are summarized in Table III.

After having decomposed the coefficient vector β in order to cope with the DC programming formulation, we see that problem (12) can be formulated as a weighted Lasso problem :

$$\min_{\beta} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \sum_{j=1}^d \lambda_j |\beta_j| \quad (13)$$

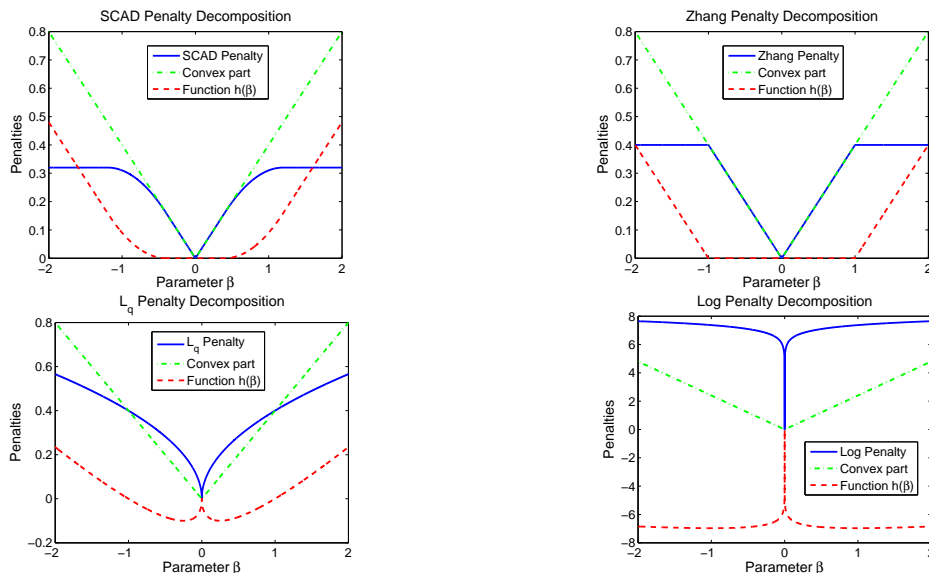


Fig. 2. Illustration of the proposed difference of convex functions decompositions. From the left to the right, Top row) SCAD penalty and Zhang penalty, Bottom row) ℓ_q and the log-penalty. For clarity sake, the plotted convex part of the log penalty is $6\lambda|\beta|$.

where $\lambda_j = \lambda - \alpha_j^t = \lambda - h'(|\beta_j^t|)$.

Recently, several authors have proposed very efficient algorithms for ℓ_1 penalized least-squares [27], [22], [19]. Among all these methods, the work of Figueiredo et al. [19] and Friedman et al. [22] can be applied or extended to problem (13).

We extended the coordinate-wise optimization idea of Friedman et al. which is known to be very efficient and competitive compared to homotopy or quadratic programming approaches [17], [34], [40]. The algorithm we developed for solving the problem (13) is based on coordinate-wise optimization but takes into account the optimality constraints for selecting the variable to optimize. This makes the algorithm still more efficient (see for instance [37]). Early comparisons of this algorithm with the gradient projection of Figueiredo et al. [19] have shown that the latter algorithm is more suited to the kind of problem we solve and thus is more efficient. For this reason, we have decided to use the code of Figueiredo et al. which is available online.

Note that although all algorithms which address a ℓ_1 penalized least-squares can be used for solving problem (13), one particularity of the DC algorithm has to be taken into account when choosing an algorithm for solving the weighted Lasso. Indeed, the DC procedure iteratively solves problem (13) several times with an update of the weights λ_j at each iteration. Thus since, it is expected that between two iterations the optimal coefficient β^t changes a little, it would be interesting from a computational demand viewpoint that the algorithm used for solving (13) can benefit from a good initialization. Both coordinate-wise and the gradient projection algorithms are able to do so. Hence after the first DC iteration, we use the solution of the previous iteration as an initialization of the current one. This simple trick, known as warm-start technique, makes the overall algorithm just slightly more expensive than a one-step Lasso resolution (see Figure 6). Furthermore, warm-

start can also be used when solving the DC problem for different values of λ .

E. Algorithm termination

For solving problem (3), we use two nested algorithms which both need a termination criterion in order to get a solution in a finite time.

By writing down the Lagrangian associated to problem (13), the KKT optimality conditions are

$$\begin{aligned} \mathbf{x}_j^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \lambda_j \text{sign}(\beta_j) &= 0 & \text{if } \beta_j \neq 0 \\ |\mathbf{x}_j^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})| &< \lambda_j & \text{if } \beta_j = 0 \end{aligned} \quad (14)$$

where \mathbf{x}_j represents the j^{th} column of the matrix \mathbf{X} . Then, we can consider a coefficient vector $\boldsymbol{\beta}$ as optimal if it satisfies the KKT conditions up to a tolerance τ . Thus, we stop the Lasso algorithm when

$$\begin{aligned} (\mathbf{x}_j^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})) - \lambda_j \text{sign}(\beta_j) &< \tau & \text{if } \beta_j \neq 0 \\ |\mathbf{x}_j^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})| &< \lambda_j + \tau & \text{if } \beta_j = 0 \end{aligned}$$

or when we reach a maximal number of iterations. In our numerical experiments, we have set $\tau = 0.001$.

For the DC algorithm, we stop iterations when either $(\|\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^t\|_\infty \leq 10^{-3})$ (the supremum norm of the coefficient vector difference between two consecutive iterations is below 10^{-4}) or the maximal number of 30 iterations is reached.

III. DISCUSSIONS

In this section, we provide some discussions about the relation between our DC algorithm and other methods for solving Lasso-type problems. We also provide some comments about the technical issues concerning our algorithm as well as theoretical perspectives related to our work.

A. Relations with other Lasso-type problems

As we stated in the introduction, recently they have been a lot of work dealing with the Lasso problem. For instance, Zhang has proposed a two-stage Lasso [47], Zou has introduced the adaptive Lasso [50], while Zou and Li improved the SCAD of Fan and Li by using Local Linear Approximation [51]. All these approaches can be considered as particular cases of our general way of solving the non-convex problem when the appropriate penalty function is used.

For instance, the two-stage Lasso of Zhang consists firstly in solving a genuine Lasso. Then according to the resulting estimate $\hat{\beta}$ and a user-defined threshold η , the second stage is a weighted Lasso where the weights are

$$\lambda_j = \begin{cases} \lambda & \text{if } |\hat{\beta}_j| \leq \eta \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that, if an initial estimation $\beta^0 = 0$ is used, according to our definition of the Zhang's penalty and to the expression of $h'_{\text{Zhang}}(\beta_j)$, the first two DC iterations of our algorithm exactly recover the two-stage approach of Zhang. This connection gives a more intuitive interpretation of Zhang's two-stage Lasso. Indeed, it makes clear that the overall penalty he suggests, is actually a non-convex penalty which is a linear approximation of SCAD penalty.

SCAD penalty function is a non-convex and differentiable function which nicely fits in our DC algorithm. The first algorithm, proposed by Fan and Li [18] is based on a local quadratic approximation. This leads to an iterative ridge regression which is computationally more expensive than the Lasso. Recently, Zou and Li advocate the use of local linear approximation (LLA) around a least-squares initial estimation $\hat{\beta}_{\text{MC}}$ and compute the final solution through a weighted Lasso. Thus, Zou and Li algorithm corresponds to one-step of our DC procedure with $\beta^0 = \hat{\beta}_{\text{MC}}$.

In the adaptive Lasso promoted by Zou [50], the authors suggest a theoretically-justified single-step weighted Lasso. The weight of each parameter β_j in the penalty is set as $\lambda_j \propto \frac{1}{|\beta_j^t|^\gamma}$ where γ is an user-defined hyperparameter and β^t is an initial vector solution. Owing to our DC approach, we can relate the adaptive Lasso approach to a non-convex penalty function. As far as our knowledge goes, this relation is novel in the literature. Indeed, when considering the log and the ℓ_q penalty, it is easy to see that at each iteration, the weights λ_j in our DC iteration are given by

$$(\text{log}): \lambda_j = \frac{\lambda}{|\beta_j^t| + \varepsilon} \quad (\ell_q): \lambda_j = \frac{\lambda q}{|\beta_j^t|^{1-q} + \varepsilon} \quad (15)$$

Therefore, the adaptive Lasso is a particular case of our DC algorithm using log penalty and considering only two iterations.

From these comparisons, we note that the above-described Lasso-type algorithms attempt to solve a non-convex problem by using few reweighted Lasso iterations (typically two steps). Using our DC approach, more iterations can occur until the stopping condition is met. Hence, our algorithm ensures a smaller optimization error owing to a better convergence of the non-convex optimization problem to a local minimum. As

stated by Bottou and Bousquet [4], optimization error may be of primary importance in statistical estimation. Although, we have not theoretically studied the impact of reducing the optimization error compared to the estimation or approximation error, we have empirically shown in the sequel, that using more iterations than two can improve the sparsity estimation.

In the literature, two recent algorithms proposed by Chartrand and Yin [12] and Candès et al. [9] share the same spirit of solving a non-convex problem using an iterative reweighted scheme until complete convergence. These algorithms were applied in the context of compressive sensing: the first paper addresses the exact recovery of sparse signal while the second paper also considers sparsity recovery in noisy situations. In [12], the authors have proposed an iterative reweighted ℓ_2 for solving the non-convex problem with a ℓ_q penalty. Section IV reports a comparative study of our algorithm with reweighted least-squares approach and shows that the computational cost of the latter algorithm is high when the size of the dictionary increases. The algorithm of Candès et al. is the one closest to our DC procedure. Indeed, they have solved the problem with a log penalty through a reweighted ℓ_1 minimization scheme. In this paper, we provide a generic framework and efficient algorithm for dealing with a more extended family of non-convex penalties. Another fact to be pointed out is the issue raised by Candès et al. concerning the convergence properties of their algorithm. By fitting such an iterative scheme into the DC framework and using the well established results of DC programming theory, we can provide a positive answer to that question.

Recently, another multi-step (typically three or four) adaptive Lasso has been proposed by Bühlmann and Meier [5]. At each iteration, they used a reweighted Lasso for which weights are similar to those provided in equation (15) for the log penalty. Furthermore, they proposed to select at each iteration the best regularization parameter λ according to some cross-validation scheme. Since, adaptive Lasso is related to the log penalty, if we omit this extra hyper-parameter selection, their algorithm is exactly the one proposed by Candès et al. and thus is equivalent to our DC approach when using log penalty.

B. Comments on initialization

Our DC algorithm is initialized with a coefficient vector β^0 so that the first iteration is a Lasso iteration. This corresponds to a coefficient vector equals to zero for some penalties like the SCAD or Zhang's penalty. However, since the problem we solve is non-convex, it is expected that different initializations lead to different optimal coefficient vectors $\hat{\beta}$. In the results section, an empirical analysis of initialization impact on the solution is carried out. We show there that starting with a Lasso iteration is a reasonable approach.

For the adaptive Lasso [50] and the one-step LLA [51], when $d < n$, these algorithms are initialized with the ordinary least-squares coefficients. Indeed, they need a consistent estimator of the β for initializing and for computing the adaptive weight in order to guarantee some theoretical properties. In the same situation, if our algorithm is initialized in the

same way, our estimate $\hat{\beta}$ shares the same properties since we exactly solve the same optimization problem but the optimization error of our solution (for finite n) is better.

C. Dealing with ε and λ

The parameter ε in the log and ℓ_q penalty definition acts as a barrier that prevents numerical instabilities. Indeed, the weights α_j^t can be large or undefined for small or null values of β_j^t . The optimal choice of this parameter can be done using the strategies suggested in [12] or [9]. The first option is to implement an annealing design of ε : begin with a large value of ε , run Algorithm 2, then decrease ε by a given factor and re-run the DC algorithm initialized with its previous output. Such a scheme is then iterated until convergence. An alternative is to set ε as a function of the magnitude of the coefficients β_j^t at the current iteration or simply at a meaningful fixed value. Section IV provides an analysis of the recovery performances when ε is fixed or optimized via an annealing scenario. The results are definitively favorable to the annealing strategy (see Figure 4).

The performance evaluation of the algorithm requires the determination of the regularization parameter λ . Using a grid search, the computation of the regularization path can be carried out efficiently owing to the warm-start property of the coordinate-wise and gradient projection algorithms devised in Section II.

IV. NUMERICAL EXPERIMENTS

In this section, we present some experiments that demonstrate the benefits of using non-convex penalties and DC programming. First, we aim at confirming the evidence that non-convex penalties are performing better than the ℓ_1 penalty for recovering sparse signals. Then, we also empirically prove that using a DC algorithm and thus a reweighted iterative scheme leads to better performance than a one-step estimator such as the adaptive Lasso [50] or the one-step SCAD [51]. We also compare our approach to an iterative reweighted least square (IRLS) method as proposed by Saab *et al.* [36]. For these purposes, we have designed several experiments for which the sparse signal to be recovered has been built as follows. For a given pre-defined values of the dictionary size d and a number k of active elements in the dictionary, the true coefficient vector β^* has been obtained as follows. The k non-zeros positions are chosen randomly and the non-zeros values are drawn from zero-mean unit variance Gaussian distribution. We have sampled a random matrix \mathbf{X} of dimension $n \times d$ with columns drawn uniformly from the surface of a unit hypersphere of dimension n . Then, the target vector is obtained as

$$\mathbf{y} = \mathbf{X}\beta + \xi$$

where ξ is a noise vector drawn from i.i.d Gaussian distribution with zero-mean and variance σ_b^2 determined from a given Signal-To-Noise as

$$\sigma_b^2 = \frac{1}{n} \|\mathbf{X}\beta\|^2 \cdot 10^{-SNR/10}$$

For all experiments, we present results averaged over a certain number of trials. For each trial, only the Gaussian noise has been resampled (this procedure corresponds to a fixed design regression setup contrary to a random design which consists in generating a new matrix \mathbf{X} at each trial). The default values we use are $d = 256$, $n = 128$.

As a performance measure, we have used the F-measure between the support of the true coefficient vector β^* and the estimated one $\hat{\beta}$. The support of a given vector β is defined as

$$\text{supp}(\beta) = \{j : \beta_j > \gamma\}$$

where γ is a threshold that allows us to neglect some true non-zero coefficients that may be obliterated by the noise (γ has been set to 0.01). The Fmeasure F_{meas} is then obtained as :

$$F_{meas} = 2 \frac{|\text{supp}(\beta^*) \cap \text{supp}(\hat{\beta})|}{|\text{supp}(\beta^*)| + |\text{supp}(\hat{\beta})|}$$

In the ideal situation where $\text{supp}(\beta^*) = \text{supp}(\hat{\beta})$, the Fmeas is equal to 1.

Note that unless specifically mentioned, for all the results presented in the following the parameter λ have been selected by 5-fold validation method according to a mean-squares error criterion.

A. Illustration of algorithm effectiveness

In this first part, we aim at checking the effectiveness of our algorithm. We briefly illustrate that several DC iterations compared to two-step methods lead to a decrease of the objective value and thus to a better convergence to a local minimum.

Figure 3 shows an example of objective value evolution with respect to the DC iterations as well as the estimated coefficient vector $\hat{\beta}$ after 2 iterations and after full convergence of the DC procedure. We can see that for both SCAD and log penalties, the objective value keeps decreasing along the DC iterations. While only slight objective decrease may be achieved, more DC iterations yield a better estimation of the true coefficient sparsity support. Indeed, the two-stage procedures tend to produce more non-zero coefficients than the ground truth. These coefficients are shrunk to zero when more iterations are performed. This example is just an illustration of how our algorithm behaves. The next paragraph gives a more detailed analysis of its performance.

B. Comparing performances

The subsection describes performance comparisons of the DC approach with the considered penalty functions along with the comparison of our algorithm with the Iteratively Reweighted Least-Squares (IRLS) approach [12].

Before examining these results, let us investigate the influence of ε in our DC approach for the log and the ℓ_q penalties. We compare the recovery performance in two cases: a DC approach with a fixed value of $\varepsilon = 0.001$ and a DC approach with an annealing strategy (as suggested by Chartrand *et al.* [12]) which starts from $\varepsilon = 1$, solves the DC problem until

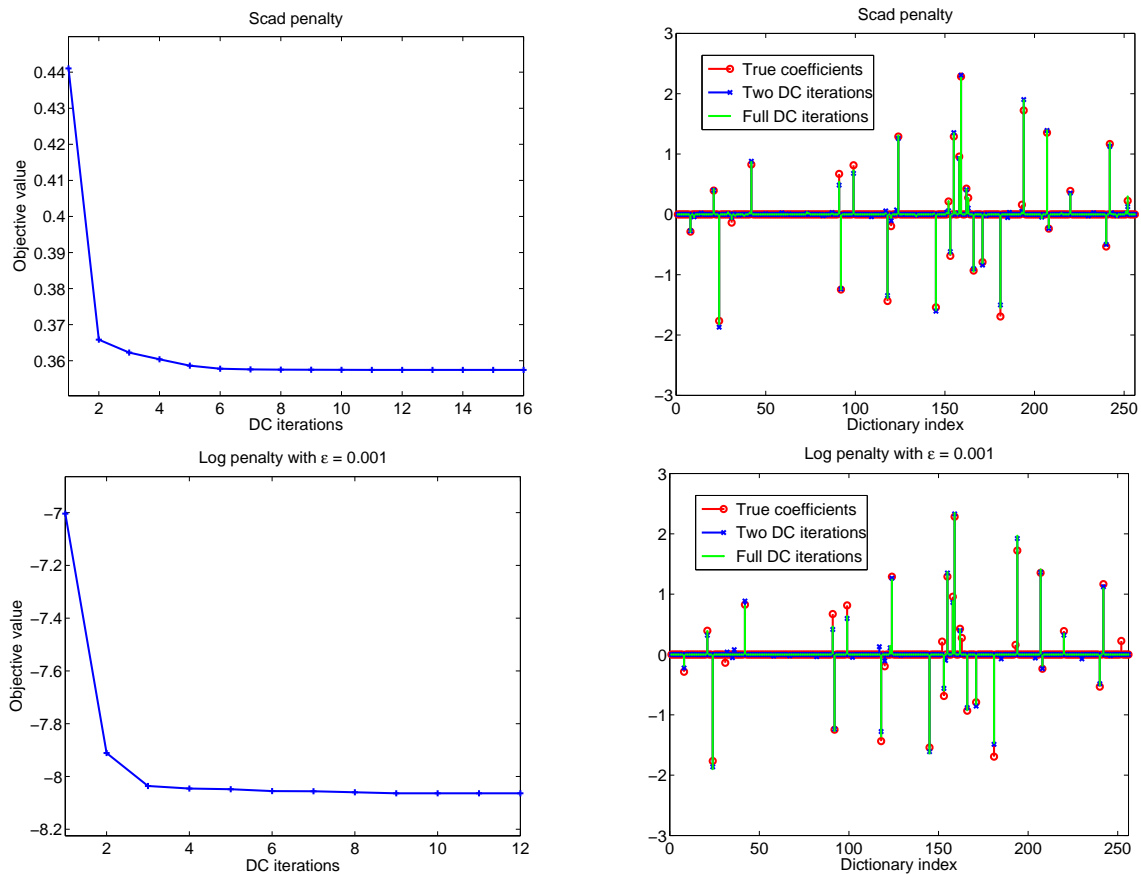


Fig. 3. Example of (left) objective value evolution along DC iterations and (right) estimated coefficient $\hat{\beta}$ after 2 iterations and full convergence of the DC procedure. These examples correspond to a signal of dimension $n = 128$ and a dictionary size of $d = 256$. The number of non-zeros coefficients k is equal to 30 and the signal-to-noise ratio is 10. We illustrate the effectiveness of our algorithm for the SCAD (top) and log with $\epsilon = 0.001$ (bottom) penalty functions. This example clearly illustrates our point that i) more DC iterations lead to better convergence, ii) more DC iterations improve the estimation of the β 's sparsity support.

convergence, then decrements ϵ by a factor of 10 and re-runs the DC Algorithm 2 with appropriate initialization. The annealing is stopped when $\epsilon = 10^{-10}$ is reached.

Figure 4 depicts the averaged (over 30 trials) F-measure obtained with the two approaches in two different Signal-to-Noise Ratio contexts. The curves clearly show that using the annealing method for dealing with ϵ leads to far better performances in most of the situations. We can note that for a low SNR, the F-measure for the annealing approach is about 10% better than the one obtained when using a fixed ϵ . Naturally, using such an iterative approach increases the computational cost of the algorithm. However, since the gain in performance is so consequent, we will consider this approach for the remaining experiments involving the log and ℓ_q penalties.

A comparison of the DC algorithm performances when using different penalty functions is given in Figure 5. For the ℓ_q penalty, we have set $q = 0.5$. In these plots, performances of the IRLS method which also considers an annealing treatment of ϵ , have been included. The IRLS algorithm actually solves problem (3) with a ℓ_q penalty. For comparison purpose, we have thus chosen $q = 0.5$ and $q = 0.05$.

In terms of sparsity recovery performances, we first note

that the Lasso performs considerably worse than all other approaches except when the SNR is high and the number of active elements is small (see top-right panel). For the Lasso, we also see that for SNR = 10, its recovery performance is poor and surprisingly, it increases with the number of active elements. This is due to the fact that the Lasso tends to keep many coefficients β to non-zero value and this phenomenon is enhanced as the SNR is low.

When comparing performances of non-convex penalties, we can note that the SCAD and Zhang's penalty functions are not as efficient as the log or the ℓ_q penalty. We also remark that for SNR = 10 (top-left panel), the log penalty with our DC approach performs slightly better than all other non-convex ones. We note that the IRLS approach with $q = 0.5$ is less efficient than its DC Lasso counterpart. The same remark holds for IRLS with $q = 0.05$ compared to the log penalty when SNR is low or for medium values of k when the SNR is high. In the latter case (SNR = 30), we see that as the number of active elements is lower than 30, the log penalty with a DC approach yields the best performance but when $|\text{supp}(\beta)| > 30$, the ℓ_q approach works better. We justify this point by the fact that the log penalty tends to produce sparser estimation than the ℓ_q penalty (with $q = 0.5$) by strongly penalizing more coefficients toward zero. In summary, we can

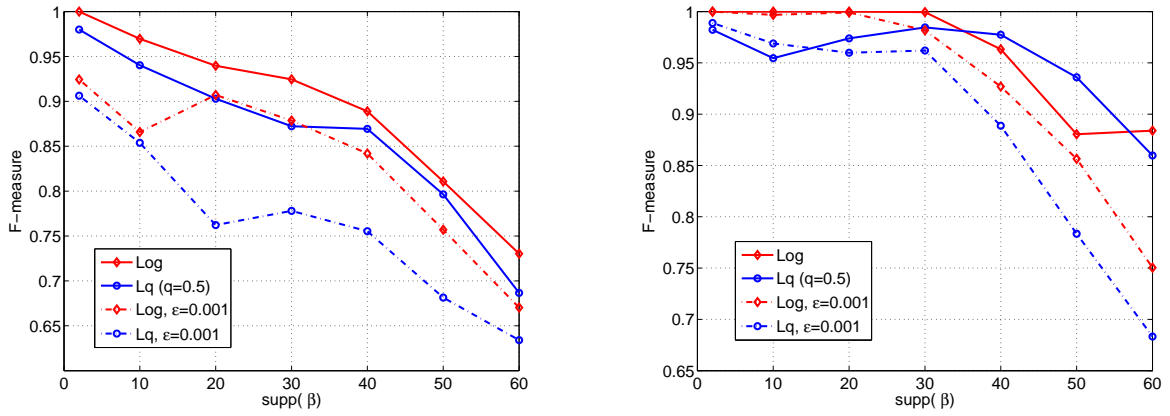


Fig. 4. Comparing performances of log and ℓ_q penalty functions when using a fixed value of ϵ or when iteratively decreasing this epsilon. Left: SNR = 10. Right: SNR = 30. The solid curves are the performances of our reweighted Lasso with an annealing design for the ϵ while the dotted curves are related to the case $\epsilon = 0.001$. The curves clearly show that one should use the annealing design.

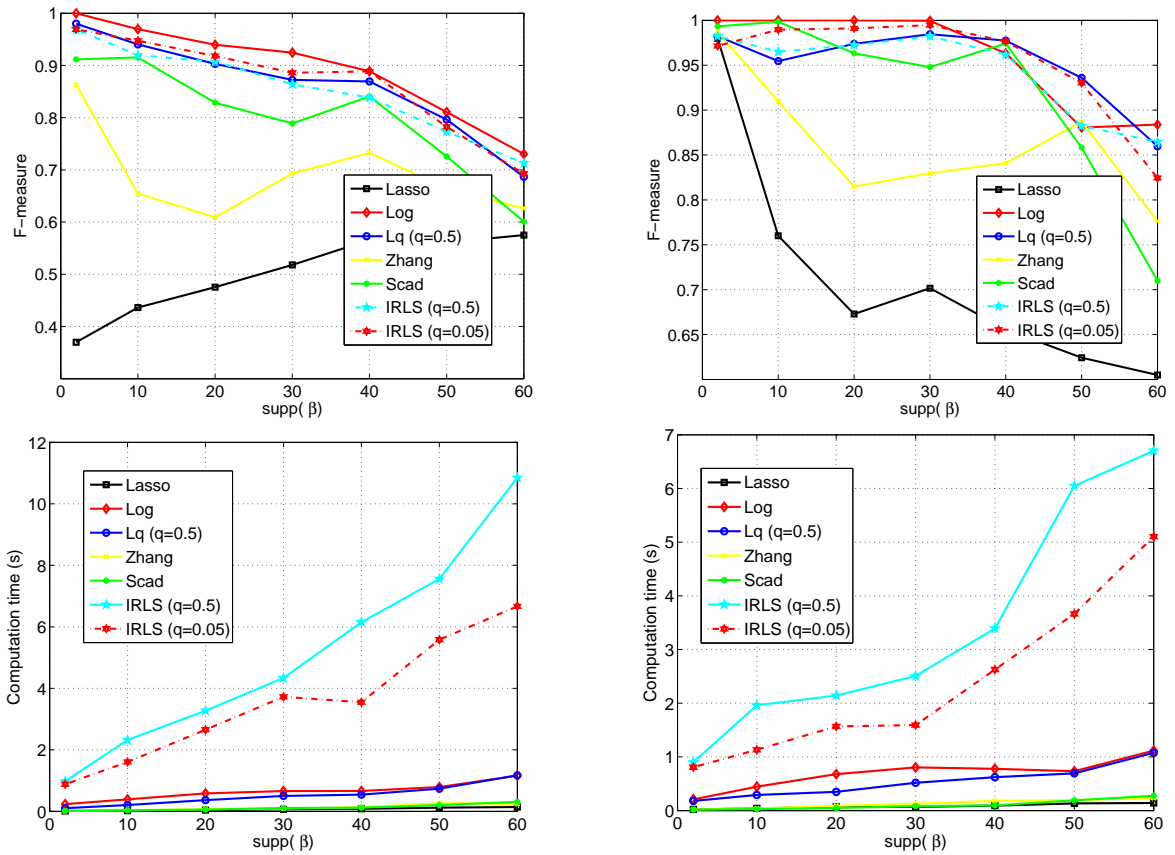


Fig. 5. Comparing F-measure performance and computational complexity of the different penalty functions with respect to the number of active elements. In this experiment, the parameter λ has been selected according to a cross-validation method. The computational time needed for estimating the model has been evaluated for this optimal λ . Top: F-measure of the compared algorithms. Bottom: computational time of the compared algorithms. Left column: SNR = 10. Right column: SNR = 30.

state that the log and the ℓ_q penalty functions with our DC approach tend to give better sparsity recovery performance than other methods. Before dealing with computation issues, let emphasize the fact that the same cross-validation procedure was used for all considered penalties. It may happen that considering customized validation criterion for each penalty could result in improved performances. Theoretical and experimental analyses of such a phenomenon are deferred to future work.

The bottom panels of Figure 5 show the averaged computational time needed by each method for computing the estimated model. Such a computational time has been evaluated for the parameter λ obtained from cross-validation. We first note that for all methods, the computational demand increases with the number of active elements. Then, we see that the IRLS approach is the most computationally expensive algorithm. DC approach with log and ℓ_q penalties is 3 to

4 times cheaper than IRLS although they both consider an annealing approach for dealing with ϵ . This gain in time compared to IRLS can be explained by the fact that reweighted ℓ_1 approaches can benefit from warm-start tricks for each DC iteration.

As expected, the Lasso is the less-demanding algorithm while our DC algorithms with SCAD and Zhang's penalties are slightly more expensive, since they involve more reweighted Lasso iterations. Note that the DC algorithms with these two latter penalties are cheaper to compute than the log and ℓ_q ones, since they are not concerned by any annealing approach for dealing with ϵ .

C. Evaluating computational complexity

This subsection reports an analysis of the computational cost as a function of dictionary size d and completes the previous computational cost analysis where d was fixed.

For this experiment, we report the averaged computational time needed for one algorithm to compute the model for the value of λ which yields the best F-measure. Furthermore, for a sake of clarity, we have reported the computational time for the Lasso, the DC algorithm with SCAD and log penalties and finally for IRLS ($q = 0.05$). For these two latter, we have considered the annealing design of ϵ .

Illustrations of the observed computation times are given in Figure 6 for two different scenarii: 1) samples size n is kept fixed to 128 (left panel) and 2) samples size evolves with d as $n = \frac{d}{4}$. We can see that computational complexities are not largely influenced by these setups.

Computational performances of the different algorithms can be ordered as follows. The Lasso is the less demanding algorithm followed by the DC SCAD. For large dictionary size ($d = 512$), the DC log is about 10 times more expensive than the DC SCAD but 10 times cheaper than the IRLS. However, we can note that for small dictionary, IRLS is cheaper than DC log whereas both algorithms are tied for medium dictionary size.

A point that it is interesting to highlight is that the Lasso, DC SCAD and DC log performance curves can be approximated by a linear model of similar slopes; this suggests that they have similar complexity up to a constant factor. This is natural since all these 3 methods are wrapped around a reweighted Lasso.

D. On the number of DC iterations

Here, we discuss the gain provided by a full convergent DC algorithm over the two-step approach. Figure 7 provides a graphical illustration of the observed performances.

Firstly, let focus on the case where the Signal-to-Noise ratio is low (left panel). We remark that the full convergence of DC SCAD definitely yields better results than its two step counterpart. Consequently, we recommend the user to prefer a full convergence algorithm when considering a SCAD penalty and a noisy signal. However this recommendation is highly mitigated for the log and ℓ_q penalties, since in this situation, one can remark that the full DC approach and the two-step methods are more or less equivalent in terms of F-measure

performances. The full DC approaches being computationally expensive than the two-step methods, the gain in F-measure is highly balanced by the calculation cost.

When considering a less noisy signal (SNR=30), the observations are in favor of the full DC algorithm when the support of the coefficient vector increases. The DC SCAD confirms its good results over two-iteration procedures whereas the DC with log and ℓ_q penalties exhibit more interesting performances for $|\text{supp}(\beta)| > 40$ showing the superiority and the improvement brought by our algorithm.

E. Influence of dictionary dependence

Here we experimentally study the sensitivity of the proposed algorithm in function of the correlation between columns of matrix \mathbf{X} . The objective is to examine the change in performances when \mathbf{X} is designed according to some weak, mild and strong dependence structures. For these purposes, we have built the target vector according to

$$\mathbf{y} = \mathbf{X}\mathbf{W}^{1/2}\beta + \xi$$

where \mathbf{W} is a Toeplitz matrix built from a vector \mathbf{v}

$$v_k = \rho^{k-1} + a_k \quad k = 1, \dots, d$$

with the vector \mathbf{a} obtained from a vector of uniform random noise of variance σ_v^2 sorted in decreasing order. Examples of covariance matrices obtained from the same \mathbf{X} but using different $\mathbf{W}^{1/2}$ are given in Figure 8.

The obtained results are reported on Figure 9. One can note that contrary to the Lasso, the reweighted Lasso performances are just slightly affected by the dependence of the dictionary atoms. Considering as a reference the performances for $\rho = 0$ (weak dependence of the atoms), we note that for mild dependence ($\rho = 0.5$, left panel), the decrease in performances is just marginal for all algorithms. However, as dependence increases ($\rho = 0.9$, right panel), we clearly see that the Lasso algorithm is subject to a performance drop. We also note that only the DC algorithm with a log penalty is robust to dictionary dependencies especially for small number of active dictionary elements. Such an empirical result is now well understood and it has notably motivated the development of the adaptive Lasso [50].

F. Influence of initialization

We have suggested that a good initialization of the DC algorithm could be so that the first iteration is the Lasso iteration. We have seen in the above experiments that such an initialization indeed leads to interesting performances. This final experiment aims at showing that this initialization works well.

The experiments carried out here are similar to those we performed above but for each trial, we have run the algorithm one hundred times with the following initialization: one initialization with $\alpha_j = 0$ (which is the standard one we used above), one initialization with $\hat{\beta} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$, one initialization with a ridge regression solution $\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda_0\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$ and

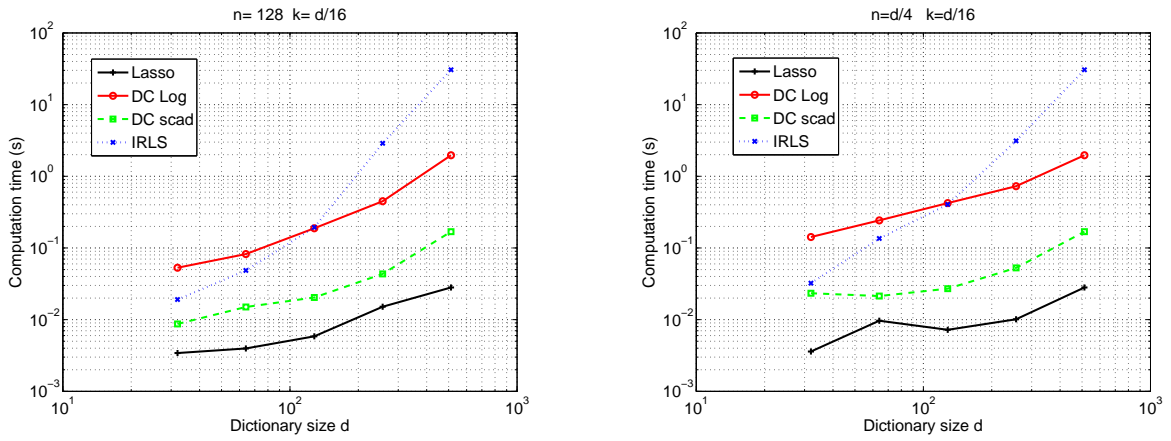


Fig. 6. Computational performance of four different algorithms with respect to dictionary size d . Left: the number of observations n is fixed. Right: n depends on the dictionary size. For these experiments, the computational performances of IRLS and DC approach with log penalty take into account the annealing tuning of ϵ .

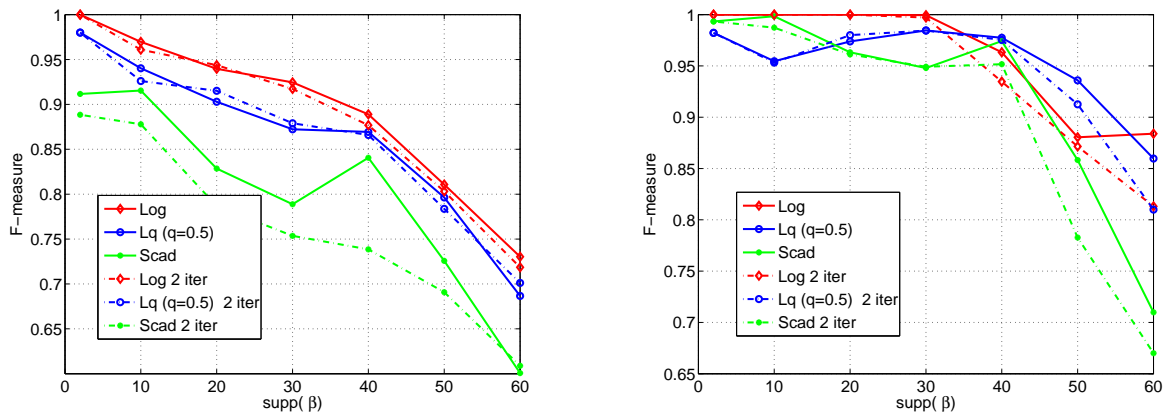


Fig. 7. Comparing performances of different penalty functions after 2 DC iterations (dotted curve) and after full DC convergence (solid curve). Left: SNR=10 and Right: SNR=30.

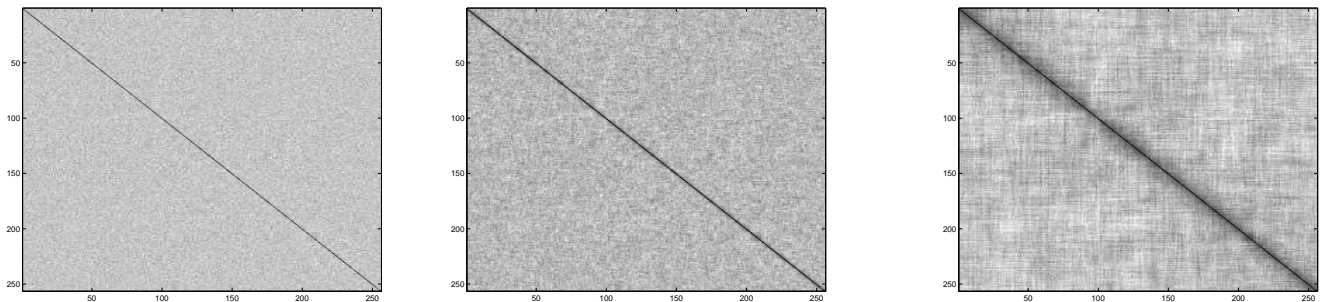


Fig. 8. Examples of different empirical covariance matrices obtained from the same \mathbf{X} but using different $\mathbf{W}^{1/2}$. Left: \mathbf{X} , middle: $\mathbf{X}\mathbf{W}_1^{1/2}$, right: $\mathbf{X}\mathbf{W}_2^{1/2}$. \mathbf{W}_1 and \mathbf{W}_2 have been respectively obtained by choosing $\{\rho = 0.5, \sigma_v = 0\}$ and $\{\rho = 0.9, \sigma_v = 0.1\}$

for the remaining initialization, we use a random zero mean and unit variance Gaussian vector. Here, we have considered $k = 10$ and $k = 50$ active elements in the dictionary.

Figure 10 depicts for each initialization the pairs (F-measure, objective value) for different penalty functions. Ideally, for the minimal objective value, we should get the highest F-measure. The results we obtained are somewhat striking. We can note that only the DC SCAD are sensitive to the initialization. Indeed, for $k = 10$ and for $k = 50$, the DC log and the DC ℓ_q achieve the same objective value regardless

of the initialization value we choose.

This suggests that by using an annealing design of the ϵ , the attraction basin that leads to the minimum objective value is sufficiently large to include all initialization values. Such an empirical result is rather interesting and it has already been noted by Chartrand *et al.* [12]. A more detailed and theoretical analysis of this point should be carried out in order to understand and clarify it.

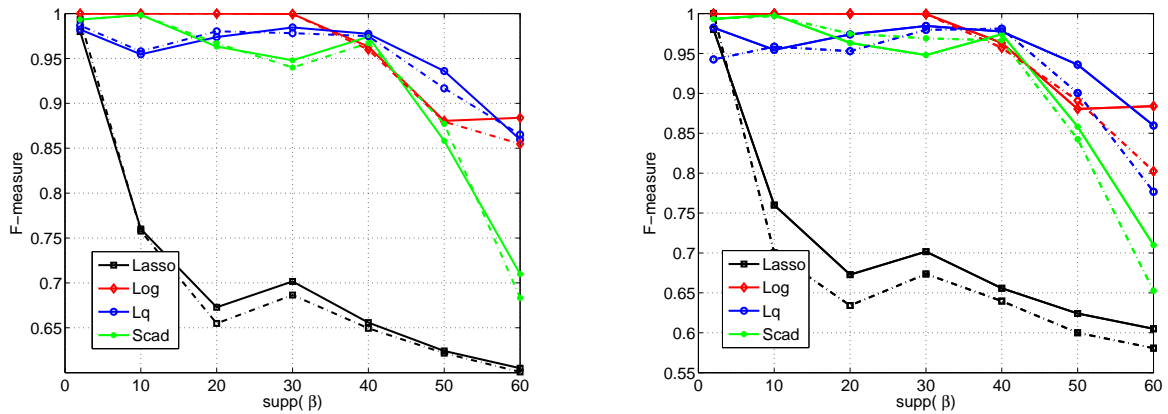


Fig. 9. Comparing performances of our DC algorithm with some penalty functions for different dependence conditions on the dictionary. For the penalty functions used, the solid curves show resulting F-measures when $\rho = 0$ and $\sigma_v = 0$ whereas the dotted curves are related to different values of ρ and σ_v . The results are reported for SNR=30. Left column: $\rho = 0.5, \sigma_v = 0$. Right column : $\rho = 0.9$ and $\sigma_v = 0.1$.

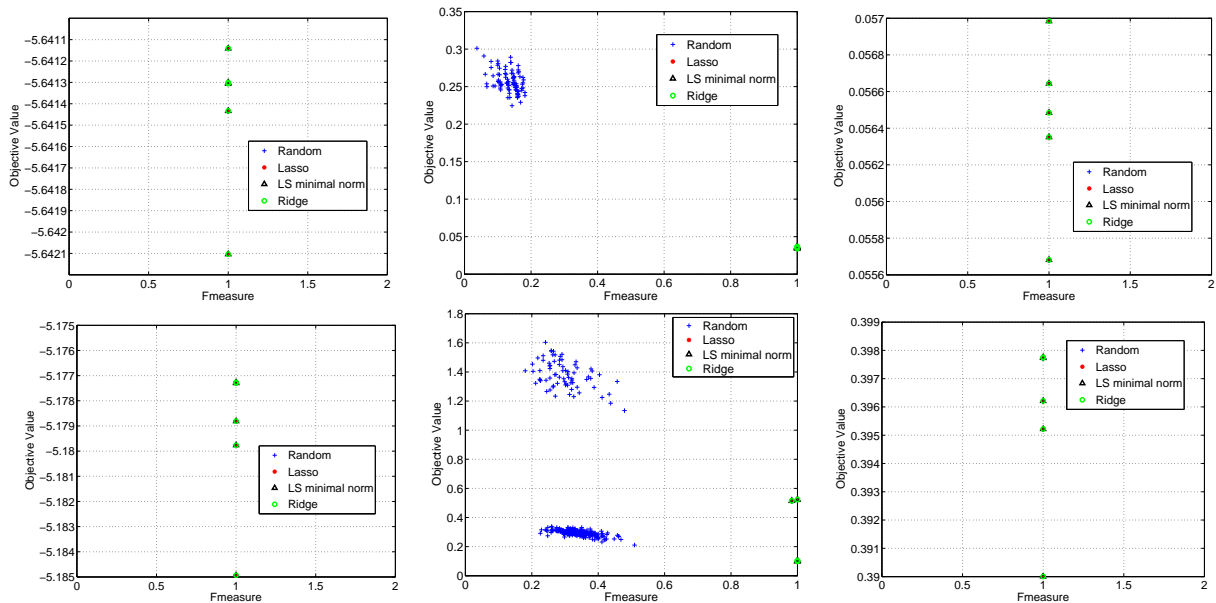


Fig. 10. Examples of couples (objective values, performance measure) for different initializations and for the following non-convex penalty functions. (From left to right) log, SCAD, ℓ_q penalty. (Top row) Number of active elements $k = 10$. (Bottom row) $k = 30$. The results involve 5 different trials and thus 5 optimal pairs (objective value, performance).

V. CONCLUSIONS

Several works in the literature have already shown that non-convex penalties instead of a ℓ_1 penalty lead to better sparsity recovery in signal or variable selection in machine learning problems. However, because of such a non-convexity, the resulting optimization problem becomes more challenging. In this paper, we have proposed a generic algorithm for solving such a problem, based on Difference of Convex functions programming. We thus proposed an iterative algorithm where at each iteration, we solve a Lasso-like problem. Hence, by fitting the resulting reweighted Lasso into the DC programming framework, we can benefit from all the theoretical and algorithmic results from these two domains i.e. fast algorithms for solving Lasso and theoretical properties of DC programming such as convergence. Besides being generic, we have shown that several algorithms of the literature correspond to

few iterations of the one we propose. Experimental results we depicted clearly show the benefit of using our DC algorithm, either in terms of sparsity recovery or in terms of computational complexity, when compared to other approaches such as IRLS.

Future researches aim at investigating the theoretical guarantees on sparsity recovery provided by our algorithm as well as extension to related problems such as the group-lasso.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their useful comments. This work was supported in part by French grants from the Agence Nationale de la Recherche (ANR KernSig Project).

REFERENCES

- [1] L. T. H. An and P. D. Tao, DC programming approach and solution algorithm to multidimensional scaling problem, ser. Nonconvex optimization and its application. Kluwer Academic Publishers, 2001, pp. 231–276.
- [2] A. Antoniadis and J. Fan, “Regularization of wavelet approximation,” Journal of the American Statistical Association, vol. 96, no. 455, pp. 939–967, 2001.
- [3] S. Balakrishnan and D. Madigan, “Algorithms for sparse linear classifiers in the massive data setting,” Journal of Machine Learning Research, vol. 9, pp. 313–337, 2008.
- [4] L. Bottou and O. Bousquet, “The tradeoffs of large scale learning,” in Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2008, vol. 20, to appear. [Online]. Available: <http://leon.bottou.org/papers/bottou-bousquet-2008>
- [5] P. Bühlmann and L. Meier, “Discussion on one-step sparse estimates in nonconcave penalized likelihood models,” The Annals of Statistics, vol. 36, no. 4, pp. 1534–1541, 2008.
- [6] F. Bunea, “Honest variable selection in linear and logistic regression models via ℓ_1 and $\ell_1 + \ell_2$ penalization,” The Electronic Journal of Statistics, vol. 2, pp. 1153–1194, 2008.
- [7] E. Candès and J. Romberg, “Sparsity and incoherence in compressive sampling,” Inverse Problems, vol. 23, no. 3, pp. 969–985, 2006.
- [8] E. Candès and T. Tao, “The dantzig selector: statistical estimation when p is much larger than n ,” The Annals of Statistics, vol. 35, no. 6, pp. 2392–2404, 2007.
- [9] E. Candès, M. Wakin, and S. Boyd, “Enhancing sparsity by reweighted ℓ_1 minimization,” J. Fourier Analysis and Applications, 2008.
- [10] R. Chartrand, “Exact reconstruction of sparse signals via nonconvex minimization,” IEEE Signal Processing Letters, vol. 14, pp. 707–710, 2007.
- [11] R. Chartrand and V. Staneva, “Restricted isometry properties and non-convex compressive sensing,” Inverse Problems, vol. 24, pp. 1–14, 2008.
- [12] R. Chartrand and W. Yin, “Iteratively reweighted algorithms for compressive sensing,” in 33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2008.
- [13] S. Chen, D. Donoho, and M. Saunders, “Atomic decomposition by basis pursuit,” SIAM Journal on Scientific Computing, vol. 20, no. 1, pp. 33–61, 1998.
- [14] D. Donoho and M. Elad, “Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 -norm minimization,” Proceedings of the National Academy of Sciences USA, vol. 1005, pp. 2197–2202, 2002.
- [15] D. Donoho and X. Huo, “Uncertainty principles and ideal atomic decompositions,” IEEE Trans. on Information Theory, vol. 47, no. 2845–2863, 2002.
- [16] D. Donoho and Y. Tsaig, “Extensions of compressed sensing,” Signal Processing, vol. 86, no. 3, pp. 533–548, 2006.
- [17] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, “Least angle regression,” Annals of Statistics, vol. 32, no. 2, pp. 407–499, 2004. [Online]. Available: citeseer.ist.psu.edu/efron02least.html
- [18] J. Fan and R. Li, “Variable selection via nonconcave penalized likelihood and its oracle properties,” Journal of the American Statistical Association, vol. 96, no. 456, pp. 1348–1360, 2001.
- [19] M. Figueiredo, R. Nowak, and S. Wright, “Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems,” IEEE Journal of Selected Topics in Signal Processing: Special Issue on Convex Optimization Methods for Signal Processing, vol. 1, no. 4, pp. 586–598, 2007.
- [20] S. Foucart and M.-J. Lai, “Sparsest solutions of underdetermined linear systems via ℓ_q -minimization for $0 < q \leq 1$,” Applied and Computational Harmonic Analysis, vol. 26, no. 3, pp. 395–407, 2009.
- [21] I. Frank and J. Friedman, “A statistical view of some chemometrics regression tools (with discussion),” Technometrics, vol. 35, no. 109–148, 1993.
- [22] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, “Pathwise coordinate optimization,” The Annals of Applied Statistics, vol. 1, no. 2, pp. 302–332, 2007.
- [23] W. J. Fu, “Penalized regression: the bridge versus the lasso,” Journal of Computational and Graphical Statistics, vol. 7, pp. 397–416, 1998.
- [24] R. Gribonval and M. Nielsen, “Highly sparse representations from dictionaries are unique and independent of the sparseness measure,” Applied and Computational Harmonic Analysis, vol. 22, no. 3, pp. 335–355, 2007.
- [25] R. Horst and N. V. Thoai, “Dc programming: overview,” Journal of Optimization Theory and Applications, vol. 103, pp. 1–41, 1999.
- [26] J. Huang, J. Horowitz, and S. Ma, “Asymptotic properties of bridge estimators in sparse high-dimensional regression models,” Annals of Statistics, vol. 36, no. 2, pp. 587–613, 2008.
- [27] S.-J. Kim, K. Koh, M. Lustig, and A. D. G. S. Boyd, “A method for large-scale ℓ_1 -regularized least squares,” IEEE Journal on Selected Topics in Signal Processing, vol. 4, no. 1, pp. 606–617, 2007.
- [28] K. Knight and W. Fu, “Asymptotics for lasso-type estimators,” Annals of Statistics, vol. 28, pp. 1356–1378, 2000.
- [29] R. M. Leahy and B. D. Jeffs, “On the design of maximally sparse beamforming arrays,” IEEE Transactions on Antennas and Propagation, vol. 39, no. 8, pp. 1178–1187, 1991.
- [30] Y. Liu, X. Shen, and H. Doss, “Multicategory ψ -learning and support vector machines: computational tools,” Journal of Computational and Graphical Statistics, vol. 14, no. 1, pp. 219–236, 2005.
- [31] N. Meinshausen and P. Bühlmann, “High dimensional graphs and variable selection with the lasso,” Annals of Statistics, vol. 34, no. 3, pp. 1436–1462, 2006.
- [32] M. Nikolova, “Local strong homogeneity of a regularized estimator,” SIAM Journal of Applied Mathematics, vol. 61, no. 2, pp. 633–658, 2000.
- [33] M. Osborne, B. Presnell, and B. Turlach, “A new approach to variable selection in least squares problems,” IMA Journal of Numerical Analysis, vol. 20, no. 3, pp. 389–403, 2000.
- [34] —, “On the lasso and its dual,” Journal of Computational and Graphical Statistics, vol. 9, no. 2, pp. 319–337, 2000.
- [35] R. Rockafellar, Convex Analysis. Princeton University Press, 1996.
- [36] R. Saab, R. Chartrand, and Özgür Yılmaz, “Stable sparse approximations via nonconvex optimization,” in 33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2008.
- [37] S. Shevade and S. Keerthi, “A simple and efficient algorithm for gene selection using sparse logistic regression,” Bioinformatics, vol. 19, no. 17, pp. 2246–2253, 2003.
- [38] P. D. Tao and L. T. H. An, “Dc optimization algorithms for solving the trust region subproblem,” SIAM Journal of Optimization, vol. 8, no. 2, pp. 476–505, 1998.
- [39] H. L. Thi and P. Tao, “A continuous approach for the concave cost supply problem via dc programming and dca,” Discrete Applied Mathematics, vol. 156, pp. 325–338, 2008.
- [40] R. Tibshirani, “Regression shrinkage and selection via the lasso,” Journal of the Royal Statistical Society, vol. 46, pp. 431–439, 1996.
- [41] M. E. Tipping, “Sparse bayesian learning and the relevance vector machine,” Journal of Machine Learning Research, vol. 1, pp. 211–244, 2001.
- [42] J. Tropp, “Just relax: Convex programming methods for identifying sparse signals,” IEEE Trans. Info. Theory, vol. 51, no. 3, pp. 1030–1051, 2006.
- [43] B. A. Turlach, W. N. Venables, and S. J. Wright, “Simultaneous variable selection,” Technometrics, vol. 27, pp. 349–363, 2005.
- [44] M. J. Wainwright, “Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1 -constrained quadratic programming,” Department of Statistics, UC Berkeley, Tech. Rep., 2006.
- [45] J. Weston, A. Elisseeff, B. Scholkopf, and M. Tipping, “The use of zero-norm with linear models and kernel methods,” Journal of Machine Learning Research, vol. 3, pp. 1439–1461, 2003.
- [46] A. L. Yuille and A. Rangarajan, “The concave-convex procedure,” in Proc. of Advances in Neural Information Processing Systems, 2001.
- [47] T. Zhang, “Some sharp performance bounds for least squares regression with ℓ_1 regularization,” Dept. of Statistics, Rutgers University, Tech. Rep., 2007, to appear in Annals of Statistics.
- [48] Z. Zhang, J. T. Kwok, and D.-Y. Yeung, “Surrogate maximization/minimization algorithms,” Machine Learning, pp. 1–33, 2007.
- [49] P. Zhao and B. Yu, “On model selection consistency of lasso,” Journal of Machine Learning Research, vol. 7, pp. 2541–2563, 2006.
- [50] H. Zou, “The adaptive lasso and its oracle properties,” Journal of the American Statistical Association, vol. 101, no. 476, pp. 1418–1429, 2006.
- [51] H. Zou and R. Li, “One-step sparse estimates in nonconcave penalized likelihood models,” The Annals of Statistics, vol. 36, no. 4, pp. 1509–1533, 2008.

Gilles GASSO

Gilles GASSO is Assistant Professor in Computer Science Department National Engineering School of Applied Sciences (INSA) in Rouen (France). His research topics concern machine learning, kernel methods, signal processing.