Recovering sparse signals with non-convex penalties and DC programming

Gilles Gasso, Alain Rakotomamonjy, Stéphane Canu

LITIS, EA 4108 - INSA / Universite 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 is usually formalized as a penalized least-squares problem in which sparsity is usually induced by a ℓ_1-norm penalty on the coefficient. Such an approach known as the *Lasso* or *Basis Pursuit Denoising* has been shown to perform reasonably well in some situations. However, it has also been proved that non-convex penalties like 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 with non-convex penalty. The main contribution is that our methodology is based on an iterative algorithm which solves at each iteration a *convex* weighted Lasso problem. It relies on the decomposition of the non-convex penalty into 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 which solve such a problem are particular cases of our algorithm. Experimental results then demonstrate that our method performs better than previously proposed algorithms.

I. Introduction

"Entia non sunt multiplicanda praeter necessitatem" is a statement attributed to the 14^{th} 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 sparsity plays a key role. Indeed, recovering sparse representation is of great interest in the signal processing and the machine learning community due to the profileration of communications 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 [6], [14].

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

Sparse approximation problems or variable selection problems are usually posed as the following mathematical programming problem :

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_0 \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^n$, $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\mathbf{X}$ a $n \times d$ matrix. Within a sparse signal approximation context, $\mathbf{X}$ would be a matrix which columns are the elements of signal dictionary and $\mathbf{y}$ the target signal, while for a machine learning problem, $\mathbf{X}$ would be the observations-variables matrix and $\mathbf{y}$ the target output.

Optimization problem (1) involves a term which measures the goodness of the approximation (the least-squares term), a term $\|\boldsymbol{\beta}\|_0$ which measures the solution sparsity by counting the number of non-zero components in $\boldsymbol{\beta}$ and a trade-off parameter $\lambda \in \mathbb{R}^+$ that balances these two terms. 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 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 [34], [28], [7], [35] have proposed to relax the problem and instead to consider the following convex optimization problem :

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_1 \tag{2}$$

This latter problem, known as the *Lasso*, has been introduced in the nineties by Tibshirani [34] and it has also been independently proposed by Chen et al. [11] as the *Basis Pursuit Denoising* problem. 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 [34], [36], homotopy

approaches [29], coordinate wise optimization [19], or gradient projection method [17]. Furthermore, some theoretical arguments support the Lasso and state that under certain conditions variable selection with the Lasso can be consistent [41], [26]. Other works such as those proposed by Donoho et al. [12], [13] and Tropp [35] have proved that in certain situations the ℓ_1 penalization is able to recover the sparsity profile of the true coefficient $\boldsymbol{\beta}^\star$.

However, Fan and Li [16] 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], [27]) and should be so that their derivatives vanish for large values. These two conditions lead then to the following (more general) optimization problem

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \frac{1}{2}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right\|^2 + \sum_{j=1}^{d} \mathrm{g}_\lambda\left(|\beta_j|\right) \tag{3}$$

which involves a non-smooth and non-convex penalty function $\mathrm{g}_\lambda\left(\cdot\right)$ instead of the ℓ_1 regularization term. For some reasons 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 $\beta_j = \beta_j^+ - \beta_j^-$

$$\begin{aligned} \min_{\boldsymbol{\beta}^+,\boldsymbol{\beta}^-\in\mathbb{R}^d} & \frac{1}{2}\left\|\mathbf{y}-\mathbf{X}(\boldsymbol{\beta}^+-\boldsymbol{\beta}^-)\right\|^2 + \sum_{j=1}^{d} \mathrm{g}_\lambda\left(\beta_j^+ + \beta_j^-\right) \\ \text{s.t.} \quad & \beta_j^+ \geq 0, \quad \beta_j^- \geq 0, \quad \forall\, j = 1,\cdots,d \end{aligned} \tag{4}$$

where the vectors $\boldsymbol{\beta}^+$ and $\boldsymbol{\beta}^-$ 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).

Historically, one of the first non-convex penalty function used for sparse approximation was the Smoothly Clipped Absolute Deviation (SCAD) penalty (see Table I). Indeed, after having highlighted the drawbacks of the ℓ_1 penalty, Fan and Li [16] proposed such a penalty to circumvents Lasso weak points. They then proved that the resulting estimate has interesting theoretical properties such as unbiasness.

Among all other penalty functions which lead to sparsity, a popular one is the so-called Bridge penalty function also known as the ℓ_q pseudo-norm when $0 < q < 1$ (see table I). This type of penalty has been

firstly used by Frank and Friedman [18] and Fu [20]. It has been shown to provide sparser solutions than the Lasso. For instance, Knight and Fu [24] 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 [9], [31]. Despite the difficulties raised by the optimization problem, some theoretical guarantees can be proved when using ℓ_q penalty [22], [24].

Another popular non-convex penalty is the logarithm penalty (named in the sequel log penalty). This penalty was used as an approximation of the $\|\boldsymbol{\beta}\|_0$ pseudo-norm by Weston et al. in a context of variable selection [37]. For sparse signal approximations, Candès et al. [8] have also investigated the use of this log penalty and empirically proves 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 value $0 < \epsilon \ll 1$ to avoid infinite value when the parameter vanishes. The constant term $-\lambda \log(\varepsilon)$ ensures that the penalty function is non-negative and avoids the trivial solution $\boldsymbol{\beta} = 0$ to problem (3).

Finally, we can consider another non-convex penalty, we named as Zhang's 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 corresponds to a Lasso for which large parameters are not penalized anymore [39]. This penalty can also be seen as a linear approximation of the SCAD penalty.

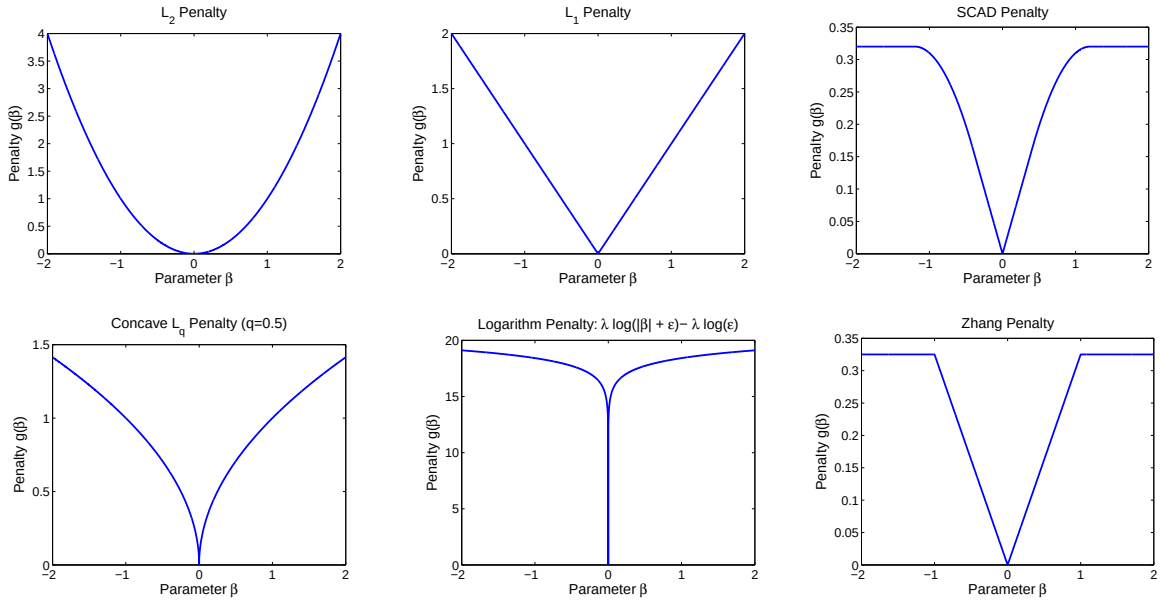

Fig. 1. Illustration of some common penalty functions described in Table I.

TABLE I

SOME EXAMPLES OF USUAL PENALTY FUNCTIONS

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

C. Previous algorithms for solving least-squares with non-convex penalties

Using non-convex penalties instead of convex ones leads to a sparser estimation $\hat{\beta}$ at the expense of introducing a more challenging optimization problem. In this subsection, we briefly review some of the algorithms available in the literature for solving (3) when $\mathrm{g}_\lambda(\cdot)$ is non-convex.

When dealing with the SCAD penalty, Fan and Li [16] 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 [43] 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. Note that since these two algorithms are initialized with the least-squares solution, they are essentially restricted to situations where $d < n$. Following a discussion on the one-step SCAD estimate of Zou and Li [43], 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 $\mathrm{g}_\lambda(\beta_j)$ is not differentiable as soon as its argument vanishes. To deal with this issue, Huang et al. [22] 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. [10], [31] use an iteratively reweighted least-squares algorithm. As a side contribution, they also investigate 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. [8] 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.

D. Our contribution

Most of the above-described algorithms are actually based on the same idea: using an iterative reweighted ℓ_1 or ℓ_2 algorithms for solving the non-convex problem. 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 disputable since convergence to a local or global minimum of the optimization problem is not guaranteed. Bühlmann et Meier [5] have proposed a full iterative scheme but they do not fit their algorithm into any optimization problem so it is not clear which kind of non-convex penalty they are using in problem (3). Furthermore, as stated by Candès et al. [8] in his 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 non-convex penalties. The approach relies on Difference of Convex (DC) functions programming [21]. 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 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. [8],
- give empirical evidence 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 [43],
- 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 our 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 author's 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 [21]. 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 that will be useful for developing our algorithm.

A. Difference of Convex functions Algorithm

Let us consider the general minimization problem

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

where $J(\cdot)$ is a non-convex (possibly non-smooth) criterion. The main idea of DC programming [21] is to decompose $J(\cdot)$ as:

$$J(\boldsymbol{\beta}) = J_1(\boldsymbol{\beta}) - J_2(\boldsymbol{\beta}) \tag{6}$$

Algorithm 1 DC Algorithm

Set $t = 0$ and an initial estimation $\boldsymbol{\beta}^0 \in dom\, J_1$

with $dom\, J_1 = \{\boldsymbol{\beta} \in \mathbb{R}^d : J_1(\boldsymbol{\beta}) < \infty\}$

repeat

(D_t) Determine $\boldsymbol{\alpha}^t \in \partial J_2(\boldsymbol{\beta}^t)$

(P_t) Determine $\boldsymbol{\beta}^{t+1} \in \partial J_1^*(\boldsymbol{\alpha}^t)$

$t = t + 1$

until convergence

where $J_1(\cdot)$ and $J_2(\cdot)$ are lower semi-continuous, proper convex functions on $\mathbb{R}^d$. Then, according to duality properties, it can be shown that the dual of the minimization problem is given by

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

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

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

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 [21] and it can be summarized as follows.

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

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

For differentiable criterion, the subdifferential is a singleton so that $\partial J_2(\boldsymbol{\beta}) = \{\nabla J_2(\boldsymbol{\beta})\}$. Using standard results on convex optimization [30], the following equations also hold for a given $\boldsymbol{\beta}'$ and $\boldsymbol{\alpha}'$

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

From these definitions, we can see that by replacing $J_1^*(\boldsymbol{\alpha})$ in (7) by its affine minorization $J_1^*(\boldsymbol{\alpha}^t) + \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^t, \boldsymbol{\beta}^t \rangle$ at the point $\boldsymbol{\alpha}^t$, the dual problem (7) becomes equivalent to find $\partial J_2(\boldsymbol{\beta}^t)$. Hence, choosing an $\boldsymbol{\alpha}^t \in \partial J_2(\boldsymbol{\beta}^t)$ is equivalent to solve an approximation of problem (7). Similarly, by replacing the

convex function $J_2(\boldsymbol{\beta})$ in (6) by its affine minorization $J_2(\boldsymbol{\beta}^t) + \langle \boldsymbol{\beta} - \boldsymbol{\beta}^t, \boldsymbol{\alpha}^t \rangle$ at the neighborhood of $\boldsymbol{\beta}^t$ 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 [33]. 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 $dom\, J_1 \subset dom\, J_2$ and $dom\, J_2^* \subset dom\, J_1^*$ (The finiteness of the minimum of the function J implies these conditions). It holds that

(i) The sequence $\{\boldsymbol{\beta}^t\}_{t \in \mathbb{N}}$ (respectively $\{\boldsymbol{\alpha}^t\}_{t \in \mathbb{N}}$) is well defined or equivalently $dom\, \partial J_1 \subset dom\, \partial J_2$ (respectively $dom\, \partial J_2^* \subset dom\, \partial J_1^*$).

(ii) The primal objective value sequence $\{J_1(\boldsymbol{\beta}^t) - J_2(\boldsymbol{\beta}^t)\}_{t \in \mathbb{N}}$ (respectively dual objective value sequence $\{J_2^*(\boldsymbol{\beta}^t) - J_1^*(\boldsymbol{\beta}^t)\}_{t \in \mathbb{N}}$) is monotonically decreasing (resp. increasing).

(iii) If the minimum of J is finite and the sequence $\{\boldsymbol{\beta}^t\}_{t \in \mathbb{N}}$ (respectively $\{\boldsymbol{\alpha}^t\}_{t \in \mathbb{N}}$) is bounded, every limit point $\hat{\boldsymbol{\beta}}^\star$ (respectively $\hat{\boldsymbol{\alpha}}^\star$) is a critical point of $J_1 - J_2$ (respectively $J_2^* - J_1^*$), that is this point satisfies the local optimality condition.

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 $J(\boldsymbol{\beta}) + \Gamma_{\mathcal{X}}(\boldsymbol{\beta})$ where $\Gamma_{\mathcal{X}}(\boldsymbol{\beta})$ is the indicator function defined as $\Gamma_{\mathcal{X}}(\boldsymbol{\beta}) = 0$ if $\boldsymbol{\beta} \in \mathcal{X}$ and $\Gamma_{\mathcal{X}}(\boldsymbol{\beta}) = +\infty$ otherwise.

(ii) Notice that Yuille and Rangarajan [38] 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. [40] nicely review 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 $\mathrm{g}_\lambda(\cdot)$ as the difference of two convex functions :

$$\mathrm{g}_\lambda(\cdot) = \mathrm{g}_{\text{vex}}(\cdot) - h(\cdot) \tag{8}$$

Then according to this decomposition,we define

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

As we will show in the sequel, for the penalty functions $\mathrm{g}_\lambda(\cdot)$ we use, the function h is explicitly known. Hence, at each iteration t, we are able to analytically determine an element of the subdifferential $\boldsymbol{\alpha}^t$ of $J_2(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-)$. Thus, if $h(\cdot)$ is differentiable, we have $\alpha_j^t = h'(\beta_j^{+t} + \beta_j^{-t})$, $j = 1, \cdots, d$ 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], [25].

Now, according to the (P_t) step of Algorithm 1 and to equation (8), once $\boldsymbol{\alpha}^t$ is known, the next value of $\boldsymbol{\beta}$ can be found by minimizing with respects to $\boldsymbol{\beta}^+$ and $\boldsymbol{\beta}^-$ the following objective function

$$\frac{1}{2}\left\|\mathbf{y} - \mathbf{X}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-)\right\|^2 + \sum_{j=1}^{d} \mathrm{g}_{\text{vex}}\left(\beta_j^+ + \beta_j^-\right) - \langle \beta_j^+ + \beta_j^-, \alpha_j^t \rangle \tag{9}$$

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 $\mathrm{g}_\lambda(\cdot)$ 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 problem. By doing so, we are then able to adapt or re-use efficient algorithms developed for the Lasso. The decomposition is as following

$$\mathrm{g}_{\text{vex}}(\cdot) = \lambda|\cdot| \tag{10}$$

$$h(\cdot) = \lambda|\cdot| - \mathrm{g}_\lambda(\cdot) \tag{11}$$

Algorithm 2 Iterative optimization based on DC programming

Set an initial estimation $\boldsymbol{\beta}^0$, $t = 0$

repeat

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

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

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

$t = t + 1$

until convergence of $\boldsymbol{\beta}$

where $h(\cdot)$ is a convex function. Due to the non-negativity of $\mathrm{g}_\lambda(\cdot)$, 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.

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, 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. 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

$$\begin{aligned}\min_{\boldsymbol{\beta}^+,\boldsymbol{\beta}^-}\ \frac{1}{2}\left\|\mathbf{y}-\mathbf{X}(\boldsymbol{\beta}^+-\boldsymbol{\beta}^-)\right\|^2+\sum_{j=1}^{d}\lambda(\beta_j^+ + \beta_j^-) \\ -\sum_{j=1}^{d}\alpha_j^t(\beta_j^+ + \beta_j^-) \\ \text{s.t.} \qquad \beta_j^+ \geq 0, \quad \beta_j^- \geq 0, \quad \forall j = 1, \cdots, d\end{aligned} \tag{12}$$

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.

TABLE II

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

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

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. [8] and Chartrand et al. [31]) 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 $\boldsymbol{\beta}$ in order to cope with the DC programming formulation, we see that problem (12) can be formulated as a weighted Lasso problem :

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \sum_{j=1}^{d} \lambda_j|\beta_j| \tag{13}$$

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 [23], [19], [17]. Among all these methods, the work of Figueiredo et al. [17] and the one of Friedman et al. [19] 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

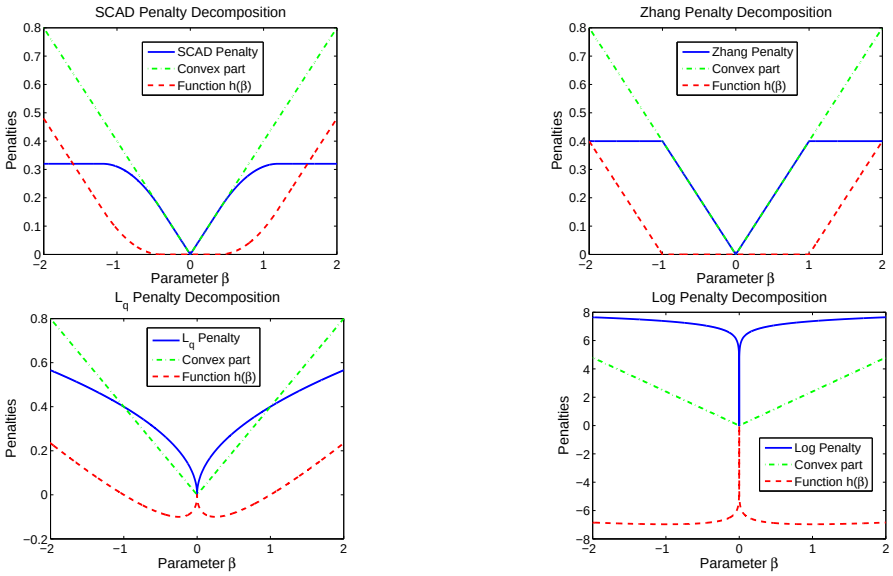

Fig. 2. Illustration of the proposed difference of convex functions decompositions. From the left to the right, Top row) SCAD penalty and Zhang penalty, Bottom row) ℓ_q and the log-penalty. For clarity sake, the plotted convex part of the log penalty is $6\lambda|\beta|$.

efficient and competitive compared to homotopy or quadratic programming approaches [15], [29], [34]. 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 [32]). Early comparisons of this algorithm with the gradient projection of Figueiredo et al. 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 on the first author website.

Note that although all algorithms which address a ℓ_1 penalized least-squares can be used for solving problem (13), one particularity of the global 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 $\boldsymbol{\beta}^t$ 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,

TABLE III

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

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

known as warm-start technique, makes the overall algorithm just slightly more expensive than a one-step Lasso resolution. 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

$$\begin{aligned} \mathsf{x}_j^\mathsf{T}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \lambda_j \text{sign}(\beta_j) = 0 \quad \text{if} \quad \beta_j \neq 0 \\ \lvert\mathsf{x}_j^\mathsf{T}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\rvert < \lambda_j \quad \text{if} \quad \beta_j = 0 \end{aligned} \tag{14}$$

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

$$\begin{aligned} (\mathsf{x}_j^\mathsf{T}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})) - \lambda_j \text{sign}(\beta_j) < \tau \quad \text{if} \quad \beta_j \neq 0 \\ \lvert\mathsf{x}_j^\mathsf{T}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\rvert < \lambda_j + \tau \quad \text{if} \quad \beta_j = 0 \end{aligned}$$

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 the infinite norm of the coefficient vector difference between two consecutive iterations is below 10^{-4} or the maximal number of 50 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 works dealing with the Lasso problem. For instance, Zhang has proposed a two-stage Lasso [39], Zou has introduced the adaptive Lasso [42], while Zou and Li improved the SCAD of Fan and Li by using Local Linear Approximation [43]. 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 $\hat{\boldsymbol{\beta}}$ and a user-defined threshold η, the second stage is a weighted Lasso where the weights are

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

It is easy to see that, if an initial estimation $\boldsymbol{\beta}^0 = 0$ 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 [16] is based on a local quadratic approximation. This leads to an iterative ridge regression which is computationally more expensive than the Lasso and is subject to some numerical instabilities. Recently, Zou and Li advocate the use of local linear approximation (LLA) around a least-squares initial estimation $\hat{\boldsymbol{\beta}}_{\text{MC}}$ and compute the final solution through a weighted Lasso. Thus, Zou and Li algorithm corresponds to one-step of our DC procedure with $\boldsymbol{\beta}^0 = \hat{\boldsymbol{\beta}}_{\text{MC}}$.

In the adaptive Lasso promoted by Zou [42], the authors suggest a theoretically-justified single-step weighted Lasso. The weight of each parameter β_j in the penalty is set as $\lambda_j \propto \frac{1}{|\beta'_j|^\gamma}$ where γ is an user-defined hyperparameter and $\boldsymbol{\beta}'$ 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 is 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

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

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

From the above 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 insures 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 considerably improve the sparsity estimation.

In the literature, two recent algorithms proposed by Chartrand and Yin [10] and Candès et al. [8] share the same spirit of solving a non-convex problem using an iterative reweighted scheme until complete convergence. These algorithms was 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 [10], the authors have proposed an iterative reweighted ℓ_2 for solving the non-convex problem with a ℓ_q penalty. To our point of view, the overall computational cost of such an approach can be high due the square scaling of the least-squares solution. 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 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 provide in equation (15) for the log penalty. Furthermore, they have added to each weight some hyper-parameters. Since, adaptive Lasso is related to the log penalty, if we omit the hyperparameters, 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 $\boldsymbol{\beta}^0$ 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 $\hat{\boldsymbol{\beta}}$. 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 [42] and the one-step LLA [43], when $d < n$, these algorithms are initialized with the ordinary least-squares coefficients. Indeed, they need a consistent estimator of the $\boldsymbol{\beta}$ for initializing and for computing the adaptive weight in order to guarantee some theoretical properties. In both cases, oracle properties of these algorithms have been proved according to these hypotheses. In the same situation, if our algorithm is initialized in the same way, our estimate $\hat{\boldsymbol{\beta}}$ shares the same oracle properties since we exactly solve the same optimization problem but the optimization error of our solution (for finite n) is better.

C. Other comments

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 [10] or [8]. The first option is to implement an annealing design of ε: beginning with a large value, ε is iteratively decreased towards zero. An alternative is to set ε as a function of the magnitude of the coefficients β_j^t at the current iteration. In our experimental analysis, we have kept this parameter fixed ($\varepsilon = 0.01$) and left for further studies its influence.

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 [42] or the one-step SCAD [43].

For these purposes, we have designed an experiment where the true coefficient vector $\boldsymbol{\beta}^\star$ is known. For each trial, we have done the following. We have selected a coefficient vector $\boldsymbol{\beta}$ of dimension $d = 256$ for which the number k of active elements in the dictionary is varying. The k non-zeros positions are chosen randomly and the non-zeros values are either drawn from zero-mean unit variance Gaussian distribution or a symmetric Bernouilli distribution with ± 1 values. We have sampled a random matrix $\mathbf{X}$ of dimension $n \times d$ with columns drawn uniformly from the surface of a unit hypersphere with $n = 128$. Then, the target vector is obtained as

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

where $\boldsymbol{\xi}$ 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

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

For each trial, we have run our algorithm with different non-convex penalties as well as the Lasso. As a performance measure, we have used the F-measure between the support of the true coefficient vector $\boldsymbol{\beta}^\star$ and the estimated one $\hat{\boldsymbol{\beta}}$. The support of a given vector $\boldsymbol{\beta}$ is defined as

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

where γ is a threshold that allows us to neglect some true non-zero coefficients that may be obliterate by the noise. The Fmeasure $Fmeas$ is then obtained as :

$$Fmeas = 2\frac{\text{Pr} \cdot \text{Re}}{\text{Pr} + \text{Re}}$$

with

$$\text{Pr} = \frac{|\text{supp}(\boldsymbol{\beta}^\star) \cap \text{supp}(\hat{\boldsymbol{\beta}})|}{|\text{supp}(\hat{\boldsymbol{\beta}})|} \quad \text{and} \quad \text{Re} = \frac{|\text{supp}(\boldsymbol{\beta}^\star) \cap \text{supp}(\hat{\boldsymbol{\beta}})|}{|\text{supp}(\boldsymbol{\beta}^\star)|}$$

In all reported results, we have used $\gamma = 0.001$. Note that all the algorithms we used share the same structure (penalized least-squares) with the same hyperparameter. Thus, the results we reported here do not consider the problem of selecting that hyperparameter λ. This makes algorithms comparison easier

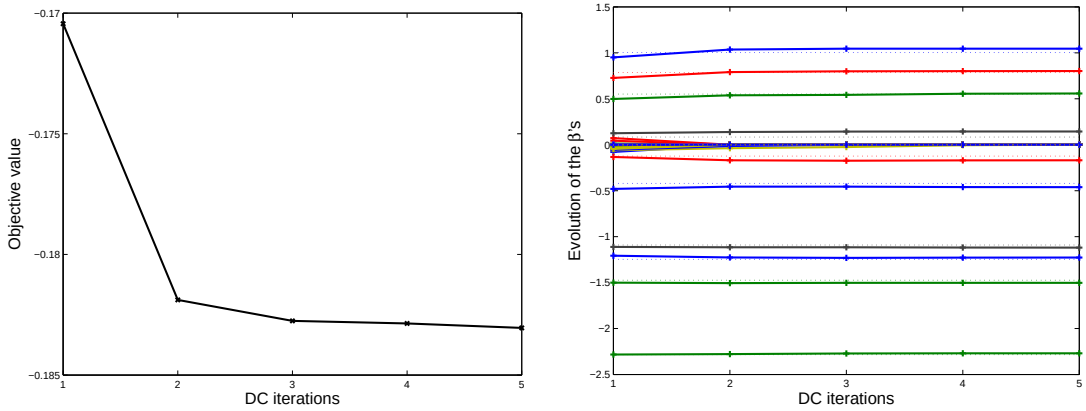

Fig. 3. Example of (left) objective value and (right) coefficient β evolution along DC iterations. The example corresponds to a signal of dimension 128 and a dictionary size of 256. The number of non-zeros coefficients is equal to 10. This example clearly illustrates our point that i) more DC iterations lead to better convergence, ii) more DC iterations improve the estimation of the β's sparsity support (see iteration 3 and 4 where the fifth coefficient from top vanishes). True coefficient values are represented in dotted lines.

since difference in performances can not be attributed to poor model selection (which in any case can be performed by using the same method and criterion).

We have also run some empirical study using a Dirac-Fourier dictionary but since the empirical results we get are similar to those obtained with random dictionary, for a sake of space saving, we do not present those results.

A. Illustration of algorithm correctness

In this first part, we aim at checking the correctness of our algorithm. Indeed, we briefly show that several (compared to two as used by the adaptive Lasso or the LLA SCAD) DC iterations 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 as well as the coefficient vector variation with respects to the DC algorithm iterations. We can see that the objective value keeps decreasing along the iterations. And while only slight objective decrease may be achieved, more DC iterations yield a better estimation of the true coefficient sparsity support. This example is just an illustration of how our algorithm behaves. The next paragraph gives a more detailed analysis of its performance.

B. Performance analysis

In this paragraph, we report the performance averaged over 30 trials of the different sparsity-inducing penalties (including the Lasso). In Figure 4, we have plotted the variation of the F-measure score with respects to the number k of active elements in the dictionary, as well as the performance gain when using two DC iterations compared to full convergence of DC iterations. Experimental results from two signal-to-noise ratio (30 dB and 10 dB) have been reported.

From these plots, we can see that compared to the Lasso, non-convex penalties allow better recovery of sparsity support of underlying signals regardless to the different experimental settings (low/high signal-to-noise ratio or Gaussian/ Bernouilli coefficient entries) and this up to a given value of k. Indeed, when the sparsity degree of the coefficient vector reaches such a value, the Lasso algorithm gives better performances on average than other non-convex penalties. We suspect that this poorer performance is essentially due to the algorithm being trapped in some "bad" local minimal. Such a hypothesis will be confirmed in the sequel. When comparing performances of the different non-convex penalties we can note that they all perform similarly with the log penalty being slightly more competitive than the others.

When analyzing the utility of a fully-optimized DC algorithm compared to two-iteration optimization (which is equivalent to the one-step estimation [43] or the adaptive Lasso [42]), we note that there exists 3 regimes: for small and large values of k, a fully-optimized DC program leads to nearly similar performances as a two-step solution. Our rationale for this is that, on one hand when the problem is easy (for small values of k), few iterations already yield good sparsity recovery performances. On the other hand, when the problem is difficult (large values of k) with many local minima, one can be trapped in "bad" ones. Usefulness of DC programming is made clear essentially for medium values of k. For these situations, gain in performance reaches 10% for some penalties.

According to these results, the log penalty performance deserves a special comment. Indeed, we can see that when the problem is "easy", this penalty leads to the best performance but as soon as the number of active elements in the dictionary is too high, the log penalty yields the poorest performance and besides, the fully optimized DC algorithm performs worse than the two-iteration solution. This again suggests that bad local minima may be in cause but more theoretical analysis and experimental studies still have to be carried out in order to fully understand this point.

Similarly to the Lasso solvers mentioned in subsection II-D, our algorithm can benefit from warm-start techniques when computing a set of different solutions (also denoted as the regularization path) depending on λ . Figure 5 shows the performance of all considered penalties as λ varies as well as the

cumulative computational time needed for obtaining all these solutions. Again we can note that if the regularization parameter is correctly selected, all algorithms perform similarly with a slight preference for the log penalty. Regarding computational time, our DC algorithm leads to an overload of about $5-7$ times the computational complexity of the Lasso, which is rather good compared to the number of Lasso iterations involved. This gain in computational time is essentially due to warm-start approaches.

C. Influence of initialization

We have suggested that a good initialization of our 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 next experiment aims at showing that this initialization is consistently better than other random ones

The experiments we carried out is similar than 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 for the remaining initialization, we use with a random zero mean and unit variance Gaussian vector. Here, we have considered $k = 10$ and $k = 50$ number of active elements in the dictionary and true coefficient vector with Gaussian non-zero entries.

Figure 6 depicts for each initialization the pairs (F-measure, objective value) for different penalty functions. Results show that initialization matters at a different degree. At first note that, in most cases, initializing with $\alpha_j = 0$ leads to very good performance while with random initialization, one can get poor results. We can also note that the Lasso initialization frequently leads to better performance than other non-random ones. For $k = 10$, we can see that the log penalty is more sensitive than the other penalties to a random initialization. Interestingly, for the Zhang's and the ℓ_q penalties, all initializations we used converge towards the same solution $\hat{\boldsymbol{\beta}}$. This suggests that all the initializations belong to the same attraction basin, and that for $k = 10$, the non-convexity of the objective function we are considering is not "severe".

However, when the number of non-zero elements in the coefficient vector β increases, all penalties suffer from random initialization. This suggests that the "convexity" of the problem depends both on the penalty function and the number of active elements in the dictionary. However, a more detailed analysis of this phenomenon should be carried out in order to understand it.

V. Conclusions

Several works in the literature have already shown that non-convex penalties instead 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 from 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.

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.

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 liner 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] E. Candès and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse Problems*, vol. 23, no. 3, pp. 969–985, 2006.

[7] 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.

[8] E. Candès, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Analysis and Applications*, 2008.

[9] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *IEEE Signal Processing Letters*, vol. 14, pp. 707–710, 2007.

[10] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2008.

[11] 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.

[12] 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.

[13] D. Donoho and X. Huo, "Uncertainty principles and ideal atomic decompositions," *IEEE Trans. on Information Theory*, vol. 47, no. 2845-2863, 2002.

[14] D. Donoho and Y. Tsaig, "Extensions of compressed sensing," *Signal Processing*, vol. 86, no. 3, pp. 533–548, 2006.

[15] 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

[16] 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.

[17] 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.

[18] I. Frank and J. Friedman, "A statistical view of som chemometrics regression tools (with discussion)," *Technometrics*, vol. 35, no. 109-148, 1993.

[19] 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.

[20] W. J. Fu, "Penalized regression: the bridge versus the lasso," *Journal of Computational and Graphical Statistics*, vol. 7, pp. 397–416, 1998.

[21] R. Horst and N. V. Thoai, "Dc programming: overview," *Journal of Optimization Theory and Applications*, vol. 103, pp. 1–41, 1999.

[22] 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.

[23] S.-J. Kim, K. Koh, M. Lustig, and a. D. G. S. Boyd, "A method for large-scale l1-regularized least squares," *IEEE Journal on Selected Topics in Signal Processing*, vol. 4, no. 1, pp. 606–617, 2007.

[24] K. Knight and W. Fu, "Asymptotics for lasso-type estimators," *Annals of Statistics*, vol. 28, pp. 1356–1378, 2000.

[25] 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.

[26] 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.

[27] M. Nikolova, "Local strong homogeneity of a regularized estimator," *SIAM Journal of Applied Mathematics*, vol. 61, no. 2, pp. 633–658, 2000.

[28] 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.

[29] ——, "On the lasso and its dual," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 319–337, 2000.

[30] R. Rockafellar, *Convex Analysis*. Princeton University Press, 1996.

[31] R. Saab, R. Chartrand, and Özgür Yilmaz, "Stable sparse approximations via nonconvex optimization," in *33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2008.

[32] 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.

[33] 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.

[34] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society*, vol. 46, pp. 431–439, 1996.

[35] J. Tropp, "Just relax: Convex programming methods for identifying sparse signals," *IEEE Trans. Info. Theory*, vol. 51, no. 3, pp. 1030–1051, 2006.

[36] B. A. Turlach, W. N. Venables, and S. J. Wright, "Simultaneous variable selection," *Technometrics*, vol. 27, pp. 349–363, 2005.

[37] 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.

[38] A. L. Yuille and A. Rangarajan, "The concave-convexe procedure," in *Proc. of Advances in Neural Information Processing Systems*, 2001.

[39] T. Zhang, "Some sharp performance bounds for least squares regression with l_1 regularization," Dept. of Statistics, Rutgers University, Tech. Rep., 2007.

[40] Z. Zhang, J. T. Kwok, and D.-Y. Yeung, "Surrogate maximixation/minimization algorithms," *Machine Learning*, pp. 1–33, 2007.

[41] P. Zhao and B. Yu, "On model selection consistency of lasso," *Journal of Machine Learning Research*, vol. 7, pp. 2541–2563, 2006.

[42] H. Zou, "The adaptive lasso and its oracle properties," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, 2006.

[43] 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.

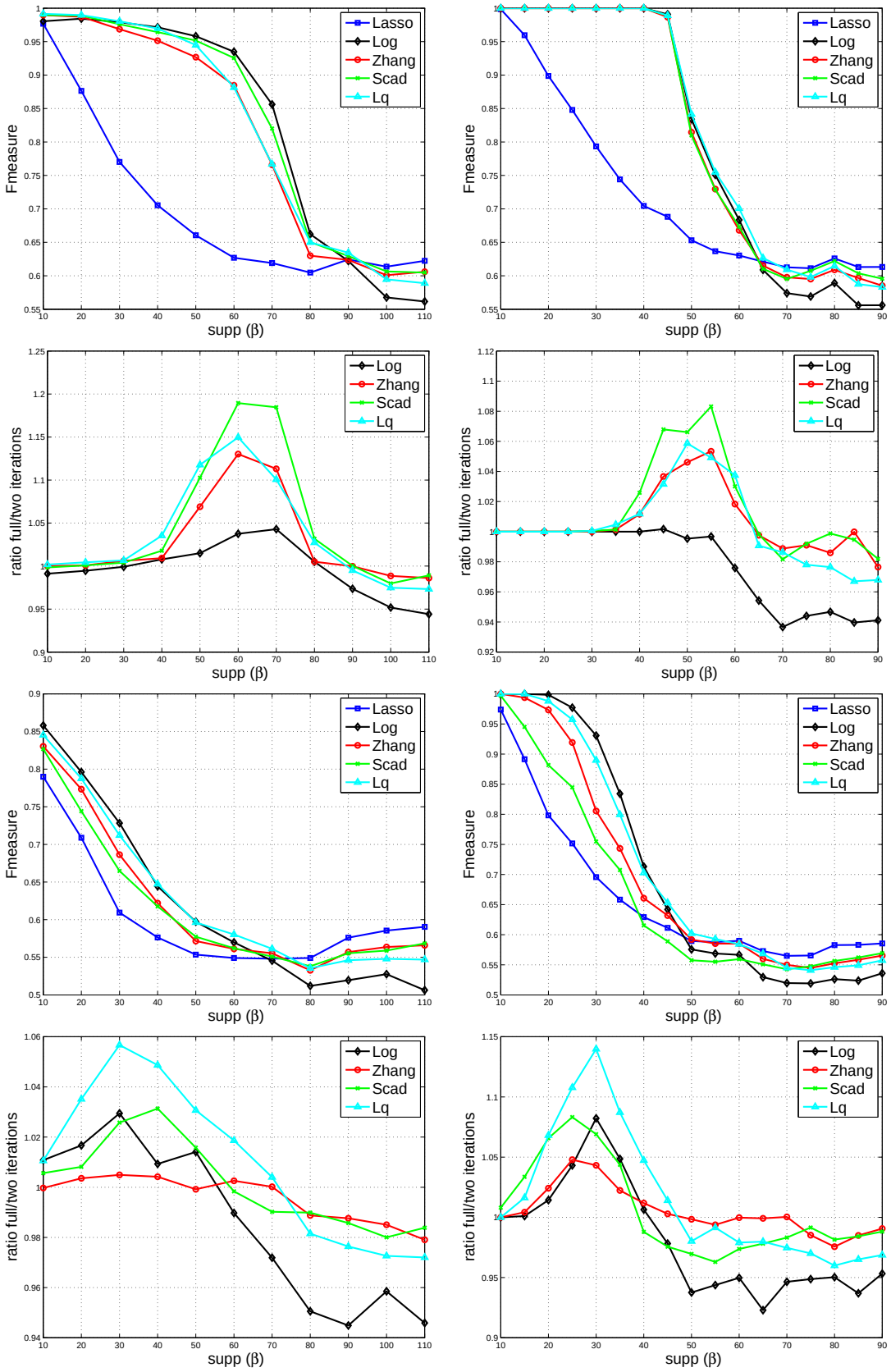

Fig. 4. F-measure of the different penalty functions with respects to the number of active elements. The four top figure are related to a signal to noise ratio of 30 dB while for the four bottom ones, we used SNR=10 dB. Left and right columns respectively depict results with Gaussian and Bernouilli entries of the non-zero elements.

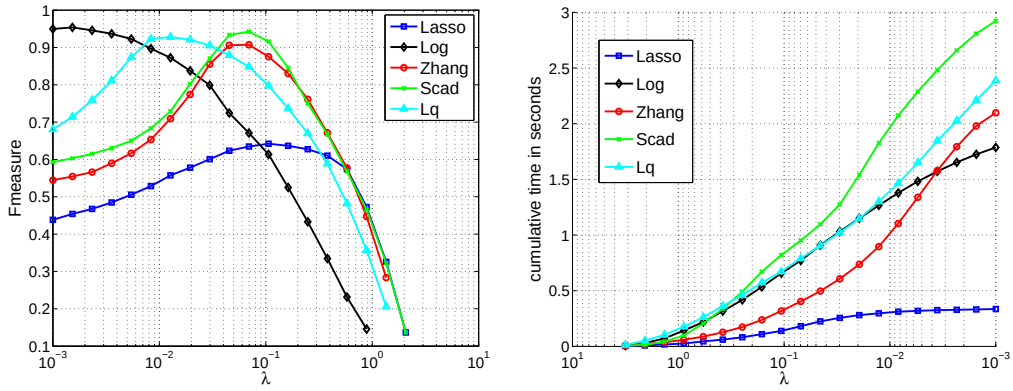

Fig. 5. Examples of regularization paths for different sparsity inducing penalties and the related cumulative computational time (averaged over 30 iterations). Note that since we start from large values of λ, for the cumulative time the x-axis has been reverted. Again in this example $n = 128$ and $d = 256$.

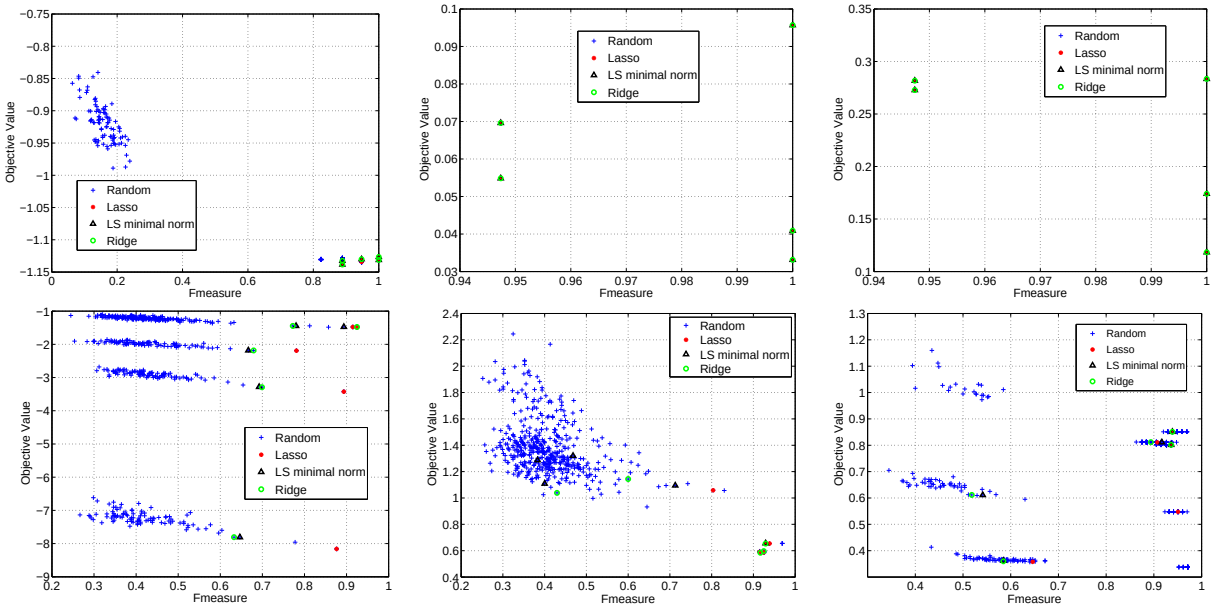

Fig. 6. Examples of couples (objective values, performance measure) for different initializations and for the following non-convex penalty functions. (From left to right) Log, Zhang, ℓ_q penalty. (Top row) Number of active elements $k = 10$. (Bottom row) $k = 50$. The results involve 5 different trials and thus 5 optimal pairs (objective value, performance).