Noname manuscript No.
(will be inserted by the editor)

Efficient Block-coordinate Descent Algorithms for the Group Lasso

Zhiwei (Tony) Qin ·
Katya Scheinberg ·
Donald Goldfarb

Received: date / Accepted: date

Abstract We present two algorithms to solve the Group Lasso problem [33]. First, we propose a general version of the Block Coordinate Descent (BCD) algorithm for the Group Lasso that employs an efficient approach for optimizing each subproblem exactly. We show that it exhibits excellent performance when the groups are of moderate size. For groups of large size, we propose an extension of ISTA/FISTA [2] based on variable step-lengths that can be viewed as a simplified version of BCD. By combining the two approaches we obtain an implementation that is very competitive and often outperforms other state-of-the-art approaches for this problem. We show how these methods fit into the globally convergent general block coordinate gradient descent framework in [28]. We also show that the proposed approach is more efficient in practice than the one implemented in [28]. In addition, we apply our algorithms to the Multiple Measurement Vector (MMV) recovery problem, which can be viewed as a special case of the Group Lasso problem, and compare their performance to other methods in this particular instance.

Keywords block coordinate descent · Group Lasso · iterative shrinkage thresholding · multiple measurement vector · line-search

1 Introduction

Parsimonious models are important in machine learning because of the common occurrence of high dimensional data. A traditional approach to enforce sparsity in the feature coefficients is l_1-regularization (Lasso) [26]. However, it has been shown that the Lasso tends to select only one variable from a group of highly correlated variables and does not care which one is selected [34]. The Group Lasso [33] is a popular extension of the Lasso, which addresses the above problem by imposing sparsity on groups of variables (features) by via $l_{2,1}$-regularization. It solves the unconstrained

Z. Qin and D. Goldfarb
Department of Industrial Engineering and Operations Research
Columbia University
New York, NY 10027
E-mail: {zq2107,goldfarb}@columbia.edu
Research supported in part by NSF Grants DMS 06-06712 and DMS 10-16571, ONR Grant N00014-08-1-1118 and DOE Grant DE-FG02-08ER25856.

K. Scheinberg
Department of Industrial and Systems Engineering
Lehigh University
Bethlehem, PA 18015
E-mail: katyas@lehigh.edu

optimization problem

$$\min_x \frac{1}{2}\|Ax-b\|^2 + \lambda \sum_{j=1}^{J} \|x_j\|, \tag{1}$$

where $A \in \mathbb{R}^{n\times m}$ is the data matrix, and $x = (x_1, \ldots, x_J) \in \mathbb{R}^m$ is the vector of feature coefficients to be estimated ($||\cdot||$ without a subscript denotes $||\cdot||_2$), with $x_j \in \mathbb{R}^{m_j}$ denoting a segment of x corresponding to the j-th group of coefficients. The penalty parameter λ determines the level of group sparsity to be enforced in the solution. We denote by A_j the submatrix of A consisting of the columns corresponding to the elements of x_j. It is assumed that the grouping/partitioning information is given. The Group Lasso model has been well studied recently (e.g. [1]) and has been shown to be effective for many applications, such as micro-array data analysis [16], gene selection [17], and birth-weight prediction [33].

Several methods have been proposed to solve problem (1) (see e.g., [5][12][15][17][23][31][32][33]). In this paper we focus on two block-coordinate descent approaches - an extension of the classical Block Coordinate Descent (BCD) method to the Group Lasso [33], where exact minimization is performed over each group of variables and a BCD version of a prox-gradient method in [2]. The BCD method relies on the fact that the objective function can be efficiently optimized over one group of variables. In [33] and [17], it is assumed that the group-blocks of A are orthonormal. In this case, each group subproblem can be solved in closed form. The gradient-based method [15] approximates the objective function using gradient information. This also generates subproblems that have closed form solutions. However, as we show in our computational results, the BCD approach (when it can be applied) often outperforms the existing gradient-based approaches.

Our contribution in this paper is the following. First, we derive the BCD algorithm for solving the Group Lasso without requiring the blocks of A to be orthonormal. This algorithm, after initially computing the eigen-decomposition of the matrices $A_j^T A_j$ for $j = 1, \cdots, J$, solves the BCD subproblems by Newton's method, which is very efficient as long as the group sizes are moderate. When the number of variables in a particular group is large, the application of Newton's method and the eigen-decomposition can become expensive. In this case we propose another method, which is a block-coordinate version of the (Fast) Iterative Shrinkage Thresholding Algorithm (ISTA/FISTA) [2]. A version of FISTA is implemented for Lasso and Group Lasso in the state-of-the-art software SLEP [15]. ISTA/FISTA are gradient-based approaches which enjoy favorable convergence rates and produce subproblems which can be solved in closed form. However, the practical performance of ISTA/FISTA is often inferior to BCD as we show in Section 7. Our proposed block-coordinate version (ISTA-BC) extends the desirable properties of the BCD method to the ISTA/FISTA approach, thus expanding the range of problems for which the block-coordinate methods can be applied. In particular, ISTA-BC steps can be applied in cases where a group with a large number of variables is present. It is also possible to apply ISTA-BC to the Group Lasso with logistic regression, where optimization over groups is more difficult. To obtain a unified approach for the Group Lasso problems, we combine BCD and ISTA-BC to yield an implementation that has good practical performance for data sets of all group sizes. We demonstrate that our proposed algorithms are very competitive and often outperform other approaches on test sets of Group Lasso problems and MMV problems [3] from signal processing. Our tests include comparisons against SLEP [15], a block coordinate gradient descent (BCGD) approach with an Armijo line-search proposed in [28], SPARSA [31], and SPG [5]. We also show that ISTA-BC and BCD steps fit into the globally convergent framework described in [28]; hence, our proposed algorithms are globally convergent.

2 Block Coordinate Descent Algorithms

2.1 BCD-GL

Block coordinate descent (BCD) algorithms optimize the objective function over one segment (group of variables) x_j at each sub-iteration, while keeping all the other segments $x_i \neq x_j$ fixed. The global convergence of the BCD iterates has been established for minimizing a convex non-differentiable function with certain separability and regularity properties [27].

For the Group Lasso (1), at the j-th sub-iteration, we need to solve

$$\min_{x_j} \frac{1}{2} x_j^\top M_j x_j + p_j^\top x_j + \lambda \|x_j\|, \tag{2}$$

where $M_j = A_j^\top A_j$, and $p_j = A_j^\top \left(\sum_{i \neq j} A_i x_i\right)$. Yuan and Lin [33] applied the BCD method to solve (1) under the restrictive assumption that $A_j^\top A_j = I$. In this case, each block subproblem has a closed-form solution (see (11) below). Tibshirani et. al. [10] dropped this restrictive assumption and proposed an alternative version where the subproblems are solved inexactly by a coordinate descent method. Here, we derive a BCD method for the general case of (1), where each subproblem is solved as a *trust-region* subproblem by an efficient application of Newton's method (we refer to our method as BCD-GL).

If $\|p_j\| \leq \lambda$, then $p_j^\top x_j + \lambda \|x_j\| \geq 0$ for all x_j. Since $M_j \succeq 0$, the objective function of (2) is non-negative in this case, and clearly, $x_j = 0$ solves (2). Conversely, if $x_j = 0$ solves (2), then from the first-order optimality conditions for (2), $p_j + \lambda g_0 = 0$ for some subgradient g_0 of $\|x_j\|$ at $x_j = 0$. Thus, $\|p_j\| = \lambda \|g_0\| \leq \lambda$ since $\|g_0\| \leq 1$. Hence, $x_j = 0$ is the optimal solution of (2) if and only if $\|p_j\| \leq \lambda$.

When the optimal solution of (2) is $x_j \neq 0$, we have from the optimality conditions for (2) that

$$\left(M_j + \frac{\lambda}{\|x_j\|} I\right) x_j = -p_j. \tag{3}$$

Now, note that there exists a $\Delta > 0$ for which the optimal solution of (2) is the optimal solution of the so-called *trust-region* subproblem

$$\begin{aligned} \min \quad & \frac{1}{2} x_j^\top M_j x_j + p_j^\top x_j \\ s.t. \quad & \|x_j\| \leq \Delta. \end{aligned} \tag{4}$$

Hence, we can apply the techniques in [20] and [18] to solve (4). Since $\lambda > 0$, $M_j + \frac{\lambda}{\|x_j\|} I \succ 0 \quad (M_j \succeq 0)$. It follows from (3) that $\|x_j^*\| = \Delta$ with x_j^* being the unique optimal solution to (4) (see Lemmas 2.1 and 2.3 in [18] and Theorem 4.3 in [20]). Hence,

$$x_j^* = -\left(M_j + \frac{\lambda}{\Delta} I\right)^{-1} p_j, \tag{5}$$

which we can write as $x_j^* = \Delta y_j(\Delta)$, where

$$y_j(\Delta) = -(\Delta M_j + \lambda I)^{-1} p_j \tag{6}$$

has norm equal to 1. The optimal solution of (2) is thus of the form (5) with Δ chosen to satisfy $\|y_j(\Delta)\| = 1$. To find the correct Δ, we have from the eigen-decomposition of M_j that

$$\|y_j(\Delta)\|^2 = \sum_i \frac{(q_i^\top p_j)^2}{(\gamma_i \Delta + \lambda)^2}, \tag{7}$$

where the γ_i's and q_i's are the eigenvalues and the respective orthogonal eigenvectors of M_j. Rather than applying Newton's method to $\|y_j(\Delta)\| = 1$ to find the root Δ, we apply Newton's method to

$$\phi(\Delta) = 1 - \frac{1}{\|y_j(\Delta)\|}, \tag{8}$$

since it is well known that (8) works better in practice (see [20] Section 4.2). Note that from (3), we are guaranteed to have a positive solution, so we are immune from the "hard case" described in [18]. In order to compute the first derivative $\phi'(\Delta)$, we use the facts that

$$\frac{d}{d\Delta}\left(\frac{1}{\|y_j(\Delta)\|}\right) = \frac{d}{d\Delta}(\|y_j(\Delta)\|^2)^{-\frac{1}{2}} = -\frac{1}{2}(\|y_j(\Delta)\|^2)^{-\frac{3}{2}}\frac{d}{d\Delta}\|y_j(\Delta)\|^2, \tag{9}$$

and

$$\frac{d}{d\Delta}\|y_j(\Delta)\|^2 = -2\sum_i \frac{(q_i^\top p_j)^2\gamma_i}{(\gamma_i\Delta+\lambda)^3}.$$

The above procedure where Δ and $y_j(\Delta)$ are determined is embodied in Algorithm 2.1 below for solving (2).

Algorithm 2.1 BCD-GL

```
Given: $x^{(0)} \in \mathbb{R}^m$. Set $k = 1$, and for $j = 1, \cdots, J$ compute the eigen-decomposition of $M_j = A_j^T A_j$.
repeat
  $x \leftarrow x^{(k-1)}$
  for $j = 1, 2, \ldots, J$ do
    $d \leftarrow \sum_{i \neq j} A_i x_i - b$, $p \leftarrow A_j^\top d$
    if $\|p\| \leq \lambda$ then
      $\tilde{x}_j \leftarrow 0$
    else
      Find the root $\Delta$ of (8), where $\|y_j(\Delta)\|$ is given by (7), using Newton's root-finding method.
      Compute $y_j$ from (6). $\tilde{x}_j \leftarrow \Delta y_j$
    end if
  end for
  $x^{(k)} \leftarrow \tilde{x}$, $k \leftarrow k + 1$
until $k = k_{max}$ or $x^{(k)}$ satisfies the stopping criterion.
```

In our experiments, the size of the problem had little effect on the number of Newton iterations required for root finding, which is usually less than five. The eigen-decomposition for each group is computed just once. The complexity of this step is $O(m_j^3)$, where m_j is the size of the j-th group. This is typically not more expensive than computing M_j itself, which takes $O(nm_j^2)$ operations. For small group sizes, this additional work is small compared to the overall per iteration cost, but when the group size is large, the task can be computationally costly. In this case, the orthogonalization proposed in [33] is also expensive ($O(nm_j^2)$). The total storage requirement for the eigen-decompositions is $O(\sum_j m_j^2)$. When the m_j's are small, it is approximately linear in m. On the other hand, the additional storage required for the orthogonalized data matrix is $O(nm)$.

In what follows we modify the ISTA algorithm to capture the good properties of the BCD approach, while trying to alleviate the negative ones.

3 Iterative Shrinkage Thresholding Algorithms

Let us write the objective function in (1) as $F(x) = g(x) + h(x)$, where $g(x) = \frac{1}{2}\|Ax - b\|^2$, and $h(x) = \lambda \sum_{j=1}^{J} \|x_j\|$. Given a point x, the ISTA approach applied to $F(x)$ minimizes the sum of $h(x)$ and a quadratic approximation of $g(x)$ at each iteration, i.e. the next iterate $x^+ = q_t(x)$ is

$$\begin{aligned} q_t(x) &= \arg\min_z \left\{ g(x) + \nabla g(x)^\top (z - x) + \frac{1}{2t}\|z - x\|^2 + h(z) \right\} \\ &= \arg\min_z \left\{ \sum_j \frac{1}{2t}\|z_j - d(x)_j\|^2 + \lambda \|z_j\| \right\}. \end{aligned} \tag{10}$$

Here, t is the step-length; $d(x) = x - tA^\top(Ax - b) = x - t\nabla g(x)$; z_j and $d(x)_j$ are the segments of the respective vectors z and $d(x)$ corresponding to the j-th segment of x. The optimization problem in (10) is separable, and the solution to each of the J subproblems is given by a *soft-thresholding operator* [5], i.e.

$$q_t(x)_j = T(d(x)_j, t) = \frac{d(x)_j}{\|d(x)_j\|} \max(0, \|d(x)_j\| - \lambda t), \quad j = 1, \cdots, J. \tag{11}$$

The step-length t is determined (usually by a backtracking line-search) so that the following holds

$$g(q_t(x)) \le g(x) + \nabla g(x)^\top (q_t(x) - x) + \frac{1}{2t}\|q_t(x) - x\|^2. \tag{12}$$

This condition ensures that the value $g(q_t(x))$ of g at the new point $q_t(x)$ is smaller than the value of the quadratic approximation to $g(z)$ at the current point x given on the right-hand-side of (12). It is easy to verify that (12) is always satisfied when $t \le 1/L$, where L is the Lipschitz constant for $\nabla g(x)$ (in our case $L = \|A^\top A\|$). However, setting $t = 1/\|A^\top A\|$ usually results in very small step sizes; hence a line search is usually necessary. The statement of the ISTA algorithm with backtracking line-search can be found in [2]. The complexity of ISTA to reach an ϵ-optimal solution is $O(L/\epsilon)$.

FISTA (Fast Iterative Shrinkage Thresholding Algorithm) is an extension of ISTA that has an improved complexity of $O(\sqrt{L/\epsilon})$ [2]. In essence, FISTA constructs the current iterate as a linear combination of the two most recent iterates.

4 ISTA/FISTA with block coordinate step-lengths

In both ISTA and FISTA, one carries out the backtracking line-search for all the segments of x at once in each iteration, so the step-length $t^{(k)}$ at iteration k is the same across the segments. The BCD-GL method computes the step for each block of variables separately. A natural extension of ISTA which mimics some of the properties of block coordinate descent is to allow different step-lengths t_j's for individual segments; i.e. we now have a vector of step lengths

$$\mathbf{t} = [\, t_1,\, t_2,\, \cdots,\, t_J \,]^\top \tag{13}$$

and for $j = 1, \cdots, J$, $d(x)_j = d_j = d_j(x, t_j) = x_j - t_j \nabla g(x)_j$, where $\nabla g(x)_j = A_j^\top (Ax - b)$. Hence using different step lengths for each block, the solution to (10) becomes

$$q_{\mathbf{t}}(x) = [\, q_1,\, q_2,\, \cdots,\, q_J \,]^\top,$$

where

$$q_j = \frac{d_j}{\|d_j\|} \max(0, \|d_j\| - \lambda t_j), \quad j = 1, \cdots, J. \tag{14}$$

4.1 ISTA with multiple-scaling on step-lengths (ISTA-MS)

A simple case in the above setting is to consider $\mathbf{t} \equiv \theta \bar{\mathbf{t}}$ where $\bar{\mathbf{t}} = (\bar{t}_1, \bar{t}_2, \cdots, \bar{t}_J)$ is a fixed vector, for instance, given by $\bar{t}_j = \frac{1}{\|A_j\|}$, and the line-search is performed on θ. We refer to this method as ISTA-MS (for multiple scaling), and show in Theorem 1 that the theoretical convergence rate of ISTA is preserved under this modification. The practical performance of this method is often better than that of the regular ISTA/FISTA, except on the Nemirovski data sets (see Figure 3). While it is still inferior to that of BCD-GL, we should keep in mind that ISTA-MS is parallelizable.

4.2 Block coordinate iterative shrinkage thresholding (ISTA-BC)

Here we propose an extension of the ISTA algorithm where, for each segment, we select t_j by a separate backtracking line-search in conjunction with soft-thresholding (14). Hence each (major) iteration now consists of J optimization steps, each of which seeks q_j that minimizes

$$g(\hat{x}) + \nabla g(\hat{x})_j^\top (q_j - \hat{x}_j) + \frac{1}{2t_j}\|q_j - \hat{x}_j\|^2 + h_j(q) \tag{15}$$

given a vector $\hat{x}$, and $h_j(q) := \lambda\|q_j\|$. The line search condition to be satisfied is

$$g(\tilde{x}) \le g(\hat{x}) + \nabla g(\hat{x})_j^\top (q_j - \hat{x}_j) + \frac{1}{2t_j}\|q_j - \hat{x}_j\|^2 \tag{16}$$

For the j-th sub-iteration, we set $\hat{x} = [\, q_1 \cdots q_{j-1}\ x_j \cdots x_J\,]^\top$.
This ensures that $\tilde{x} = [\, q_1 \cdots q_{j-1}\ q_j\ x_{j+1} \cdots x_J\,]^\top$ yields at least as good an objective value as the quadratic approximation at the most recently computed point $\hat{x}$. We refer to this approach as ISTA-BC (for block-coordinate steps) and state it as Algorithm 4.1. The global convergence of

Algorithm 4.1 ISTA-BC

```
Given: $x^{(0)} \in \mathbb{R}^m, k = 1, t_0 > 0$.
repeat
  $\hat{x} \leftarrow x^{(k-1)}$
  for $j = 1, 2, \dots, J$ do
    Find $t_j \le t_0$ for which $\tilde{x}^{(j)} = [\hat{x}_1, \cdots, \hat{x}_{j-1}, q_j, \hat{x}_{j+1}, \cdots, \hat{x}_J]^\top$ satisfies the line-search condition (16), with $q_j$ given by (14).
    $\hat{x} \leftarrow \tilde{x}^{(j)}$
  end for
  $x^{(k)} \leftarrow \tilde{x}^{(J)}$
  $k \leftarrow k + 1$
until $k = k_{max}$ or $x^{(k)}$ satisfies the stopping criterion.
```

ISTA-BC is guaranteed by Theorem 2 in Section 4.6 based on the results in [28].

Compared to ISTA, a disadvantage of ISTA-BC (and BCD-GL) is that we cannot solve the subproblems in parallel, since the j-th sub-iteration depends on the previous sub-iteration through $\nabla g(\hat{x}_j)$. Nevertheless, as demonstrated in Section 7.5, ISTA-BC enjoys considerable computational advantage over ISTA/FISTA in a non-parallel setting. We note that a parallelizable Jacobi-like version of the ISTA-BC algorithm could be implemented by replacing $\hat{x}$ with $x^{(k-1)}$ in (15) and (16) for all sub-iterations, but we found in our non-parallel experiments that this approach also yielded performance inferior to ISTA-BC.

The FISTA update can be incorporated into the ISTA-BC algorithm. We have implemented such an update (FISTA-BC), but it did not exhibit any computational advantage in our tests. It is also unclear if there is any theoretical advantage in using an acceleration step in the block coordinate framework.

4.3 Hybrid Implementation (BCD-HYB)

We have also implemented a hybrid version of BCD-GL and ISTA-BC to overcome BCD-GL's weakness in scalability when a data set has large groups. Specifically, we switch from a BCD sub-iteration to an ISTA-BC sub-iteration when a group contains more than N_{BC} variables, so that we no longer need to perform an eigen-decomposition for that group. Clearly, this decision can be made a priori. We call this hybrid version BCD-HYB in our experiments. The global convergence of BCD-HYB is also established in Section 4.6.

Note that ISTA-BC and BCD-GL are identical when $A_j^\top A_j = L_j I$ and $t_j = \frac{1}{L_j}$, where L_j is a scalar, as in the MMV problem in Section 5. The proofs of the theoretical rates of convergence of the ISTA/FISTA algorithms do not readily extend to ISTA-BC or BCD-GL.

4.4 Randomized Scheme for Coordinate Selection

Nesterov recently proposed a randomized coordinate selection scheme for the BCD algorithms solving problems with a smooth objective function [19]. Specifically, the index i of the group of variables x_i to be optimized over in the next iteration is selected with probability

$$p_\alpha^{(i)} = \frac{L_i^\alpha}{\sum_{j=1}^J L_j^\alpha}, \quad i = 1, \cdots, J, \tag{17}$$

where $\alpha \in \mathbb{R}$, and L_j is the Lipschitz constant of $\nabla_{x_j} g(x)$, the gradient of $g(x)$ with respect to the j-th group. When $\alpha = 0$, the group indices are sampled from a uniform distribution, and when $\alpha = 1$, the probability of selecting $j \in [1, J]$ is proportional to L_j. The convergence rate results in [19] have been extended to the case of a composite function with a non-smooth block-separable component [22], such as the Group Lasso. We implemented several versions of ISTA-BC, BCD-GL, and BCD-HYB with this randomized scheme for the special cases of $\alpha = 0$ and 1. Note that the case of $\alpha = 1$ only applies for BCD-GL, where the entire set $\{L_j\}_{j=1}^J$ is explicitly computed. We denote by ISTA-RBC, RUBCD-GL, RBCD-HYB, the randomized versions of ISTA-BC, BCD-GL, and BCD-HYB respectively with a uniform distribution, and by RBCD-GL, the randomized version of BCD-GL for the case where $\alpha = 1$. We compare the performance of the randomized versions with the original versions on selected problems in Section 7, and we observe that none of the versions with a uniform distribution provides any computational advantage. On the other hand, RBCD-GL sometimes appears to be more efficient than BCD-GL.

4.5 Convergence Rate Analysis for ISTA-MS

We prove that ISTA-MS with backtracking line-search has a global convergence rate of $O(\frac{1}{k})$, where k is the number of iterations. Our arguments follow closely those in [2].

Define

$$\mathbf{L} = [L_1, L_2, \cdots, L_J] = \left[\frac{1}{t_1}, \frac{1}{t_2}, \cdots, \frac{1}{t_J}\right]. \tag{18}$$

Corresponding to $g(\cdot)$ and $h(\cdot)$ defined in the previous sections, we define

$$g_j : \mathbb{R}^{m_j} \to \mathbb{R}, \quad g_j(z_j; x) = \frac{1}{2} z_j^\top A_j^\top A_j z_j + (\sum_{i \neq j} A_i x_i - b)^\top A_j z_j, \tag{19}$$

$$h_j : \mathbb{R}^{m_j} \to \mathbb{R}, \quad h_j(z_j) = \lambda \|z_j\|. \tag{20}$$

For each j, let $L(g_j) = \|A_j\|^2$ be the Lipschitz constant for ∇g_j. Let $\mathbf{L}(g)$ be the vector of the J Lipschitz constants. Define $\overline{L}(g) = \max_j L(g_j)$, $\underline{L}(g) = \min_j L(g_j)$, $R_{\mathbf{L}} = \frac{\overline{L}(g)}{\underline{L}(g)}$, and $s_j = \sqrt{L(g_j)}$.

For ISTA-MS, the optimization problem that we solve in each iteration is

$$\min_z Q_{\mathbf{L}}^{\mathrm{MS}}(z, x) \equiv g(x) + \nabla g(x)^\top (z - x) + \sum_{j=1}^{J} \frac{L_j}{2} \|z_j - x_j\|^2 + h(z), \tag{21}$$

whose solution is

$$p_{\mathbf{L}}^{\mathrm{MS}}(x) = S_{\mathbf{t}}(d(x)) = \arg\min_z Q_{\mathbf{L}}^{\mathrm{MS}}(z, x). \tag{22}$$

Hence, $x^{(k+1)} = p_{\mathbf{L}}^{\mathrm{MS}}(x^{(k)})$. Since (21) is strongly convex, we have the following property for any $x \in \mathbb{R}^m$ from the optimality conditions for $p_{\mathbf{L}}^{\mathrm{MS}}(x)$:

$$\nabla g(x) + diag(\mathbf{L})^\top (p_{\mathbf{L}}^{\mathrm{MS}}(x) - x) + \partial h(p_{\mathbf{L}}^{\mathrm{MS}}(x)) = 0. \tag{23}$$

From the definitions (19) and (20), it is easy to see that the objective function in the j-th subproblems of ISTA-MS is a quadratic approximation of $g_j(z_j, x^{(k)}) + h_j(z_j)$, where $x^{(k)}$ is the k-th iterate. Similar to Lemma 2.3 in [2], we have the following result.

Lemma 1 *Let $x \in \mathbb{R}^m$ and $\mathbf{L} \succ 0$ satisfy*

$$F(p_{\mathbf{L}}^{\mathrm{MS}}(x)) \leq Q_{\mathbf{L}}^{\mathrm{MS}}(p_{\mathbf{L}}^{\mathrm{MS}}(x), x). \tag{24}$$

Then, for any $y \in \mathbb{R}^m$,

$$F(y) - F(p_{\mathbf{L}}^{\mathrm{MS}}(x)) \geq \sum_{j=1}^{J} \frac{L_j}{2} \|p_{\mathbf{L}}^{\mathrm{MS}}(x)_j - x_j\|^2 + L_j (x_j - y_j)^T (p_{\mathbf{L}}^{\mathrm{MS}}(x)_j - x_j). \tag{25}$$

Proof The proof follows close that in [2], except that a vector $\mathbf{L}$ instead of a scalar L is used, and Lemma 2.2 in [2] is replaced with (23). □

To obtain a lower bound t_{min} for the unscaled step-length in the k-th iteration $t^{(k)}(= \theta$ in Section 4.1), we observe that since $\nabla g(x)$ is Lipschitz continuous with Lipschitz constant $L(g)$,

$$g(p_{\mathbf{L}}^{\mathrm{MS}}(x)) \leq g(x) + \nabla g(x)^\top (p_{\mathbf{L}}^{\mathrm{MS}}(x) - x) + \frac{L(g)}{2} \|p_{\mathbf{L}}^{\mathrm{MS}}(x) - x\|^2.$$

If we set $t^{(k)}$ such that $L_j^{(k)} = \frac{s_j}{t^{(k)}} \geq \frac{\min\{s_j\}}{t^{(k)}} = L(g) \quad \forall j$, then the line-search condition is guaranteed to be satisfied, since $\frac{L(g)}{2} \|p_{\mathbf{L}}^{\mathrm{MS}}(x^{(k+1)}) - x^{(k)}\|^2 \leq \sum_j \frac{L_j^{(k)}}{2} \|p_{\mathbf{L}}^{\mathrm{MS}}(x^{(k+1)})_j - x_j^{(k)}\|^2$. Hence, $t_{min} = \frac{\min\{s_j\}}{L(g)} = \frac{\sqrt{\underline{L}(g)}}{L(g)}$.

With the above intermediate results, we now prove the $O(\frac{1}{k})$ global convergence rate for ISTA-MS.

Theorem 1 *Let $\{x^{(k)}\}$ be the sequence generated by ISTA-MS. Then for any $k \geq 1$,*

$$F(x^{(k)}) - F(x^*) \leq \frac{C}{k}, \tag{26}$$

where x^* is the optimal solution, and $C = \frac{L(g)}{2} \sqrt{R_{\mathbf{L}}} \|x^* - x^{(0)}\|^2$.

Proof Applying Lemma 1 with $y = x^*$, $x = x^{(n)}$, and $\mathbf{L} = \mathbf{L}^{(n)}$, we have

$$2t^{(n)}(F(x^*) - F(x^{(n+1)})) \geq \sum_j s_j(\|x^* - x^{(n+1)}\|^2 - \|x^* - x^{(n)}\|^2)$$

Since $F(x^*) - F(x^{(n+1)}) \leq 0$, we can replace $t^{(n)}$ with t_{min}. Summing the resulting inequality over $n = 0, \cdots, k-1$ yields

$$\frac{2\sqrt{\underline{L}(g)}}{L(g)} \left(kF(x^*) - \sum_{n=0}^{k-1} F(x^{(n+1)}) \right) \geq \sum_j s_j \left(\|x_j^* - x_j^{(k)}\|^2 - \|x_j^* - x_j^{(0)}\|^2 \right). \tag{27}$$

Now, applying Lemma 1 again with $x = y = x^{(n)}$ and $\mathbf{L} = \mathbf{L}^{(n)}$, we get

$$2t^{(n)} \left(F(x^{(n)}) - F(x^{(n+1)}) \right) \geq \sum_j s_j \|x_j^{(n)} - x_j^{(n+1)}\|^2.$$

Since this implies that $F(x^{(n)}) \geq F(x^{(n+1)})$ for all n, it follows that $\sum_{n=0}^{k-1} F(x^{(n+1)}) \geq kF(x^{(k)})$. Combining this with (27) yields

$$\frac{2k\sqrt{\underline{L}(g)}}{L(g)} \left(F(x^*) - F(x^{(k)}) \right) \geq \sum_j s_j \left(\|x_j^* - x_j^{(k)}\|^2 - \|x_j^* - x_j^{(0)}\|^2 \right).$$

Ignoring the non-negative term on the RHS of the above inequality and noting that $\max_j\{s_j\} = \sqrt{\overline{L}(g)}$, we obtain (26) where $C = \frac{\beta L(g)}{2}\sqrt{R_{\mathbf{L}}}\|x^* - x^{(0)}\|^2$, and $L(g)$ is the Lipschitz constant of $\nabla g(x)$. □

4.6 Global Convergence of BCD-HYB

We prove the global convergence of BCD-HYB by demonstrating that ISTA-BC and BCD-GL steps satisfy the conditions on the steps of the general convergent BCGD framework in [28].

First, we briefly describe the BCGD algorithm. The blocks x_j's are chosen in a cyclic manner. We slightly abuse the notation by calling the iteration in which x_j is updated the j-th iteration. In the j-th iteration, we compute the search direction (in the subspace of x_j)

$$d_{H_{jj}} = \arg\min_{d_j} \left(\nabla g(x_j)^\top d_j + \frac{1}{2} d_j^\top H_{jj} d_j + \lambda\|x_j + d_j\| \right). \tag{28}$$

Here, H_{jj} is some approximation of the j-th diagonal block of $\nabla^2 g(x)$, and we denote by $H^{(j)}$ the approximate Hessian used in the j-th iteration with $H_{jj}^{(j)} = H_{jj}$. Once the search direction $d_{H_{jj}}$ is computed, BCGD determines an appropriate step-length $\alpha^{(j)}$ such that the new iterate $x^{(j)} = x^{(j-1)} + \alpha^{(j)} d^{(j)}$ satisfies the following modified Armijo rule:

$$F(x^{(j)}) = F(x^{(j-1)} + \alpha^{(j)} d^{(j)}) \leq F(x^{(j-1)}) + \alpha^{(j)} \sigma \Delta^{(j)}, \tag{29}$$

where $d^{(j)}$ has its j-th segment equal to $d_{H_{jj}}$ and zeros everywhere else, $0 < \sigma < 1$, $0 \leq \gamma < 1$, and

$$\Delta^{(j)} = \nabla g(x)^\top d^{(j)} + \gamma (d^{(j)})^\top H^{(j)} d^{(j)} + \lambda\|x^{(j-1)} + d^{(j)}\| - \lambda\|x^{(j-1)}\|. \tag{30}$$

It is shown in [28] that the BCGD algorithm converges globally for any choice of $H^{(j)}$, if the above Armijo rule holds and if

$$\bar{\theta} I \succeq H^{(j)} \succeq \underline{\theta} I \quad \forall j, \text{ where } 0 < \underline{\theta} \leq \bar{\theta}. \tag{31}$$

Our BCD-HYB algorithm differs from BCGD in that while BCGD computes the search direction and the step-length in two separate stages, BCD-HYB accomplishes both tasks in one single stage. However, as we show below, both the ISTA-BC and BCD-GL steps in BCD-HYB can be viewed as instances of a BCGD step, and they satisfy the two conditions ((29) and (31)) above. Hence, the convergence results of BCGD apply to our BCD-HYB algorithm.

Let us first consider the ISTA-BC steps. We observe that in this case the optimization problem (15) solved during the ISTA-BC steps is essentially identical to (28) solved by BCGD steps with $H_{jj} = \frac{1}{t_j} I$. In addition, we see that our ISTA-BC line-search descent condition (16) agrees with the Armijo rule descent condition (29), with $\alpha^{(j)} = 1\ \forall j, \sigma = 1$, and $\gamma = 0.5$. Hence, from the viewpoint of BCGD, ISTA-BC chooses in the j-th iteration an appropriate t_j so that, with the resulting $H_{jj} = \frac{1}{t_j} I$, the search direction $d_{H_{jj}}$ from (28) satisfies the descent condition (16) automatically. The line-search effort in this case is shifted towards choosing an appropriate t_j.

In the case of BCD-GL, the steps (2) are equivalent to (28) with $H_{jj} = M_j$, which is the true Hessian for the j-th block. Hence, $F(x^{(j)}) = F(x^{(j-1)}) + \Delta^{(j)}$. From the previous paragraph, it is apparent that BCD-GL always satisfies the Armijo rule with $\alpha^{(j)} = 1\ \forall j, \sigma = 1$, and $\gamma = 0.5$.

In both cases, it is easy to see that $\Delta \leq 0$ in (29), which means that if (29) is satified with $\sigma = 1$ then it also holds for any $0 < \sigma < 1$. Hence, both ISTA-BC and BCD-GL steps satisfy the modified Armijo rule.

Now we show that steps of both types satisfy (31). Without loss of generality, assume $t_0 = 1$ in Algorithm 4.1. Then, t_j obtained from the backtracking line-search in the j-th iteration of ISTA-BC satisfies $\max\{L(g_j), 1\} \geq \frac{1}{t_j} \geq 1$. From the BCGD perspective, we effectively set $H^{(j)} = \frac{1}{t_j} I_m$ as explained above. Hence, we have $\max\{\bar{L}(g), 1\} I_m \succeq H^{(j)} \succeq \min\{\underline{L}(g), 1\} I_m\ \forall j$, and condition (31) is satisfied.

For the BCD-GL steps, we essentially set $H^{(j)} = diag(M_1, \cdots, M_J)$. Unlike the ISTA-BC case, we only have $\bar{L}(g) I_m \succeq H^{(j)} \succeq 0$, which may not satisfy (31). Hence, in the cases when M_j is not positive definite (which only happens if columns of A from the same group are linearly dependent), we can apply the following simple modification: $H_{jj} = M_j + \delta I$, where δ is a small positive constant. The solution to the perturbed subproblem still satisfies the Armijo rule, and we now have $(\bar{L}(g) + \delta) I_m \succeq H^{(j)} \succeq \delta I$, which satisfies (31). Hence we have shown that both types of steps can be viewed as those of the BCGD framework in [28], and thus the convergence results for BCGD apply. The computational performance of BCD-HYB is, however, significantly better than that of BCGD. We summarize the global convergence of BCD-HYB as follows.

Lemma 2 *For any $t_j > 0$, $x^{(j-1)}$ is a stationary point of F if and only if $d^{(j)} = 0$.*

Theorem 2 *Let $\{x^{(j)}\}, \{d^{(j)}\}, \{t_j\}$ be sequences generated by BCD-HYB. Then the following results hold.*

1. *$\{F(x^{(j)})\}$ is non-increasing, and $\Delta^{(j)}$ satisfies*

$$-\Delta^{(j)} \geq \frac{1}{2}(d^{(j)})^T H^{(j)} d^{(j)} \geq \frac{1}{2}\|d^{(j)}\|^2 \quad \forall j, \tag{32}$$

$$F(x^{(j+1)}) - F(x^{(j)}) \leq \Delta^{(j+1)} \leq 0 \quad \forall j. \tag{33}$$

2. *If $\{x^{(j)}\}_{\mathcal{K}}$ is a convergent subsequence of $\{x^{(j)}\}$, then $\{\Delta^{(j)}\} \to 0$ and $\{d^{(j)}\}_{\mathcal{K}} \to 0$.*
3. *If the blocks are chosen in a cyclic manner, then every cluster point of $\{\tilde{x}^{(j)}\}$ is a stationary point of F.*
4. *If $\lim_{j\to\infty} F(x^{(j)}) \geq -\infty$, then $\{\Delta^{(j)}\} \to 0$ and $\{d^{(j)}\} \to 0$.*

The condition in the last point of Theorem 2 is always satisfied since $F(x^{(k)}) = \frac{1}{2}\|Ax^{(k)} - b\|^2 + \lambda \sum_{j=1}^{J} \|x_j^{(k)}\| \geq 0$.

5 Multiple Measurement Vector Recovery (MMV)

The basic compressed sensing aims to recover an unknown sparse signal vector from a single measurement vector [6][9] and is called the single measurement vector (SMV) model. The MMV model [7] extends the SMV model to reconstruct a signal matrix $X = \begin{pmatrix} X_1 & \cdots & X_K \end{pmatrix}$ from the data matrix $A \in \mathbb{R}^{n\times m}$ and a matrix of multiple measurement vectors $B = \begin{pmatrix} B_1 & \cdots & B_K \end{pmatrix} \in \mathbb{R}^{n\times K}$, which are related by $AX = B$. The signal (column) vectors $\{X_i\}_{i=1}^K$ are assumed to be jointly sparse, i.e. they have non-zero entries concentrated in common rows. The convex relaxed version of the MMV model solves the optimization problem

$$\begin{aligned} \min_X \ & \|X\|_{1,2} \\ s.t. \ & AX = B, \end{aligned} \tag{34}$$

where $\|X\|_{1,2} = \sum_i \|x^i\|_2$ and x^i is the i-th row of X. Several methods have been proposed for solving the MMV problem (34); e.g., see [3][4][25]. In [4], the noisy version of (34),

$$\begin{aligned} \min_X \ & \|X\|_{1,2} \\ s.t. \ & \|AX - B\|_F \le \sigma \end{aligned} \tag{35}$$

is considered.

5.1 Link to the Group Lasso

We can solve (34) by solving a series of sub-problems [1]

$$\min_X \|X\|_{1,2} + \frac{1}{2\mu}\|AX - B\|_F^2, \tag{36}$$

when $\mu \to 0$. The solution $X^{(k)}$ to the k-th sub-problem serves as the starting point for the $(k+1)$-st sub-problem. By defining $g(X) = \frac{1}{2}\|AX - B\|_F^2$, it is straightforward to apply ISTA-BC to solve (36). The optimal solution to the j-th subproblem of (36) is given by the soft-thresholding operator $T(d^j, t_j) = \frac{d^j}{\|d^j\|}\max(0, \|d^j\| - \mu t_j)$, where d^j is the j-th row of the matrix $X - diag(\mathbf{t})A^\top(AX - B)$.

We can also cast (36) directly as an instance of the Group Lasso. Define $\tilde{A}$ to be the block-diagonal matrix with each diagonal block equal to the matrix A and $\tilde{x}$ and $\tilde{b}$ to be the column-vectorized form of the matrices X and B respectively:

$$\tilde{A} = \begin{pmatrix} A & & & \\ & A & & \\ & & \ddots & \\ & & & A \end{pmatrix}, \tilde{x} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{pmatrix}, \tilde{b} = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_K \end{pmatrix}.$$

It is easy to see that $\tilde{A}$ and $\tilde{b}$ are the input data if we are to solve (36) as the Group Lasso problem, and the corresponding solution is $\tilde{x}$. Since the rows of X are the segments, the j-th segment in $\tilde{x}$ consists of the j-th, the $(n+j)$-th, ..., and the $((K-1)n+j)$-th entries. We can re-arrange the columns of $\tilde{A}$ so that the entries in $\tilde{x}$ belonging to the same segment are contiguous. $\tilde{b}$ remains the

[1] This is equivalent to a special case of what is recently known as multi-task regression with structured sparsity [13].

same. Now, denoting the j-th segment of $\tilde{x}$ by $\tilde{x}_j = (\tilde{x}_{j1}, \tilde{x}_{j2}, \cdots, \tilde{x}_{jK})^\top$ and letting $\tilde{A}_j$ be the corresponding block in $\tilde{A}$, we have

$$\tilde{A}_j = \begin{pmatrix} A_j & & & \\ & A_j & & \\ & & \ddots & \\ & & & A_j \end{pmatrix} \text{ and } \tilde{A}_j \tilde{x}_j = \begin{pmatrix} A_j x_{j1} \\ A_j x_{j2} \\ \vdots \\ A_j x_{jK} \end{pmatrix},$$

where A_j is the j-th column of A.

5.2 Equivalence of ISTA-BC and BCD-GL

Since $g(X)$ is a quadratic function in X, $H = \nabla^2 g(X) = \tilde{A}^\top \tilde{A}$. Because of the special structure of $\tilde{A}$ as we have shown above, the j-th diagonal block of H, denoted H_{jj}, is a multiple of the identity matrix:

$$H_{jj} = \tilde{A}_j^\top \tilde{A}_j = \|A_j\|^2 I_K. \tag{37}$$

H_{jj} is the true Hessian of the j-th block-coordinate of $g(X)$. Hence, the BCD subproblem (2) has the same simple closed-form solution $T(d^j, t_j)$ as ISTA-BC with $t_j = \frac{1}{\|A_j\|^2}$. In fact, it is also equivalent to BCGD with least squares loss. The equivalence relationships imply that the resulting algorithm does not require the eigenvalue decompositions in BCD-GL or the backtracking line-search in ISTA-BC, and hence, it can be very efficient. We state this specialized version in Algorithm 5.1 (BCD-MMV).[2]

Algorithm 5.1 BCD-MMV

```
1: Given: $X^{(0)} \in \mathbb{R}^m, A \in \mathbb{R}^{n\times m}, B \in \mathbb{R}^{n\times K}$.
2: $s_j \leftarrow \|A_j\|^2, \quad j = 1, \dots, m$.
3: repeat
4:   $\hat{X} \leftarrow X^{(k-1)}$
5:   for $j = 1, 2, \dots, m$ do
6:     $w \leftarrow s_j \hat{x}^j - A_j^\top (A\hat{X} - B)$, where $\hat{x}^j$ is the $j$-th row of $\hat{X}$.
7:     $z \leftarrow T(\frac{\|w\|}{s_j}, \frac{1}{s_j})$, where $T(\cdot,\cdot)$ is defined in (11).
8:     $\tilde{X}^{(j)} \leftarrow \begin{pmatrix} \hat{x}^1 \\ \vdots \\ \hat{x}^{j-1} \\ z \\ \hat{x}^{j+1} \\ \vdots \\ \hat{x}^m \end{pmatrix}$
9:     $\hat{X} \leftarrow \tilde{X}^{(j)}$
10:  end for
11:  $X^{(k)} \leftarrow \tilde{X}^{(m)} (= \hat{X})$
12:  $k \leftarrow k + 1$
13: until $k = k_{max}$ or $X^{(k)}$ satisfies the stopping criterion.
```

[2] This algorithm coincides with the M-BCD method proposed in [21] recently while the first version of this paper was in preparation.

6 Implementation

In ISTA-BC, we use back-tracking line search to determine the value(s) for $\mathbf{t}$. At the beginning of each sub-iteration for ISTA-BC, we allow the initial step-length to increase from the previous step size for that segment by a factor of $\frac{1}{\beta}$, where $0 < \beta < 1$.

The major computational work of the line-search performed by ISTA-BC (and FISTA-BC) lies in computing the residual $r = Ax - b$. In the j-th sub-iteration only the j-th segment of x is updated. Hence, we can update the residual incrementally as $A\tilde{x} - b = A_j(z_j - x_j) + (Ax - b)$. The old residual $Ax - b$ is available before the sub-iteration, so the total work of this scheme is on average $\frac{1}{J}$ of the work required to compute $A\tilde{x} - b$ from scratch, where J is the total number of segments. To avoid the accumulation of arithmetic errors over many iterations, we compute the actual residual every J sub-iterations.

We set $N_{BC} = 200$ for our BCD-HYB implementation discussed in Section 4.3. In Algorithm 2.1, the initial value of Δ was set at 0 for the first iteration of Newton's method. The result of each iteration was then used as the initial value for the next iteration.

7 Numerical Experiments

All numerical experiments were run in Matlab on a laptop with an Intel Core 2 Duo CPU and 4G memory. We recorded the CPU times, the number of matrix-vector multiplications (Aprods), and the number of major iterations (Iters) required by each of the algorithms discussed in the previous sections on the test sets. We implemented a version of BCGD [28] in which the blocks are chosen by the Gauss-Seidel rule, to compare with our methods. We also ran publicly available software for SPARSA [31], SLEP[3] [15], and SPOR-SPG[4] [4] on our test problems. Both SPARSA and SLEP solve the unconstrained formulation of the Group Lasso, while SPOR-SPG solves the constrained version. Since the SLEP solver has its core sub-routine (the Euclidean projection [14]) implemented in the C language, and so does SPOR-SPG for its one-norm projection [5], the Aprods serves as a better measure of performance than CPU times.

The stopping criterion $\|G_{\mathbf{t}}(x)\| = \|\frac{1}{\mathbf{t}}(x - p_{\mathbf{t}}(x))\| \leq \epsilon$ is suggested in [30] and can be applied to all ISTA/FISTA versions. We observed that $\epsilon = 10^{-2}$ was sufficient to recover the sparse solutions in our experiments. However, for comparison we used a different criterion which we explain below in Sections 7.3 and 7.4. The penalty parameter λ was chosen as $\gamma\lambda_{max}$, where λ_{max} is the upper bound derived in [17]. We chose $\gamma = 0.2$, which we found to yield reasonable level of group sparsity in general.

For the ease of recalling the names of the algorithms introduced in Sections 2 - 4, we summarize the abbreviated names in Table 1 below.

7.1 Synthetic Data Sets

We tested the algorithms on simulated problems of various sizes. Tables 2, 3, and 4 present the attributes of the Group Lasso and the MMV standard data sets respectively. Data sets yl1 to yl4 are from [33]; mgb1 and mgb2 are from [17]; ljy is adapted from a test set in [15]; nemirovski1-4 were created by Nemirovski and made challenging for the first-order methods, in the spirit of the worst case complexity examples. The MMV data sets in Table 4 are the ones in the online appendix of [4], and the comprehensive scalability test sets are adapted from the one in [25]. We refer the readers to Appendix A for the details of the simulated data sets.

[3] We ran only the Group Lasso experiments on SLEP.

[4] We ran the MMV experiments on SPOR-SPG [4] and the Group Lasso experiments on SPG in [5].

Abbrev. names	Algorithms
ISTA-BC	Algorithm 4.1
FISTA-BC	ISTA-BC with the FISTA update
ISTA-MS	ISTA with multi-scaling on step-lengths (Section 4.1)
BCD-GL	Algorithm 2.1
BCD-HYB	Hybrid version of BCD-GL and ISTA-BC (Section 4.3)
RBCD-GL	Randomized BCD-GL with probability distribution proportional to block-wise Lipschitz constants
RUCBD-GL	Randomized BCD-GL with uniform distribution
ISTA-RBC	Randomized ISTA-BC with uniform distribution
RBCD-HYB	Randomized BCD-HYB with uniform distribution

Table 1 Summary of the abbreviated names of the algorithms.

Data set	n	J	m (no. features)	Data type
yl1	50	15	15×3	categorical
yl2	100	10	4×2+6×4	categorical
yl3	100	16	16×2	continuous
yl4	500	100	50×3+50×2	mixed
mgb1	200	4	1+3×3	continuous
mgb2	100	251	1+250×4	continuous
ljy	1000	100	100×10	continuous
nemirovski1	1036	52	1036	continuous
nemirovski2	2062	103	2062	continuous
nemirovski3	2062	103	2062	continuous
nemirovski4	2062	207	4124	continuous

Table 2 Synthetic data sets. n = no. samples, J = no. of segments. The fourth column indicates the group sizes and the number of groups for each size.

7.2 Breast Cancer Data Set

To demonstrate the efficiency of our proposed algorithms on real-world data, we used the breast cancer data set [29] to compare their performance with that of existing algorithms on a regression task of predicting the patients' lengths of survival. This data set contains gene expression profiling data for 8141 genes in 295 breast cancer tumors. We considered only those genes with known symbols. There are various approaches in the literature for grouping the genes by their functions. We followed [11] and used the canonical pathways from MSigDB [24], which contains 880 gene groups. 878 out of these groups involve the genes studied in the breast cancer data set, accounting for 3510 genes. Note that the gene groups may overlap. However, by adopting the model proposed in [11], we were able to still solve a classical Group Lasso problem by augmenting the design matrix as in [11], increasing the total number of features to 28459.

7.3 Group Lasso experiments

To obtain a fair comparison of the computational performance among all the algorithms, we pre-computed a reference solution x^* for each problem and stopped the algorithms when the current

Data set	n	J	m (no. features)	Data type
yl1L	5000	50	47×10+3×1000	categorical
yl4L	2000	114	50×3+4×1000+50×3+10×50	mixed
mgb2L	2000	6	1+5×1000	continuous
ljyL	2000	15	5×200+10×500	continuous
glasso5	1000	400	3000	mixed
glassoL1	2000	242	10000	continuous
glassoL2	4000	438	20000	continuous

Table 3 Synthetic data sets. These test sets have much larger groups and more samples than the ones in Table 2.

Data set	A	K	No. nnz rows	ν	σ
mmv1a	60 × 100	5	12	0.01	$1.0\|R\|_F$
mmv1d	200 × 400	5	20	0.01	$1.2\|R\|_F$
mmv1e	200 × 400	10	20	0.02	$1.0\|R\|_F$
mmv1f	300 × 800	3	50	0.01	$0.9\|R\|_F$

Table 4 MMV data sets. The noisy measurement matrix $B = AX_0 + R$, where X_0 is the ground truth signal matrix and R is the noise matrix with $\|R\| = \nu\|AX_0\|_F$.

Data set	n (no. meas)	m (no. groups)	K (group size)	sparsity
spa	100	500	10	25:25:250
meas	100:100:1000	600	20	30
compreh	10,40:40:200	4n	2n	10% × m

Table 5 MMV scalability test data sets. The measurement matrices are noiseless. We adopt the Matlab syntax $a : b : c$ to represent the sequence from a to c with increments of b.

solution x satisfied $F(x) - F(x^*) \le 10^{-5}$. For SPG, we used x^* to compute the appropriate parameter value $\tau(= \|x^*\|_{1,2})$ required by the solver, and we set the projected gradient tolerance at 10^{-3}. The reference solutions were obtained by running ISTA-BC to the point where the relative change in the objective value is less than 10^{-10}. We also performed a convergence rate test on the breast cancer data set.

Since the matrix-vector multiplications performed by ISTA-BC, BCD-HYB, and BCGD involve only a block of the matrix A each time, e.g. $A_j^\top r(r \in \mathbb{R}^n)$ and $A_j x_j$, we treated each of these multiplications as a fraction of a unit Aprods. Hence, the Aprods counts for these algorithms were fractional.

7.4 The MMV experiments

For all the algorithms except SPOR-SPG, we used continuation for solving the MMV test problems. The problems in Table 4 contain noisy measurements. Hence, the noisy version (35) has to be solved. We used ISTA-BC to compute the reference solutions for the MMV data sets in Table 4 in the same way as for the Group Lasso experiments (with continuation), and we recorded the corresponding μ's for which the solution obtained to problem (36) satisfies $\|AX - B\|_F \leq \sigma$, for the given σ's in Table 4. We compared the performance of the algorithms in the same way as in the Group Lasso experiments. Besides the standard comparisons, we performed several scalability tests to see how the test statistics changed with variations in individual dimensions of the problem input data. A summary of the scalability test attributes is given in Table 5. The measurements are all noiseless. For these scalability tests, we allowed the algorithms to run until the objective function values $\|X\|_{1,2}$ converged to a relative accuracy of 10^{-5}. A reference solution is not required in this case. The results for the MMV tests are summarized in Figure 6.

The number of matrix-vector multiplications is counted in a slightly different way from that in the Group Lasso experiments. Here, the basic computations involved in ISTA-BC/FISTA-BC and BCD-MMV are $A_j^\top R$ for computing the gradient and $A_j x^j$ for the incremental residual update, while SPARSA and SPOR-SPG both have $A^\top R$ and AX as basic computations. Recall that A_j is the j-th column of A and x^j is the j-th row of X in the MMV case. Since $A^\top R = [A^\top R_1 \dots A^\top R_K]$, and $AX = [AX_1 \dots AX_K]$, each $A^\top R$ and AX actually involves K matrix-vector multiplications. So, Aprods $= K(\#(A^\top R) + \#(AX)) = \frac{K}{m}(\#(A_j^\top R) + \#(A_j x^j))$.

7.5 Results

Because of space consideration and the number of algorithms that we compared, we present our results using the performance profiles proposed by Dolan and Moré [8]. For any given algorithm $\mathcal{A}$, these profiles plot for each value of a positive factor τ the percentage of problems (from the given problem set) on which the performance of $\mathcal{A}$ is within a factor of τ of the best performance of any algorithm on this problem. Hence, when the plot of algorithm $\mathcal{A}$ is shifted more towards the left in the figure than another algorithm's plot, it indicates that algorithm $\mathcal{A}$ solves more problems quicker. If the plot is shifted more towards the top, this indicates that algorithm $\mathcal{A}$ is more robust.

We present the performance profiles for the Group Lasso experiments in Figure 1. We also present convergence plots in Figure 2. It should be noted that the eigen-decompositions and Newton's root-finding steps for BCD-GL are not reflected in Aprods. The cost of the two procedures is small compared to the total cost per iteration when the group sizes are small. For data sets with large groups, Aprods for BCD-HYB gives a more accurate representation of the CPU performance. Hence, we have included BCD-HYB instead in Figure 1 for better comparison. Figure 6 summarizes the MMV test results.

7.5.1 Group Lasso

The performance profile plots show that BCD-HYB and ISTA-BC clearly outperform FISTA and BCGD overall. BCD-HYB yields some of the best results in terms of Aprods and the number of iterations for the small data sets. The results for the large data sets are less clear-cut, but BCD-HYB is competitive against the other existing algorithms. This is evident from Figures 3 and 4.

Although ISTA-BC carries out a separate line-search for each segment of x, the considerable gain in computational performance often justifies the extra cost of the line-search. This is, however, not the case on the Nemirovski data sets, where ISTA-BC, ISTA-MS, and BCGD performed much worse than the rest of the group. It is interesting to observe that applying the FISTA step in the block coordinate case does not benefit computational performance on a number of Group Lasso data sets.

On the real-world breast cancer data set, we see from the convergence plot that BCD-HYB, ISTA-BC, and SPG are best and comparable to each other, with BCD-HYB and ISTA-BC having a marginal advantage (see also Figure 5). FISTA-BC, FISTA, and SLEP show roughly the same rate of convergence, while the remaining algorithms form another group exhibiting a different

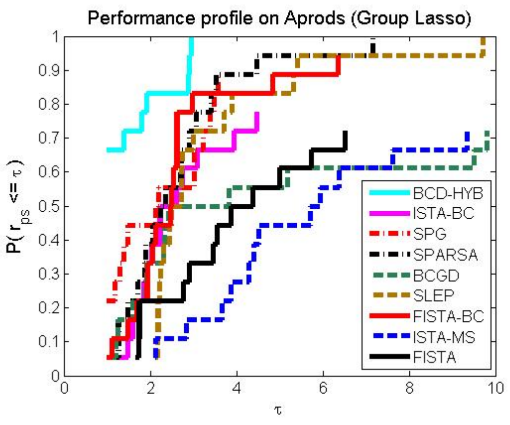


Fig. 1 Performance profiles graph [8] for the Group Lasso data sets.

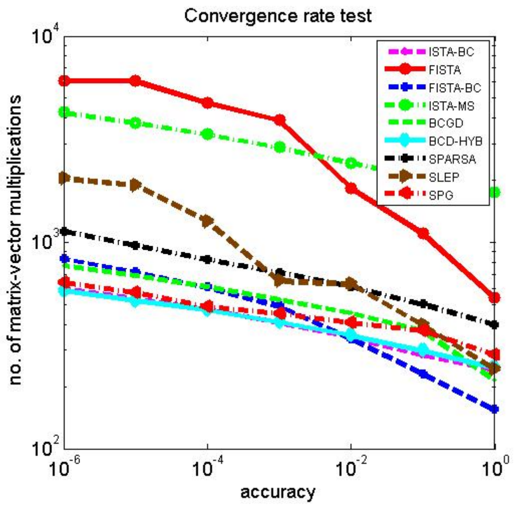


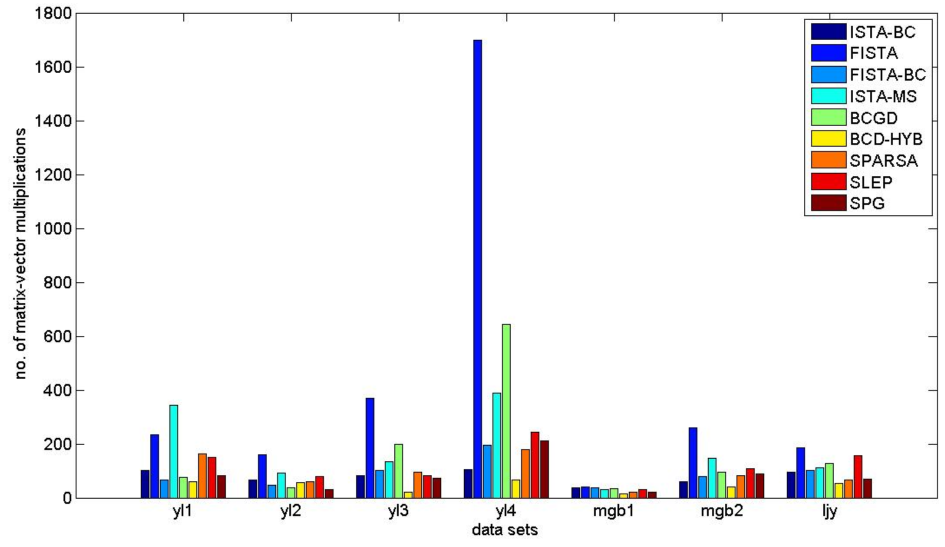
Fig. 2 Group Lasso convergence rate test on the breast cancer gene expression data set. The performances of BCD-HYB and ISTA-BC are almost identical.

convergence rate. Again, it is an interesting observation that the FISTA group actually exhibits a lower empirical rate of convergence.

We tested the randomized algorithms discussed in Section 4.4 on selected problems in Tables 2 and 3 as well as the breast cancer data set. From the results in Table 6, it appears that the randomized algorithms with a uniform distribution have no computational advantage over their original counterparts. RBCD-GL with a distribution where the probabilities are proportional to the corresponding group-wise Lipschitz constants shows better performance in terms of the number of iterations, but it has the same disadvantage of having to perform eigen-decomposition as BCD-GL on instances with large groups.

7.5.2 MMV

We did not include the results of SPARSA in Figure 6 because it consistently stopped at suboptimal solutions for the scalability tests. From Figure 6, it is clear that FISTA-BC and BCD-MMV have an



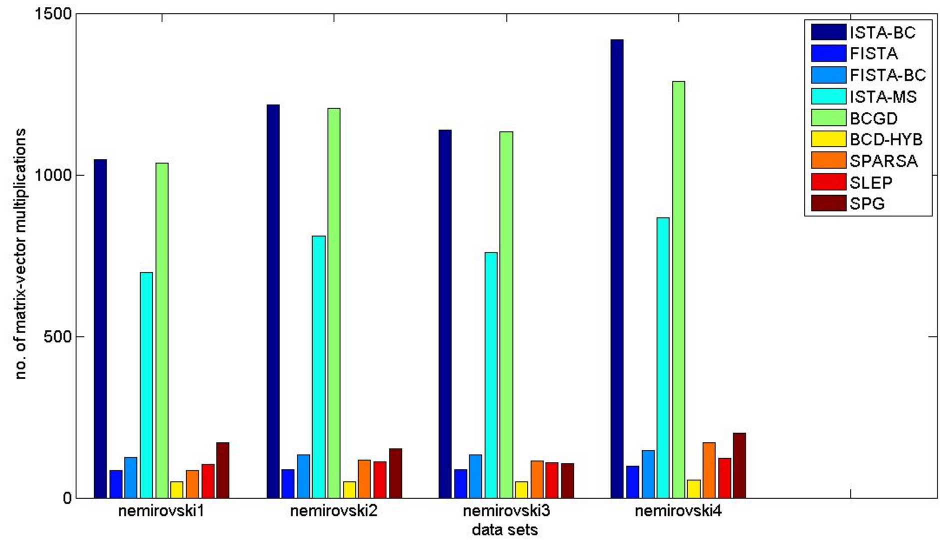


Fig. 3 Performance comparison for the small Group Lasso data sets.

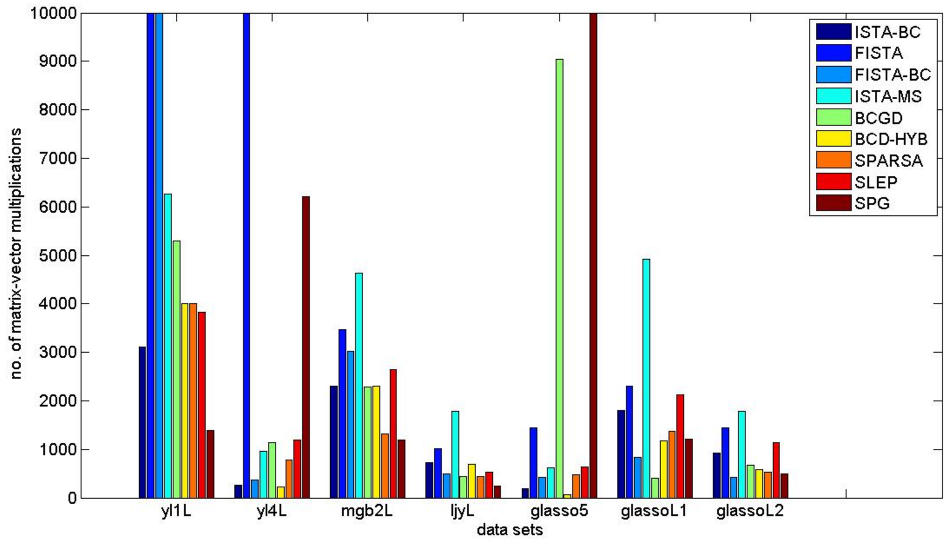


Fig. 4 Performance comparison for the large Group Lasso data sets. The bars corresponding to the algorithms that did not terminate within the maximum number of iterations are assigned to a value of 10,000.

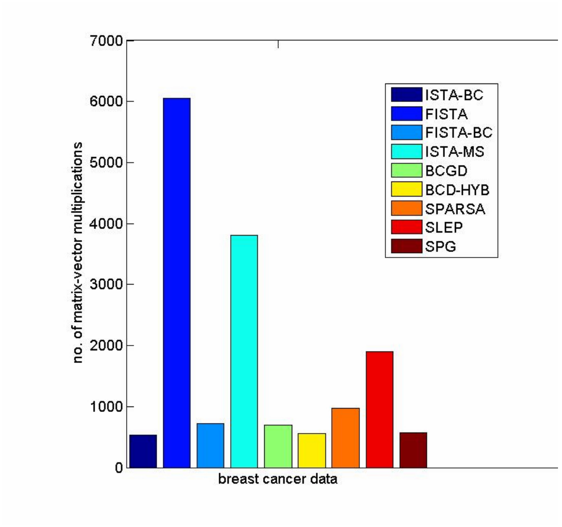


Fig. 5 Breast cancer data test results. The reference solution is computed by ISTA-BC. The target accuracy here is 10^{-5}.

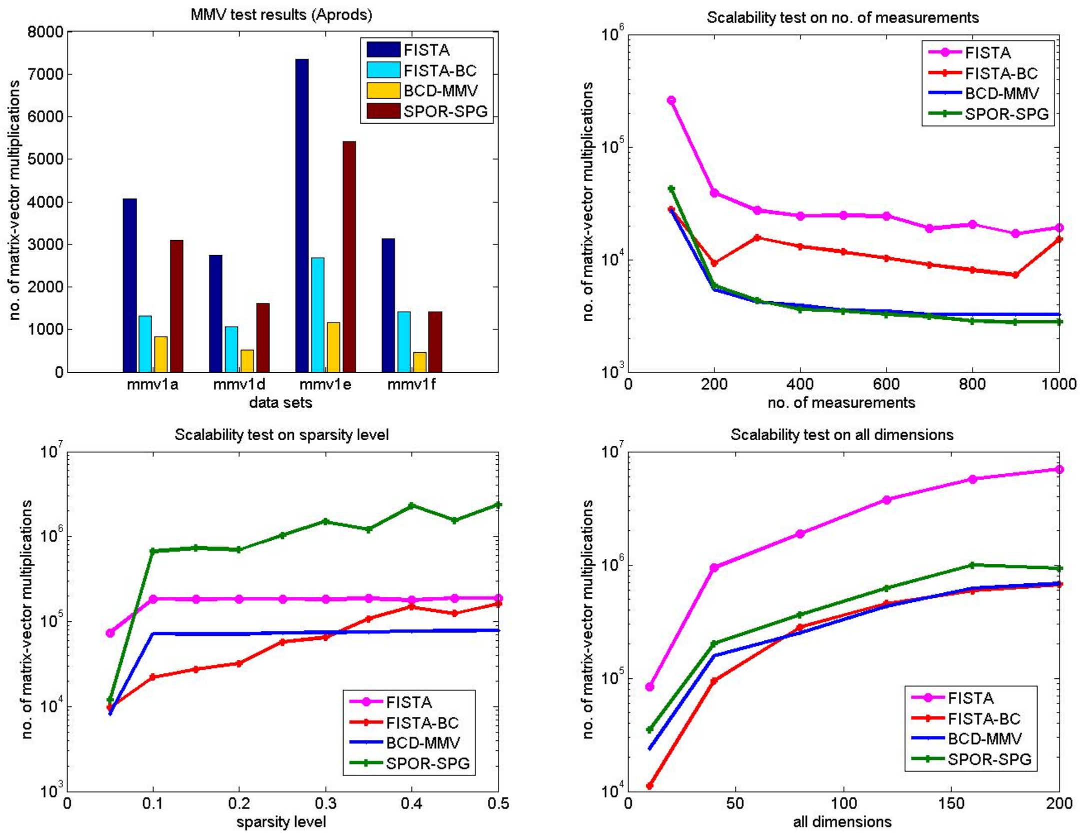


Fig. 6 MMV test results.

Data Sets	Algs	CPU (s)	Iters	Aprods
yl3	ISTA-BC	1.82e-001	20	8.30e+001
	BCD-HYB	1.02e-001	6	2.13e+001
	BCD-GL	3.29e-001	8	2.84e+001
	RBCD-GL	2.08e-001	15	4.97e+001
	RUBCD-GL	1.85e-001	16	5.28e+001
	ISTA-RBC	3.27e-001	20	8.13e+001
	RBCD-HYB	5.75e-001	10	3.34e+001
yl4	ISTA-BC	6.63e-001	27	1.02e+002
	BCD-HYB	5.21e-001	21	6.57e+001
	BCD-GL	5.01e-001	21	6.67e+001
	RBCD-GL	5.39e-001	15	5.45e+001
	RUBCD-GL	6.67e-001	32	9.97e+001
	ISTA-RBC	1.12e+000	33	1.26e+002
	RBCD-HYB	7.92e-001	30	9.27e+001
glassoL1	ISTA-BC	7.63e+001	451	1.79e+003
	BCD-HYB	5.86e+001	349	1.17e+003
	BCD-GL	5.26e+001	316	1.07e+003
	RBCD-GL	1.17e+002	544	1.90e+003
	RUBCD-GL	1.05e+002	604	1.96e+003
	ISTA-RBC	1.45e+002	810	3.20e+003
	RBCD-HYB	1.17e+002	665	2.17e+003
glassoL2	ISTA-BC	1.53e+002	232	9.31e+002
	BCD-HYB	9.41e+001	164	5.84e+002
	BCD-GL	9.97e+001	147	5.51e+002
	RBCD-GL	2.07e+002	268	9.74e+002
	RUBCD-GL	1.85e+002	285	9.74e+002
	ISTA-RBC	2.39e+002	355	1.42e+003
	RBCD-HYB	1.97e+002	307	1.04e+003
BreastCancerData	ISTA-BC	2.60e+001	156	5.38e+002
	BCD-HYB	2.59e+001	153	5.57e+002
	BCD-GL	2.44e+001	165	6.09e+002
	RBCD-GL	1.78e+001	44	3.00e+002
	RUBCD-GL	3.33e+001	202	7.32e+002
	ISTA-RBC	4.80e+001	241	8.34e+002
	RBCD-HYB	4.34e+001	214	7.57e+002

Table 6 Numerical results on selected problems for the randomized algorithms and their original counterparts. ISTA-RBC, RUBCD-GL, RBCD-HYB are the randomized versions of ISTA-BC, BCD-GL, and BCD-HYB respectively with a uniform distribution, and RBCD-GL is the randomized version of BCD-GL with a distribution where the probabilities are proportional to the group-wise Lipschitz constants $\{L_j\}_{j=1}^{J}$.

advantage over the FISTA in terms of Aprods. BCD-MMV is very competitive against SPOR-SPG and, in many cases, performed better. In the scalability test on sparsity level, however, both FISTA and BCD-MMV appear to be invariant to the increase in the sparsity level, while FISTA-BC and SPOR-SPG require increasing amount of computation.

8 Conclusion

We have proposed a general version of the BCD algorithm for the Group Lasso as well as an extension of ISTA based on variable step-lengths. The combination of the two yields an efficient and scalable method for solving the Group Lasso problems with various group sizes. As a special case, we have also considered the MMV problem and applied the ISTA-BC/BCD algorithm to it. Our numerical results show that the proposed algorithms compare favorably to current state-of-the-art methods in terms of computational performance.

Acknowledgements The authors would like to thank Shiqian Ma for valuable discussions on the MMV problems.

References

1. Bach, F.: Consistency of the group Lasso and multiple kernel learning. The Journal of Machine Learning Research **9**, 1179–1225 (2008)
2. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences **2**(1), 183–202 (2009)
3. van den Berg, E., Friedlander, M.: Joint-sparse recovery from multiple measurements. arXiv **904** (2009)
4. van den Berg, E., Friedlander, M.: Sparse Optimization With Least-squares Constraints. Tech. rep., Technical Report TR-2010-02, Department of Computer Science, University of British Columbia (2010)
5. van den Berg, E., Schmidt, M., Friedlander, M., Murphy, K.: Group sparsity via linear-time projection. Tech. rep., Technical Report TR-2008-09, Department of Computer Science, University of British Columbia (2008)
6. Candès, E., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. Information Theory, IEEE Transactions on **52**(2), 489–509 (2006)
7. Chen, J., Huo, X.: Theoretical results on sparse representations of multiple-measurement vectors. IEEE Transactions on Signal Processing **54**, 12 (2006)
8. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**(2), 201–213 (2002)
9. Donoho, D.: Compressed sensing. Information Theory, IEEE Transactions on **52**(4), 1289–1306 (2006)
10. Friedman, J., Hastie, T., Tibshirani, R.: A note on the group lasso and a sparse group lasso. preprint (2010)
11. Jacob, L., Obozinski, G., Vert, J.: Group Lasso with overlap and graph Lasso. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 433–440. ACM (2009)
12. Kim, D., Sra, S., Dhillon, I.: A scalable trust-region algorithm with application to mixed-norm regression. In: Int. Conf. Machine Learning (ICML), vol. 1 (2010)
13. Kim, S., Xing, E.: Tree-guided group lasso for multi-task regression with structured sparsity. In: Proceedings of the 27th Annual International Conference on Machine Learning (2010)
14. Liu, J., Ji, S., Ye, J.: Multi-task feature learning via efficient l 2, 1-norm minimization. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 339–348. AUAI Press (2009)
15. Liu, J., Ji, S., Ye, J.: SLEP: Sparse Learning with Efficient Projections. Arizona State University (2009)
16. Ma, S., Song, X., Huang, J.: Supervised group Lasso with applications to microarray data analysis. BMC bioinformatics **8**(1), 60 (2007)
17. Meier, L., Van De Geer, S., Buhlmann, P.: The group lasso for logistic regression. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **70**(1), 53–71 (2008)
18. Moré, J., Sorensen, D.: Computing a trust region step. SIAM Journal on Scientific and Statistical Computing **4**, 553 (1983)
19. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. CORE Discussion Papers (2010)
20. Nocedal, J., Wright, S.: Numerical optimization. Springer verlag (1999)
21. Rakotomamonjy, A.: Surveying and comparing simultaneous sparse approximation (or group-lasso) algorithms. Signal processing (2011)
22. Richtárik, P., Takáč, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Arxiv preprint arXiv:1107.2848 (2011)
23. Roth, V., Fischer, B.: The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In: Proceedings of the 25th international conference on Machine learning, pp. 848–855. ACM (2008)
24. Subramanian, A., Tamayo, P., Mootha, V., Mukherjee, S., Ebert, B., Gillette, M., Paulovich, A., Pomeroy, S., Golub, T., Lander, E., et al.: Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences of the United States of America **102**(43), 15,545 (2005)
25. Sun, L., Liu, J., Chen, J., Ye, J.: Efficient Recovery of Jointly Sparse Vectors. NIPS (2009)

26. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) **58**(1), 267–288 (1996)
27. Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. Journal of optimization theory and applications **109**(3), 475–494 (2001)
28. Tseng, P., Yun, S.: A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming **117**(1), 387–423 (2009)
29. Van De Vijver, M., He, Y., van't Veer, L., Dai, H., Hart, A., Voskuil, D., Schreiber, G., Peterse, J., Roberts, C., Marton, M., et al.: A gene-expression signature as a predictor of survival in breast cancer. New England Journal of Medicine **347**(25), 1999 (2002)
30. Vandenberghe, L.: Gradient methods for nonsmooth problems. EE236C course notes (2008)
31. Wright, S., Nowak, R., Figueiredo, M.: Sparse reconstruction by separable approximation. IEEE Transactions on Signal Processing **57**(7), 2479–2493 (2009)
32. Yang, H., Xu, Z., King, I., Lyu, M.: Online learning for group lasso. In: 27th Intl Conf. on Machine Learning (ICML2010). Citeseer
33. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **68**(1), 49–67 (2006)
34. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**(2), 301–320 (2005)

A Simulated Data Sets

For the specifications of the data sets in Table 2, interested readers can refer to the references provided at the beginning of Section 7.1. Here, we provide the details of the data sets in Table 3.

A.1 yl1L

50 latent variables $Z_1, \dots, Z_{50}$ are simulated from a centered multivariate Gaussian distribution with covariance between Z_i and Z_j being $0.5^{|i-j|}$. The first 47 latent variables are encoded in $\{0, \dots, 9\}$ according to their inverse cdf values as done in [33]. The last three variables are encoded in $\{0, \dots, 999\}$. Each latent variable corresponds to one segment and contributes L columns in the design matrix with each column j containing values of the indicator function $I(Z_i = j)$. L is the size of the encoding set for Z_i. The responses are a linear combination of a sparse selection of the segments plus a Gaussian noise. We simulate 5000 observations.

A.2 yl4L

110 latent variables are simulated in the same way as the third data set in [33]. The first 50 variables contribute 3 columns each in the design matrix A with the i-th column among the three containing the i-th power of the variable. The next 50 variables are encoded in a set of 3, and the final 10 variables are encoded in a set of 50, similar to yl1L. In addition, 4 groups of 1000 Gaussian random numbers are also added to A. The responses are constructed in a similar way as in yl1L. 2000 observation are simulated.

A.3 mgb2L

5001 variables are simulated as in yl1L without categorization. They are then divided into six groups, with first containing one variable and the rest containing 1000 each. The responses are constructed in a similar way as in yl1L. We collect 2000 observations.

A.4 ljyL

We simulate 15 groups independent standard Gaussian random variables. The first five groups are of a size 5 each, and the last 10 groups contain 500 variables each. The responses are constructed in a similar way as in yl1L, and we simulate 2000 observations.

A.5 glassoL1,glassoL2

The design matrix A is drawn from the standard Gaussian distribution. The length of each group is sampled from the uniform distribution $U(10, 50)$ with probability 0.9 and from the uniform distribution $U(50, 300)$ with probability 0.1. We continue to generate the groups until the specified number of features m is reached. The ground truth feature coefficients x^* is generated from $\mathcal{N}(2, 4)$ with approximately 5% non-zero values, and we assume a $\mathcal{N}(0.5, 0.25)$ noise.