Fast Block Coordinate Descent for Non-Convex Group Regularizations

Yasutoshi Ida[†] **Sekitoshi Kanai**[†] **Atsutoshi Kumagai**[†]
[†]NTT Computer and Data Science Laboratories

Abstract

Non-convex sparse regularizations with group structures are useful tools for selecting important feature groups. For optimization with these regularizations, block coordinate descent (BCD) is a standard solver that iteratively updates each parameter group. However, it suffers from high computation costs for a large number of parameter groups. The state-of-the-art method prunes unnecessary updates in BCD by utilizing bounds on the norms of the parameter groups. Unfortunately, since it computes the bound for each iteration, the computation cost still tends to be high when the updates are not sufficiently pruned. This paper proposes a fast BCD for non-convex group regularizations. Specifically, it selects a small subset of the parameter groups from all the parameter groups on the basis of the bounds and performs BCD on the subset. The subset grows step by step in accordance with the bounds during optimization. Since it computes the bounds only when selecting and growing the subsets, the total cost for computing the bounds is smaller than in the previous method. In addition, we theoretically guarantee the convergence of our method. Experiments show that our method is up to four times faster than the state-of-the-art method and 68 times faster than the original BCD without any loss of accuracy.

1 INTRODUCTION

Sparse optimization problems with group regularizations are useful for selecting important feature groups. Among the various methods, such as Group Lasso (Yuan and Lin, 2006), Group SCAD (smoothly clipped absolute deviation) (Wang et al., 2007) and Group MCP (minimax concave penalty) (Breheny and Huang, 2009; Huang et al., 2012) have received attention due to their theoretical foundations and desirable properties. Specifically, they utilize non-convex group regularizations and overcome the theoretical drawbacks of the statistical properties of convex ones, such as bias and inconsistency of the estimated parameters (Zhang, 2010; Mazumder et al., 2010; Loh and Wainwright, 2017). Owing to these advantageous properties, they have been successfully used in various domains, including bioinformatics to analyze gene expression data/genetic markers (Wang et al., 2007; Breheny and Huang, 2009; Ayers and Cordell, 2010; Ogutu and Piepho, 2014) and for causal inference (Greenewald et al., 2021).

Although we can use several methods to solve the optimization problems of Group SCAD/MCP (Wang et al., 2007; Bühlmann and Meier, 2008), block coordinate descent (BCD) is currently considered to be the standard iterative method for its high efficiency (Breheny and Huang, 2015). It utilizes a thresholding function derived from the regularization terms, and the thresholding function shrinks the parameter groups. As a result, a lot of parameter groups are expected to be zeros and a sparse solution can be obtained. In the thresholding function, the magnitude of shrinkage for each parameter group is determined in accordance with the norm of the parameter group. Intuitively, since parameter groups with small norms do not contribute much to predictions, the thresholding function imposes a large shrinkage on such groups. Unfortunately, BCD tends to become slower as the number of parameter groups increases because it iteratively updates each parameter group on the basis of the thresholding function.

To overcome the above problem, bounding techniques (Ida et al., 2019, 2020) skip unnecessary updates in BCD on the basis of bounds on the norms of the parameter groups. Since the thresholding function shrinks the parameter groups to zeros in accordance with the norms, the bounds of these norms can identify parameter groups that become zeros. If the sizes of a parameter group is p_g, the computation cost of the bound is $\mathcal{O}(p_g)$ time (Ida et al., 2019). On the other hand, the update requires $\mathcal{O}(pp_g + p_g^2)$ time, where p is the total number of parameters. Therefore, these bounds efficiently skip the updates for the parameter groups that turn out to be zeros. However, since these bounding techniques compute a bound for each iteration of

Proceedings of the 26[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

BCD until convergence, the computation cost still tends to be high when the updates are not sufficiently skipped, e.g. when many parameter groups turn out to be nonzero during optimization.

This paper proposes a fast BCD for non-convex group regularizations. To prune unnecessary parameter groups, it first selects a small subset of parameter groups from all the parameter groups and performs BCD on the subset. The parameters in the subset have a small amount of shrinkage since such parameters largely contribute to predictions (Fujiwara et al., 2016a). To find such a subset, we utilize a novel bounding technique that effectively deals with thresholding functions of non-convex group regularizations. The subset grows step by step during optimization on the basis of the bounds. This strategy avoids the limitation of the existing bounding techniques: our method computes bounds only when selecting and growing the subset whereas the existing techniques compute a bound for each iteration. As a result, our method can reduce the cost for computing the bound. In addition, we theoretically guarantee the convergence of our method. Experiments show that our method reduces the number of the computations for the bounds by a factor of 10^4 to 10^5 compared with the existing method. We also demonstrate that our method is up to four times faster than the state-of-the-art method and 68 times faster than the original BCD without any loss of accuracy.

2 PRELIMINARY

This section reviews Group SCAD, the thresholding function, and the bounding technique. We will mainly handle Group SCAD for simplicity; the extension to Group MCP is described in Section 4.

2.1 Group SCAD

Let $\overline{X} \in \mathbb{R}^{n \times p}$ be a matrix of features, where n is the number of data points and each data point is represented as a p-dimensional feature. $y \in \mathbb{R}^n$ is a set of continuous responses. Group SCAD (Wang et al., 2007) finds a set of parameter groups that are related to the response from the dataset where n is much smaller than p. If G is the total number of feature groups, $\overline{X}$ is represented as $\overline{X} = [\overline{X}^{(1)}, ..., \overline{X}^{(G)}]$, where $\overline{X}^{(g)} \in \mathbb{R}^{n \times p_g}$ is the submatrix of $\overline{X}$ corresponding to the g-th feature group with p_g features. Here, in accordance with the previous work (Breheny and Huang, 2015), each group is orthonormalized as $X^{(g)} = \overline{X}^{(g)} Q^{(g)} (\Lambda^{(g)})^{-1/2}$ where $Q^{(g)}$ and $\Lambda^{(g)}$ are obtained by singular value decomposition: $n^{-1}(\overline{X}^{(g)})^\top \overline{X}^{(g)} = Q^{(g)} \Lambda^{(g)} (Q^{(g)})^\top$. In the same way, a parameter vector $\beta \in \mathbb{R}^p$ is represented as $\beta = [\beta^{(1)\top}, ..., \beta^{(G)\top}]^\top$, where $\beta^{(g)} \in \mathbb{R}^{p_g}$ is the parameter vector of group g. The solution of Group SCAD is obtained by solving the following problem:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \left\| y - \sum_{g=1}^G X^{(g)} \beta^{(g)} \right\|_2^2 + \sum_{g=1}^G p_{\lambda,\gamma}(\|\beta^{(g)}\|_2), \quad (1)$$

where $\| \cdot \|_2$ represents the l_2 norm and the regularization term $p_{\lambda,\gamma}(\|\beta^{(g)}\|_2)$ is defined as follows (Wang et al., 2007):

$$p_{\lambda,\gamma}(\|\beta^{(g)}\|_2) = \begin{cases} \lambda \|\beta^{(g)}\|_2 & \text{if } \|\beta^{(g)}\|_2 \le \lambda \\ \frac{\gamma\lambda\|\beta^{(g)}\|_2 - 0.5(\|\beta^{(g)}\|_2^2 + \lambda^2)}{\gamma - 1} & \text{if } \lambda < \|\beta^{(g)}\|_2 \le \gamma\lambda \\ \frac{\lambda^2(\gamma^2 - 1)}{2(\gamma - 1)} & \text{if } \|\beta^{(g)}\|_2 > \gamma\lambda. \end{cases} \quad (2)$$

$\lambda > 0$ is a hyperparameter that controls the degree of sparsity of the solution. $\gamma > 2$ is another one that determines the magnitude of the regularization. The recommended setting is $\gamma = 3.7$ (Fan and Li, 2001).

2.2 Thresholding Function

To obtain a solution to problem (1), the solver usually utilizes a thresholding function (Fan and Li, 2001; Breheny and Huang, 2015; Gong et al., 2013), which is derived from the regularization of Equation (2). Since it shrinks parameter groups to zeros, a sparse solution can be obtained by using solvers with the thresholding function. For the g-th group, the thresholding function $F(z^{(g)}, \lambda, \gamma)$ of Group SCAD is computed as

$$F(z^{(g)}, \lambda, \gamma) = \begin{cases} S(\|z^{(g)}\|_2, \lambda) \frac{z^{(g)}}{\|z^{(g)}\|_2} & \text{if } \|z^{(g)}\|_2 \le 2\lambda \\ \frac{\gamma - 1}{\gamma - 2} S(\|z^{(g)}\|_2, \frac{\gamma\lambda}{\gamma - 1}) \frac{z^{(g)}}{\|z^{(g)}\|_2} & \text{if } 2\lambda < \|z^{(g)}\|_2 \le \gamma\lambda \\ z^{(g)} & \text{if } \gamma\lambda < \|z^{(g)}\|_2. \end{cases} \quad (3)$$

$z^{(g)} \in \mathbb{R}^{p_g}$ in the function is computed as follows:

$$z^{(g)} = X^{(g)\mathrm{T}} (y - \sum_{i \neq g}^G \mathbf{X}^{(i)} \beta^{(i)}). \quad (4)$$

$S(\cdot, \cdot)$ is computed as $S(\|z^{(g)}\|_2, \lambda) = \max(0, \|z^{(g)}\|_2 - \lambda)$. By using Equation (3), the g-th updated parameter vector is represented as $\beta^{(g)} = F(z^{(g)}, \lambda, \gamma)$. Equation (3) suggests that the magnitude of shrinkage of $\beta^{(g)}$ depends on the magnitude of the norm of $z^{(g)}$; e.g., $\beta^{(g)}$ has no shrinkage for the case of $\gamma\lambda < \|z^{(g)}\|_2$, small shrinkage for $2\lambda < \|z^{(g)}\|_2 \le \gamma\lambda$, and large shrinkage for $\|z^{(g)}\|_2 \le 2\lambda$. Specifically, the parameter groups with smaller shrinkage contribute more to predictions.

2.3 BCD and Bounding Technique

Algorithm 1 is a BCD solver that uses the thresholding function of Equation (3) (Breheny and Huang, 2015). It is implemented in the *grpreg* package of R[1], and it has been

[1] https://cran.r-project.org/web/packages/grpreg/index.html

Algorithm 1 BCD of Group SCAD

1: $\mathbb{A} = \{1, ..., G\}$, $\beta_0 \leftarrow \mathbf{0}$, $t \leftarrow 0$
2: **repeat**
3: **for** each $g \in \mathbb{A}$ **do**
4: compute $z_t^{(g)}$ by Equation (4);
5: $\beta_t^{(g)} \leftarrow F(z_t^{(g)}, \lambda, \gamma)$ by Equation (3);
6: **end for**
7: $t \leftarrow t + 1$
8: **until** β_t converges

used as a standard solver for Group SCAD/MCP because of its high efficiency. If $\mathbb{A} := \{1, ..., G\}$ and t is the number of the outer loops (lines 2–8), the inner loop (lines 3–6) updates $\beta_t^{(g)}$ for each group by using Equations (3) and (4). The outer loop (lines 2–8) performs the inner loop until β_t converges. Unfortunately, BCD tends to slow down when the number of groups $|\mathbb{A}|$ is large because the total number of iterations becomes large.

To accelerate BCD, the bounding technique (Ida et al., 2019) skips unnecessary computations of lines 4 and 5 in Algorithm 1 by utilizing the upper bound $\overline{z}_t^{(g)}$ of $\|z_t^{(g)}\|_2$ such that $\overline{z}_t^{(g)} \geq \|z_t^{(g)}\|_2$. In particular, we can skip lines 4 and 5 when $\overline{z}_t^{(g)} \leq \lambda$ holds because $S(\cdot, \cdot)$ in Equation (3) clearly shrinks such parameter groups to zeros. The point is that the computation cost of $\overline{z}_t^{(g)}$ is $\mathcal{O}(p_g)$ time, whereas that of $\|z_t^{(g)}\|_2$ requires $\mathcal{O}(np_g)$ or $\mathcal{O}(pp_g + p_g^2)$ time. As a result, it efficiently skips lines 4 and 5 when $\overline{z}_t^{(g)} \leq \lambda$ holds. However, the method computes $\overline{z}_t^{(g)}$ for each iteration and must compute both $\overline{z}_t^{(g)}$ and $\|z_t^{(g)}\|_2$ when $\overline{z}_t^{(g)} \leq \lambda$ does not hold. Therefore, its cost for computing $\overline{z}_t^{(g)}$ may be dominant when $\overline{z}_t^{(g)} \leq \lambda$ does not hold for many parameter groups. If T is the total number of the outer loops for lines 2–8 in Algorithm 1, it requires $\mathcal{O}(Tp)$ time in addition to the computation cost of the original BCD when $\overline{z}_t^{(g)} \leq \lambda$ does not hold for all the groups. Specifically, if the parameters are not very sparse, it is slower than the original BCD since it cannot skip the computations of the nonzero parameter groups.

3 PROPOSED APPROACH

This section presents our approach. First, we overview our ideas in Section 3.1. Next, we introduce our key techniques of our method in Sections 3.2, 3.3, and 3.4. Finally, we describe our algorithm in Section 3.5. The omitted proofs can be found in Appendix.

3.1 Idea

Although the bounding technique can accelerate BCD, it tends to be slow in some cases, as described in Section 2.3. This is because it computes an upper bound $\overline{z}_t^{(g)}$ of $\|z_t^{(g)}\|_2$ for each iteration to skip parameter groups that become zeros, as shown in Figure 1 (b). As a result, the cost for computing $\overline{z}_t^{(g)}$ is dominant when the parameter groups are not very sparse, as it cannot skip the nonzero parameter groups.

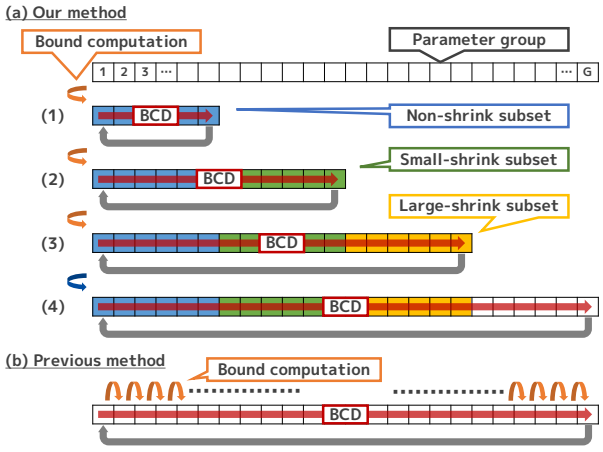


Figure 1: Procedures of our method and previous method.

To overcome these problems, we propose a method that reduces the number of the computations for $\overline{z}_t^{(g)}$. It performs BCD on a growing sequence of subproblems that are restricted to small subsets of parameter groups as a way to prune unnecessary parameter groups. To find these subsets, we utilize a novel bounding technique based on thresholding functions of non-convex group regularizations. Our method computes bounds only when selecting and growing the subsets, whereas the existing method computes them at every iteration. As a result, our method only requires $\mathcal{O}(G^2 + p)$ time for computing bounds where $G \ll p$ for many use cases, while the previous bounding technique requires $\mathcal{O}(Tp)$ time.

Figure 1 (a) is an overview of our method. It selects three types of subset: (i) a non-shrink subset, (ii) a small-shrink subset, and (iii) a large-shrink subset. These subsets are derived from the thresholding function of Equation (3). We start the optimization with the non-shrink subset. Then we add the small-shrink and large-shrink subsets to the problem step by step. Finally, we optimize all the parameter groups until convergence.

3.2 Subsets of Parameter Groups

This section describes the three types of subset shown in Figure 1 (a). Let $\overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)}$ be upper and lower bounds of $\|z_t^{(g)}\|_2$ such that $\overline{z}_t^{(g)} \geq \|z_t^{(g)}\|_2$ and $\underline{z}_t^{(g)} \leq \|z_t^{(g)}\|_2$ hold, respectively. These bounds are given in Section 3.3. We define the three types of subset by utilizing these bounds.

Definition 1. *The subsets* $\mathbb{N}$, $\mathbb{S}$ *and* $\mathbb{L}$ *are given as follows:*

$$\mathbb{N} = \{g \in \mathbb{A} | \underline{z}_t^{(g)} > \gamma\lambda\}, \quad (5)$$

$$\mathbb{S} = \{g \in \mathbb{A} | (\gamma\lambda \geq \overline{z}_t^{(g)}) \wedge (\underline{z}_t^{(g)} > 2\lambda)\}, \quad (6)$$

$$\mathbb{L} = \{g \in \mathbb{A} | (2\lambda \geq \overline{z}_t^{(g)}) \wedge (\underline{z}_t^{(g)} > \lambda)\}. \quad (7)$$

The subsets $\mathbb{N}$, $\mathbb{S}$, and $\mathbb{L}$ have the following properties:

Lemma 1 (Non-Shrink Subset $\mathbb{N}$)**.** *Each* $\beta_t^{(g)}$ *such that* $g \in \mathbb{N}$ *must be* $z_t^{(g)}$.

Lemma 2 (Small-Shrink Subset $\mathbb{S}$)**.** *Each* $\beta_t^{(g)}$ *such that* $g \in \mathbb{S}$ *must be* $\frac{\gamma-1}{\gamma-2} S(\|z_t^{(g)}\|_2, \frac{\gamma\lambda}{\gamma-1}) \frac{z_t^{(g)}}{\|z_t^{(g)}\|_2}$.

Lemma 3 (Large-Shrink Subset $\mathbb{L}$)**.** *Each* $\beta_t^{(g)}$ *such that* $g \in \mathbb{L}$ *must be* $(\|z_t^{(g)}\|_2 - \lambda) \frac{z_t^{(g)}}{\|z_t^{(g)}\|_2}$.

Lemma 1 suggests that subset $\mathbb{N}$ contains parameter groups that do not shrink. On the other hand, Lemmas 2 and 3 show that subsets $\mathbb{S}$ and $\mathbb{L}$ contain parameter groups that shrink to $\frac{\gamma-1}{\gamma-2} S(\|z_t^{(g)}\|_2, \frac{\gamma\lambda}{\gamma-1}) \frac{z_t^{(g)}}{\|z_t^{(g)}\|_2}$ and $(\|z_t^{(g)}\|_2 - \lambda) \frac{z_t^{(g)}}{\|z_t^{(g)}\|_2}$, respectively. Specifically, the parameter groups in subset $\mathbb{S}$ shrink more than the parameter groups in $\mathbb{N}$. In the same way, the parameter groups in $\mathbb{L}$ shrink more than those in $\mathbb{N}$ and $\mathbb{S}$. We should note that our method also deals with subsets not included in $\mathbb{N}$, $\mathbb{S}$ and $\mathbb{L}$ in Section 3.5.

3.3 Upper and Lower Bounds

This section describes a way of computing upper and lower bounds of $\|z_t^{(g)}\|_2$ that appear in Equations (5), (6), and (7) for selecting $\mathbb{N}$, $\mathbb{S}$, and $\mathbb{L}$. First, we introduce $\overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)}$, as follows:

Definition 2. *Let* t' *be* $0 \leq t' < t$. *Then,* $\overline{z}_t^{(g)}$ *and* $\underline{z}_t^{(g)}$ *are computed as*

$$\overline{z}_t^{(g)} = \|z_{t'}^{(g)}\|_2 + \textstyle\sum_{l \in \mathbb{A} \setminus \{g\}} \Lambda(g, l), \quad (8)$$

and

$$\underline{z}_t^{(g)} = \|z_{t'}^{(g)}\|_2 - \textstyle\sum_{l \in \mathbb{A} \setminus \{g\}} \Lambda(g, l), \quad (9)$$

where $\Lambda(g,l) = \|\hat{K}^{(g)}[l]\|_2 \|\beta_t^{(l)} - \beta_{t'}^{(l)}\|_2$. *The* i*-th element of* $\hat{K}^{(g)}[l] \in \mathbb{R}^{p_g}$ *is given as* $\|K^{(g,l)}[i,:]\|_2$, *that is, the* l_2 *norm of the* i*-th row vector in the block matrix* $K^{(g,l)} \in \mathbb{R}^{p_g \times p_l}$ *of* $K := X^{\mathrm{T}} X \in \mathbb{R}^{p \times p}$.

We call $z_{t'}^{(g)}$ and $\beta_{t'}^{(g)}$ snapshots. Note that we can precompute $\|\hat{K}^{(g)}[\cdot]\|_2$ before beginning optimization. The following lemma shows that $\overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)}$ are, respectively, upper and lower bounds of $\|z_t^{(g)}\|_2$:

Lemma 4 (Upper and Lower Bounds)**.** *When* $\overline{z}_t^{(g)}$ *and* $\underline{z}_t^{(g)}$ *are respectively computed using Equations (8) and (9) in Definition 2,* $\overline{z}_t^{(g)} \geq \|z_t^{(g)}\|_2$ *and* $\|z_t^{(g)}\|_2 \geq \underline{z}_t^{(g)}$ *hold.*

On the basis of Lemma 4, we can estimate to which subset of $\mathbb{N}$, $\mathbb{S}$, and $\mathbb{L}$ the g-th parameter group belongs in accordance with Equations (5), (6), and (7). The computation cost of selecting these subsets is as follows:

Lemma 5 (Computation Cost for Subset Selection)**.** *The computation of selecting any subset of* $\mathbb{N}$, $\mathbb{S}$, *and* $\mathbb{L}$ *requires* $\mathcal{O}(G^2 + p)$ *time given snapshots* $z_{t'}^{(g)}$, $\beta_{t'}^{(g)}$, *and precomputed* $\|\hat{K}^{(g)}[\cdot]\|_2$ *for* $g \in \{1, ..., G\}$.

Our method requires $\mathcal{O}(G^2 + p)$ time while the existing bounding technique (Ida et al., 2019) requires $\mathcal{O}(Tp)$ time, where T is the number of the outer loops. Since T is often large in BCD and $G \ll p$ usually holds for many use cases, our method is expected to be more efficient than the existing method.

Regarding the accuracy of the subset selection, since these upper and lower bounds are approximations for $\|z_t^{(g)}\|_2$, the error bounds of the approximations are important. If $|\cdot|$ represents the absolute value, they are given as follows:

Lemma 6 (Error Bound)**.** *Let* ϵ *be computed as* $2\sum_{l \in \mathbb{A} \setminus \{g\}} \Lambda(g, l)$ *for the* g*-th group. Then, it satisfies* $|\overline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq \epsilon$ *and* $|\underline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq \epsilon$.

If the error bounds become large, the subset selection becomes inefficient because the inequalities in Equations (5), (6), and (7) are difficult to satisfy. The next section shows that the gradual parameter addition strategy can reduce the error bound and further decreases the computation cost.

3.4 Gradual Parameter Addition

Our method starts by optimizing parameters in subset $\mathbb{N}$ because it largely contributes to predictions. Then, subsets $\mathbb{S}$ and $\mathbb{L}$ are added to the optimization problem step by step. That is, we first extract subset $\mathbb{N}$ from $\mathbb{A} := \{1, ..., G\}$ and optimize the parameters in $\mathbb{N}$. After convergence, subset $\mathbb{S}$ is selected from $\mathbb{A} \setminus \mathbb{N}$ and added to the optimization problem. Then, we again optimize the problem and extract subset $\mathbb{L}$ from $\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})$ to be added to the problem.

The advantage of the strategy is that we can expect to obtain a small error bound ϵ in Lemma 6 for extracting $\mathbb{S}$ and $\mathbb{L}$. The error bounds are given as follows:

Lemma 7 (Error Bounds for Subsets $\mathbb{S}$ and $\mathbb{L}$)**.** *Let* ϵ_S *and* ϵ_L *be computed as* $2\sum_{l \in \mathbb{N} \setminus \{g\}} \Lambda(g, l)$ *and* $2\sum_{l \in (\mathbb{N} \cup \mathbb{S}) \setminus \{g\}} \Lambda(g, l)$, *respectively. Then, if we utilize the gradual parameter addition strategy, we have* $|\overline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq \epsilon_S$ *and* $|\underline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq \epsilon_S$ *for the* g*-th group in subset* $\mathbb{A} \setminus \mathbb{N}$, *and* $|\overline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq \epsilon_L$ *and* $|\underline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq \epsilon_L$ *for the* g*-th group in subset* $\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})$.

We can expect ϵ_S and ϵ_L to be smaller than ϵ in Lemma 6 because $\Lambda(\cdot,\cdot) \geq 0$, $|\mathbb{A}| \geq |\mathbb{N}|$, and $|\mathbb{A}| \geq |\mathbb{N} \cup \mathbb{S}|$. That is, the upper and lower bounds can approximate $\|z_t^{(g)}\|_2$ with small error bounds for extracting $\mathbb{S}$ and $\mathbb{L}$.

The another advantage of the strategy is that it can further reduce the computation cost of extracting $\mathbb{S}$ and $\mathbb{L}$. The computation costs are as follows:

Lemma 8 (Costs for Extracting $\mathbb{S}$ and $\mathbb{L}$)**.** *If we utilize the gradual parameter addition strategy, the selections of subsets $\mathbb{S}$ and $\mathbb{L}$ respectively require $\mathcal{O}(|\mathbb{N}||\mathbb{A} \setminus \mathbb{N}| + p)$ and $\mathcal{O}(|\mathbb{N} \cup \mathbb{S}||\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})| + p)$ times given snapshots $z_{t'}^{(g)}$, $\beta_{t'}^{(g)}$, and precomputed $\|\hat{K}^{(g)}[\cdot]\|_2$ for $g \in \{1, ..., G\}$.*

From Lemma 8, the selections of $\mathbb{S}$ and $\mathbb{L}$ require fewer computations than that of Lemma 5, because $|\mathbb{N}|$, $|\mathbb{A} \setminus \mathbb{N}|$, $|\mathbb{N} \cup \mathbb{S}|$, and $|\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})|$ are less than or equal to G.

3.5 Algorithm

This section describes our fast BCD, which is called AccelGSCAD (Accelerated Group SCAD). Algorithm 2 is the pseudo-code. $\mathbb{C}$ is a subset that contains indices of groups optimized by BCD. Algorithm 2 first precomputes $\|\hat{K}^{(g)}[l]\|_2$, which is used for computing bounds (lines 2–6). Next, it extracts $\mathbb{N}$ from $\mathbb{A}$ by using Equation (5) (lines 9–17). To select $\mathbb{N}$, we need $\beta_{t'}$ and $z_{t'}$. Therefore, BCD of Algorithm 3 is executed for m iterations to obtain them (line 10). $\mathbb{N}$ is added to $\mathbb{C}$ (line 17), and Algorithm 3 optimizes the parameters corresponding to $\mathbb{C}$ until convergence (line 38). $\beta_{t'}$ and $z_{t'}$ are also updated by Algorithm 3 in order to select other subsets. Then, $\mathbb{S}$ is extracted from $\mathbb{A} \setminus \mathbb{N}$ by using Equation (6) (lines 18–25) and it is added to $\mathbb{C}$ (line 25). Algorithm 3 with $\mathbb{C}$ is executed until convergence (line 38). In the same way, $\mathbb{L}$ is extracted from $\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})$ by using Equation (7) (lines 26–33) and the parameters are optimized (line 38). Then, $\mathbb{A} \setminus \mathbb{C}$ is added to $\mathbb{C}$ (line 35) and the parameters are optimized (line 38).

As a result, the computation cost of selecting the subsets in the case of Algorithm 2 is as follows:

Theorem 1 (Computation Cost)**.** *The total computation cost of selecting the subsets in the case of Algorithm 2 is $\mathcal{O}(G^2 + |\mathbb{N}||\mathbb{A} \setminus \mathbb{N}| + |\mathbb{N} \cup \mathbb{S}||\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})| + p)$ time given snapshots $z_{t'}^{(g)}$, $\beta_{t'}^{(g)}$ and precomputed $\|\hat{K}^{(g)}[\cdot]\|_2$ for $g \in \{1, ..., G\}$.*

Obviously, we obtain the following worst time complexity:

Corollary 1 (Worst-case Computation Cost)**.** *The worst time complexity of selecting the subsets in the case of Algorithm 2 is $\mathcal{O}(G^2 + p)$ time given snapshots $z_{t'}^{(g)}$, $\beta_{t'}^{(g)}$ and precomputed $\|\hat{K}^{(g)}[\cdot]\|_2$ for $g \in \{1, ..., G\}$.*

The existing BCD with the bounding technique requires $\mathcal{O}(Tp)$ time for computing bounds (Ida et al., 2019). This cost may be dominant when the parameter groups are not very sparse and T is large, as it cannot skip the nonzero parameter groups. On the other hand, the cost of our method for computing the bounds does not depend on T and is $\mathcal{O}(G^2 + p)$ time even in the worst case. Since $G \ll p$ holds for many use cases, our method accelerates Group SCAD even when the parameters are not very sparse.

As for convergence, our method has the following property:

Algorithm 2 AccelGSCAD

1: $\mathbb{A} = \{1, ..., G\}, \mathbb{N} = \emptyset, \mathbb{S} = \emptyset, \mathbb{L} = \emptyset, \mathbb{C} = \emptyset, \beta \leftarrow 0$;
2: **for** each $g \in \mathbb{A}$ **do**
3: **for** each $l \in \mathbb{A}$ **do**
4: compute $\|\hat{K}^{(g)}[l]\|_2$;
5: **end for**
6: **end for**
7: $step \leftarrow 0$;
8: **while** $step < 4$ **do**
9: **if** $step == 0$ **then**
10: execute Algorithm 3 on $\mathbb{A}$ for m iterations;
11: **for** each $g \in \mathbb{A}$ **do**
12: compute $\underline{z}_t^{(g)}$ by Equation (9);
13: **if** $\underline{z}_t^{(g)} > \gamma\lambda$ **then**
14: add g to $\mathbb{N}$;
15: **end if**
16: **end for**
17: add $\mathbb{N}$ to $\mathbb{C}$;
18: **else if** $step == 1$ **then**
19: **for** each $g \in \mathbb{A} \setminus \mathbb{C}$ **do**
20: compute $\overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)}$ by Equations (8) and (9);
21: **if** $\gamma\lambda \geq \overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)} > 2\lambda$ **then**
22: add g to $\mathbb{S}$;
23: **end if**
24: **end for**
25: add $\mathbb{S}$ to $\mathbb{C}$;
26: **else if** $step == 2$ **then**
27: **for** each $g \in \mathbb{A} \setminus \mathbb{C}$ **do**
28: compute $\overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)}$ by Equations (8) and (9);
29: **if** $2\lambda \geq \overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)} > \lambda$ **then**
30: add g to $\mathbb{L}$;
31: **end if**
32: **end for**
33: add $\mathbb{L}$ to $\mathbb{C}$;
34: **else if** $step == 3$ **then**
35: add $\mathbb{A} \setminus \mathbb{C}$ to $\mathbb{C}$;
36: **end if**
37: $step++$;
38: execute Algorithm 3 on $\mathbb{C}$;
39: **end while**

Algorithm 3 BCD with snapshots

1: Given: $\mathbb{C}$
2: $t \leftarrow 0$; $t' \leftarrow 0$;
3: **repeat**
4: $\beta_{t'} \leftarrow \beta_t$; $z_{t'} \leftarrow z_t$; $t' \leftarrow t$;
5: **for** each $g \in \mathbb{C}$ **do**
6: compute $z_t^{(g)}$ by Equation (4);
7: $\beta_t^{(g)} \leftarrow F(z_t^{(g)}, \lambda, \gamma)$ by Equation (3);
8: **end for**
9: $t \leftarrow t + 1$;
10: **until** β converges

Theorem 2 (Convergence Property)**.** *Suppose that $\lambda > 0$, $\gamma > 2$ and each group is orthonormalized in problem (1). Then, Algorithm 2 converges to a stationary point of the objective function in problem (1).*

The orthonormalization assumption in the theorem is the same as in the previous work, which proves convergence of BCD for Group SCAD (Breheny and Huang, 2015).

4 EXTENSION

This section describes the extension of our method to Group MCP (Breheny and Huang, 2009). Since it utilizes a thresholding function similarly to Group SCAD, our method is relatively easy to extend to Group MCP. The thresholding function is as follows:

$$F_{\mathrm{MCP}}(z^{(g)}, \lambda, \gamma) = \begin{cases} \frac{\gamma}{\gamma-1} S(\|z^{(g)}\|_2, \lambda) \frac{z^{(g)}}{\|z^{(g)}\|_2} & \text{if } \|z^{(g)}\|_2 \leq \gamma\lambda \\ z^{(g)} & \text{if } \gamma\lambda < \|z^{(g)}\|_2. \end{cases} \quad (10)$$

Note that $\gamma > 1$ for the above thresholding function. For Equation (10) of Group MCP, we define the non-shrink and large-shrink subsets in the same way as in Group SCAD.

Definition 3. *For the thresholding function of Equation (10), the subset $\mathbb{N}$ is as follows:*

$$\mathbb{N} = \{g \in \mathbb{A} | \underline{z}_t^{(g)} > \gamma\lambda\}. \quad (11)$$

Definition 4. *For the thresholding function of Equation (10), the subset $\mathbb{L}$ is as follows:*

$$\mathbb{L} = \{g \in \mathbb{A} | (\gamma\lambda \geq \overline{z}_t^{(g)}) \wedge (\underline{z}_t^{(g)} > \lambda)\}. \quad (12)$$

Subsets $\mathbb{N}$ and $\mathbb{L}$ have the following properties:

Lemma 9 (Non-Shrink Subset $\mathbb{N}$ for Group MCP)**.** *Each $\beta_t^{(g)}$ such that $g \in \mathbb{N}$ must be $z_t^{(g)}$.*

Lemma 10 (Large-Shrink Subset $\mathbb{L}$ for Group MCP)**.** *Each $\beta_t^{(g)}$ such that $g \in \mathbb{L}$ must be $(\|z_t^{(g)}\|_2 - \lambda)\frac{\gamma z_t^{(g)}}{(\gamma-1)\|z_t^{(g)}\|_2}$.*

The proofs are similar to those of Lemma 1 and 3 of Group SCAD. For the algorithm, we start by optimizing the parameters corresponding to $\mathbb{N}$; then we sequentially add the parameters in $\mathbb{L}$ and the other parameters to the optimization problem similarly to Algorithm 2.

5 RELATED WORK

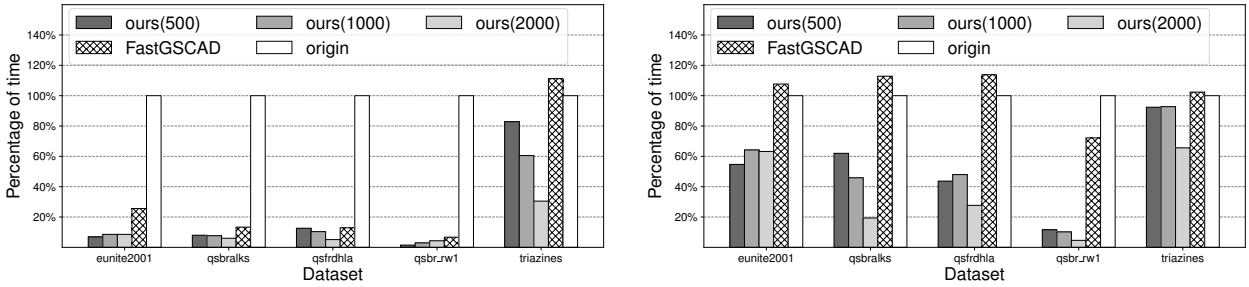
Screening Method. Screening methods are commonly used to speed up solvers for sparsity-inducing regularizations (Ghaoui et al., 2012; Tibshirani et al., 2012; Wang et al., 2013; Bonnefoy et al., 2015; Rakotomamonjy et al., 2019). They eliminate several parameters before solving the problem, and they have been extended to group regularizations. For non-convex group regularizations, Lee and Breheny (2015) proposed a screening method for Group SCAD/MCP. The performance of the screening method deteriorates when the initial parameters are far from the optimal parameters (Johnson and Guestrin, 2016, 2017). Therefore, it usually requires a sequential rule (Ndiaye et al., 2017), which must investigate a sequence of hyperparameters and cannot be parallelized.

Working Set Algorithm. In the context of convex regularizations, working set algorithms can overcome the above drawback of screening methods and have achieved state-of-the-art performance (Johnson and Guestrin, 2015; Ndiaye et al., 2017; Massias et al., 2018). They solve a growing sequence of subproblems that are restricted to small subsets of parameters during optimization to accelerate the solvers (Yuan et al., 2011). A recent paper showed that a working set method based on gap safe rules (Ndiaye et al., 2017) achieves state-of-the-art performance in accelerating solvers of Lasso (Massias et al., 2018). However, since it utilizes the duality gap in the convex optimization literature to select the subsets, it is difficult to be used for non-convex regularizations. Although Boisbunon et al. (2014) and Rakotomamonjy et al. (2022) proposed working set methods for non-convex regularizations, they do not handle group structures.

Bounding Technique. Apart from the above approaches, bounding techniques have been proposed to accelerate coordinate descent methods (Fujiwara et al., 2016a,b; Ida et al., 2019, 2020). They safely skip unnecessary updates in coordinate descent by utilizing a similar upper bound to that described in Section 2.3. They have speeded up many methods, such as Lasso (Fujiwara et al., 2016b), l_1-graph construction (Fujiwara et al., 2016a), Sparse Group Lasso (Ida et al., 2019), and CUR matrix decomposition (Ida et al., 2020). However, the cost of computing bounds for our baseline method (Ida et al., 2019) may be large, as we described in Section 2.3.

6 EXPERIMENT

We evaluated the processing times and values of the objectives to assess the efficiency and accuracy of our method. In Sections 6.1 and 6.2, we evaluate the performance for the settings of $\lambda = \lambda'/10^3$ and $\lambda = \lambda'/10^4$, where λ' is the smallest λ for which the optimal parameters are all zeros (Tibshirani et al., 2012). Namely, we evaluated algorithms in both sparse and dense settings, as large and small λ induce sparse and dense parameters, respectively. In Section 6.3, we compare our method with the existing method in terms of the number of computations for bounds. In Section 6.4, we evaluate the number of updates in BCD to confirm that our method does not slow down convergence. In Section 6.5, we evaluate the performance of com-



(a) $\lambda = \lambda'/10^3$ (b) $\lambda = \lambda'/10^4$

Figure 2: Percentages of times relative to the wall clock times of origin for (a) sparse and (b) dense settings. Our method could accelerate BCD even in the dense setting while the existing bounding technique (FastGSCAD) could not.

puting the regularization path, where a sequence of solutions is computed while changing the hyperparameter λ. We evaluated the processing times with a warm start strategy for a non-increasing sequence of $(\lambda_q)_{q=0}^{99}$ defined as $\lambda_q = \lambda_{\max} 10^{-\gamma q/99}$, where $\lambda_{\max} = \lambda'/10^2$ and $\gamma = 4$.

We compared our method with the original BCD with Group SCAD (*origin*) (Breheny and Huang, 2015) and BCD with the existing bounding technique (*FastGSCAD*) (Ida et al., 2019). All the parameters were initialized to zero. We stopped the algorithms when the relative tolerance of the parameter vector dropped below 10^{-5} for all the methods (Johnson and Guestrin, 2016, 2017). We tried $m = 500, 1000, 2000$ as the hyperparameter of our method. Note that our algorithm runs until all the parameters are optimized (see lines 35 and 38 in Algorithm 2).

We conducted the experiments on five datasets from the LIBSVM (Chang and Lin, 2011) and OpenML (Vanschoren et al., 2013) websites: their names are eunite2001, qsbralks, qsfrdhla, qsbr_rw1, and triazines. These datasets are real-world datasets: eunite2001 is for electricity load forecasting, and the other datasets are for drug design. We created polynomial features for these datasets to obtain group structures in the features by using a polynomial kernel of second-degree (Pavlidis et al., 2001; Roth and Fischer, 2008). As a result, the numbers of groups for each dataset were 120, 210, 528, 1225, and 1770, respectively. To standardize the group size, we replaced λ with $\sqrt{p_g}\lambda$ for each group (Huang et al., 2012). The total numbers of features were 720, 1260, 3168, 7350 and 10,620, respectively. The numbers of data points were 336, 13, 16, 14, and 186, respectively. Each experiment was conducted with one CPU core and 264 GB of main memory on a 2.20 GHz Intel Xeon server running Linux.

6.1 Processing Time

We evaluated the processing times of origin, FastGSCAD, and our method on the five datasets. Figure 2 (a) and (b) show the percentages of times relative to the wall clock times of origin for the settings $\lambda = \lambda'/10^3$ and $\lambda = \lambda'/10^4$, respectively. In the figure, ours(500), ours(1000), and ours(2000) correspond to hyperparameters $m = 500$, 1000, and 2000. We obtained the same ratio of sparsity in parameters among all the methods for the settings of $\lambda = \lambda'/10^3$ and $\lambda = \lambda'/10^4$, as shown in Table 1.

Table 1: Ratio of sparsity in parameters. A large value corresponds to high sparsity.

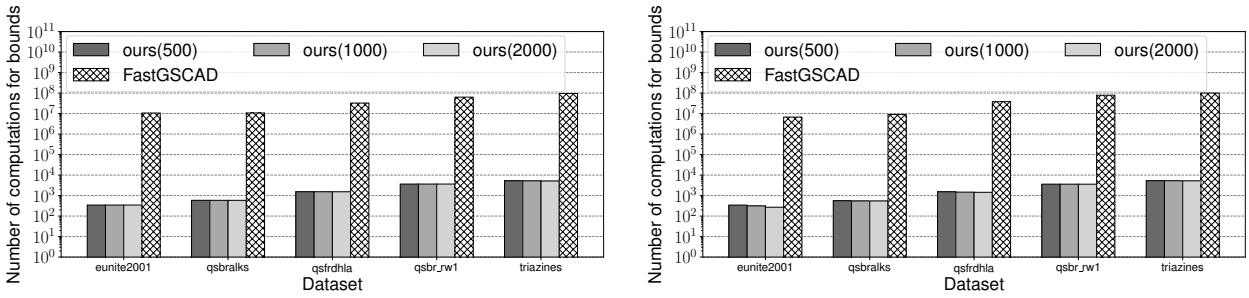
Dataset	Sparsity [%]	
	$\lambda'/10^3$	$\lambda'/10^4$
eunite2001	89.72	44.44
qsbralks	89.52	80.00
qsfrdhla	95.27	85.98
qsbr_rw1	98.78	96.57
triazines	94.58	67.06

For Figure 2 (a), the setting of λ induces sparse parameters. Our method was up to four times faster than FastGSCAD and 68 times faster than the original method. Although FastGSCAD skips unnecessary updates of BCD for groups during optimization, it sometimes incurs a large additional cost in determining whether each parameter group can be skipped or not for every iteration, as shown in the dataset of triazines. On the other hand, our method has a small additional cost since our method computes the bound only when the subsets are selected.

The setting in Figure 2 (b) induces dense parameters as shown in Table 1. In such a setting, the additional cost of FastGSCAD is dominant because it cannot skip unnecessary parameter groups. FastGSCAD was actually slower than the original method on most datasets. In contrast, our method was faster than the original method on all the datasets. These results suggest that our method can speed up optimization of Group SCAD even if the hyperparameter setting incurs dense parameters.

Table 2: Comparison of objective values. Lower is better.

Dataset	λ	origin	FastGSCAD	ours(500)	ours(1000)	ours(2000)
eunite2001	$\lambda'/10^3$	1.644×10^2	1.644×10^2	1.644×10^2	1.644×10^2	1.644×10^2
	$\lambda'/10^4$	1.180×10^2	1.180×10^2	1.180×10^2	1.180×10^2	1.180×10^2
qsbralks	$\lambda'/10^3$	2.491×10^{-5}	2.491×10^{-5}	2.491×10^{-5}	2.491×10^{-5}	2.491×10^{-5}
	$\lambda'/10^4$	4.781×10^{-7}	4.781×10^{-7}	4.781×10^{-7}	4.781×10^{-7}	4.781×10^{-7}
qsfrdhla	$\lambda'/10^3$	1.094×10^{-5}	1.094×10^{-5}	1.094×10^{-5}	$\mathbf{1.052 \times 10^{-5}}$	1.094×10^{-5}
	$\lambda'/10^4$	3.115×10^{-7}	3.115×10^{-7}	3.115×10^{-7}	3.115×10^{-7}	3.115×10^{-7}
qsbr_rw1	$\lambda'/10^3$	1.664×10^{-5}	1.664×10^{-5}	1.664×10^{-5}	1.664×10^{-5}	1.664×10^{-5}
	$\lambda'/10^4$	4.658×10^{-7}	4.658×10^{-7}	4.658×10^{-7}	4.658×10^{-7}	4.658×10^{-7}
triazines	$\lambda'/10^3$	8.224×10^{-4}	8.224×10^{-4}	$\mathbf{8.219 \times 10^{-4}}$	$\mathbf{8.205 \times 10^{-4}}$	$\mathbf{8.208 \times 10^{-4}}$
	$\lambda'/10^4$	5.090×10^{-4}	5.090×10^{-4}	5.090×10^{-4}	5.090×10^{-4}	$\mathbf{5.087 \times 10^{-4}}$



(a) $\lambda = \lambda'/10^3$ (b) $\lambda = \lambda'/10^4$

Figure 3: Numbers of computations for bounds on a logarithmic scale. (a) and (b) are sparse and dense cases, respectively.

6.2 Accuracy

We examined the value of the objective function, which appears in problem (1), to assess the effectiveness of our approach. Since the convergence property of Theorem 2 is based on the objective function on the training dataset, we focused on the evaluation of the converged objective value. The results for the settings of $\lambda = \lambda'/10^3$ and $\lambda = \lambda'/10^4$ are shown in Table 2. Our approach achieved the same or better objective values compared with those of the original method and FastGSCAD. This is probably because it intensively updates parameter groups in the non-shrink subset that largely contributes to predictions while it is theoretically guaranteed to converge, as shown by Theorem 2. The result shows that our method is faster than the existing methods without degrading accuracy.

6.3 Number of Computations for Bounds

We also compared our method with FastGSCAD in terms of the number of computations for bounds. Figure 3 shows the results on a logarithmic scale. Our method reduces the number of the computations by a factor of 10^4 to 10^5 compared with FastGSCAD. As we described in Sections 1, 2.3, Figure 1, and Algorithm 1, since FastGSCAD computes a bound for each inner iteration of BCD, its overhead for computing bounds tends to be high. On the other hand, our method reduces the overhead by computing bounds only when selecting the subsets. Figure 3 supports the effectiveness of the above strategy and it contributes to the reduction of the processing time as shown in Figure 2.

6.4 Number of Updates

Although our method focuses on the reducing the cost of computing bounds, the speed of convergence would also change since the order of updates is different from that of the previous method. Therefore, to confirm that our method does not slow down convergence, we evaluated the number of updates until convergence in BCD for FastGSCAD and our method. Note that the number of updates is the number of un-skipped inner loops in BCD. Figure 4 shows the result for the setting of $\lambda = \lambda'/10^3$ and $m = 1000$. In our method, the total number of updates in BCD is the sum of the numbers of updates in each stage of (1), (2), (3), and (4) in Figure 1 (a). It reduced the number of updates to about 10% compared with FastGSCAD. The result suggests that our method also accelerates the convergence of BCD.

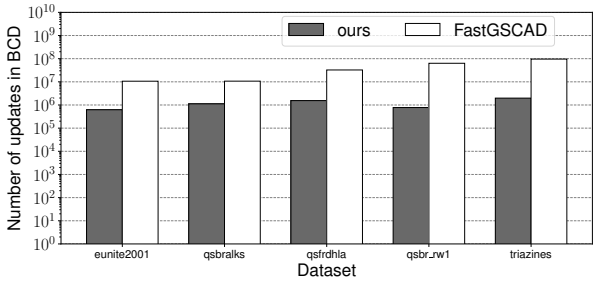


Figure 4: Total numbers of updates in BCD for FastGSCAD and our method.

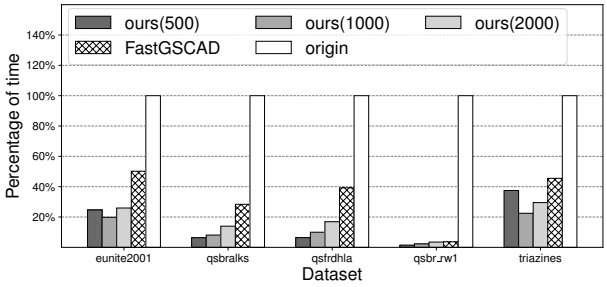


Figure 5: Percentages of times relative to the wall clock times of origin for computing regularization path.

6.5 Regularization Path

Finally, we evaluated the processing time of computing the regularization path. Figure 5 shows the results. Our method was faster than the original method and FastGSCAD on all the datasets. Figure 2 (a) for the sparse setting suggests that our method performs competitively with FastGSCAD on some datasets, while Figure 2 (b) shows that it is faster than the other methods in the dense setting. As a result, our method is faster than the other methods in terms of the total processing time of the regularization path, as shown in Figure 5. We should also note that we eventually obtained the same values of the objective function for all the methods.

7 DISCUSSION

Bounding Technique as a Working Set Algorithm. Although our method centers on the bounding technique that skips unnecessary updates in BCD, it can be regarded as a working set algorithm. In this regard, while the recent working set algorithms use the duality gap as the criterion for selecting subsets, the bounding technique uses the changes in parameters during optimization. Owing to its simplicity, the bounding technique can be extended to various non-convex group regularizers. Although this paper mainly deals with Group SCAD/MCP, our method is applicable to other non-convex regularizers if they have thresholding functions divided into multiple cases like Equation (3). Examples include hard thresholding (Donoho and Johnstone, 1994), ℓ_q-norm (Marjanovic and Solo, 2012), q-shrinkage (Woodworth and Chartrand, 2016), and firm thresholding (Gao and Bruce, 1997). Since our method selects subsets based on parameter changes and the thresholding function, the extension to these regularizers is straightforward, as we show in Section 4 that applies our method to Group MCP.

Application to Convex Group Regularizers. Although our bounding technique focuses on non-convex group regularizers, it can also be applied to convex ones such as Group Lasso (Yuan and Lin, 2006). If our method is adapted to Group Lasso, its procedure is similar to what a screening method does (Ghaoui et al., 2012). This is because our method divides all the parameter groups into a shrink subset, a subset whose parameters turn out to be zero, and one other subset. In this case, we can relatively easily to prove convergence of our method as in the proof of Theorem 2 by utilizing the existing convergence proof of BCD for Group Lasso (Tseng., 2001). It should be noted that the procedure of our method is different from the usual screening method if it is used with non-convex group regularizers: the screening method usually focuses on eliminating parameters that turn out to be zero while our method prioritizes the order of parameter updates by selecting various subsets such as non-, small-, and large-shrink subsets.

Overlapping Groups. The treatment of our method in this paper focused on non-overlapping groups. On the other hand, our method can be extended for overlapping groups by using the latent group lasso scheme (Jacob et al., 2009), which decomposes observed groups into latent groups. Since the plain BCD can be used with the scheme, our method is also available for overlapping groups.

8 CONCLUSION

We proposed a fast BCD for non-convex group regularizations. It solves a growing sequence of subproblems that are restricted to small subsets of parameter groups as a way to prune unnecessary parameter groups. To select the subsets, we developed a novel bounding technique with a small computation cost, which can deal with non-convex group regularizations. In theory, our method is guaranteed to converge to a stationary point. Experiments showed that our method was up to four times faster than the state-of-the-art method and 68 times faster than the original method without any loss of accuracy.

References

Kristin L Ayers and Heather J Cordell. SNP Selection in Genome-wide and Candidate Gene Studies via Penalized

Logistic Regression. *Genetic epidemiology*, 34(8):879–891, 2010.

Aurelie Boisbunon, Rémi Flamary, and Alain Rakotomamonjy. Active Set Strategy for High-dimensional Non-Convex Sparse Optimization Problems. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1517–1521. IEEE, 2014.

Antoine Bonnefoy, Valentin Emiya, Liva Ralaivola, and Rémi Gribonval. Dynamic Screening: Accelerating First-Order Algorithms for the Lasso and Group-lasso. *IEEE Trans. Signal Processing*, 63(19):5121–5132, 2015.

Patrick Breheny and Jian Huang. Penalized Methods for Bi-level Variable Selection. *Statistics and its interface*, 2(3):369, 2009.

Patrick Breheny and Jian Huang. Group Descent Algorithms for Nonconvex Penalized Linear and Logistic Regression Models with Grouped Predictors. *Statistics and Computing*, 25(2):173–187, 2015.

Peter Bühlmann and Lukas Meier. Discussion: One-Step Sparse Estimates in Nonconcave Penalized Likelihood Models. *Ann. Statist.*, 36(4):1534–1541, 2008.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM TIST*, 2(3):27:1–27:27, 2011.

David L Donoho and Iain M Johnstone. Ideal Spatial Adaptation by Wavelet Shrinkage. *Biometrika*, 81(3):425–455, 1994.

Jianqing Fan and Runze Li. Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties. *Journal of the American statistical Association*, 96(456): 1348–1360, 2001.

Yasuhiro Fujiwara, Yasutoshi Ida, Junya Arai, Mai Nishimura, and Sotetsu Iwamura. Fast Algorithm for the Lasso based L1-Graph Construction. *Proc. VLDB Endow.*, 10(3):229–240, 2016a.

Yasuhiro Fujiwara, Yasutoshi Ida, Hiroaki Shiokawa, and Sotetsu Iwamura. Fast Lasso Algorithm via Selective Coordinate Descent. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1561–1567, 2016b.

Hong-Ye Gao and Andrew G. Bruce. Waveshrink with Firm Shrinkage. *Statistica Sinica*, 7(4):855–874, 1997.

Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe Feature Elimination for the Lasso and Sparse Supervised Learning Problems. *Pacific Journal of Optimization*, 8(4):667–698, 2012.

Pinghua Gong, Changshui Zhang, Zhaosong Lu, Jianhua Huang, and Jieping Ye. A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 37–45, 2013.

Kristjan H. Greenewald, Karthikeyan Shanmugam, and Dmitriy Katz. High-Dimensional Feature Selection for Sample Efficient Treatment Effect Estimation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2224–2232, 2021.

Jian Huang, Patrick Breheny, and Shuangge Ma. A Selective Review of Group Selection in High-Dimensional Models. *Statistical Science*, 27(4):481–499, 2012.

Yasutoshi Ida, Yasuhiro Fujiwara, and Hisashi Kashima. Fast Sparse Group Lasso. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1702–1710, 2019.

Yasutoshi Ida, Sekitoshi Kanai, Yasuhiro Fujiwara, Tomoharu Iwata, Koh Takeuchi, and Hisashi Kashima. Fast Deterministic CUR Matrix Decomposition with Accuracy Assurance. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 4594–4603, 2020.

Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group Lasso with Overlap and Graph Lasso. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 433–440, 2009.

Tyler Johnson and Carlos Guestrin. Blitz: A Principled Meta-Algorithm for Scaling Sparse Optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1171–1179, 2015.

Tyler B. Johnson and Carlos Guestrin. Unified Methods for Exploiting Piecewise Linear Structure in Convex Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4754–4762, 2016.

Tyler B. Johnson and Carlos Guestrin. StingyCD: Safely Avoiding Wasteful Updates in Coordinate Descent. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1752–1760, 2017.

Sangin Lee and Patrick Breheny. Strong Rules for Nonconvex Penalties and Their Implications for Efficient Algorithms in High-Dimensional Regression. *Journal of Computational and Graphical Statistics*, 24(4):1074–1091, 2015.

Po-Ling Loh and Martin J. Wainwright. Support Recovery without Incoherence: A Case for Nonconvex Regularization. *The Annals of Statistics*, 45(6):2455 – 2482, 2017.

Goran Marjanovic and Victor Solo. On l_q Optimization and Matrix Completion. *IEEE Transactions on Signal Processing*, 60(11):5714–5724, 2012.

Mathurin Massias, Joseph Salmon, and Alexandre Gramfort. Celer: a Fast Solver for the Lasso with Dual Extrapolation. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 3321–3330, 2018.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of Machine Learning Research (JMLR)*, 11:2287–2322, 2010.

Eugène Ndiaye, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Gap Safe Screening Rules for Sparsity Enforcing Penalties. *Journal of Machine Learning Research (JMLR)*, 18(1):4671–4703, 2017.

Joseph O. Ogutu and Hans-Peter Piepho. Regularized Group Regression Methods for Genomic Prediction: Bridge, MCP, SCAD, Group Bridge, Group Lasso, Sparse Group Lasso, Group MCP and Group SCAD. In *BMC proceedings*, volume 8, page S7, 2014.

Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene Functional Classification from Heterogeneous Data. In *Proceedings of the Fifth Annual International Conference on Computational Biology*, pages 249–255. ACM, 2001.

Alain Rakotomamonjy, Gilles Gasso, and Joseph Salmon. Screening Rules for Lasso with Non-Convex Sparse Regularizers. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 5341–5350, 2019.

Alain Rakotomamonjy, Rémi Flamary, Joseph Salmon, and Gilles Gasso. Convergent Working Set Algorithm for Lasso with Non-Convex Sparse Regularizers. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 5196–5211, 2022.

Volker Roth and Bernd Fischer. The Group-Lasso for Generalized Linear Models: Uniqueness of Solutions and Efficient Algorithms. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 848–855, 2008.

Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, and Ryan J. Tibshirani. Strong Rules for Discarding Predictors in Lasso-type Problems. *Journal of the Royal Statistical Society Series B*, 74(2):245–266, 2012.

Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494, 2001.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

Jie Wang, Jiayu Zhou, Peter Wonka, and Jieping Ye. Lasso Screening Rules via Dual Polytope Projection. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1070–1078, 2013.

Lifeng Wang, Guang Chen, and Hongzhe Li. Group SCAD Regression Analysis for Microarray Time Course Gene Expression Data. *Bioinformatics*, 23(12):1486–1494, 2007.

Joseph Woodworth and Rick Chartrand. Compressed Sensing Recovery via Nonconvex Shrinkage Penalties. *Inverse Problems*, 32(7):075004, 2016.

Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. An Improved GLMNET for L1-Regularized Logistic Regression. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 33–41. ACM, 2011.

Ming Yuan and Yi Lin. Model Selection and Estimation in Regression with Grouped Variables. *Journal of the Royal Statistical Society*, 68(1):49–67, 2006.

Tong Zhang. Analysis of Multi-stage Convex Relaxation for Sparse Regularization. *Journal of Machine Learning Research (JMLR)*, 11:1081–1107, 2010.

A PROOFS

A.1 Proof of Lemma 1

Proof. From Equation (5), we have $\underline{z}_t^{(g)} > \gamma\lambda$ for each $g \in \mathbb{N}$. If $\underline{z}_t^{(g)} > \gamma\lambda$ holds, we obtain $\|z_t^{(g)}\|_2 > \gamma\lambda$ because $\|z_t^{(g)}\|_2 \geq \underline{z}_t^{(g)}$. Therefore, we obtain Lemma 1 from Equation (3) because $\beta_t^{(g)} = F(z_t^{(g)}, \lambda, \gamma) = z_t^{(g)}$ if $\|z_t^{(g)}\|_2 > \gamma\lambda$. □

A.2 Proof of Lemma 2

Proof. From Equation (6), we have $\gamma\lambda \geq \overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)} > 2\lambda$ for each $g \in \mathbb{S}$. In this case, we have $\gamma\lambda \geq \|z_t^{(g)}\|_2 > 2\lambda$ because $\gamma\lambda \geq \overline{z}_t^{(g)} \geq \|z_t^{(g)}\|_2$ and $\|z_t^{(g)}\|_2 \geq \underline{z}_t^{(g)} > 2\lambda$. Therefore, we obtain Lemma 2 from Equation (3) because $\beta_t^{(g)} = F(z_t^{(g)}, \lambda, \gamma) = \frac{\gamma-1}{\gamma-2} S(\|z_t^{(g)}\|_2, \frac{\gamma\lambda}{\gamma-1}) \frac{z_t^{(g)}}{\|z_t^{(g)}\|_2}$ if $\gamma\lambda \geq \|z_t^{(g)}\|_2 > 2\lambda$. □

A.3 Proof of Lemma 3

Proof. From Equation (7), we have $2\lambda \geq \overline{z}_t^{(g)}$ and $\underline{z}_t^{(g)} > \lambda$ for each $g \in \mathbb{L}$. In this case, we have $2\lambda \geq \|z_t^{(g)}\|_2 > \lambda$ because $2\lambda \geq \overline{z}_t^{(g)} \geq \|z_t^{(g)}\|_2$ and $\|z_t^{(g)}\|_2 \geq \underline{z}_t^{(g)} > \lambda$. Since we obtain $S(\|z_t^{(g)}\|_2, \lambda) = (\|z_t^{(g)}\|_2 - \lambda)$ in Equation (3) when $\|z_t^{(g)}\|_2 > \lambda$, we have $\beta_t^{(g)} = (\|z_t^{(g)}\|_2 - \lambda) \frac{z_t^{(g)}}{\|z_t^{(g)}\|_2}$ for each $g \in \mathbb{L}$, which completes the proof. □

A.4 Proof of Lemma 4

Proof. For the upper bound, let $K^{(g,:)} \in \mathbb{R}^{p_g \times p}$ be a block matrix of K that corresponds to group g. By using $z_{t'}^{(g)}$ and $\beta_{t'}^{(g)}$, we transform Equation (4) into the following equation:

$$z_t^{(g)} = z_{t'}^{(g)} - \textstyle\sum_{l \in \mathbb{A} \setminus \{g\}} K^{(g,l)} \Delta\beta_t^{(l)},$$

where $\Delta\beta_t^{(l)} = \beta_t^{(l)} - \beta_{t'}^{(l)}$. From the aforementioned equation, we obtain the following inequality:

$$\begin{aligned} \|z_t^{(g)}\|_2 &\leq \|z_{t'}^{(g)}\|_2 + \textstyle\sum_{l \in \mathbb{A} \setminus \{g\}} \|K^{(g,l)} \Delta\beta_t^{(l)}\|_2 \\ &\leq \|z_{t'}^{(g)}\|_2 + \textstyle\sum_{l \in \mathbb{A} \setminus \{g\}} \|\hat{K}^{(g)}[l]\|_2 \|\Delta\beta_t^{(l)}\|_2. \end{aligned}$$

The triangle inequality and the Cauchy–Schwarz inequality are used to obtain the first and second inequalities, respectively. Since $\|\hat{K}^{(g)}[l]\|_2 \|\Delta\beta_t^{(l)}\|_2 = \Lambda(g,l)$, we obtain the following upper bound in Lemma 4 from the aforementioned inequality:

$$\|z_t^{(g)}\|_2 \leq \|z_{t'}^{(g)}\|_2 + \textstyle\sum_{l \in \mathbb{A} \setminus \{g\}} \Lambda(g,l) = \overline{z}_t^{(g)}.$$

Similarly to the proof of the upper bound, we obtain the following lower bound:

$$\|z_t^{(g)}\|_2 \geq \|z_{t'}^{(g)}\|_2 - \textstyle\sum_{l \in \mathbb{A} \setminus \{g\}} \Lambda(g,l) = \underline{z}_t^{(g)},$$

which completes the proof. □

A.5 Proof of Lemma 5

Proof. Suppose that we first compute $\|\beta_t^{(l)} - \beta_{t'}^{(l)}\|_2$ for all the $l \in \{1, ..., G\}$. The computation cost is $\mathcal{O}(p)$ time. For the upper bound of Equation (8), we need $\mathcal{O}(p_g)$ time to compute $\|z_{t'}^{(g)}\|_2$. The computations of $\sum_{l \in \mathbb{A} \setminus \{g\}} \Lambda(g,l)$ require $\mathcal{O}(G)$ because $\|\beta_t^{(l)} - \beta_{t'}^{(l)}\|_2$ for all of the $l \in \{1, ..., G\}$ has already been computed. The computation cost of the lower bound is the same as that of the upper bound. Since we need to compute the upper and lower bounds for G groups in order to select a subset, the total computation cost is $\mathcal{O}(G^2 + p)$ time. □

A.6 Proof of Lemma 6

Proof. From Lemma 4, we have $\underline{z}_t^{(g)} \leq \|z_t^{(g)}\|_2 \leq \overline{z}_t^{(g)}$. Therefore, the error bound of the upper bound is $|\overline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq |\overline{z}_t^{(g)} - \underline{z}_t^{(g)}| = 2 \sum_{l \in \mathbb{A} \setminus \{g\}} \Lambda(g,l) = \epsilon$. For the lower bound, we obtain $|\underline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq |\underline{z}_t^{(g)} - \overline{z}_t^{(g)}| = \epsilon$. □

A.7 Proof of Lemma 7

Proof. Since the parameters in $\mathbb{A} \setminus \mathbb{N}$ have not been updated when we extract $\mathbb{S}$ from $\mathbb{A} \setminus \mathbb{N}$, we have $\|\beta_t^{(l)} - \beta_{t'}^{(l)}\|_2 = 0$ for $l \in \mathbb{A} \setminus \mathbb{N}$. As a result, $\sum_{l \in \mathbb{A} \setminus \mathbb{N}} \Lambda(g, l) = 0$. Therefore, we obtain $|\overline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq 2 \sum_{l \in \mathbb{N}} \Lambda(g, l) = \epsilon_S$ for the upper bound from Lemma 6. Similarly to the upper bound, we obtain $|\underline{z}_t^{(g)} - \|z_t^{(g)}\|_2| \leq \epsilon_S$ for the lower bound. The manner of proof of the error bound ϵ_L is the same as that of ϵ_S above. □

A.8 Proof of Lemma 8

Proof. Let p_{n} and p_{s} be $\sum_{g \in \mathbb{N}} p_g$ and $\sum_{g \in \mathbb{A} \setminus \mathbb{N}} p_g$, respectively. If we utilize the gradual parameter addition strategy, we only compute the upper and lower bounds for $\mathbb{A} \setminus \mathbb{N}$ when we select the subset $\mathbb{S}$. As the parameters in $\mathbb{A} \setminus \mathbb{N}$ have not been updated yet in this case, $\|\beta_t^{(l)} - \beta_{t'}^{(l)}\|_2 = 0$ for $l \in \mathbb{A} \setminus \mathbb{N}$. Therefore, we only need to compute $\|\beta_t^{(l)} - \beta_{t'}^{(l)}\|_2$ for all the $l \in \mathbb{N}$ first. The computation cost is $\mathcal{O}(p_{\mathrm{n}})$. For the upper bound, it takes $\mathcal{O}(p_{\mathrm{s}})$ time to compute $\|z_{t'}^{(g)}\|_2$ for $g \in \mathbb{A} \setminus \mathbb{N}$. Since $\Lambda(\cdot, l) = 0$ for $l \in \mathbb{N}$, we only need to compute $\sum_{l \in \mathbb{N} \setminus \{g\}} \Lambda(g, l)$ in Equation (8). The computation cost of $\sum_{l \in \mathbb{N} \setminus \{g\}} \Lambda(g, l)$ for $g \in \mathbb{A} \setminus \mathbb{N}$ is $\mathcal{O}(|\mathbb{N}||\mathbb{A} \setminus \mathbb{N}|)$ time. The computation cost of the lower bound is the same as that of the upper bounds. Therefore, the total computation cost for selecting $\mathbb{S}$ is $\mathcal{O}(p + |\mathbb{N}||\mathbb{A} \setminus \mathbb{N}|)$ time. Similarly to the case of $\mathbb{S}$, we obtain $\mathcal{O}(p + |\mathbb{N} \cup \mathbb{S}||\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})|)$ time for selecting $\mathbb{L}$. □

A.9 Proof of Theorem 1

Proof. The extraction of $\mathbb{N}$ (lines 9–17) requires $\mathcal{O}(G^2 + p)$ time from Lemma 5. The algorithm needs $\mathcal{O}(|\mathbb{N}||\mathbb{A} \setminus \mathbb{N}| + p)$ time for extracting $\mathbb{S}$ (lines 18–25) according to Lemma 8. It extracts $\mathbb{L}$ (lines 26–33) at $\mathcal{O}(|\mathbb{N} \cup \mathbb{S}||\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})| + p)$ time from Lemma 8. Therefore, the total additional computation cost is $\mathcal{O}(G^2 + |\mathbb{N}||\mathbb{A} \setminus \mathbb{N}| + |\mathbb{N} \cup \mathbb{S}||\mathbb{A} \setminus (\mathbb{N} \cup \mathbb{S})| + p\})$ time. □

A.10 Proof of Theorem 2

First, we consider convergence for BCD of Algorithm 3 on a subset of all the groups. We have the following lemma for this setting:

Lemma K (Convergence of BCD for Group SCAD on a subset of all the groups)**.** *Consider a subset of groups $\mathbb{D} \in \mathbb{A}$. Suppose that $\lambda > 0$, $\gamma > 2$ and each group in $\mathbb{D}$ is orthonormalized. Then, BCD of Algorithm 3 on $\mathbb{D}$ converges to a stationary point of the following problem with respect to $\beta^{(g)}$ where $g \in \mathbb{D}$:*

$$\min_{\beta^{(g)} | g \in \mathbb{D}} \frac{1}{2n} \Big\| y - \sum_{g \in \mathbb{D}} X^{(g)} \beta^{(g)} \Big\|_2^2 + \sum_{g \in \mathbb{D}} p_{\lambda,\gamma}(\|\beta^{(g)}\|_2). \tag{A.1}$$

Proof. Let $Q(t)$ be the objective function in problem (A.1) for t-th iteration in Algorithm 3. Since BCD of Algorithm 3 is performed on $\mathbb{D}$, terms in problem (1) with respect to $g \notin \mathbb{D}$ can be seen as constants and ignored. Therefore, Algorithm 3 can be regarded as dealing with problem (A.1) instead of problem (1). From Lemma 1 in (Breheny and Huang, 2015), $Q(t)$ with respect to $\beta^{(g)}$ is a strictly convex function when all the parameters except $\beta^{(g)}$ are fixed under the assumptions of Lemma K. In this case, $Q(t + 1) \leq Q(t)$ holds for every iteration in BCD of Algorithm 3 because it optimizes the objective function for each $\beta^{(g)}$ in one iteration. Therefore, every limit point of the sequence of parameters for each iteration in Algorithm 3 is a stationary point of $Q(\cdot)$. □

Almost the same result as above is given in the previous work (Breheny and Huang, 2015). Then, we prove Theorem 2 as follows:

Proof. From Lemma K, line 38 in Algorithm 2 converges with respect to $\beta^{(g)}$ for $g \in \mathbb{C}$. In addition, since we have $\mathbb{C} = \mathbb{A}$ after line 35 in Algorithm 2, it finally converges to a stationary point of the objective function in problem (1). □