An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems

Yangjing Zhang* Ning Zhang† Defeng Sun ‡ Kim-Chuan Toh §

August 2, 2018

Abstract. The sparse group Lasso is a widely used statistical model which encourages the sparsity both on a group and within the group level. In this paper, we develop an efficient augmented Lagrangian method for large-scale non-overlapping sparse group Lasso problems with each subproblem being solved by a superlinearly convergent inexact semismooth Newton method. Theoretically, we prove that, if the penalty parameter is chosen sufficiently large, the augmented Lagrangian method converges globally at an arbitrarily fast linear rate for the primal iterative sequence, the dual infeasibility, and the duality gap of the primal and dual objective functions. Computationally, we derive explicitly the generalized Jacobian of the proximal mapping associated with the sparse group Lasso regularizer and exploit fully the underlying second order sparsity through the semismooth Newton method. The efficiency and robustness of our proposed algorithm are demonstrated by numerical experiments on both the synthetic and real data sets.

Keywords. Sparse group Lasso, generalized Jacobian, augmented Lagrangian method, semismooth Newton method

Mathematics Subject Classification. 90C25, 90C06, 62J05

1 Introduction

In this paper, we aim to design a fast algorithm for solving the following sparse group Lasso (SGLasso) problem:

$$\min_{x\in\mathbb{R}^n} \ \frac{1}{2}\|\mathcal{A}x-b\|^2+\lambda_1\|x\|_1+\lambda_2\sum_{l=1}^{g} w_l\|x_{G_l}\|, \tag{1}$$

*Department of Mathematics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 (zhangyangjing@u.nus.edu).

†Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong (ningzhang_2008@yeah.net).

‡Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong (defeng.sun@polyu.edu.hk).

§Department of Mathematics, and Institute of Operations Research and Analytics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 (mattohkc@nus.edu.sg).

where $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map, $b \in \mathbb{R}^m$ is the given response vector, $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are regularization parameters. For $l = 1, 2, \ldots, g$, $w_l > 0$, and the set $G_l \subseteq \{1, 2, \ldots, n\}$ contains the indices corresponding to the l-th group of features. We denote the restriction of the vector x to the index set G_l as x_{G_l}. Here $\|\cdot\|$ and $\|\cdot\|_1$ denote the ℓ_2 norm and ℓ_1 norm, respectively. For convenience, we denote the SGLasso regularizer by the proper closed convex function $p(x) := \lambda_1\|x\|_1 + \lambda_2 \sum_{l=1}^{g} w_l \|x_{G_l}\|,\ \forall\, x \in \mathbb{R}^n$.

In recent decades, high dimensional feature selection problems have become increasingly important, and the penalized regression models have been proven to be particularly useful for these feature selection problems. For many such problems in real applications, the number of predictors n is much larger than the number of observations m. A notable example of the penalized regression model is the Lasso model that was first proposed by Tibshirani [51]. The problem (1) contains the Lasso problem as a special case if we take the parameter $\lambda_2 = 0$. On the other hand, by assuming that some prior information about the group structure of the underlying solution x is known, Yuan and Lin [55] proposed the group Lasso model, i.e., in problem (1) with parameters $\lambda_1 = 0$ and $\lambda_2 \geq 0$. The group Lasso can select a small set of groups. However, it does not ensure sparsity within each group. For the purpose of achieving sparsity of groups and within each group, Friedman et al. [18] proposed the SGLasso model (1), potentially with overlaps between groups. Apart from the above penalized regression models, there exist a number of variants with different regularizers, such as the fused Lasso [52] and the network Lasso [21].

The SGLasso model has been widely applied to different fields, such as text processing, bioinformatics, signal interpretation, and object tracking (e.g., [14, 23, 24, 27, 40, 58]). Its wide ranging applications have inspired many researchers to design various algorithms for solving the SGLasso problem. These algorithms include the (accelerated) proximal gradient method (see e.g., [2, 55]), (randomized) block coordinate descent algorithm (see e.g., [43, 44, 48]), and alternating direction method of multipliers (see e.g., [4]). To the best of our knowledge, these existing algorithms are first order methods that are applied directly to the primal problem (1) and they hardly utilize any second order information. In contrast, we aim to design an efficient second order information based algorithm for solving the dual problem of the SGLasso problem (1). For problem (1) with $\lambda_2 = 0$, i.e., the Lasso problem, there exist a number of algorithms with second order information being incorporated, such as block active set methods [5, 26], orthant based methods [1, 5, 12], and the semismooth Newton augmented Lagrangian method (SSNAL) [30], to name only a few. In this paper, we extend the SSNAL established in [30] for solving the SGLasso problem with three major reasons. First of all, unlike the other methods, the SSNAL does not require the uniqueness of solution. Secondly, the SSNAL does not need to identify the active sets explicitly, which is critical for our SGLasso setting where the regularizer is no longer piecewise linear. Thirdly and more importantly, the SSNAL has excellent numerical performance for solving the Lasso problem.

Solving the SGLasso problem is especially challenging when there are overlapping groups because of the complex structure of the SGLasso regularizer p. The complicated composite structure of p generally makes it impossible to compute its proximal mapping analytically. However, the efficient computation of such a proximal mapping is indispensable to a number of algorithms, and many of the papers mentioned in the last paragraph thus considered the simpler case of the non-overlapping SGLasso problem. As a first attempt to design a

Hessian based algorithm for the SGLasso problem, we will also focus on the simpler case of the non-overlapping SGLasso problem. The non-overlapping case can be treated as a preliminary study towards the final goal of designing a Hessian based algorithm of solving the overlapping SGLasso problem. For the rest of this paper, we make the following blanket assumption.

Assumption 1.1. *The different groups G_l, $l = 1, 2, \ldots, g$ form a partition of $\{1, 2, \ldots, n\}$, i.e., $G_i \cap G_j = \emptyset$ for all $1 \le i < j \le g$, and $\cup_{l=1}^{g} G_l = \{1, 2, \ldots, n\}$.*

In order to solve the non-overlapping SGLasso problem, we aim to use the semismooth Newton (SSN) augmented Lagrangian (SSNAL) framework for solving the dual problem of (1). This approach is motivated by the success of the SSNAL when applied to the dual of the Lasso problem [30] and that of the fused Lasso problem [31]. We note that the objective functions of the Lasso and fused Lasso problems are piecewise linear-quadratic, and therefore as proven in [30, 31], both the primal and dual iterates generated by the augmented Lagrangian method (ALM) are asymptotically superlinearly convergent. It is this attractive convergence property that leads to the impressive numerical performance of the SSNAL. However, the regularizer p in the objective function of the SGLasso problem (1) is no longer a polyhedral function due to the presence of the ℓ_2 norm. As a result, the asymptotic superlinear convergence of both the primal and dual iterative sequences generated by the ALM are no longer guaranteed to hold by the existing theoretical results. Fortunately, by leveraging on the recent advances made in Cui, Sun, and Toh [11] on the analysis of the asymptotic R-superlinear convergence of the ALM for convex composite conic programming, we are able to establish the global linear convergence (with an arbitrary rate) of the primal iterative sequence, the dual infeasibility, and the dual function values generated by the ALM for the SGLasso problem. With this convergence result, we could expect the ALM to be highly efficient for solving the SGLasso problem.

The remaining challenge of designing an efficient ALM to solve (1) is in solving the subproblem in each iteration. As inspired by the success in [30, 31], we will design a highly efficient SSN method for solving the subproblem in each ALM iteration. The effectiveness of the SSN method relies critically on the efficient computation of the generalized Jacobian of the proximal mapping associated with the SGLasso regularizer p. Thus a major contribution of this paper is to analyse the structure of the generalized Jacobian and its efficient computation. As far as we know, the elements in the generalized Jacobian of the proximal mapping of p have not been derived before, and this paper aims to derive an explicit formula for them. We note that the SGLasso regularizer p enjoys the "prox-decomposition" property [53], similar to the fused Lasso regularizer (see [31]). With the "prox-decomposition" property and some necessary properties for the ℓ_1 norm and ℓ_2 norm, we are able to derive an explicit formula for the generalized Jacobian of the proximal mapping of p. Based on the structure of the generalized Jacobian of the proximal mapping of p, we can derive a certain structured sparsity (which we name as the second order sparsity) of the Hessians associated with the objective function in each ALM subproblem to implement the SSN method efficiently. We should emphasize that the efficiency of the SSN method depends critically on the second order sparsity and the sparsity of the primal iterates. Moreover, the SSN method will be proven to have superlinear/quadratic convergence. In a nutshell, the globally fast linear convergence (with an arbitrary linear rate) of the ALM and the superlinear/quadratic

convergence of the SSN method for solving each ALM subproblem can guarantee that our SSNAL is highly efficient and robust for solving large-scale SGLasso problems.

The rest of this paper is organized as follows. Section 2 demonstrates the decomposition property of the SGLasso regularizer and provides theoretical conditions for ensuring the global fast linear convergence of the ALM. The explicit formulation of the generalized Jacobian of the proximal mapping of the SGLasso regularizer is derived in section 3. In section 4, we design the semismooth Newton based augmented Lagrangian method (SSNAL) for solving the dual of the SGLasso problem (1) and derive our main convergence results. We will also present efficient techniques for implementing SSNAL. Section 5 evaluates the performance of SSNAL on both the synthetic and real data sets. Finally, concluding remarks are given in section 6.

Notation. For a linear map $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^m$, we denote its adjoint by $\mathcal{A}^*$. For any convex function p, we denote its conjugate function by p^*, i.e., $p^*(x) = \sup_z\{\langle x, z\rangle - p(z)\}$. For each $l \in \{1, 2, \ldots, g\}$, we define the linear operator $\mathcal{P}_l : \mathbb{R}^n \to \mathbb{R}^{|G_l|}$ by $\mathcal{P}_l x = x_{G_l}$. Let $s := \sum_{l=1}^g |G_l|$. Define $\mathcal{P} := [\mathcal{P}_1; \mathcal{P}_2; \ldots; \mathcal{P}_g] : \mathbb{R}^n \to \mathbb{R}^s$ and $\mathcal{B}_2 := \mathcal{B}_2^{\lambda_{2,1}} \times \cdots \times \mathcal{B}_2^{\lambda_{2,g}}$, where $\mathcal{B}_2^{\lambda_{2,l}} := \{u_l \in \mathbb{R}^{|G_l|} \,|\, \|u_l\| \le \lambda_{2,l}\}$ and $\lambda_{2,l} := \lambda_2 w_l$. For a given closed convex set Ω and a vector x, denote the distance of x to Ω by $\mathrm{dist}(x, \Omega) := \inf_{x' \in \Omega}\{\|x - x'\|\}$ and the Euclidean projection of x onto Ω by $\Pi_\Omega(x) := \arg\min_{x' \in \Omega}\{\|x - x'\|\}$. We define $\mathrm{sign}(\cdot)$ in a component-wise fashion such that $\mathrm{sign}(t) = 1$ if $t > 0$, $\mathrm{sign}(t) = 0$ if $t = 0$, and $\mathrm{sign}(t) = -1$ if $t < 0$. For any functions f and g, define $(f \circ g)(\cdot) := f(g(\cdot))$. We denote the Hadamard product by $\odot$. For a given vector x, $\mathrm{supp}(x)$ denotes the support of x, i.e., the set of indices such that $x_i \neq 0$. We denote the vector of all ones by e. For a matrix A and a vector a, we denote by $\mathrm{diag}(A)$ and $\mathrm{Diag}(a)$ the diagonal vector of A and the diagonal matrix whose diagonal elements are the components of a, respectively.

2 Preliminaries

In this section, we establish the decomposition property of the SGLasso regularizer p and present some general error bound results. The SGLasso problem (1) can be written equivalently as follows:

$$\text{(P)} \qquad \min_{x \in \mathbb{R}^n} \; h(x) := f(x) + p(x),$$

where $f(x) := \frac{1}{2}\|\mathcal{A}x - b\|^2$, $p(x) := \varphi(x) + \phi(x)$, $\varphi(x) := \lambda_1\|x\|_1$, and $\phi(x) := \sum_{l=1}^g \lambda_{2,l}\|x_{G_l}\|$ with $\lambda_{2,l} := \lambda_2 w_l$, $l = 1, 2, \ldots, g$. The dual problem [3, Theorem 3.3.5] of (P) takes the following form:

$$\text{(D)} \qquad \begin{array}{ll} \max & g(y, z) := -\langle b, y\rangle - \frac{1}{2}\|y\|^2 - p^*(z) \\ \text{s.t.} & \mathcal{A}^* y + z = 0. \end{array}$$

In addition, the Karush-Kuhn-Tucker (KKT) optimality system associated with (P) and (D) is given by

$$\mathcal{A}x - y - b = 0, \; \mathrm{Prox}_p(x + z) - x = 0, \; \mathcal{A}^* y + z = 0, \tag{2}$$

where the proximal mapping of p is defined by:

$$\mathrm{Prox}_p(u) := \arg\min_x \left\{ p(x) + \frac{1}{2}\|x - u\|^2 \right\}, \; \forall\, u \in \mathbb{R}^n. \tag{3}$$

For any given parameter $t > 0$ and a closed proper convex function h (for its definition, see e.g. [47, Page 52]), the following Moreau identity will be frequently used:

$$\mathrm{Prox}_{th}(u) + t\mathrm{Prox}_{h^*/t}(u/t) = u. \tag{4}$$

It is well known that the proximal mappings of ℓ_1 norm and ℓ_2 norm can be expressed as follows: for any given $c > 0$,

$$\mathrm{Prox}_{c\|\cdot\|_1}(u) = \mathrm{sign}(u) \odot \max\{|u| - ce, 0\},$$
$$\mathrm{Prox}_{c\|\cdot\|}(u) = \begin{cases} \frac{u}{\|u\|}\max\{\|u\| - c, 0\}, & \text{if } u \neq 0, \\ 0, & \text{otherwise.} \end{cases}$$

The following definition of "semismoothness with respect to a multifunction", which is adopted from [28, 31, 38, 42], will play an important role in the subsequent analysis.

Definition 2.1. *Let $\mathcal{O} \subseteq \mathbb{R}^n$ be an open set, $\mathcal{K} : \mathcal{O} \subseteq \mathbb{R}^n \rightrightarrows \mathbb{R}^{m\times n}$ be a nonempty and compact valued, upper-semicontinuous set-valued mapping and $\mathcal{F} : \mathcal{O} \to \mathbb{R}^m$ be a locally Lipschitz continuous function. $\mathcal{F}$ is said to be semismooth at $x \in \mathcal{O}$ with respect to the multifunction $\mathcal{K}$ if $\mathcal{F}$ is directionally differentiable at x and for any $V \in \mathcal{K}(x + \Delta x)$ with $\Delta x \to 0$,*

$$\mathcal{F}(x + \Delta x) - \mathcal{F}(x) - V\Delta x = o(\|\Delta x\|).$$

Let γ be a positive constant. $\mathcal{F}$ is said to be γ-order (strongly, if $\gamma = 1$) semismooth at x with respect to $\mathcal{K}$ if $\mathcal{F}$ is directionally differentiable at x and for any $V \in \mathcal{K}(x + \Delta x)$ with $\Delta x \to 0$,

$$\mathcal{F}(x + \Delta x) - \mathcal{F}(x) - V\Delta x = O(\|\Delta x\|^{1+\gamma}).$$

$\mathcal{F}$ is said to be a semismooth (respectively, γ-order semismooth, strongly semismooth) function on $\mathcal{O}$ with respect to $\mathcal{K}$ if it is semismooth (respectively, γ-order semismooth, strongly semismooth) everywhere in $\mathcal{O}$ with respect to $\mathcal{K}$.

We will show in the next lemma that the proximal mappings of the ℓ_1 norm and ℓ_2 norm are strongly semismooth with respect to the Clarke generalized Jacobian (for its definition, see [9, Definition 2.6.1]).

Lemma 2.1. *For any $c > 0$, the proximal mappings $\mathrm{Prox}_{c\|\cdot\|_1}(\cdot)$ and $\mathrm{Prox}_{c\|\cdot\|}(\cdot)$ are strongly semismooth with respect to the Clarke generalized Jacobian $\partial\mathrm{Prox}_{c\|\cdot\|_1}(\cdot)$ and $\partial\mathrm{Prox}_{c\|\cdot\|}(\cdot)$, respectively.*

Proof. Since $\mathrm{Prox}_{c\|\cdot\|_1}(\cdot)$ is a Lipschitz continuous piecewise affine function, it follows from [15, Proposition 7.4.7] that $\mathrm{Prox}_{c\|\cdot\|_1}(\cdot)$ is strongly semismooth everywhere. Next, we focus on the proximal mapping $\mathrm{Prox}_{c\|\cdot\|}(\cdot)$. From the definition of $\mathrm{Prox}_{c\|\cdot\|}(\cdot)$ and the fact that the projection of any vector onto the second order cone, i.e., the epigraph of the ℓ_2 norm function, is strongly semismooth [8, Proposition 4.3], we can obtain the conclusion directly from [37, Theorem 4]. □

Next, we analyse the vital decomposition property, which is termed as "prox-decomposition" in [53], of the SGLasso regularizer p. In the next proposition, we show that the proximal

mapping $\mathrm{Prox}_p(\cdot)$ of $p = \varphi + \phi$ can be decomposed into the composition of the proximal mappings $\mathrm{Prox}_\varphi(\cdot)$ and $\mathrm{Prox}_\phi(\cdot)$. With this decomposition property, we are able to compute $\mathrm{Prox}_p(\cdot)$ in a closed form. This decomposition result was proved in [54, Theorem 1], which is mainly an extension of that for the fused Lasso regularizer in [17]. Here, we give another short proof based on the systematic investigation in [53].

Proposition 2.1. *Under Assumption 1.1, it holds that*

$$\mathrm{Prox}_p(u) = \mathrm{Prox}_\phi \circ \mathrm{Prox}_\varphi\,(u),\ \forall\, u \in \mathbb{R}^n.$$

Proof. Under Assumption 1.1, the function p has a separable structure. Hence, the problem (3) is separable for each group. Therefore, it is sufficient to prove that

$$\mathrm{Prox}_{\lambda_1\|\cdot\|_1+\lambda_{2,l}\|\cdot\|}(u_l) = \mathrm{Prox}_{\lambda_{2,l}\|\cdot\|} \circ \mathrm{Prox}_{\lambda_1\|\cdot\|_1}(u_l),\ \forall\, u_l \in \mathbb{R}^{|G_l|},\ l = 1, 2, \ldots, g.$$

By [53, Theorem 1], for each $l \in \{1, 2, \ldots, g\}$, it suffices to show that

$$\partial(\lambda_1\|u_l\|_1) \subseteq \partial(\lambda_1\|v_l\|_1),\ v_l := \mathrm{Prox}_{\lambda_{2,l}\|\cdot\|}(u_l),\ \forall\, u_l \in \mathbb{R}^{|G_l|}.$$

For any given $u_l \in \mathbb{R}^{|G_l|}$, we discuss the following two cases.

Case 1: If $\|u_l\| \le \lambda_{2,l}$, then $v_l = 0$. It follows that $\partial(\lambda_1\|v_l\|_1) = [-\lambda_1,\, \lambda_1]^{|G_l|}$, which obviously contains $\partial(\lambda_1\|u_l\|_1)$.

Case 2: If $\|u_l\| > \lambda_{2,l}$, then $v_l = (1 - \lambda_{2,l}/\|u_l\|)u_l$, which implies that $\mathrm{sign}(v_l) = \mathrm{sign}(u_l)$. Thus, it holds that $\partial(\lambda_1\|u_l\|_1) = \partial(\lambda_1\|v_l\|_1)$.

Hence, the proof is completed. □

Consider an arbitrary point $u \in \mathbb{R}^n$. Based on the above proposition, we are now ready to compute $\mathrm{Prox}_p(u)$ explicitly. Let $v := \mathrm{Prox}_\varphi(u)$. For each group G_l, $l = 1, 2, \ldots, g$, it holds that

$$\arg\min_{x_{G_l}} \left\{ \lambda_{2,l}\|x_{G_l}\| + \frac{1}{2}\|x_{G_l} - v_{G_l}\|^2 \right\} = v_{G_l} - \Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_{G_l}).$$

That is, $\mathcal{P}_l\mathrm{Prox}_\phi(v) = \mathcal{P}_l v - \Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(\mathcal{P}_l v)$. Therefore, we have

$$\mathrm{Prox}_p(u) = \mathrm{Prox}_\phi(v) = v - \mathcal{P}^*\Pi_{\mathcal{B}_2}(\mathcal{P}v). \tag{5}$$

For the rest of this section, we introduce some error bound results that will be used later in the convergence rate analysis. Define the proximal residual function $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}^n$ by

$$\mathcal{R}(x) := x - \mathrm{Prox}_p(x - \nabla f(x)), \quad \forall\, x \in \mathbb{R}^n. \tag{6}$$

Since the $\mathrm{dom} f \cap \mathrm{dom} p \neq \emptyset$, we know from [47, Theorem 23.8] that the generalized Jacobian $\partial h : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ takes the following form:

$$\partial h(x) := \{v \in \mathbb{R}^n \,|\, v \in \nabla f(x) + \partial p(x)\}, \quad \forall\, x \in \mathbb{R}^n. \tag{7}$$

Suppose that $\lambda_1 + \lambda_2 > 0$. Let Ω_P be the optimal solution set of (P). Since f is nonnegative on $\mathbb{R}^n$, it is easy to obtain that $h(x) \to +\infty$ as $\|x\| \to +\infty$. Thus, Ω_P is a compact

convex set. The first order optimality condition of (P) implies that $\bar{x} \in \Omega_P$ is equivalent to $0 \in \partial h(\bar{x})$, which in turn is equivalent to $\mathcal{R}(\bar{x}) = 0$. It is proved in [56, Theorem 1] that the local error bound condition (in the sense of Luo and Tseng [35]) holds around the optimal set Ω_P, i.e., for every $\xi \geq \inf_x h(x)$, there exist positive scalars κ_0 and δ_0 such that

$$\operatorname{dist}(x, \Omega_P) \leq \kappa_0 \|\mathcal{R}(x)\|, \ \forall\, x \in \mathbb{R}^n \ \textit{satisfying}\ h(x) \leq \xi \ \textit{and}\ \|\mathcal{R}(x)\| \leq \delta_0. \tag{8}$$

Therefore, by using the facts that Ω_P is compact and that $\mathcal{R}$ is continuous, we know that for any $r_1 > 0$, there exists $\kappa_1 > 0$ such that

$$\operatorname{dist}(x, \Omega_P) \leq \kappa_1 \|\mathcal{R}(x)\|, \ \ \forall\, x \in \mathbb{R}^n \ \textit{satisfying}\ \operatorname{dist}(x, \Omega_P) \leq r_1. \tag{9}$$

Furthermore, by mimicking the proofs in [13, Theorem 3.1] or [10, Proposition 2.4] and noting that Ω_P is a compact set, we can obtain the following result with no difficulty.

Proposition 2.2. *For any $r > 0$, there exists $\kappa > 0$ such that*

$$\operatorname{dist}(x, \Omega_P) \leq \kappa \operatorname{dist}(0, \partial h(x)), \ \forall\, x \in \mathbb{R}^n \ \textit{satisfying}\ \operatorname{dist}(x, \Omega_P) \leq r.$$

3 Generalized Jacobian of $\mathrm{Prox}_p(\cdot)$

In this section, we shall analyse the generalized Jacobian of the proximal mapping $\mathrm{Prox}_p(\cdot)$ of the SGLasso regularizer p. From Proposition 2.1, for any $u \in \mathbb{R}^n$, we have

$$\mathrm{Prox}_p(u) = \mathrm{Prox}_\phi(\mathrm{Prox}_\varphi(u)).$$

At the first glance, we may try to apply the chain rule in deriving the generalized Jacobian of $\mathrm{Prox}_p(\cdot)$. Indeed it was illustrated in [49] that under certain conditions, the generalized Jacobian for composite functions can be obtained by the chain rule in a similar fashion as in finding the ordinary Jacobian for composite smooth functions. Specifically, if the conditions in [49, Lemma 2.1] hold, then we could have obtained by the chain rule the following B-subdifferential (for its definition, see [41, Equation (2.12)]), which is a subset of the Clarke generalized Jacobian,

$$\partial_B \mathrm{Prox}_p(u) = \left\{ \tilde{\Theta} \cdot \Theta \,\middle|\, \tilde{\Theta} \in \partial_B \mathrm{Prox}_\phi(v),\ \Theta \in \partial_B \mathrm{Prox}_\varphi(u),\ v = \mathrm{Prox}_\varphi(u) \right\}.$$

However, the conditions in [49, Lemma 2.1] may not hold in our context, and consequently the above equation is usually invalid. Therefore, the B-subdifferential of $\mathrm{Prox}_p(\cdot)$ is non-trivial to obtain, and we have to find an alternative surrogate to bypass this difficulty. The challenge just highlighted also appeared in [31] when analysing the generalized Jacobian of the proximal mapping of the fused Lasso regularizer. In that work, the general definition "semismoothness with respect to a multifunction" was adopted, and such a multifunction was constructed to play the role of the Clarke generalized Jacobian. Here, we shall use the same strategy, and our task now is to identify such a multifunction.

Before characterizing the multifunction relating to the semismoothness, based on the fact in (5) that $\mathrm{Prox}_\phi(v) = v - \mathcal{P}^* \Pi_{\mathcal{B}_2}(\mathcal{P}v)$, $\forall\, v \in \mathbb{R}^n$, we define the following alternative for the generalized Jacobian of $\mathrm{Prox}_\phi(\cdot)$:

$$\hat{\partial} \mathrm{Prox}_\phi(v) := \left\{ I - \mathcal{P}^* \Sigma \mathcal{P} \,\middle|\, \Sigma = \mathrm{Diag}(\Sigma_1, \ldots, \Sigma_g), \Sigma_l \in \partial \Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_{G_l}), l = 1, 2, \ldots, g \right\}.$$

It can be observed that the main part of $\hat{\partial}\mathrm{Prox}_{\phi}(\cdot)$ is the block diagonal matrix Σ, of which each block is the Clarke generalized Jacobian of a projection operator onto an ℓ_2-norm ball. Since $\partial\Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(\cdot)$ admits a closed form expression, so does $\hat{\partial}\mathrm{Prox}_{\phi}(\cdot)$. Now, we are in a position to present the following multifunction $\mathcal{M}:\mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times n}$ and regard it as the surrogate generalized Jacobian of $\mathrm{Prox}_p(\cdot)$ at any $u\in\mathbb{R}^n$:

$$\mathcal{M}(u) := \left\{ (I-\mathcal{P}^*\Sigma\mathcal{P})\Theta \;\middle|\; \begin{array}{l} \Sigma=\mathrm{Diag}(\Sigma_1,\ldots,\Sigma_g), \Sigma_l\in\partial\Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_{G_l}), l=1,2,\ldots,g, \\ v=\mathrm{Prox}_{\varphi}(u),\ \Theta\in\partial\mathrm{Prox}_{\varphi}(u) \end{array} \right\}. \tag{10}$$

Remark 3.1. *For $l=1,2,\ldots,g$ and $v_l\in\mathbb{R}^{|G_l|}$, the projection onto an ℓ_2-norm ball and its Clarke generalized Jacobian are given as follows, respectively:*

$$\Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_l) = \begin{cases} \lambda_{2,l}\frac{v_l}{\|v_l\|}, & \text{if } \|v_l\|>\lambda_{2,l}, \\ v_l, & \text{otherwise}, \end{cases} \tag{11}$$

$$\partial\Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_l) = \begin{cases} \{\frac{\lambda_{2,l}}{\|v_l\|}(I-\frac{v_lv_l^T}{\|v_l\|^2})\}, & \text{if } \|v_l\|>\lambda_{2,l}, \\ \{I-t\frac{v_lv_l^T}{(\lambda_{2,l})^2}\,|\,0\le t\le 1\}, & \text{if } \|v_l\|=\lambda_{2,l}, \\ \{I\}, & \text{if } \|v_l\|<\lambda_{2,l}. \end{cases} \tag{12}$$

In numerical computations, for any $u\in\mathbb{R}^n$, one needs to construct at least one element in $\mathcal{M}(u)$ explicitly. This can be done as follows. For $l=1,2,\ldots,g$, choose

$$\Sigma_l = \begin{cases} \frac{\lambda_{2,l}}{\|v_l\|}(I-\frac{v_lv_l^T}{\|v_l\|^2}), & \text{if } \|v_l\|>\lambda_{2,l}, \\ I, & \text{if } \|v_l\|\le\lambda_{2,l}. \end{cases}$$

In addition, the Clarke generalized Jacobian of Prox_{φ} are given as follows:

$$\partial\mathrm{Prox}_{\varphi}(u) = \left\{ \mathrm{Diag}(\theta) \;\middle|\; \theta\in\mathbb{R}^n, \theta_i\in\begin{cases} \{1\}, & \text{if } |u_i|>\lambda_1, \\ \{t\,|\,0\le t\le 1\}, & \text{if } |u_i|=\lambda_1,\ i=1,\ldots,n \\ \{0\}, & \text{if } |u_i|<\lambda_1, \end{cases} \right\}. \tag{13}$$

Define a vector $\theta\in\mathbb{R}^n$ and construct a matrix $\Theta=\mathrm{Diag}(\theta)$ with

$$\theta_i = \begin{cases} 0, & \text{if } |u_i|\le\lambda_1, \\ 1, & \text{otherwise}, \ i=1,\ldots,n. \end{cases} \tag{14}$$

We also construct one element for numerical implementations:

$$\Theta=\mathrm{Diag}(\theta)\in\partial\mathrm{Prox}_{\varphi}(u). \tag{15}$$

Therefore, it holds that $(I-\mathcal{P}^*\Sigma\mathcal{P})\Theta\in\mathcal{M}(u)$.

The following main theorem of this section justifies why $\mathcal{M}(u)$ in (10) can be treated as the surrogate generalized Jacobian of $\mathrm{Prox}_p(\cdot)$ at u. That is, it shows that the proximal mapping Prox_p is strongly semismooth on $\mathbb{R}^n$ with respect to the multifunction $\mathcal{M}$ defined in (10).

Theorem 3.1. *Assume that Assumption 1.1 holds. Let $u \in \mathbb{R}^n$. Then the multifunction $\mathcal{M}$, defined in* (10)*, is a nonempty compact valued upper-semicontinuous multifunction, and for any $M \in \mathcal{M}(u)$, M is symmetric and positive semidefinite. Moreover, for any $M \in \mathcal{M}(w)$ with $w \to u$,*

$$\mathrm{Prox}_p(w) - \mathrm{Prox}_p(u) - M(w-u) = O(\|w-u\|^2). \tag{16}$$

Proof. By Lemma 2.1, Proposition 2.1, and [15, Theorem 7.5.17], one can deduce that the point-to-set map $\mathcal{M}$ has nonempty compact images and is upper-semicontinuous, and equation (16) holds. It remains to show that M is symmetric and positive semidefinite for any $M \in \mathcal{M}(u)$. Denote $v := \mathrm{Prox}_\varphi(u)$. Take $M \in \mathcal{M}(u)$ arbitrarily. Then, there exist $\Sigma_l \in \partial \Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_{G_l})$, $l = 1, 2, \ldots, g$ and $\Theta = \mathrm{Diag}(\theta) \in \partial \mathrm{Prox}_\varphi(u)$, given by (12) and (13), respectively, such that

$$M = \sum_{l=1}^{g} \mathcal{P}_l^*(I - \Sigma_l)\mathcal{P}_l\Theta.$$

It suffices to show that $\mathcal{P}_l^*(I - \Sigma_l)\mathcal{P}_l\Theta$ is symmetric and positive semidefinite for any $l \in \{1, 2, \ldots, g\}$. Denote the index sets

$$\Xi_l := \{i \in G_l \,|\, \theta_i = 1\},\ l = 1, 2, \ldots, g. \tag{17}$$

For simplicity, we write v_{G_l} as v_l in the following proof.

Case 1: $\|v_l\| < \lambda_{2,l}$. By (12), $I - \Sigma_l = 0$.

Case 2: $\|v_l\| = \lambda_{2,l}$. By (12), there exists some $t \in [0, 1]$ such that

$$\mathcal{P}_l^*(I - \Sigma_l)\mathcal{P}_l\Theta = \frac{t}{(\lambda_{2,l})^2}(\mathcal{P}_l^* v_l)(\mathcal{P}_l^* v_l)^T\Theta.$$

By the definition of $\mathcal{P}_l$, we deduce that $\mathrm{supp}(\mathcal{P}_l^* v_l) \subseteq \Xi_l$. It follows from (17) that $(\mathcal{P}_l^* v_l)^T\Theta = (\mathcal{P}_l^* v_l)^T$. That is,

$$\mathcal{P}_l^*(I - \Sigma_l)\mathcal{P}_l\Theta = \frac{t}{(\lambda_{2,l})^2}(\mathcal{P}_l^* v_l)(\mathcal{P}_l^* v_l)^T,$$

which is symmetric and positive semidefinite.

Case 3: $\|v_l\| > \lambda_{2,l}$. From (12) and the proof in case 2, we have

$$\begin{aligned}
\mathcal{P}_l^*(I - \Sigma_l)\mathcal{P}_l\Theta &= \mathcal{P}_l^*\Big(I - \frac{\lambda_{2,l}}{\|v_l\|}\big(I - \frac{v_l v_l^T}{\|v_l\|^2}\big)\Big)\mathcal{P}_l\Theta \\
&= \Big(1 - \frac{\lambda_{2,l}}{\|v_l\|}\Big)\mathcal{P}_l^*\mathcal{P}_l\Theta + \frac{\lambda_{2,l}}{\|v_l\|^3}(\mathcal{P}_l^* v_l)(\mathcal{P}_l^* v_l)^T\Theta \\
&= \Big(1 - \frac{\lambda_{2,l}}{\|v_l\|}\Big)\mathcal{P}_l^*\mathcal{P}_l\Theta + \frac{\lambda_{2,l}}{\|v_l\|^3}(\mathcal{P}_l^* v_l)(\mathcal{P}_l^* v_l)^T.
\end{aligned}$$

Since both $\mathcal{P}_l^*\mathcal{P}_l$ and Θ are diagonal, it holds that $\mathcal{P}_l^*(I - \Sigma_l)\mathcal{P}_l\Theta$ is symmetric. Furthermore, it is obvious that $\mathcal{P}_l^*\mathcal{P}_l\Theta$ is positive semidefinite. Therefore, the last equality implies that $\mathcal{P}_l^*(I - \Sigma_l)\mathcal{P}_l\Theta$ is positive semidefinite. In summary, we have shown that M is symmetric and positive semidefinite. □

4 An inexact semismooth Newton based augmented Lagrangian method

In this section, we shall design an inexact semismooth Newton based augmented Lagrangian method for solving problem (D), the dual of the SGLasso problem (1). Compared with the previous work [30], this work adopts the same algorithmic framework of ALM and SSN in this section. As we know, the most important issue in implementing the algorithm lies in finding the explicit expression of the generalized Jacobian, i.e., a matrix $M \in \mathcal{M}(u)$. This matrix admits a diagonal form (with zero or one in the diagonal) in the previous paper [30] whereas it has a much more complicated structure than a diagonal structure in our current work. In particular, any matrix $M \in \mathcal{M}(u)$ in the set of generalized Jacobian will consists of two parts, since the sparse group Lasso regularizer contains two parts. As the generalized Jacobians here have more complex structures, the efficient implementation of the algorithm for solving a SGLasso problem is naturally more difficult than that for solving a Lasso problem [30].

Here we always assume that $\lambda_1 + \lambda_2 > 0$. Write (D) equivalently in the following

$$\begin{array}{ll} \min & \langle b, y\rangle + \frac{1}{2}\|y\|^2 + p^*(z) \\ \text{s.t.} & \mathcal{A}^* y + z = 0. \end{array} \tag{18}$$

For $\sigma > 0$, the augmented Lagrangian function associated with (18) is given by

$$\mathcal{L}_\sigma(y, z; x) = \langle b, y\rangle + \frac{1}{2}\|y\|^2 + p^*(z) + \frac{\sigma}{2}\|\mathcal{A}^* y + z - \sigma^{-1}x\|^2 - \frac{1}{2\sigma}\|x\|^2. \tag{19}$$

The k-th iteration of the augmented Lagrangian method is given as follows:

$$\begin{cases} (y^{k+1}, z^{k+1}) \approx \arg\min_{y,z}\{\mathcal{L}_{\sigma_k}(y, z; x^k)\}, \\ x^{k+1} = x^k - \sigma_k(\mathcal{A}^* y^{k+1} + z^{k+1}), \ k \geq 0. \end{cases}$$

In each iteration, the most expensive step is to solve the following subproblem:

$$\min_{y,z}\{\mathcal{L}_{\sigma_k}(y, z; x^k)\}. \tag{20}$$

Since for any given $x^k \in \mathbb{R}^n$ and $\sigma_k > 0$, $\mathcal{L}_{\sigma_k}(y, z; x^k)$ is a strongly convex function, the subproblem (20) admits a unique optimal solution. For any $y \in \mathbb{R}^m$, define

$$\begin{aligned} &\psi_k(y) := \inf_z \mathcal{L}_{\sigma_k}(y, z; x^k) \\ = \ &\langle b, y\rangle + \frac{1}{2}\|y\|^2 + p^*(\mathrm{Prox}_{p^*/\sigma_k}(\sigma_k^{-1}x^k - \mathcal{A}^* y)) + \frac{\sigma_k}{2}\|\mathrm{Prox}_p(\sigma_k^{-1}x^k - \mathcal{A}^* y)\|^2 \\ &-\frac{1}{2\sigma_k}\|x^k\|^2. \end{aligned} \tag{21}$$

Then, $(y^{k+1}, z^{k+1}) \approx \arg\min_{y,z}\{\mathcal{L}_{\sigma_k}(y, z; x^k)\}$ can be computed as follows:

$$y^{k+1} \approx \arg\min_y \psi_k(y) \quad \text{and} \quad z^{k+1} = \mathrm{Prox}_{p^*/\sigma_k}(\sigma_k^{-1}x^k - \mathcal{A}^* y^{k+1}). \tag{22}$$

Now, we propose an inexact augmented Lagrangian method for solving (18).

Algorithm 1 An inexact augmented Lagrangian method for solving (18)

Let $\sigma_0 > 0$ be a given parameter. Choose $(y^0, z^0, x^0) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$. Iterate the following steps for $k = 0, 1, \ldots$.

Step 1. Compute

$$(y^{k+1}, z^{k+1}) \approx \arg\min_{y,z}\{\mathcal{L}_{\sigma_k}(y, z; x^k)\} \tag{23}$$

via (22).

Step 2. Compute

$$x^{k+1} = x^k - \sigma_k(\mathcal{A}^* y^{k+1} + z^{k+1}) \;=\; \sigma_k \mathrm{Prox}_p(\sigma_k^{-1} x^k - \mathcal{A}^* y^{k+1}). \tag{24}$$

Step 3. Update $\sigma_{k+1} \uparrow \sigma_\infty \leq \infty$.

Given nonnegative summable sequences $\{\varepsilon_k\}$ and $\{\delta_k\}$ such that $\delta_k < 1$ for all $k \geq 0$, we estimate the accuracy of the approximate solution (y^{k+1}, z^{k+1}) of (23) via the standard stopping criteria studied in [45]:

$$\begin{aligned} &\text{(A)} \quad \mathcal{L}_{\sigma_k}(y^{k+1}, z^{k+1}; x^k) - \inf_{y,z} \mathcal{L}_{\sigma_k}(y, z; x^k) \;\leq\; \varepsilon_k^2/2\sigma_k, \\ &\text{(B)} \quad \mathcal{L}_{\sigma_k}(y^{k+1}, z^{k+1}; x^k) - \inf_{y,z} \mathcal{L}_{\sigma_k}(y, z; x^k) \;\leq\; (\delta_k^2/2\sigma_k)\|x^{k+1} - x^k\|^2. \end{aligned}$$

Since ψ_k is strongly convex with modulus 1, one has the estimate

$$\mathcal{L}_{\sigma_k}(y^{k+1}, z^{k+1}; x^k) - \inf_{y,z} \mathcal{L}_{\sigma_k}(y, z; x^k) = \psi_k(y^{k+1}) - \inf \psi_k \leq \frac{1}{2}\|\nabla \psi_k(y^{k+1})\|^2.$$

Therefore, the above stopping criteria (A) and (B) can be replaced by the following easy-to-check criteria, respectively,

$$\begin{aligned} &\text{(A}') \quad \|\nabla \psi_k(y^{k+1})\| \leq \varepsilon_k/\sqrt{\sigma_k}, \\ &\text{(B}') \quad \|\nabla \psi_k(y^{k+1})\| \leq (\delta_k/\sqrt{\sigma_k})\|x^{k+1} - x^k\|. \end{aligned}$$

4.1 Convergence rates for Algorithm 1

Note that the superlinear convergence of the primal and dual sequences generated by the semismooth Newton ALM for solving Lasso and fused Lasso problems ([30, 31]) heavily relies on the polyhedral properties of the Lasso and fused Lasso regularizers. However, the sparse group Lasso regularizer is non-polyhedral. Therefore, one has to modify the convergence analysis to obtain the fast linear convergence property under a suitable error bound assumption. Next, we shall analyse the global linear convergence at an arbitrarily fast rate of the inexact augmented Lagrangian method for solving problem (18).

For the nonnegative summable sequence $\{\varepsilon_k\}$ in the stopping criterion (A'), we introduce a scalar α such that

$$\sum_{k=0}^{\infty} \varepsilon_k \leq \alpha. \tag{25}$$

Let r be any given positive scalar satisfying $r > \alpha$. It follows from Proposition 2.2 that there exists a positive scalar κ such that

$$\operatorname{dist}(x, \Omega_P) \leq \kappa \operatorname{dist}(0, \partial h(x)), \ \forall\, x \in \mathbb{R}^n \ \textit{satisfying} \ \operatorname{dist}(x, \Omega_P) \leq r. \tag{26}$$

The next lemma measures the distance of each primal iterate generated by Algorithm 1 to the optimal solution set Ω_P. The proof of Lemma 4.1 is mainly based on [11, Proposition 1(c)], which itself is an extension of [46, Theorem 2] and [36, Theorem 2.1]. Compared to the proof in [11, Proposition 1(c)], the following lemma uses (26) instead of the calmness condition of $(\partial h)^{-1}$ at the origin for some $\bar{x} \in \Omega_P$.

Lemma 4.1. *Suppose that the initial point $x^0 \in \mathbb{R}^n$ satisfies $\operatorname{dist}(x^0, \Omega_P) \leq r - \alpha$, where α is given by (25). Let $\{x^k\}$ be any infinite sequence generated by Algorithm 1 under criteria (A') and (B') simultaneously. Then for all $k \geq 0$, one has*

$$\operatorname{dist}(x^{k+1}, \Omega_P) \leq \mu_k \operatorname{dist}(x^k, \Omega_P),$$

where $\mu_k := [\delta_k + (1+\delta_k)\kappa/\sqrt{\kappa^2 + \sigma_k^2}]/(1-\delta_k)$ and κ is from (26).

Proof. Denote the proximal point mapping by $P_k := (\mathcal{I} + \sigma_k \partial h)^{-1}$. Then, it follows from [45, Proposition 6] and criterion (A') that

$$\|x^{k+1} - P_k(x^k)\|^2/2\sigma_k \leq \mathcal{L}_{\sigma_k}(y^{k+1}, z^{k+1}; x^k) - \inf_{y,z} \mathcal{L}_{\sigma_k}(y, z; x^k) \ \leq \ \varepsilon_k^2/2\sigma_k.$$

This, together with the fact that $\Pi_{\Omega_P}(x^0) = P_k(\Pi_{\Omega_P}(x^0))$, implies that

$$\|x^{k+1} - \Pi_{\Omega_P}(x^0)\| \leq \|x^{k+1} - P_k(x^k)\| + \|P_k(x^k) - \Pi_{\Omega_P}(x^0)\| \leq \|x^k - \Pi_{\Omega_P}(x^0)\| + \varepsilon_k.$$

Therefore, one has

$$\|x^k - \Pi_{\Omega_P}(x^0)\| \leq \|x^0 - \Pi_{\Omega_P}(x^0)\| + \sum_{i=0}^{k-1} \varepsilon_i \leq \|x^0 - \Pi_{\Omega_P}(x^0)\| + \alpha, \ \forall\, k \geq 0.$$

Consequently, $\operatorname{dist}(x^k, \Omega_P) \leq \operatorname{dist}(x^0, \Omega_P) + \alpha \leq r, \ \forall\, k \geq 0$. Moreover, one has

$$\|P_k(x^k) - \Pi_{\Omega_P}(x^k)\| = \|P_k(x^k) - P_k(\Pi_{\Omega_P}(x^k))\| \leq \|x^k - \Pi_{\Omega_P}(x^k)\| \leq r,$$

which implies that

$$\operatorname{dist}(P_k(x^k), \Omega_P) \leq r, \ \forall\, k \geq 0.$$

Additionally, it was shown in [46, Proposition 1(a)] that

$$P_k(x^k) \in (\partial h)^{-1}((x^k - P_k(x^k))/\sigma_k), \ \forall\, k \geq 0.$$

Then it follows from (26) that

$$\mathrm{dist}(P_k(x^k), \Omega_P) \le \kappa \mathrm{dist}(0, \partial h(P_k(x^k)) \le (\kappa/\sigma_k)\|x^k - P_k(x^k)\|, \ \forall\, k \ge 0.$$

Therefore, from the proof in [11, Proposition 1 (c)], for all $k \ge 0$, we obtain that

$$\mathrm{dist}(P_k(x^k),\, \Omega_P) \le (\kappa/\sqrt{\kappa^2 + \sigma_k^2})\, \mathrm{dist}(x^k,\, \Omega_P)$$

and that

$$\begin{aligned}
&\|x^{k+1} - \Pi_{\Omega_P}(P_k(x^k))\| \\
&\le \delta_k \|x^{k+1} - \Pi_{\Omega_P}(P_k(x^k)\| + \left(\delta_k + (1+\delta_k)\kappa/\sqrt{\kappa^2 + \sigma_k^2}\right)\, \mathrm{dist}(x^k, \Omega_P).
\end{aligned}$$

This, together with the fact that $\mathrm{dist}(x^{k+1},\, \Omega_P) \le \|x^{k+1} - \Pi_{\Omega_P}(P_k(x^k))\|, \ \forall\, k \ge 0$, completes the proof. □

While the global convergence of Algorihm 1 follows from [36, 45] directly, the conditions required in [36, 45] to guarantee the local linear convergence of both $\{x^k\}$ and $\{(y^k, z^k)\}$ may no longer hold for the SGLasso problem due to the non-polyhedral property of the ℓ_2 norm function. Fortunately, the new results established in [11] on the convergence rates of the ALM allow us to establish the following theorem, which proves the global Q-linear convergence of the primal sequence $\{x^k\}$ and the global R-linear convergence of the dual infeasibility and the dual objective values. Furthermore, the linear rates can be arbitrarily fast if the penalty parameter σ_k is chosen sufficiently large.

Theorem 4.1. *Let $\{(y^k, z^k, x^k)\}$ be an infinite sequence generated by Algorithm 1 under stopping criterion (A'). Then, the sequence $\{x^k\}$ converges to some $\bar{x} \in \Omega_P$, and the sequence $\{(y^k,\, z^k)\}$ converges to the unique optimal solution of (D).*

Furthermore, if criterion (B') is also executed in Algorithm 1 and the initial point $x^0 \in \mathbb{R}^n$ satisfies $\mathrm{dist}(x^0,\, \Omega_P) \le r - \alpha$, then for all $k \ge 0$, we have

$$\mathrm{dist}(x^{k+1}, \Omega_P) \le \mu_k\, \mathrm{dist}(x^k, \Omega_P), \tag{27a}$$

$$\|\mathcal{A}^* y^{k+1} + z^{k+1}\| \le \mu_k'\, \mathrm{dist}(x^k, \Omega_P), \tag{27b}$$

$$\sup(D) - g(y^{k+1}, z^{k+1}) \le \mu_k''\, \mathrm{dist}(x^k, \Omega_P), \tag{27c}$$

where

$$\begin{aligned}
\mu_k &:= \left[\delta_k + (1+\delta_k)\kappa/\sqrt{\kappa^2 + \sigma_k^2}\right]/(1-\delta_k), \\
\mu_k' &:= 1/[(1-\delta_k)\sigma_k], \\
\mu_k'' &:= [\delta_k^2 \|x^{k+1} - x^k\| + \|x^{k+1}\| + \|x^k\|]/[2(1-\delta_k)\sigma_k],
\end{aligned}$$

and κ is from (26). Moreover, μ_k, μ_k', and μ_k'' go to 0 if $\sigma_k \uparrow \sigma_\infty = +\infty$.

Proof. The statements on the global convergence just follow from [45, Theorem 5] or [11, Proposition 2]. Inequality (27a) is a direct consequence of Lemma 4.1. From the updating formula (24) of x^{k+1}, we deduce that

$$\|\mathcal{A}^* y^{k+1} + z^{k+1}\| = \sigma_k^{-1} \|x^{k+1} - x^k\|,$$

which, together with [11, Lemma 3], i.e.

$$\|x^{k+1} - x^k\| \leq (1 - \delta_k)^{-1}\mathrm{dist}(x^k, \Omega_p), \tag{28}$$

implies that (27b) holds. Finally, it follows from [11, Proposition 2 (5b)] that

$$\sup(D) - g(y^{k+1}, z^{k+1}) \leq \mathcal{L}_{\sigma_k}(y^{k+1}, z^{k+1}; x^k) - \inf_{y,z} \mathcal{L}_{\sigma_k}(y, z; x^k) + (1/2\sigma_k)(\|x^k\|^2 - \|x^{k+1}\|^2).$$

This, together with criterion (B) and (28), shows that (27c) holds. The proof of this theorem is completed. □

Remark 4.1. *Assume that all the conditions in Theorem 4.1 are satisfied. Since the primal objective function h is Lipschitz continuous on any compact set, there exists a constant $L > 0$ such that h is Lipschitz continuous on the set $\{x \in \mathbb{R}^n \,|\, \mathrm{dist}(x, \Omega_P) \leq r\}$ with modulus L. Therefore, one can obtain from Theorem 4.1 that for all $k \geq 0$,*

$$h(x^{k+1}) - \inf(P) \leq L\mathrm{dist}(x^{k+1}, \Omega_P) \leq L\mu_k \mathrm{dist}(x^k, \Omega_P).$$

This inequality, together with (27c) *and the strong duality theorem, implies that*

$$h(x^{k+1}) - g(y^{k+1}, z^{k+1}) \leq (L\mu_k + \mu_k'')\mathrm{dist}\,(x^k, \Omega_P),$$

which means that the duality gap converges to zero R-linearly at an arbitrary linear rate if σ_k is sufficiently large and R-superlinearly if $\sigma_k \uparrow \sigma_\infty = +\infty$.

4.2 A semismooth Newton method for solving the subproblem (22)

In this subsection, we propose an efficient semismooth Newton (SSN) method for solving the subproblem (22). As already mentioned earlier, having an efficient method for solving (22) is critical to the efficiency of Algorithm 1. In each iteration, we have to solve the following problem, for any given $\sigma > 0$ and fixed $\tilde{x}$,

$$\min_y \left\{ \psi(y) := \langle b, y\rangle + \frac{1}{2}\|y\|^2 + p^*(\mathrm{Prox}_{p^*/\sigma}(\sigma^{-1}\tilde{x} - \mathcal{A}^* y)) + \frac{\sigma}{2}\|\mathrm{Prox}_p(\sigma^{-1}\tilde{x} - \mathcal{A}^* y)\|^2 \right\}. \tag{29}$$

Note that $\psi(\cdot)$ is strongly convex and continuously differentiable with

$$\nabla\psi(y) = b + y - \sigma\mathcal{A}\mathrm{Prox}_p(\sigma^{-1}\tilde{x} - \mathcal{A}^* y).$$

Thus, the unique solution $\bar{y}$ of (29) can be obtained by solving the following nonsmooth equation

$$\nabla\psi(y) = 0. \tag{30}$$

Generally, to solve

$$F(x) = 0,$$

where $F : \mathbb{R}^m \to \mathbb{R}^m$ is a locally Lipschitz continuous function, one can employ the following SSN method:

$$x^{k+1} = x^k - V_k^{-1} F(x^k),$$

where $V_k \in \partial F(x^k)$, and $\partial F(x^k)$ denotes the Clarke generalized Jacobian [9, Definition 2.6.1] of F at x^k. For more details about the SSN method, we refer the reader to [28, 29, 42, 50, 57] and the references therein. In particular, existing studies such as [42, 50] used the Clarke generalized Jacobian $V_k \in \partial F(x^k)$ in the updating scheme and established correspondingly the convergence results of the SSN method.

We should point out again that characterizing $\partial(\nabla\psi)(\cdot)$ is a difficult task to accomplish. In section 3, we have constructed a multifunction $\mathcal{M}$, which is used as a surrogate of the generalized Jacobian $\partial \mathrm{Prox}_p$. Besides, it is illustrated in Theorem 3.1 that Prox_p is strongly semismooth with respect to the multifunction $\mathcal{M}$. Likewise, we define a multifunction $\mathcal{V} : \mathbb{R}^m \rightrightarrows \mathbb{R}^{m\times m}$ as follows:

$$\mathcal{V}(y) := \left\{ V \,|\, V = I + \sigma\mathcal{A}M\mathcal{A}^*,\ M \in \mathcal{M}(\sigma^{-1}\tilde{x} - \mathcal{A}^*y) \right\},$$

where $\mathcal{M}(\cdot)$ is defined in (10). It follows from Theorem 3.1 and [15, Theorem 7.5.17] that (i) $\mathcal{V}$ is a nonempty compact valued upper-semicontinuous multifunction; (ii) $\nabla\psi$ is strongly semismooth on $\mathbb{R}^m$ with respect to the multifunction $\mathcal{V}$; (iii) every matrix in the set $\mathcal{V}(\cdot)$ is symmetric and positive definite. With the above analysis, we are ready to design the following SSN method for solving (30).

Algorithm 2 A semismooth Newton method for solving (30)

Given $\mu \in (0, 1/2)$, $\bar{\eta} \in (0, 1)$, $\tau \in (0, 1]$, and $\beta \in (0, 1)$. Choose $y^0 \in \mathbb{R}^m$. Iterate the following steps for $j = 0, 1, \ldots$.

Step 1. Choose $M_j \in \mathcal{M}(\sigma^{-1}\tilde{x} - \mathcal{A}^*y^j)$. Let $V_j = I + \sigma\mathcal{A}M_j\mathcal{A}^*$. Solve the following linear system

$$V_j d = -\nabla\psi(y^j) \tag{31}$$

exactly or by the conjugate gradient (CG) algorithm to find d^j such that $\|V_j d^j + \nabla\psi(y^j)\| \le \min(\bar{\eta}, \|\nabla\psi(y^j)\|^{1+\tau})$.

Step 2. (Line search) Set $\alpha_j = \beta^{m_j}$, where m_j is the smallest nonnegative integer m for which

$$\psi(y^j + \beta^m d^j) \le \psi(y^j) + \mu\beta^m\langle\nabla\psi(y^j), d^j\rangle.$$

Step 3. Set $y^{j+1} = y^j + \alpha_j d^j$.

The following convergence theorem for Algorithm 2 can be obtained directly from [31, Theorem 3].

Theorem 4.2. *Let $\{y^j\}$ be the sequence generated by Algorithm 2. Then $\{y^j\}$ is well-defined and converges to the unique solution $\bar{y}$ of (29). Moreover, the convergence rate is at least superlinear:*

$$\|y^{j+1} - \bar{y}\| = O(\|y^j - \bar{y}\|^{1+\tau}),$$

where $\tau \in (0, 1]$ is the parameter given in Algorithm 2.

4.3 Efficient techniques for solving the linear system (31)

In this section, we analyse the sparsity structure of the matrix in the linear system (31) and design sophisticated numerical techniques for solving the large-scale linear systems involved in the SSN method. These techniques were first applied in [30] which took full advantage of the second order sparsity of the underlying problem. The numerical techniques also rely heavily on the sparsity of the primal iterative sequence.

As can be seen, the most expensive step in each iteration of Algorithm 2 is in solving the linear system (31). Let $(\tilde{x}, y) \in \mathbb{R}^n \times \mathbb{R}^m$ and $\sigma > 0$ be given. The linear system (31) has the following form:

$$(I + \sigma AMA^T)d = -\nabla\psi(y), \tag{32}$$

where A denotes the matrix representation of the linear operator $\mathcal{A}$, and $M \in \mathcal{M}(u)$ with $u = \sigma^{-1}\tilde{x} - A^T y$. With the fact that A is an m by n matrix and M is an n by n matrix, the cost of naively computing AMA^T is $O(mn(m+n))$. Similarly, for any vector $d \in \mathbb{R}^m$, the cost of naively computing the matrix-vector product $AMA^T d$ is $O(mn)$. Since the cost of naively computing the coefficient matrix $I + \sigma AMA^T$ and that of multiplying a vector by the coefficient matrix $I + \sigma AMA^T$ are excessively demanding, common linear system solvers, such as the Cholesky decomposition and the conjugate gradient method, will be extremely slow (if possible at all) in solving the linear system (32) arising from large-scale problems. Therefore, it is critical for us to extract and exploit any structures present in the matrix AMA^T to dramatically reduce the cost of solving (32).

Next, we analyse the proof in Theorem 3.1 in detail in order to find the special structure of AMA^T, thereby reducing the computational cost mentioned above. Let $v := \mathrm{Prox}_{\varphi}(u)$. From the proof in Theorem 3.1, case 1 and case 2 (taking $t = 0$) are simple since the set $\mathcal{M}(u)$ contains a zero matrix. We can choose $M = 0$ so that

$$I + \sigma AMA^T = I.$$

The sole challenge lies in case 3. Here, we shall consider

$$\Big(1 - \frac{\lambda_{2,l}}{\|v_l\|}\Big) A\mathcal{P}_l^*\mathcal{P}_l\Theta A^T + \frac{\lambda_{2,l}}{\|v_l\|^3} A(\mathcal{P}_l^* v_l)(\mathcal{P}_l^* v_l)^T A^T.$$

Note that both $\mathcal{P}_l^*\mathcal{P}_l$ and Θ are diagonal matrices whose diagonal elements are either 0 or 1. Therefore, the product $\mathcal{P}_l^*\mathcal{P}_l\Theta$ enjoys the same property. Moreover, we have $\mathrm{supp}(\mathrm{diag}(\mathcal{P}_l^*\mathcal{P}_l)) = G_l$ and $\mathrm{supp}(\mathrm{diag}(\Theta)) = \mathrm{supp}(v)$ by the definition of $\mathcal{P}_l$, (14), and (15). Therefore,

$$\mathrm{supp}(\mathrm{diag}(\mathcal{P}_l^*\mathcal{P}_l\Theta)) = \Xi_l,$$

where Ξ_l is the index set defined by (17) that corresponds to the non-zero elements of v in the l-th group. In other words, the diagonal matrix $\mathcal{P}_l^*\mathcal{P}_l\Theta$ is expected to contain only a few 1's in the diagonal. Consequently, the computational cost of $A\mathcal{P}_l^*\mathcal{P}_l\Theta A^T$ can be greatly reduced. Next, we observe that $\mathrm{supp}(\mathcal{P}_l^* v_l) \subseteq \Xi_l$. Thus to compute $A(\mathcal{P}_l^* v_l)$, one just needs to consider those columns of A corresponding to the index set Ξ_l, thereby reducing the cost of computing $A(\mathcal{P}_l^* v_l)$ and that of $A(\mathcal{P}_l^* v_l)(\mathcal{P}_l^* v_l)^T A^T$. The following notations are introduced to express these techniques clearly. Denote the index set $\Xi_> := \{l \,|\, \|v_l\| > \lambda_{2,l}, l = 1, 2, \ldots, g\}$, which corresponds to case 3 in Theorem 3.1. For each $l = 1, 2, \ldots, g$, let $A_l \in \mathbb{R}^{m \times |\Xi_l|}$ be the

sub-matrix of A with those columns in Ξ_l and $s_l := (\mathcal{P}_l^* v_l)_{\Xi_l} \in \mathbb{R}^{|\Xi_l|}$ be the sub-vector of $\mathcal{P}_l^* v_l$ restricted to Ξ_l. Then, we deduce that

$$\begin{aligned} AMA^T &= \sum_{l\in\Xi_>} \Big(1 - \frac{\lambda_{2,l}}{\|v_l\|}\Big) A\mathcal{P}_l^*\mathcal{P}_l\Theta A^T + \frac{\lambda_{2,l}}{\|v_l\|^3} A(\mathcal{P}_l^* v_l)(\mathcal{P}_l^* v_l)^T A^T \\ &= \sum_{l\in\Xi_>} \Big(1 - \frac{\lambda_{2,l}}{\|v_l\|}\Big) A_l A_l^T + \frac{\lambda_{2,l}}{\|v_l\|^3} (A_l s_l)(A_l s_l)^T. \end{aligned} \tag{33}$$

Therefore, the cost of computing AMA^T and that of the matrix-vector product $AMA^T d$ for any $d \in \mathbb{R}^m$ are $O(m^2(r + r_2))$ and $O(m(r + r_2))$, respectively, where $r := \sum_{l\in\Xi_>} |\Xi_l| \le |\mathrm{supp}(v)|$ and $r_2 := |\Xi_>| \le g$. We may refer to r as the overall sparsity and r_2 as the group sparsity. In other words, the computational cost depends on the overall sparsity r, the group sparsity r_2, and the number of observations m. The number r is presumably much smaller than n due to the fact that $v = \mathrm{Prox}_{\varphi}(u)$. Besides, the number of observations m is usually smaller than the number of predictors n in many applications. Even if n happens to be extremely large (say, larger than 10^7), one can still solve the linear system (32) efficiently via the (sparse) Cholesky factorization as long as r, r_2, and m are moderate (say, less than 10^4).

In addition, if the optimal solution is so sparse that $r + r_2 \ll m$, then the cost of solving (32) can be reduced further. In this case, the coefficient matrix can be written as follows:

$$I + \sigma AMA^T = I + DD^T,$$

where $D = [B,\, C] \in \mathbb{R}^{m\times(r+r_2)}$ with $B_l := \sqrt{\sigma\Big(1 - \frac{\lambda_{2,l}}{\|v_l\|}\Big)} A_l \in \mathbb{R}^{m\times|\Xi_l|}$, $B := [B_l]_{l\in\Xi_>} \in \mathbb{R}^{m\times r}$, $c_l := \sqrt{\sigma\frac{\lambda_{2,l}}{\|v_l\|^3}}(A_l s_l) \in \mathbb{R}^m$ and $C = [c_l]_{l\in\Xi_>} \in \mathbb{R}^{m\times r_2}$. By the Sherman-Morrison-Woodbury formula, it holds that

$$(I + \sigma AMA^T)^{-1} = (I + DD^T)^{-1} = I - D(I + D^T D)^{-1} D^T.$$

In this case, the main cost is in computing $I + D^T D$ at $O(m(r + r_2)^2)$ operations, as well as to factorize the $r + r_2$ by $r + r_2$ matrix $I + D^T D$ at the cost of $O((r + r_2)^3)$ operations.

Based on the above arguments, one can claim that the linear system (31) in each SSN iteration can be solved efficiently at low costs. In fact based on our experience gathered from the numerical experiments in the next section, the computational costs are so low that the time taken to perform indexing operations, such as obtaining the sub-matrix A_l from A and the sub-vector $(\mathcal{P}_l^* v_l)_{\Xi_l}$ from $\mathcal{P}_l^* v_l$ for $l \in \Xi_>$, may become noticeably higher than the time taken to compute the matrix AMA^T itself. Fortunately, the group sparsity r_2 generally limits the number of such indexing operations needed when computing AMA^T.

Note that in the unlikely event that computing the Cholesky factorization of AMA^T or that of $I + D^T D$ is expensive, such as when $r + r_2$ and m are both large (say more than 10^4), one can employ the preconditioned conjugate gradient (PCG) method to solve the linear system (32) efficiently through exploiting the fast computation of the matrix-vector product $AMA^T d$ for any given vector d.

5 Numerical experiments

In this section, we compare the performance of our semismooth Newton augmented Lagrangian (Ssnal) method with the semi-proximal alternating direction method of multipliers (sPADMM) and the state-of-the-art solver SLEP[1][33] for solving the SGLasso problem. Specifically, the function "sgLeastR" in the solver SLEP is used for comparison. For the details of "sgLeastR" the reader is referred to the paper [34]. ADMM was first proposed in [19, 20], and the implementation will be illustrated in section 5.1. In addition, we also compare with the block coordinate descent (BCD) algorithm when testing on the climate data set in section 5.5. The BCD method we used is the highly efficient method proposed in [39] with a gap safe screening rule, and a Python implementation is available as gl_path.py[2]. Therefore, we can test the performance of the BCD algorithm by running the Python codes sgl_path.py. For fair comparison, we directly run the Python codes instead of translating them into Matlab codes.

Since the primal problem (1) is unconstrained, it is reasonable to measure the accuracy of an approximate optimal solution (y, z, x) for problem (18) and problem (1) by the relative duality gap and dual infeasibility. Specifically, let

$$\text{pobj} := \frac{1}{2}\|\mathcal{A}x - b\|^2 + \lambda_1\|x\|_1 + \lambda_2 \sum_{l=1}^{g} w_l \|x_{G_l}\| \text{ and } \text{dobj} := -\langle b, y\rangle - \frac{1}{2}\|y\|^2$$

be the primal and dual objective function values. Then the relative duality gap and the relative dual infeasibility are defined by

$$\eta_G := \frac{|\text{pobj} - \text{dobj}|}{1 + |\text{pobj}| + |\text{dobj}|}, \ \eta_D := \frac{\|\mathcal{A}^* y + z\|}{1 + \|z\|}.$$

For given error tolerances $\varepsilon_D > 0$ and $\varepsilon_G > 0$, our algorithm Ssnal will be terminated if

$$\eta_D < \varepsilon_D \quad \text{and} \quad \eta_G < \varepsilon_G, \tag{34}$$

while the sPADMM will be terminated if the above conditions hold or the maximum number of $10,000$ iterations is reached. By contrast, since SLEP does not produce the dual sequences $\{(y^k, z^k)\}$, the relative dual infeasibility cannot be used as a stopping criterion for SLEP. Therefore, we terminate SLEP if the relative difference of the optimal objective values between SLEP and Ssnal is less than ε_G, i.e.,

$$\eta_P := \frac{\text{obj}_\text{P} - \text{obj}_\text{S}}{1 + |\text{obj}_\text{P}| + |\text{obj}_\text{S}|} < \varepsilon_G,$$

or the maximum number of $10,000$ iterations is reached. Here obj_P and obj_S denote the objective values obtained by SLEP and Ssnal respectively. Note that the parameters for SLEP are set to their default values unless otherwise specified. BCD is terminated by its default stopping condition.

[1] http://www.public.asu.edu/~jye02/Software/SLEP

[2] The source codes can be found in https://github.com/EugeneNdiaye/GAPSAFE_SGL

In our numerical experiments, we choose $\varepsilon_D = \varepsilon_G = 10^{-6}$ unless otherwise specified. That is, the condition (34) for SSNAL becomes

$$\eta_S := \max\{\eta_G, \eta_D\} < 10^{-6}.$$

Similarly, the stopping condition for sPADMM becomes

$$\eta_A := \max\{\eta_G, \eta_D\} < 10^{-6}.$$

In addition, we adopt the following weights: $w_l = \sqrt{|G_l|}$, $\forall\, l = 1, 2, \ldots, g$ for the model (1). In the following tables, "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM; "nnz" denotes the number of non-zero entries in the solution x obtained by SSNAL using the following estimation:

$$\text{nnz} := \min\{k \mid \sum_{i=1}^{k} |\hat{x}_i| \geq 0.999\|x\|_1\},$$

where $\hat{x}$ is obtained via sorting x by magnitude in a descending order. We display the number of outer ALM iterations (in Algorithm 1) and the total number of inner SSN iterations (in Algorithm 2) of SSNAL in the format of "outer iteration (inner iteration)" under the iteration column. The computation time is in the format of "hours:minutes:seconds", and "00" in the time column means that the elapsed time is less than 0.5 second.

All our numerical results are obtained by running MATLAB (version 9.0) on a windows workstation (24-core, Intel Xeon E5-2680 @ 2.50GHz, 128 Gigabytes of RAM) except that the PYTHON codes spl_path.py is implemented in Anaconda 2.

5.1 Dual based semi-proximal ADMM

In this section, we study the implementation of the (inexact) semi-proximal alternating direction method of multipliers (sPADMM), which is an extension of the classic ADMM [19, 20]. This method is one of the most natural methods for solving (18) due to its separable structure. Generally, the framework of the sPADMM consists of the following iterations:

$$\begin{cases} y^{k+1} \approx \arg\min_y \mathcal{L}_\sigma(y, z^k; x^k) + \frac{1}{2}\|y - y^k\|^2_{\mathcal{S}_1}, \\ z^{k+1} \approx \arg\min_z \mathcal{L}_\sigma(y^{k+1}, z; x^k) + \frac{1}{2}\|z - z^k\|^2_{\mathcal{S}_2}, \\ x^{k+1} = x^k - \tau\sigma(\mathcal{A}^* y^{k+1} + z^{k+1}), \end{cases} \tag{35}$$

where $\tau \in (0, (1+\sqrt{5})/2)$, $\mathcal{S}_1$ and $\mathcal{S}_2$ are self-adjoint positive semidefinite linear operators, and $\mathcal{L}_\sigma$ is the augmented Lagrangian function defined in (19). The sPADMM is convergent under some mild conditions, and we refer the reader to [7, 16] for the convergence results. However, due to the lack of error bound conditions for the KKT system (2), the linear convergence rate of the sPADMM cannot be established from existing results.

In each iteration of (35), the first step is to minimize a function of y. In particular, y^{k+1} can be obtained by solving the following $m \times m$ linear system of equations:

$$(\sigma^{-1} I + \mathcal{A}\mathcal{A}^* + \mathcal{S}_1) y^{k+1} = -\sigma^{-1} b - \mathcal{A}(z^k - \sigma^{-1} x^k) + \mathcal{S}_1 y^k.$$

As the dimension m is a moderate number in many statistical applications. Thus, in our implementation, equation (5.1) was solved via the Cholesky factorization, and the proximal term $\mathcal{S}_1$ was taken to be the zero matrix. In the event that computing the Cholesky factorization of $\sigma^{-1}I+\mathcal{A}\mathcal{A}^*$ is expensive, one can choose $\mathcal{S}_1$ judiciously to make the coefficient matrix to be a positive definite diagonal matrix plus a low-rank matrix that one can invert efficiently via the Sherman-Morrison-Woodbury formula. We refer the reader to [7, section 7.1] for the details on how to choose $\mathcal{S}_1$ appropriately.

The second step in (35) is to minimize a function of z. For the SGLasso problem, one would simply choose $\mathcal{S}_2 = 0$. In this case, by the Moreau identity (4), z^{k+1} is updated by the following scheme:

$$z^{k+1} = \sigma^{-1}x^k - \mathcal{A}^*y^{k+1} - \mathrm{Prox}_p(\sigma^{-1}x^k - \mathcal{A}^*y^{k+1}),$$

where Prox_p is computable by Proposition 2.1. In summary, two subproblems of (35) are solvable and consequently the framework (35) is easily implementable. Moreover, in order to improve the convergence speed numerically, we set the step-length τ in (35) to be 1.618 and tune the parameter σ according to the progress between primal feasibility and dual feasibility in the implementation.

5.2 Synthetic data

This section presents the tests of the three algorithms Ssnal, sPADMM, and SLEP on various synthetic data constructed in the same way as in [48]. The data matrix A is generated randomly as an $m \times n$ matrix of normally distributed random numbers, and the number of groups g is chosen manually to be 100, 1000, and 10000. Then we partition $\{1, 2, \ldots, n\}$ into g groups such that the indices of components in each group are adjacent, for example, $G_1 = \{1, 2, \ldots, 25\}$, $G_2 = \{26, 27, \ldots, 53\}$, etc. The group sizes $\{|G_i|, i = 1, 2, \ldots, g\}$ are determined randomly such that each $|G_i|$ is expected to be around the mean value of $\frac{n}{g}$. Subsequently, the response vector b is constructed as

$$b = Ax + \epsilon,$$

where ϵ is normally distributed random noise, $x_{G_l} = (1, 2, \ldots, 10, 0, \ldots, 0)^T$ for $l = 1, 2, \ldots, 10$, and $x_{G_l} = 0$ for all other groups. That is, the first 10 groups are the non-trivial groups, and the true number of non-zero elements of the underlying solution x is 100. The regularization parameters $\lambda_1 = \lambda_2$ are chosen to make the number of non-zero elements of the resulting solution close to the true number of 100.

Table 1 compares the numerical results of the three algorithms Ssnal, sPADMM, and SLEP tested on different synthetic data. As can be seen from the table, the computational time of Ssnal is less than that of sPADMM and SLEP for most cases. The overall advantage of computational time suggests that our algorithm Ssnal is efficient for solving the SGLasso problem with randomly generated data. Moreover, we observe from the table that sPADMM is inefficient in solving the SGLasso problem with randomly generated large-scale data. A possible reason is that the first order method sPADMM requires a large number of iterations to solve the problem to the required accuracy of 10^{-6}. The table also shows that our algorithm Ssnal can significantly outperform SLEP on problems with a large number of

groups. In particular, SSNAL is more than 5 times faster than SLEP for the high dimensional instance with problem size $(m, n) = (1e4, 1e6)$ and group number $g = 10,000$. For this instance, the number of non-zero entries in the solution x is small, and we have highly conducive second order sparsity which we can fully exploit in the numerical computations outlined in section 4.3.

Table 1: The performances of SSNAL, sPADMM, and SLEP on synthetic data. Regularization parameters are set as follows: $\lambda_1 = \lambda_2$. "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM.

size (m,n)	g	λ_1	nnz	iteration S|A|P	time S|A|P
(1e3,1e5)	100	1338	166	1(3) | 1246 | 1	00 | 01:23 | 00
	1000	1736	154	2(13) | 1247 | 26	01 | 01:25 | 01
	10000	983	84	4(27) | 1185 | 239	02 | 01:21 | 13
(1e4,1e6)	100	3775	43	1(3) | 2228 | 17	13 | 03:01:49 | 59
	1000	7229	167	1(3) | 2232 | 1	11 | 03:05:03 | 08
	10000	4000	109	3(28) | 2104 | 148	01:38 | 03:06:31 | 08:37

5.3 UCI data sets with random groups

This section presents the performances of the three algorithms SSNAL, sPADMM, and SLEP on large-scale UCI data sets [32] $(\mathcal{A}, b)$ that are originally obtained from the LIBSVM data sets [6]. In our numerical experiments, we follow [30] and apply the method in [22] to expand the original features of the data sets *bodyfat*, *pyrim*, and *triazines* using polynomial basis functions. For example, a polynomial basis function of order 7 is used to expand the features of the data set *bodyfat*, and then the expanded data set is named as *bodyfat7*. This naming convention is also used for *pyrim5*, *triazines4*, and *housing 7*. As noted in [30, Table 1], these data sets are quite different in terms of the problem dimension and the largest eigenvalue of $\mathcal{A}\mathcal{A}^*$. For example, for a relatively high-dimensional instance *log1p.E2006.train*, the dimension of $\mathcal{A}$ is 16087×4272227 and the largest eigenvalue of $\mathcal{A}\mathcal{A}^*$ is 5.86×10^7.

Next, we describe how the groups in each problem are specified. By reordering the components of the variable x if necessary, without loss of generality, we assume that the vector x can be partitioned into g groups where the indices of components in each group are adjacent. The group sizes $\{|G_l|, l = 1, 2, \ldots, g\}$ are determined randomly such that each $|G_l|$ is around the mean value of $\frac{n}{g}$. In the experiment, the average group size is about 300.

We tested the SGLasso problems with two different sets of regularization parameters which are chosen manually:

$$\text{(S1)} \quad \lambda_1 = \lambda_2 = \gamma \|\mathcal{A}^* b\|_\infty;$$

$$\text{(S2)} \quad \lambda_1 = 0.5\gamma \|\mathcal{A}^* b\|_\infty, \ \lambda_2 = 9.5\gamma \|\mathcal{A}^* b\|_\infty.$$

The parameter γ is chosen to produce a reasonable number of non-zero elements in the resulting solution x. Three values of γ are used for each UCI data set in our experiments.

Table 2 presents the comparison results of the three algorithms SSNAL, sPADMM, and SLEP on 8 selected UCI data sets with regularization parameters specified as in (S1). As shown in the table, SSNAL has succeeded in solving all instances within 1 minute, while SLEP failed to solve 10 cases. Although sPADMM has also succeeded in solving all instances, its running time for each case is much longer than that of SSNAL. In majority of the cases,

SSNAL outperformed the first order methods sPADMM and SLEP by a large margin. For example, for the instance *E2006.train* with $\gamma = 1e\text{-}7$, SSNAL solved it to the desired accuracy in 3 seconds, sPADMM took more than 8 minutes, while SLEP failed to solve it within 10000 steps. The numerical results show convincingly that our algorithm SSNAL can solve SGLasso problems highly efficiently and robustly. Again, the superior performance of our SSNAL algorithm can be attributed to our ability to extract and exploit the second order sparsity structure (in the SGLasso problem) within the SSN method to solve each ALM subproblem very efficiently.

Table 3 is the same as Table 2 but for the regularization parameters specified as in (S2). This table also shows that the computational time of SSNAL is far less than that of sPADMM and SLEP for almost all cases. Furthermore, for more difficult cases, such as those with large problem dimension (m, n) and large number of non-zero entries (nnz), the superiority of SSNAL is even more striking compared to sPADMM and SLEP. The results again demonstrate that our algorithm SSNAL is highly efficient for solving SGLasso problems.

Figure 1 presents the performance profiles of SSNAL, sPADMM, and SLEP for all 48 tested problems, which are presented in Table 2 and Table 3. The meaning of the performance profiles is given as follows: a point (x, y) is on the performance curve of a particular method if and only if this method can solve up to desired accuracy $(100y)\%$ of all the tested instances within at most x times of the fastest method for each instance. As can be seen, SSNAL outperforms sPADMM and SLEP by a large margin for all tested UCI data sets with randomly generated groups. In particular, focusing on $y = 40\%$, we can see from Figure 1 that SSNAL is around 30 times faster compared to sPADMM and SLEP for over 60% of the tested instances.

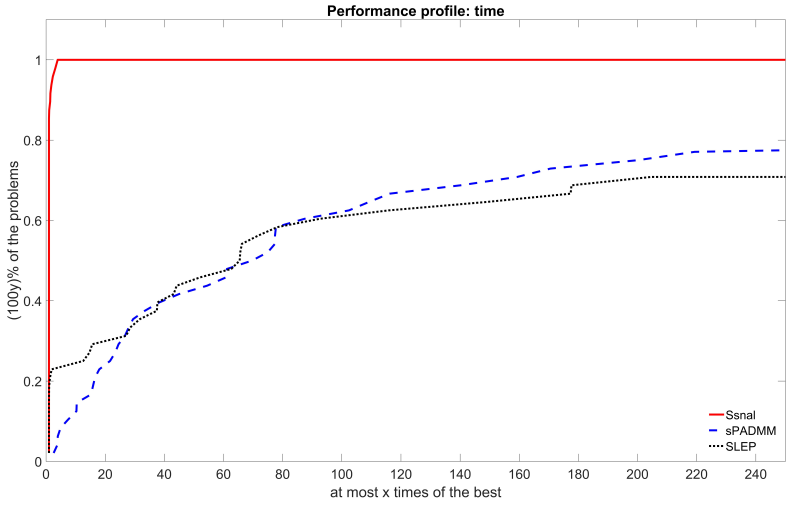

Figure 1: Performance profiles of SSNAL, sPADMM, and SLEP on UCI data sets with randomly generated groups.

Table 2: The performances of SSNAL, sPADMM, and SLEP on 8 selected UCI data sets with randomly generated groups. The regularization parameters are specified as in (S_1). "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM.

problem name (m,n); g	γ	nnz	iteration S|A|P	time S|A|P	error $\eta_S\|\eta_A\|\eta_P$
E2006.train	1e-05	1	3(7) | 44 | 4	01 | 07:50 | 00	1.4e-09 | 7.4e-07 | 6.2e-09
(16087,150360)	1e-06	1	4(8) | 23 | 11	01 | 07:03 | 01	7.9e-11 | 7.8e-07 | 9.9e-07
501	1e-07	46	20(40) | 70 | 10000	03 | 08:30 | 08:59	1.2e-07 | 9.3e-07 | 2.5e-04
E2006.test	1e-05	1	4(8) | 36 | 4	00 | 18 | 00	1.0e-10 | 8.6e-07 | 4.9e-13
(3308,150358)	1e-06	1	4(9) | 29 | 21	00 | 18 | 00	1.2e-08 | 8.3e-07 | 9.7e-07
501	1e-07	150	23(48) | 205 | 10000	02 | 28 | 02:17	8.0e-07 | 1.0e-06 | 3.1e-03
log1p.E2006.train	1e-03	2	3(10) | 2360 | 882	09 | 01:01:06 | 04:45	4.9e-09 | 1.0e-06 | 9.7e-07
(16087,4272227)	1e-04	3	3(10) | 2490 | 5260	08 | 01:02:49 | 28:21	7.0e-07 | 1.0e-06 | 9.8e-07
14241	1e-05	1005	5(22) | 876 | 7553	37 | 33:57 | 40:48	3.3e-07 | 9.9e-07 | 1.0e-06
log1p.E2006.test	1e-03	4	4(13) | 1549 | 2107	08 | 10:22 | 05:12	3.5e-07 | 9.9e-07 | 9.8e-07
(3308,4272226)	1e-04	5	4(12) | 1749 | 2693	06 | 11:39 | 06:41	6.2e-07 | 9.9e-07 | 9.9e-07
14241	1e-05	5009	7(34) | 464 | 10000	45 | 03:42 | 25:04	2.1e-07 | 9.7e-07 | 7.5e-06
bodyfat7	1e-04	7	11(29) | 878 | 3246	01 | 28 | 54	7.7e-07 | 9.9e-07 | 9.6e-07
(252,116280)	1e-05	13	15(37) | 918 | 10000	03 | 29 | 02:48	7.9e-07 | 9.9e-07 | 2.1e-05
388	1e-06	237	21(58) | 917 | 10000	07 | 29 | 02:51	6.9e-07 | 1.0e-06 | 2.4e-04
pyrim5	1e-02	279	8(32) | 3074 | 4736	02 | 02:04 | 01:55	1.3e-07 | 1.0e-06 | 1.0e-06
(74,201376)	1e-03	606	11(39) | 2003 | 10000	02 | 01:21 | 04:05	3.9e-07 | 1.0e-06 | 8.9e-06
671	1e-04	937	17(50) | 1969 | 10000	05 | 01:23 | 04:05	6.1e-07 | 1.0e-06 | 1.2e-04
triazines4	1e-02	406	8(35) | 5038 | 9374	09 | 24:37 | 25:38	8.5e-08 | 1.0e-06 | 1.0e-06
(186,635376)	1e-03	1396	9(43) | 4020 | 10000	17 | 19:43 | 27:25	4.2e-07 | 1.0e-06 | 5.8e-04
2118	1e-04	3574	16(58) | 4287 | 10000	55 | 22:33 | 27:23	6.6e-07 | 1.0e-06 | 5.5e-03
housing7	1e-02	220	7(31) | 813 | 3366	01 | 25 | 59	2.5e-07 | 9.9e-07 | 9.9e-07
(506,77520)	1e-03	817	9(37) | 816 | 5199	02 | 25 | 01:32	8.9e-08 | 9.9e-07 | 1.0e-06
258	1e-04	2134	14(47) | 618 | 10000	07 | 19 | 02:56	7.1e-07 | 1.0e-06 | 3.7e-06

Table 3: The performances of SSNAL, sPADMM, and SLEP on 8 selected UCI data sets with randomly generated groups. The regularization parameters are specified as in (S_2). "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM.

problem name (m,n);g	γ	nnz	iteration S|A|P	time S|A|P	error $\eta_S\|\eta_A\|\eta_P$
E2006.train	1e-05	1	3(7) | 73 | 1	01 | 08:40 | 00	1.6e-10 | 9.7e-07 | 3.0e-08
(16087,150360)	1e-06	1	3(7) | 44 | 4	01 | 07:49 | 00	1.5e-09 | 7.9e-07 | 7.3e-07
501	1e-07	16	8(16) | 31 | 13	01 | 07:33 | 01	3.3e-07 | 9.9e-07 | 9.9e-07
E2006.test	1e-05	1	3(7) | 52 | 16	00 | 20 | 00	3.7e-09 | 7.1e-07 | 3.8e-08
(3308,150358)	1e-06	1	4(8) | 32 | 16	00 | 18 | 00	4.9e-10 | 9.4e-07 | 4.9e-08
501	1e-07	15	9(19) | 29 | 25	01 | 18 | 00	4.8e-07 | 6.9e-07 | 8.7e-07
log1p.E2006.train	1e-03	1	2(6) | 2664 | 202	05 | 01:06:05 | 01:06	5.1e-08 | 9.9e-07 | 2.0e-07
(16087,4272227)	1e-04	7	3(11) | 2379 | 1286	10 | 01:00:42 | 06:53	4.2e-09 | 9.9e-07 | 9.6e-07
14241	1e-05	32	3(11) | 2474 | 4375	09 | 01:02:34 | 23:23	6.8e-07 | 1.0e-06 | 9.9e-07
log1p.E2006.test	1e-03	2	2(7) | 1961 | 379	04 | 12:55 | 57	5.5e-07 | 9.9e-07 | 9.0e-07
(3308,4272226)	1e-04	10	4(13) | 1567 | 1459	08 | 10:30 | 03:40	3.9e-07 | 9.9e-07 | 9.6e-07
14241	1e-05	95	5(15) | 1749 | 4800	08 | 11:37 | 12:18	5.6e-08 | 9.9e-07 | 9.9e-07
bodyfat7	1e-04	111	9(20) | 363 | 1711	00 | 12 | 33	1.7e-08 | 9.8e-07 | 1.0e-06
(252,116280)	1e-05	208	13(30) | 438 | 8460	01 | 14 | 02:42	2.5e-07 | 9.6e-07 | 1.0e-06
388	1e-06	264	17(37) | 555 | 10000	05 | 18 | 03:10	7.9e-07 | 9.9e-07 | 2.3e-06
pyrim5	1e-02	230	4(17) | 1258 | 1489	01 | 51 | 37	3.7e-07 | 4.5e-07 | 9.7e-07
(74,201376)	1e-03	626	8(34) | 1413 | 6038	02 | 57 | 02:32	9.0e-07 | 1.0e-06 | 1.0e-06
671	1e-04	1178	13(43) | 1684 | 10000	04 | 01:10 | 04:16	1.1e-07 | 1.0e-06 | 4.8e-05
triazines4	1e-02	577	6(27) | 10000 | 4422	06 | 48:29 | 12:01	1.1e-07 | 2.7e-06 | 1.0e-06
(186,635376)	1e-03	1171	8(36) | 4875 | 10000	10 | 23:49 | 27:19	5.3e-07 | 1.0e-06 | 3.2e-06
2118	1e-04	4346	11(48) | 3343 | 10000	28 | 17:51 | 28:42	9.1e-07 | 1.0e-06 | 2.7e-04
housing7	1e-02	206	3(11) | 1097 | 29	00 | 34 | 01	9.0e-09 | 9.7e-07 | 2.9e-07
(506,77520)	1e-03	839	8(30) | 754 | 936	01 | 23 | 17	1.3e-07 | 1.0e-06 | 9.8e-07
258	1e-04	1689	10(36) | 837 | 5510	03 | 26 | 01:38	1.0e-07 | 1.0e-06 | 1.0e-06

5.4 UCI datesets with simulated groups

This section also makes uses of the UCI data sets mentioned in section 5.3. Instead of specifying the groups randomly, we attempt to generate more meaningful groups in the following manner. Firstly, the classical Lasso (model (1) with $\lambda_2 = 0$) is solved with the accuracy of 10^{-4} to obtain a sparse solution x, and the computed solution x is sorted in a descending order. Then, the first $|G_1|$ largest variables are allocated to group 1, and the next $|G_2|$ variables are allocated to group 2, etc. Since this group membership is determined by the magnitude of each variable of the computed solution from the classical Lasso, we believe that this kind of group structure is more natural than that constructed randomly in the previous section. Besides, the group sizes $\{|G_l|,\, l = 1, 2, \ldots, g\}$ are determined randomly such that each $|G_l|$ is around the mean value of $\frac{n}{g}$. Compared to the last section, a different value 30 is taken as the average group size for the diversity of experiments.

To generate the solution from the classical Lasso to decide on the group membership mentioned above, we take the medium value of γ in Table 4, e.g., $\gamma = 1e\text{-}6$ for the instance *E2006.train*. And the regularization parameters for the classical Lasso are set as follow: $\lambda_1 = \gamma\|\mathcal{A}^*b\|_\infty$, $\lambda_2 = 0$. For the SGLasso problem, the regularization parameters follow three different strategies: (S1) and (S2) given in the previous section, and

$$\text{(S3)} \quad \lambda_1 = \gamma\|\mathcal{A}^*b\|_\infty,\ \lambda_2 = \sqrt{\lambda_1} \text{ if } \lambda_1 > 1 \text{ and } \lambda_2 = \lambda_1^2 \text{ if } \lambda_1 \le 1.$$

The comparison results with parameter sets (S1), (S2), and (S3) are presented in Table 4, Table 5, and Table 6, respectively. As shown in these three tables, SSNAL has succeeded in solving all the 72 instances highly efficiently, while sPADMM failed in 5 instances, and SLEP failed in 58 instances. Moreover, for those failed instances, we observe from the tables that SLEP terminated when the errors are still relatively large, which is 10^{-2} for most cases. The results may suggest that using only first order information is not enough for computing high accuracy solution, while second order information can contribute to the fast convergence and high computational efficiency of a well designed second order SSN method. For the vast majority of the instances, the computational time of SSNAL is far less than that of sPADMM and SLEP. Again, the results have demonstrated convincingly that our algorithm SSNAL is capable of solving large-scale SGLasso problems to high accuracy very efficiently and robustly.

Figure 2 presents the performance profiles of SSNAL, sPADMM, and SLEP for all 72 tested problems, which are presented in Table 4, Table 5, and Table 6. From the figure, we find that SSNAL not only solves all the tested instances to the desired accuracy, but also outperforms sPADMM and SLEP by an obvious margin for these tested UCI data sets with simulated groups. Within 250 times of the running time of SSNAL, sPADMM can only solve approximately 80% of all the tested instances, while SLEP can only solve 20% of all the tested instances. We can safely claim that our algorithm SSNAL can solve large-scale SGLasso problems to high accuracy very efficiently and robustly.

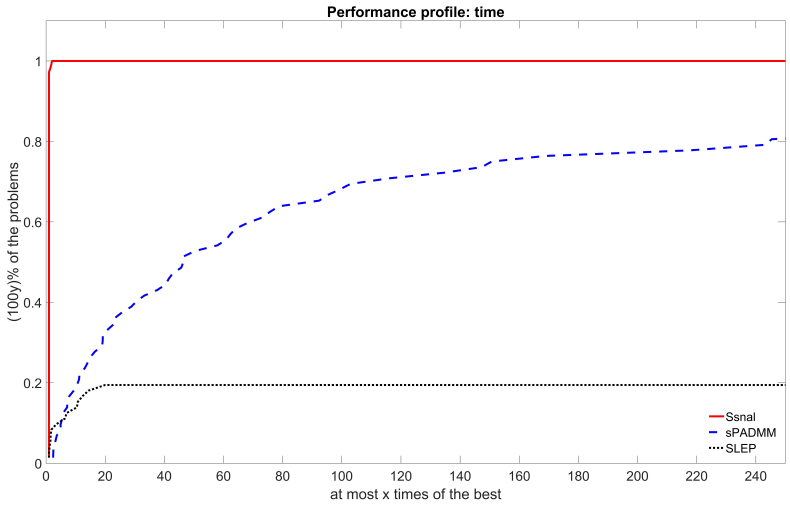

Figure 2: Performance profiles of SSNAL, sPADMM, and SLEP on UCI data sets with simulated groups.

Table 4: The performances of SSNAL, sPADMM, and SLEP on 8 selected UCI data sets with simulated groups. The regularization parameters are specified as in (S_1). "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM.

problem name (m,n);g	γ	nnz	iteration S|A|P	time S|A|P	error $\eta_S\|\eta_A\|\eta_P$
E2006.train	1e-05	1	4(9) | 36 | 16	01 | 07:30 | 01	6.4e-10 | 9.0e-07 | 4.1e-08
(16087,150360)	1e-06	34	14(28) | 33 | 10000	02 | 07:14 | 08:56	9.2e-07 | 6.9e-07 | 3.7e-05
5012	1e-07	210	25(52) | 110 | 10000	08 | 09:17 | 08:58	5.8e-08 | 7.7e-07 | 1.2e-02
E2006.test	1e-05	1	4(9) | 31 | 9	00 | 17 | 00	5.4e-09 | 9.7e-07 | 7.7e-07
(3308,150358)	1e-06	38	20(41) | 70 | 10000	01 | 20 | 02:35	3.1e-07 | 8.2e-07 | 1.3e-03
5012	1e-07	275	27(56) | 324 | 10000	02 | 33 | 02:38	1.9e-07 | 9.5e-07 | 9.8e-02
log1p.E2006.train	1e-03	15	3(13) | 2249 | 290	09 | 57:09 | 01:37	5.5e-07 | 9.9e-07 | -1.5e-04
(16087,4272227)	1e-04	253	4(25) | 1194 | 10000	29 | 38:34 | 54:46	1.4e-07 | 1.0e-06 | 1.7e-02
142408	1e-05	7821	6(32) | 391 | 10000	03:27 | 24:39 | 54:48	2.4e-07 | 9.8e-07 | 1.5e-02
log1p.E2006.test	1e-03	13	4(14) | 1572 | 284	07 | 10:16 | 47	4.6e-07 | 9.9e-07 | -2.4e-04
(3308,4272226)	1e-04	546	5(21) | 627 | 10000	14 | 04:33 | 26:56	2.1e-07 | 9.9e-07 | 4.0e-02
142408	1e-05	4874	8(33) | 300 | 10000	43 | 02:37 | 27:04	6.2e-07 | 9.8e-07 | 1.1e-01
bodyfat7	1e-04	11	12(32) | 930 | 10000	01 | 30 | 02:54	8.1e-07 | 9.9e-07 | 6.7e-04
(252,116280)	1e-05	26	19(53) | 2394 | 10000	03 | 01:16 | 02:56	4.1e-07 | 1.0e-06 | 2.2e-04
3876	1e-06	166	23(75) | 1201 | 10000	08 | 38 | 02:58	2.5e-08 | 9.9e-07 | 2.0e-04
pyrim5	1e-02	98	7(27) | 1243 | 10000	02 | 51 | 03:39	3.6e-07 | 9.9e-07 | 9.4e-02
(74,201376)	1e-03	201	12(43) | 2080 | 10000	02 | 01:27 | 03:39	2.1e-07 | 1.0e-06 | 4.2e-02
6713	1e-04	644	18(66) | 2351 | 10000	06 | 01:43 | 03:46	3.4e-07 | 1.0e-06 | 1.0e-02
triazines4	1e-02	261	10(42) | 8439 | 10000	11 | 44:25 | 26:35	4.1e-08 | 9.6e-07 | 6.8e-02
(186,635376)	1e-03	737	15(62) | 10000 | 10000	18 | 50:16 | 26:36	3.4e-08 | 1.2e-05 | 6.5e-02
21179	1e-04	1510	20(79) | 9466 | 10000	36 | 01:20:53 | 39:04	4.3e-08 | 1.0e-06 | 5.7e-02
housing7	1e-02	91	6(25) | 862 | 10000	01 | 30 | 03:00	3.6e-08 | 9.9e-07 | 2.5e-02
(506,77520)	1e-03	150	9(34) | 596 | 10000	02 | 21 | 03:01	8.0e-07 | 9.9e-07 | 9.4e-02
2584	1e-04	807	15(49) | 638 | 10000	09 | 23 | 03:01	6.0e-07 | 1.0e-06 | 3.8e-02

Table 5: The performances of SSNAL, sPADMM, and SLEP on 8 selected UCI data sets with simulated groups. The regularization parameters are specified as in (S_2). "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM.

problem name (m,n);g	γ	nnz	iteration S|A|P	time S|A|P	error $\eta_S\|\eta_A\|\eta_P$
E2006.train	1e-05	1	3(7) | 54 | 16	01 | 08:02 | 01	6.4e-10 | 7.1e-07 | 3.9e-08
(16087,150360)	1e-06	11	8(17) | 34 | 10000	01 | 07:28 | 08:46	3.0e-07 | 8.6e-07 | 1.6e-05
5012	1e-07	40	21(47) | 68 | 10000	03 | 08:15 | 08:54	8.7e-08 | 9.9e-07 | 3.3e-03
E2006.test	1e-05	2	5(11) | 42 | 10000	00 | 18 | 02:08	2.9e-07 | 7.0e-07 | 1.5e-06
(3308,150358)	1e-06	22	9(19) | 29 | 10000	01 | 17 | 02:26	3.3e-07 | 7.3e-07 | 6.6e-05
5012	1e-07	66	25(51) | 138 | 10000	01 | 24 | 02:30	5.1e-07 | 8.4e-07 | 1.5e-02
log1p.E2006.train	1e-03	11	2(12) | 2399 | 92	09 | 01:00:29 | 30	6.5e-08 | 1.0e-06 | -2.9e-04
(16087,4272227)	1e-04	39	3(16) | 2088 | 578	13 | 55:02 | 03:14	4.5e-07 | 1.0e-06 | -1.8e-05
142408	1e-05	597	4(22) | 862 | 10000	29 | 33:09 | 55:12	2.7e-07 | 9.9e-07 | 3.2e-02
log1p.E2006.test	1e-03	7	2(12) | 1567 | 60	07 | 10:20 | 10	8.6e-07 | 1.0e-06 | -4.4e-04
(3308,4272226)	1e-04	47	4(17) | 1260 | 327	08 | 08:28 | 53	1.3e-07 | 1.0e-06 | -1.3e-04
142408	1e-05	1079	5(23) | 467 | 10000	17 | 03:37 | 27:49	9.8e-07 | 9.9e-07 | 1.2e-01
bodyfat7	1e-04	26	10(24) | 748 | 10000	01 | 24 | 03:23	3.7e-07 | 1.0e-06 | 2.1e-02
(252,116280)	1e-05	43	15(37) | 1266 | 10000	01 | 41 | 03:22	6.1e-07 | 1.0e-06 | 2.4e-03
3876	1e-06	52	20(53) | 1188 | 10000	04 | 38 | 03:25	2.4e-07 | 1.0e-06 | 3.9e-04
pyrim5	1e-02	42	6(19) | 1672 | 10000	01 | 01:05 | 04:02	6.4e-08 | 1.0e-06 | 1.1e-01
(74,201376)	1e-03	136	8(32) | 1518 | 10000	01 | 59 | 04:24	1.5e-07 | 9.9e-07 | 1.2e-01
6713	1e-04	342	13(50) | 1879 | 10000	04 | 01:49 | 04:27	1.6e-07 | 1.0e-06 | 3.7e-02
triazines4	1e-02	40	8(20) | 6085 | 10000	04 | 30:37 | 26:40	1.6e-08 | 9.0e-07 | 1.1e-01
(186,635376)	1e-03	544	10(43) | 6473 | 10000	11 | 32:06 | 26:29	7.4e-08 | 9.7e-07 | 7.0e-02
21179	1e-04	964	17(63) | 10000 | 10000	18 | 49:47 | 26:52	4.1e-07 | 2.2e-06 | 8.2e-02
housing7	1e-02	51	4(15) | 1242 | 10000	00 | 38 | 03:36	5.4e-08 | 9.8e-07 | 5.1e-02
(506,77520)	1e-03	153	7(26) | 853 | 10000	01 | 26 | 03:36	1.2e-07 | 1.0e-06 | 5.0e-02
2584	1e-04	175	10(34) | 577 | 10000	02 | 18 | 03:34	1.2e-07 | 9.8e-07 | 1.3e-01

Table 6: The performances of SSNAL, sPADMM, and SLEP on 8 selected UCI data sets with simulated groups. The regularization parameters are specified as in (S_3). "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM.

problem name (m,n);g	γ	nnz	iteration S|A|P	time S|A|P	error $\eta_S\|\eta_A\|\eta_P$
E2006.train	1e-05	1	4(9) | 34 | 16	01 | 06:57 | 01	8.9e-10 | 7.8e-07 | 8.3e-07
(16087,150360)	1e-06	27	22(45) | 69 | 10000	03 | 07:52 | 10:14	1.7e-07 | 8.7e-07 | 3.8e-03
5012	1e-07	1399	30(79) | 395 | 10000	01:27 | 11:23 | 09:48	8.5e-07 | 9.8e-07 | 8.6e-02
E2006.test	1e-05	1	4(10) | 29 | 12	00 | 16 | 00	8.8e-09 | 8.8e-07 | 7.4e-07
(3308,150358)	1e-06	48	25(51) | 182 | 10000	01 | 25 | 02:26	5.1e-08 | 9.5e-07 | 2.3e-02
5012	1e-07	1325	38(103) | 1432 | 10000	31 | 01:17 | 02:42	8.7e-09 | 8.6e-07 | 3.3e-01
log1p.E2006.train	1e-03	5	4(17) | 2451 | 626	13 | 58:57 | 04:11	7.8e-09 | 9.8e-07 | -7.5e-06
(16087,4272227)	1e-04	510	5(26) | 823 | 10000	29 | 31:05 | 01:01:43	5.5e-07 | 9.9e-07 | 1.1e-02
142408	1e-05	9772	7(33) | 340 | 10000	04:37 | 22:48 | 01:02:05	3.6e-08 | 1.0e-06 | 1.2e-02
log1p.E2006.test	1e-03	8	5(22) | 1688 | 732	12 | 10:22 | 02:09	5.0e-09 | 9.9e-07 | -3.1e-05
(3308,4272226)	1e-04	909	6(27) | 470 | 10000	18 | 03:25 | 29:31	1.0e-07 | 9.8e-07 | 5.4e-02
142408	1e-05	4956	9(35) | 288 | 10000	44 | 02:22 | 30:13	6.6e-08 | 9.6e-07 | 1.2e-01
bodyfat7	1e-04	3	12(34) | 1101 | 1163	02 | 34 | 28	2.4e-07 | 9.9e-07 | 9.5e-07
(252,116280)	1e-05	24	19(58) | 1586 | 10000	05 | 49 | 05:16	8.6e-07 | 1.0e-06 | 2.1e-06
3876	1e-06	106	25(94) | 2231 | 10000	12 | 01:09 | 03:39	5.2e-07 | 1.0e-06 | 4.7e-03
pyrim5	1e-02	87	9(32) | 1382 | 10000	01 | 55 | 03:51	4.3e-08 | 1.0e-06 | 7.1e-02
(74,201376)	1e-03	176	16(56) | 5642 | 10000	04 | 03:42 | 03:46	5.6e-07 | 1.0e-06 | 5.8e-03
6713	1e-04	129	26(95) | 10000 | 10000	09 | 06:31 | 03:52	5.3e-07 | 4.1e-05 | 1.2e-03
triazines4	1e-02	246	10(37) | 8369 | 10000	09 | 43:00 | 26:58	5.1e-08 | 9.2e-07 | 6.6e-02
(186,635376)	1e-03	803	20(72) | 10000 | 10000	23 | 50:09 | 27:06	8.2e-09 | 3.7e-06 | 3.4e-02
21179	1e-04	333	27(115) | 10000 | 10000	01:04 | 46:21 | 27:21	3.9e-07 | 1.4e-04 | 5.5e-02
housing7	1e-02	50	7(30) | 976 | 10000	01 | 30 | 04:07	1.3e-07 | 1.0e-06 | 1.2e-02
(506,77520)	1e-03	157	11(41) | 620 | 10000	03 | 19 | 04:03	1.1e-07 | 9.9e-07 | 6.7e-02
2584	1e-04	838	16(51) | 685 | 10000	08 | 21 | 04:11	8.7e-08 | 1.0e-06 | 3.8e-02

5.5 NCEP/NCAR reanalysis 1 dataset

This section evaluates the performance of SSNAL, sPADMM, SLEP and BCD on the NCEP/NCAR reanalysis 1 dataset [25] . The data set contains the monthly means of climate data measurements spread across the globe in a grid of $2.5^o \times 2.5^o$ resolutions (longitude and latitude 144×73) from 1948/1/1 to 2018/5/31. Each grid point (location) constitutes a group of 7 predictive variables (Air Temperature, Precipitable Water, Relative Humidity, Pressure, Sea Level Pressure, Horizontal Wind Speed and Vertical Wind Speed). Such data sets have a natural group structure: 144×73 groups, where each group is of length 7, and the corresponding data matrix $\mathcal{A}$ is of dimension 845×73584.

Following the numerical experiment in [39], we also consider as target variable $b \in \mathbb{R}^{845}$, the values of Air Temperature in a neighborhood of Dakar. We also take a decreasing sequence of 100 regularization parameters defined as follows:

$$\bar{\lambda}_t = \lambda_{\max} 10^{-3(t-1)/(100-1)},\ (\lambda_1, \lambda_2) \in \{(0.4\bar{\lambda}_t, 0.6\bar{\lambda}_t) \,|\, t = 1, 2, \ldots, 100\},$$

where $\lambda_{\max} = \Omega^D(\mathcal{A}^T b)$, and Ω^D is the dual norm of p that is defined by $\Omega^D(y) := \max_{p(x)\le 1} x^T y$. In total, there are 100 pairs of decreasing λ_1 and λ_2 that will lead to a solution path.

We find that BCD is terminated if

$$\text{pobj} - \text{dobj} < \varepsilon \|b\|^2,$$

or the default maximum number of 29,999 iterations is reached. In the same way, we terminate SSNAL if

$$\frac{\|\mathcal{A}^* y + z\|}{1 + \|z\|} < \varepsilon,\ \text{pobj} - \text{dobj} < \varepsilon \|b\|^2. \tag{36}$$

We terminate sPADMM if (36) holds or the maximum number of 10,000 iterations is reached. Besides, we terminate SLEP if the difference of the optimal objective values between SLEP and SSNAL is less than ε, i.e.,

$$\text{obj}_\text{P} - \text{obj}_\text{S} < \varepsilon \|b\|^2,$$

or the maximum number of 10,000 iterations is reached.

Table 7 presents the comparison of SSNAL, sPADMM, SLEP, and BCD on the climate data along a solution path. As revealed by Table 7, for the case $\varepsilon = 10^{-4}$ where the accuracy is relatively low ($\|b\|^2 \approx 5 \times 10^5$), both BCD and SSNAL have successfully solve all cases along the path; while sPADMM took more than 3 hours and solved 85% of all cases, and SLEP took more than 5 hours and merely solved 21% of all cases. One might notice that in this case the duality gap is allowed to be about 50. For low accuracy requirement, the BCD algorithm in [39] is highly efficient, but our algorithm SSNAL can also make it within 10 minutes. In addition, for the cases with tolerance 10^{-6} and 10^{-8}, SSNAL has successfully solved all cases within 12 minutes; while all the other algorithms failed to solve some cases along the solution path. We can see that our algorithm SSNAL has a clear advantage over the other first order algorithms when one wants moderate or high accuracy solutions. We can safely conclude that SSNAL is efficient and robust on the real climate data set.

Table 7: The performances of SSNAL, ADMM, SLEP, BCD on climate data along a solution path. "success" denotes the number of cases which are solved successfully among all the 100 cases along the solution path. "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM; "B" for BCD.

tolerance ε	time S|A|P|B	success S|A|P|B
1e-4	09:02 | 03:34:49 | 05:49:56 | 03:51	100 | 85 | 21 | 100
1e-6	11:27 | 09:42:48 | 06:21:05 | 01:10:54	100 | 21 | 18 | 93
1e-8	11:40 | 10:40:17 | 06:36:49 | 01:39:01	100 | 16 | 16 | 93

6 Conclusion

In this paper, we have developed a highly efficient semismooth Newton based augmented Lagrangian method SSNAL for solving large-scale non-overlapping sparse group Lasso problems. The elements in the generalized Jacobian of the proximal mapping associated with the sparse group Lasso regularizer were first derived, and the underlying second order sparsity structure was thoroughly analysed and utilised to achieve superior performance in the numerical implementations of SSNAL. Extensive numerical experiments have demonstrated that the proposed algorithm is highly efficient and robust, even on high-dimensional real data sets. Based on the superior performance of SSNAL for solving non-overlapping sparse group Lasso problems, we can expect the effectiveness of our algorithmic framework for solving overlapping sparse group Lasso problems and other large-scale convex composite problems in future studies.

Acknowledgments

The authors would like to thank Dr. Xudong Li and Ms. Meixia Lin for their help in the numerical implementations. We also thank the referees for their valuable suggestions which have helped to improve the paper.

References

[1] G. Andrew and J. Gao. Scalable training of L_1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM, 2007.

[2] A. Argyriou, C. A. Micchelli, M. Pontil, L. Shen, and Y. Xu. Efficient first order methods for linear composite regularizers. *arXiv preprint arXiv:1104.1436*, 2011.

[3] J. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization: theory and examples*. Springer Science & Business Media, 2010.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[5] R. H. Byrd, G. M. Chin, J. Nocedal, and F. Oztoprak. A family of second-order methods for convex ℓ_1-regularized optimization. *Mathematical Programming*, 159(1-2):435–467, 2016.

[6] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[7] L. Chen, D. F. Sun, and K.-C. Toh. An efficient inexact symmetric Gauss–Seidel based majorized ADMM for high-dimensional convex composite conic programming. *Mathematical Programming*, 161(1-2):237–270, 2017.

[8] X. D. Chen, D. F. Sun, and J. Sun. Complementarity functions and numerical experiments on some smoothing Newton methods for second-order-cone complementarity problems. *Computational Optimization and Applications*, 25(1):39–56, 2003.

[9] F. H. Clarke. *Optimization and nonsmooth analysis*. SIAM, 1990.

[10] Y. Cui, D. F. Sun, and K.-C. Toh. On the asymptotic superlinear convergence of the augmented Lagrangian method for semidefinite programming with multiple solutions. *arXiv preprint arXiv:1610.00875*, 2016.

[11] Y. Cui, D. F. Sun, and K.-C. Toh. On the R-superlinear convergence of the KKT residues generated by the augmented Lagrangian method for convex composite conic programming. *Mathematical Programming (arXiv preprint arXiv:1706.08800)*, 2018.

[12] J. C. De Los Reyes, E. Loayza, and P. Merino. Second-order orthant-based methods with enriched Hessian information for sparse ℓ_1-optimization. *Computational Optimization and Applications*, 67(2):225–258, 2017.

[13] Y. Dong. An extension of Luque's growth condition. *Applied Mathematics Letters*, 22(9):1390–1393, 2009.

[14] Y. C. Eldar and M. Mishali. Robust recovery of signals from a structured union of subspaces. *IEEE Transactions on Information Theory*, 55(11):5302–5316, 2009.

[15] F. Facchinei and J.-S. Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007.

[16] M. Fazel, T. K. Pong, D. F. Sun, and P. Tseng. Hankel matrix rank minimization with applications to system identification and realization. *SIAM Journal on Matrix Analysis and Applications*, 34(3):946–977, 2013.

[17] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[18] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[19] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40, 1976.

[20] R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):41–76, 1975.

[21] D. Hallac, J. Leskovec, and S. Boyd. Network lasso: Clustering and optimization in large graphs. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 387–396. ACM, 2015.

[22] L. Huang, J. Jia, B. Yu, B.-G. Chun, P. Maniatis, and M. Naik. Predicting execution time of computer programs using sparse polynomial regression. In *Advances in Neural Information Processing Systems*, pages 883–891, 2010.

[23] L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual International Conference on Machine Learning*, pages 433–440. ACM, 2009.

[24] R. Jenatton, J. Mairal, F. R. Bach, and G. R. Obozinski. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 487–494, 2010.

[25] E. Kalnay, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen, Y. Zhu, M. Chelliah, W. Ebisuzaki, W. Higgins, J. Janowiak, K. C. Mo, C. Ropelewski, J. Wang, A. Leetmaa, R. Reynolds, R. Jenne, and D. Joseph. The NCEP/NCAR 40-year reanalysis project. *Bulletin of the American Meteorological Society*, 77(3):437–472, 1996.

[26] J. Kim and H. Park. Fast active-set-type algorithms for l_1-regularized linear regression. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 397–404, 2010.

[27] D. Kong and C. Ding. Efficient algorithms for selecting features with arbitrary group constraints via group lasso. In *2013 IEEE 13th International Conference on Data Mining*, pages 379–388, 2013.

[28] B. Kummer. Newton's method for non-differentiable functions. *Advances in Mathematical Optimization*, 45:114–125, 1988.

[29] B. Kummer. Newton's method based on generalized derivatives for nonsmooth functions: convergence analysis. In *Advances in Optimization*, pages 171–194. Springer, 1992.

[30] X. Li, D. F. Sun, and K.-C. Toh. A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems. *SIAM Journal on Optimization*, 28(1):433–458, 2018.

[31] X. Li, D. F. Sun, and K.-C. Toh. On efficiently solving the subproblems of a level-set method for fused lasso problems. *SIAM Journal on Optimization*, 28(2):1842–1862, 2018.

[32] M. Lichman. UCI machine learning repository, 2013.

[33] J. Liu, S. Ji, and J. Ye. SLEP: Sparse Learning with Efficient Projections. *Arizona State University*, 6:491, 2009.

[34] J. Liu and J. Ye. Moreau-Yosida regularization for grouped tree structure learning. In *Advances in Neural Information Processing Systems*, pages 1459–1467, 2010.

[35] Z.-Q. Luo and P. Tseng. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46(1):157–178, 1993.

[36] F. J. Luque. Asymptotic convergence analysis of the proximal point algorithm. *SIAM Journal on Control and Optimization*, 22(2):277–293, 1984.

[37] F. Meng, D. F. Sun, and G. Zhao. Semismoothness of solutions to generalized equations and the Moreau-Yosida regularization. *Mathematical Programming*, 104(2):561–581, 2005.

[38] R. Mifflin. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977.

[39] E. Ndiaye, O. Fercoq, A. Gramfort, and J. Salmon. Gap safe screening rules for sparse-group lasso. In *Advances in Neural Information Processing Systems*, pages 388–396, 2016.

[40] J. Peng, J. Zhu, A. Bergamaschi, W. Han, D.-Y. Noh, J. R. Pollack, and P. Wang. Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer. *The Annals of Applied Statistics*, 4(1):53, 2010.

[41] L. Qi. Convergence analysis of some algorithms for solving nonsmooth equations. *Mathematics of Operations Research*, 18(1):227–244, 1993.

[42] L. Qi and J. Sun. A nonsmooth version of Newton's method. *Mathematical Programming*, 58(1):353–367, 1993.

[43] Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2):143–169, 2013.

[44] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[45] R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2):97–116, 1976.

[46] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[47] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1997.

[48] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

[49] D. F. Sun. The strong second-order sufficient condition and constraint nondegeneracy in nonlinear semidefinite programming and their implications. *Mathematics of Operations Research*, 31(4):761–776, 2006.

[50] D. F. Sun and J. Sun. Semismooth matrix-valued functions. *Mathematics of Operations Research*, 27(1):150–169, 2002.

[51] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[52] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[53] Y.-L. Yu. On decomposing the proximal map. In *Advances in Neural Information Processing Systems*, pages 91–99, 2013.

[54] L. Yuan, J. Liu, and J. Ye. Efficient methods for overlapping group lasso. In *Advances in Neural Information Processing Systems*, pages 352–360, 2011.

[55] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[56] H. Zhang, J. Jiang, and Z.-Q. Luo. On the linear convergence of a proximal gradient method for a class of nonsmooth convex minimization problems. *Journal of the Operations Research Society of China*, 1(2):163–186, 2013.

[57] X. Y. Zhao, D. F. Sun, and K.-C. Toh. A Newton-CG augmented Lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20(4):1737–1765, 2010.

[58] Y. Zhou, J. Han, X. Yuan, Z. Wei, and R. Hong. Inverse sparse group lasso model for robust object tracking. *IEEE Transactions on Multimedia*, 2017.