SIAM J. OPTIM.
Vol. 29, No. 3, pp. 2026–2052

EFFICIENT SPARSE SEMISMOOTH NEWTON METHODS FOR THE CLUSTERED LASSO PROBLEM*

MEIXIA LIN†, YONG-JIN LIU‡, DEFENG SUN§, AND KIM-CHUAN TOH¶

Abstract. We focus on solving the clustered Lasso problem, which is a least squares problem with the ℓ_1-type penalties imposed on both the coefficients and their pairwise differences to learn the group structure of the regression parameters. Here we first reformulate the clustered Lasso regularizer as a weighted ordered-Lasso regularizer, which is essential in reducing the computational cost from $O(n^2)$ to $O(n\log(n))$. We then propose an inexact semismooth Newton augmented Lagrangian (SSNAL) algorithm to solve the clustered Lasso problem or its dual via this equivalent formulation, depending on whether the sample size is larger than the dimension of the features. An essential component of the SSNAL algorithm is the computation of the generalized Jacobian of the proximal mapping of the clustered Lasso regularizer. Based on the new formulation, we derive an efficient procedure for its computation. Comprehensive results on the global convergence and local linear convergence of the SSNAL algorithm are established. For the purpose of exposition and comparison, we also summarize/design several first-order methods that can be used to solve the problem under consideration, but with the key improvement from the new formulation of the clustered Lasso regularizer. As a demonstration of the applicability of our algorithms, numerical experiments on the clustered Lasso problem are performed. The experiments show that the SSNAL algorithm substantially outperforms the best alternative algorithm for the clustered Lasso problem.

Key words. clustered Lasso, augmented Lagrangian method, semismooth Newton method, convex minimization

AMS subject classifications. 90C06, 90C25, 90C90

DOI. 10.1137/18M1207752

1. Introduction.

We consider a minimization problem of the following form:

$$\min_{x\in\Re^n}\left\{\frac{1}{2}\|Ax-b\|^2+\beta\|x\|_1+\rho\sum_{1\le i<j\le n}|x_i-x_j|\right\}, \tag{1.1}$$

where $A\in\Re^{m\times n}$, $b\in\Re^m$ are given data, and $\beta,\rho>0$ are given positive parameters. For $x\in\Re^n$, $\|x\|_1=\sum_{i=1}^n|x_i|$. Obviously, *the optimal solution set of problem* (1.1)*, denoted by* Ω_p*, is nonempty and bounded.* Problems of the form (1.1) are called the clustered Lasso problems, which are motivated by the desire to learn the group

*Received by the editors August 16, 2018; accepted for publication (in revised form) April 24, 2019; published electronically July 30, 2019.
https://doi.org/10.1137/18M1207752

Funding: The research of the second author was supported in part by the National Natural Science Foundation of China under grants 11371255 and 11871153, and the Natural Science Foundation of Fujian Province of China under grant 2019J01644. The research of the third author was supported in part by a start-up research grant from the Hong Kong Polytechnic University. The research of the fourth author was supported in part by the Ministry of Education, Singapore, Academic Research Fund under grant R-146-000-257-112.

†Department of Mathematics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore (lin_meixia@u.nus.edu).

‡Corresponding author. Key Laboratory of Operations Research and Control of Universities in Fujian, College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, People's Republic of China (yjliu@fzu.edu.cn).

§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 (mattohkc@nus.edu.sg).

structure of the regression parameters $\{x_i\}$ in the statistical context [30, 24]. Two types of sparsity are desirable: zero-sparsity and equi-sparsity. The clustered Lasso model is proposed with ℓ_1-type penalties imposed on both the coefficients and their pairwise differences.

It is worthwhile mentioning several other popular models for group sparsity of the regression parameters. The fused Lasso model [35, 38, 20], which was developed for ordered predictors, penalizes the differences between adjacent predictors. The group Lasso model [40, 10, 15] assumes that the grouping of the predictors is known, say from the underlying background, and then penalizes the ℓ_2-norm of the coefficients within the same predictor group. The OSCAR model [3, 44] penalizes the combination of the ℓ_1-norm and a pairwise ℓ_∞-norm for the coefficients. OSCAR is similar to the clustered Lasso since it seeks zero-sparsity and equi-sparsity in $\{|x_i|\}$. All these models are extended from the original Lasso model [34, 36, 37] to obtain minimal prediction error and also to recover the true underlying specific structure of the model.

The clustered Lasso model has been applied in microarray data analysis. In addition, the clustered Lasso can be used as a pre-processing step for the fused Lasso or the group Lasso for uncovering the group structure of the predictors. Researchers have designed some algorithms for solving (1.1) through reformulating (1.1) as a constrained Lasso problem by introducing new variables in [30, 24, 33]. Unfortunately, these methods can hardly be applied to large-scale problems due to huge computational cost.

In real applications, one may need to run the clustered Lasso problem (1.1) many times with different (β, ρ) when tuning parameters to get reasonable sparsity structure of the predictors. Therefore, it is important for us to design an efficient and robust algorithm, especially for the high-dimensional and/or high-sample cases. In order to achieve fast convergence, we aim to solve the clustered Lasso problem by designing a method which exploits the second-order information. Specifically, we will design a semismooth Newton augmented Lagrangian method, which has already been demonstrated to be extremely efficient for Lasso [17], fused Lasso [18], group Lasso [42], and OSCAR [21].

The main contributions of our paper can be summarized as follows.

1. We reformulate the clustered Lasso regularizer as a weighted ordered-Lasso regularizer, which is crucial to reducing the cost of computing the regularizer from $O(n^2)$ to $O(n\log(n))$ operations. Based on this reformulation, we are able to compute the proximal mapping of the clustered Lasso regularizer by using the pool-adjacent-violators algorithm in $O(n\log(n))$ operations. As far as we are aware of, this is the first time that the proximal mapping of the clustered Lasso regularizer is shown to be computable in $O(n\log(n))$ operations.
2. The new formulation is also critical for us to obtain a well-structured generalized Jacobian of the corresponding proximal mapping so that it can be computed explicitly and efficiently with the structure to be mentioned in section 2.3.
3. We propose a semismooth Newton augmented Lagrangian (Ssnal) method for solving problem (1.1) or its dual depending on whether the sample size is larger than the dimension of the features. Since the objective function in (1.1) is piecewise linear-quadratic, the augmented Lagrangian method (Alm) is proved to have the asymptotic superlinear convergence property from [28, 29, 17]. For the Alm subproblem, we employ a semismooth Newton method that exploits the second-order sparsity of the generalized Jacobian of the

proximal mapping of the clustered Lasso regularizer to get fast superlinear or even quadratic convergence.

4. As first-order methods have been very popular in solving various Lasso-type problems in recent years, we summarize two first-order algorithms which can be used to solve problem (1.1). The computation of the key projection step is highly improved due to our new formulation of the clustered Lasso regularizer.
5. We conduct comprehensive numerical experiments to demonstrate the efficiency and robustness of the SSNAL method against different parameter settings. We also demonstrate the superior performance of our algorithm over other first-order methods for large-scale instances with $n \gg m$.

The remaining parts of this paper are organized as follows. The next section is devoted to computing and analyzing the proximal mapping of the clustered Lasso regularizer and its generalized Jacobian. In sections 3 and 4, we develop semismooth Newton-based augmented Lagrangian algorithms to solve the clustered Lasso problem and its dual problem, respectively. We employ various numerical techniques to efficiently exploit the second-order sparsity and special structure of the generalized Jacobian when implementing the SSNAL algorithms. For the purpose of evaluating the efficiency of our SSNAL algorithms, in section 5.1 we summarize two first-order algorithms which are conducive to solving the general problem (1.1). By using the proposed proximal mapping of the clustered Lasso regularizer to be given in section 2.1, one can compute the key projection step in these two first-order methods efficiently in $O(n\log(n))$ operations. This is already a significant improvement over the current methods in [30, 24, 33], which require $O(n^2)$ just to evaluate the clustered Lasso regularizer. The numerical performance of our SSNAL algorithms for the clustered Lasso problems on large-scale real data and synthetic data against other state-of-the-art algorithms is presented in section 5. We conclude our paper in the final section.

Notation. Throughout the paper, we use "diag(X)" to denote the vector consisting of the diagonal entries of the matrix X and "Diag(x)" to denote the diagonal matrix whose diagonal is given by the vector x. We denote by I_n, $\mathbf{O}_n$, and $\mathbf{E}_n$ the $n\times n$ identity matrix, the $n\times n$ zero matrix, and the $n\times n$ matrix of all ones, respectively. For given matrix C, we also use $C^\dagger$ to represent its Moore–Penrose pseudoinverse. As usual, f^* is the Fenchel conjugate of an arbitrary function f.

2. Computing the proximal mapping of the clustered Lasso regularizer and its generalized Jacobian. For convenience, we denote the clustered Lasso regularizer in (1.1) by

$$p(x) = \beta\|x\|_1 + \rho \sum_{1\le i<j\le n} |x_i - x_j| \ \forall x \in \Re^n.$$

Let $f : \Re^n \to (-\infty, \infty]$ be any given proper closed convex function. Then, the proximal mapping $\mathrm{Prox}_f(\cdot)$ of f is defined as

$$\mathrm{Prox}_f(y) = \underset{x\in\Re^n}{\mathrm{argmin}}\Big\{\frac{1}{2}\|x-y\|^2 + f(x)\Big\} \ \forall y \in \Re^n.$$

We have the following important Moreau identity:

$$\mathrm{Prox}_{tf}(x) + t\mathrm{Prox}_{f^*/t}(x/t) = x,$$

where $t > 0$ is a given parameter.

In this section, we shall develop some useful results on calculating the proximal mapping of the clustered Lasso regularizer $p(\cdot)$ and the corresponding generalized Jacobian.

2.1. The computation of the proximal mapping $\mathbf{Prox}_{p}(\cdot)$. Define

$$S_\rho(y) := \operatorname*{argmin}_{x\in\Re^n} \left\{ \frac{1}{2}\|x-y\|^2 + \rho \sum_{1\le i<j\le n} |x_i - x_j| \right\} \ \forall y \in \Re^n$$

and $\mathcal{D} = \{x \in \Re^n \mid Bx \ge 0\}$, where B is a matrix such that

$$Bx = [x_1 - x_2; \ldots; x_{n-1} - x_n] \in \Re^{n-1}.$$

We shall reformulate the clustered Lasso regularizer as a weighted ordered-Lasso regularizer, which enables us to reduce the cost of computing the regularizer from $O(n^2)$ to $O(n\log(n))$ operations. For any $x \in \Re^n$, we define $x^\downarrow$ to be the vector whose components are those of x sorted in a nonincreasing order, i.e., $x_1^\downarrow \ge x_2^\downarrow \ge \cdots \ge x_n^\downarrow$.

PROPOSITION 2.1. *Let $x \in \Re^n$ be an arbitrarily given vector. Then it holds that*

$$g(x) := \sum_{1\le i<j\le n} |x_i - x_j| = \langle w, x^\downarrow \rangle,$$

where the vector $w \in \Re^n$ is defined by

$$w_k = n - 2k + 1, \ k = 1, \ldots, n. \tag{2.1}$$

Proof. By noting that $g(x) = g(Px)$ for any permutation matrix P, one has that

$$\begin{aligned} g(x) &= \sum_{1\le i<j\le n} |x_i^\downarrow - x_j^\downarrow| = \sum_{1\le i<j\le n} (x_i^\downarrow - x_j^\downarrow) \\ &= \sum_{i=1}^{n-1} (n-i) x_i^\downarrow - \sum_{j=2}^{n} (j-1) x_j^\downarrow = \sum_{k=1}^{n} (n-2k+1) x_k^\downarrow, \end{aligned}$$

which completes the proof. ∎

Remark 2.2. As a side note, the result in Proposition 2.1 is *not* valid for a nonuniformly weighted sum.

The next proposition shows that if a vector $y \in \Re^n$ is sorted in a nonincreasing order, $S_\rho(y)$ can be computed by a single metric projection onto $\mathcal{D}$.

PROPOSITION 2.3. *Suppose that $y \in \Re^n$ is given such that $y_1 \ge y_2 \ge \cdots \ge y_n$. Then it holds that*

$$S_\rho(y) = \Pi_{\mathcal{D}}(y - \rho w),$$

where $w \in \Re^n$ is given in (2.1). *The metric projection onto $\mathcal{D}$ can be computed via the pool-adjacent-violators algorithm* [2].

Proof. Let $g(\cdot)$ be defined in Proposition 2.1. We first note that $g(x) = g(Px)$ for any permutation matrix P and $x \in \Re^n$. For convenience, let $x^* = S_\rho(y)$. Next we show that the components of x^* must be arranged in a nonincreasing order. Suppose

on the contrary that there exists $i < j$ such that $x_i^* < x_j^*$. We define $\bar{x} \in \Re^n$ by $\bar{x}_i = x_j^*$, $\bar{x}_j = x_i^*$, $\bar{x}_k = x_k^*$ for all $k \neq i, j$. Then, we derive that

$$\begin{aligned}&\frac{1}{2}\|x^* - y\|^2 + \rho g(x^*) - \Big(\frac{1}{2}\|\bar{x} - y\|^2 + \rho g(\bar{x})\Big)\\ &= \frac{1}{2}\Big((x_i^* - y_i)^2 + (x_j^* - y_j)^2 - (x_j^* - y_i)^2 - (x_i^* - y_j)^2\Big) = (x_j^* - x_i^*)(y_i - y_j) \geq 0,\end{aligned}$$

which implies that $\bar{x}$ is also a minimizer. By the uniqueness of the minimizer, we have that $\bar{x} = x^*$ and hence $x_j^* = \bar{x}_i = x_i^*$, which is a contradiction. Hence, we obtain that

$$\begin{aligned}x^* &= \underset{x\in\Re^n}{\operatorname{argmin}} \Big\{\frac{1}{2}\|x - y\|^2 + \rho g(x) \;\Big|\; x_1 \geq x_2 \geq \cdots \geq x_n\Big\}\\ &= \underset{x\in\Re^n}{\operatorname{argmin}} \Big\{\frac{1}{2}\|x - y\|^2 + \rho\langle w, x\rangle \;\Big|\; x_1 \geq x_2 \geq \cdots \geq x_n\Big\}\\ &= \underset{x\in\Re^n}{\operatorname{argmin}} \Big\{\frac{1}{2}\|x - (y - \rho w)\|^2 \;\Big|\; x_1 \geq x_2 \geq \cdots \geq x_n\Big\}\\ &= \Pi_{\mathcal{D}}(y - \rho w).\end{aligned}$$

The proof is complete. ∎

Combining Proposition 2.1 with Proposition 2.3, we can get an explicit formula for $S_\rho(\cdot)$. Let $y \in \Re^n$ be given. Then there exists a permutation matrix $P_y \in \Re^{n\times n}$ such that $\tilde{y} = P_y y$ and $\tilde{y}_1 \geq \tilde{y}_2 \geq \cdots \geq \tilde{y}_n$. Thus,

$$S_\rho(y) = P_y^T S_\rho(\tilde{y}) = P_y^T \Pi_{\mathcal{D}}(\tilde{y} - \rho w) = P_y^T \Pi_{\mathcal{D}}(P_y y - \rho w).$$

Next we recall an important result on computing $\mathrm{Prox}_p(\cdot)$, which comes from [39, Corollary 4].

PROPOSITION 2.4. *Let $y \in \Re^n$ be given. Then, we have that*

$$\mathrm{Prox}_p(y) = \mathrm{Prox}_{\beta\|\cdot\|_1}(S_\rho(y)) = \mathrm{sign}(S_\rho(y)) \circ \max(|S_\rho(y)| - \beta, 0),$$

where "$\circ$" denotes the Hadamard product.

The above proposition states that the proximal mapping of the clustered Lasso regularizer can be decomposed into the composition of the proximal mapping of $\beta\|\cdot\|_1$ and the proximal mapping of $\rho g(\cdot)$.

2.2. The computation of the generalized Jacobian of $\mathrm{Prox}_p(\cdot)$. We first present some results on the generalized HS-Jacobian of $\Pi_{\mathcal{D}}$, which can be obtained directly from the previous work in [14], wherein Han and Sun constructed a theoretically computable generalized Jacobian of the metric projector over a polyhedral set. Recently, Li, Sun, and Toh [18] further derived an efficient formula for computing a special HS-Jacobian of the solution mapping of a parametric strongly convex quadratic programming. In this section, we will adapt the ideas in [18] to efficiently compute the generalized Jacobian of $\Pi_{\mathcal{D}}(\cdot)$.

Since $\Pi_{\mathcal{D}}$ is the metric projection onto the nonempty polyhedral set $\mathcal{D}$, for any given $y \in \Re^n$, there exists a multiplier $\lambda \in \Re^{n-1}$ such that the following KKT system holds:

$$\begin{cases}\Pi_{\mathcal{D}}(y) - y + B^T\lambda = 0,\\ B\Pi_{\mathcal{D}}(y) \geq 0,\ \lambda \leq 0,\\ \lambda^T B\Pi_{\mathcal{D}}(y) = 0.\end{cases} \tag{2.2}$$

Let $\mathcal{M}_{\mathcal{D}}(y) := \{\lambda \in \Re^{n-1} \mid (y,\lambda) \text{ satisfies (2.2)}\}$. Since $\mathcal{M}_{\mathcal{D}}(y)$ is a nonempty polyhedral convex set which contains no lines, it has at least one extreme point [27, Corollary 18.5.3]. Denote the active index set by

$$\mathcal{I}_{\mathcal{D}}(y) := \{i \mid B_i \Pi_{\mathcal{D}}(y) = 0,\ i = 1, 2, \ldots, n-1\}, \tag{2.3}$$

where B_i is the ith row of B. Define a collection of index subsets of $\{1, \ldots, n-1\}$ as follows:

$$\mathcal{K}_{\mathcal{D}}(y) := \Big\{ K \ \Big|\ \exists \lambda \in \mathcal{M}_{\mathcal{D}}(y) \text{ s.t. } \mathrm{supp}(\lambda) \subseteq K \subseteq \mathcal{I}_{\mathcal{D}}(y),\ B_K \text{ is of full row rank} \Big\},$$

where $\mathrm{supp}(\lambda)$ denotes the support of λ and B_K is the matrix consisting of the rows of B indexed by K. It should be noted that $\mathcal{K}_{\mathcal{D}}(y)$ is nonempty due to the existence of an extreme point of $\mathcal{M}_{\mathcal{D}}(y)$, as stated in [14]. Han and Sun in [14] introduced the multifunction $\mathcal{Q}_{\mathcal{D}} : \Re^n \rightrightarrows \Re^{n\times n}$ defined by

$$\mathcal{Q}_{\mathcal{D}}(y) := \Big\{ \widehat{Q} \in \Re^{n\times n} \ \Big|\ \widehat{Q} = I_n - B_K^T (B_K B_K^T)^{-1} B_K,\ K \in \mathcal{K}_{\mathcal{D}}(y) \Big\},$$

which is called the generalized HS-Jacobian of $\Pi_{\mathcal{D}}$ at y. From [19, Proposition 1 and Theorem 1], we can readily get the following proposition, whose proof is omitted for brevity.

PROPOSITION 2.5. *For any $y \in \Re^n$, there exists a neighborhood $\mathcal{Y}$ of y such that*

$$\mathcal{K}_{\mathcal{D}}(u) \subseteq \mathcal{K}_{\mathcal{D}}(y),\ \mathcal{Q}_{\mathcal{D}}(u) \subseteq \mathcal{Q}_{\mathcal{D}}(y)\ \forall u \in \mathcal{Y}$$

and

$$\Pi_{\mathcal{D}}(u) = \Pi_{\mathcal{D}}(y) + \widehat{Q}(u - y)\ \forall \widehat{Q} \in \mathcal{Q}_{\mathcal{D}}(u).$$

Thus, $\partial_B \Pi_{\mathcal{D}}(y) \subseteq \mathcal{Q}_{\mathcal{D}}(y)$, where $\partial_B \Pi_{\mathcal{D}}(y)$ is the B-subdifferential of $\Pi_{\mathcal{D}}$ at y. In particular, $\widehat{Q}_{\mathcal{D},0}(y) \in \mathcal{Q}_{\mathcal{D}}(y)$, where

$$\widehat{Q}_{\mathcal{D},0}(y) := I_n - B_{\mathcal{I}_{\mathcal{D}}(y)}^T \left(B_{\mathcal{I}_{\mathcal{D}}(y)} B_{\mathcal{I}_{\mathcal{D}}(y)}^T \right)^{\dagger} B_{\mathcal{I}_{\mathcal{D}}(y)}.$$

Next, we propose a simple and useful result for our further discussions. Given $y \in \Re^n$ and $K \subseteq \{1, \ldots, n-1\}$, we provide an alternative way to compute $I_n - B_K^T (B_K B_K^T)^{\dagger} B_K$. Let $\Sigma_K = \mathrm{Diag}(\sigma_K) \in \Re^{(n-1)\times(n-1)}$ be defined by

$$(\sigma_K)_i = \begin{cases} 1 & \text{if } i \in K, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 1, 2, \ldots, n-1.$$

By using the fact that there exists a permutation matrix P_K such that

$$\begin{bmatrix} B_K \\ \mathbf{0} \end{bmatrix}_{(n-1)\times n} = P_K \Sigma_K B = \begin{bmatrix} I_{|K|} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{(n-1)\times(n-1)} P_K B,$$

one can easily prove the following proposition, which will be used later.

PROPOSITION 2.6. *It holds that*

$$I_n - B_K^T (B_K B_K^T)^{\dagger} B_K = I_n - B^T (\Sigma_K B B^T \Sigma_K)^{\dagger} B.$$

For convenience, below we state Lemma 2.7 and Proposition 2.8, which are discussed in [18, Lemma 2 and Proposition 6]. For $2 \le j \le n$, we define the linear mapping $\mathbf{B}_j : \Re^j \to \Re^{j-1}$ such that $\mathbf{B}_j x = [x_1 - x_2; \ldots; x_{j-1} - x_j]\ \forall x \in \Re^j$. With this notation, we can write $B = \mathbf{B}_n$.

LEMMA 2.7. *For $2 \le j \le n$, it holds that*

$$T_j := I_j - \mathbf{B}_j^T(\mathbf{B}_j\mathbf{B}_j^T)^{-1}\mathbf{B}_j = \frac{1}{j}\mathbf{E}_j.$$

PROPOSITION 2.8. *Let $\Sigma \in \Re^{(n-1)\times(n-1)}$ be an N-block diagonal matrix with $\Sigma = \mathrm{Diag}(\Lambda_1, \ldots, \Lambda_N)$, where, for $i = 1, \ldots, N$, Λ_i is either $\mathbf{O}_{n_i}$ or I_{n_i}, and any two consecutive blocks are not of the same type. Define $J := \{j \mid \Lambda_j = I_{n_j},\ j = 1, \ldots, N\}$. Then, it holds that*

$$\Gamma := I_n - B^T(\Sigma B B^T \Sigma)^\dagger B = \mathrm{Diag}(\Gamma_1, \ldots, \Gamma_N),$$

where, for $i = 1, \ldots, N$,

$$\Gamma_i = \begin{cases} \frac{1}{n_i+1}\mathbf{E}_{n_i+1} & \text{if } i \in J, \\ I_{n_i} & \text{if } i \notin J \text{ and } i \in \{1, N\}, \\ I_{n_i-1} & \text{otherwise,} \end{cases}$$

with the convention $I_0 = \emptyset$. Moreover, $\Gamma = H + UU^T = H + U_J U_J^T$, where $H \in \Re^{n\times n}$ is an N-block diagonal matrix given by $H = \mathrm{Diag}(\Upsilon_1, \ldots, \Upsilon_N)$ with

$$\Upsilon_i \ = \ \begin{cases} \mathbf{O}_{n_i+1} & \text{if } i \in J, \\ I_{n_i} & \text{if } i \notin J \text{ and } i \in \{1, N\}, \\ I_{n_i-1} & \text{otherwise.} \end{cases}$$

Here the (k, j)th entry of the matrix $U \in \Re^{n\times N}$ is given by

$$U_{k,j} = \begin{cases} \frac{1}{\sqrt{n_j+1}} & \text{if } \sum_{t=1}^{j-1} n_t + 1 \le k \le \sum_{t=1}^{j} n_t + 1 \text{ and } j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

and U_J consists of the nonzero columns of U, i.e., the columns whose indices are in J.

Based on the above preliminaries, we define the multifunction $\mathcal{Q}_{S_\rho} : \Re^n \rightrightarrows \Re^{n\times n}$ by

$$\mathcal{Q}_{S_\rho}(y) := \Big\{ Q \in \Re^{n\times n} \ \Big|\ Q = P_y^T \widehat{Q} P_y,\ \widehat{Q} \in \mathcal{Q}_{\mathcal{D}}(P_y y - \rho w) \Big\}.$$

The following proposition shows that $\mathcal{Q}_{S_\rho}(y)$ can be viewed as the generalized Jacobian of $S_\rho(\cdot)$ at y.

PROPOSITION 2.9. *For any $y \in \Re^n$, there exists a neighborhood $\mathcal{Y}$ of y such that, for all $u \in \mathcal{Y}$,*

$$\mathcal{K}_{\mathcal{D}}(P_y u - \rho w) \subseteq \mathcal{K}_{\mathcal{D}}(P_y y - \rho w),\ \mathcal{Q}_{\mathcal{D}}(P_y u - \rho w) \subseteq \mathcal{Q}_{\mathcal{D}}(P_y y - \rho w),\ \mathcal{Q}_{S_\rho}(u) \subseteq \mathcal{Q}_{S_\rho}(y),$$

and

$$\begin{cases} \Pi_{\mathcal{D}}(P_y u - \rho w) = \Pi_{\mathcal{D}}(P_y y - \rho w) + \widehat{Q} P_y (u - y)\ \forall \widehat{Q} \in \mathcal{Q}_{\mathcal{D}}(P_y u - \rho w), \\ S_\rho(u) \ = \ S_\rho(y) + Q(u - y)\ \forall Q \in \mathcal{Q}_{S_\rho}(u). \end{cases}$$

Proof. The desired results can be easily derived from Proposition 2.5 together with simple manipulations. ∎

Define the multifunction $\mathcal{M} : \Re^n \rightrightarrows \Re^{n\times n}$ by

$$\mathcal{M}(y) := \Big\{ M \in \mathcal{S}^n \;\Big|\; M = \Theta Q,\ \Theta \in \partial_B \mathrm{Prox}_{\beta\|\cdot\|_1}(S_\rho(y)),\ Q \in \mathcal{Q}_{S_\rho}(y) \Big\}, \tag{2.4}$$

where the B-subdifferential of $\mathrm{Prox}_{\beta\|\cdot\|_1}(\cdot)$ at $\eta \in \Re^n$ is given by

$$\partial_B \mathrm{Prox}_{\beta\|\cdot\|_1}(\eta) = \left\{ \mathrm{Diag}(q) \;\middle|\; \begin{array}{ll} q_i = 0 & \text{if } |\eta_i| < \beta, \\ q_i \in \{0,1\} & \text{if } |\eta_i| = \beta, \\ q_i = 1 & \text{otherwise} \end{array} \right\}.$$

We can view $\mathcal{M}(y)$ as the generalized Jacobian of $\mathrm{Prox}_p(\cdot)$ at y. The reason for this is shown in the following theorem, which is similar to what was done in [18, Theorem 1] for the fused Lasso proximal mapping.

THEOREM 2.10. *Let $\beta, \rho > 0$ and $y \in \Re^n$ be given. Then, the multifunction $\mathcal{M}$ is nonempty, compact, and upper-semicontinuous. For any $M \in \mathcal{M}(y)$, M and $I - M$ are both symmetric and positive semidefinite. Moreover, there exists a neighborhood $\mathcal{Y}$ of y such that for all $u \in \mathcal{Y}$,*

$$\mathrm{Prox}_p(u) - \mathrm{Prox}_p(y) - M(u - y) = 0 \ \forall M \in \mathcal{M}(u). \tag{2.5}$$

Proof. From the definition of $\mathcal{M}$, we easily see that it is nonempty and compact. We know that $\partial_B \mathrm{Prox}_{\beta\|\cdot\|_1}(\cdot)$ is upper semicontinuous, which, together with the property on $S_\rho(\cdot)$ in Proposition 2.9, implies that $\mathcal{M}$ is upper-semicontinuous. In addition, by noting that $\mathrm{Prox}_{\beta\|\cdot\|_1}(\cdot)$ is piecewise affine, we have that (2.5) follows from [8, Theorem 7.5.17].

Next we only need to prove that any $M \in \mathcal{M}(y)$ is symmetric and positive semidefinite. The symmetry follows directly from the definition. From (2.4) and Lemma 2.6, one knows that for any $M \in \mathcal{M}(y)$, there exists a 0-1 diagonal matrix $\Theta \in \partial_B \mathrm{Prox}_{\beta\|\cdot\|_1}(S_\rho(y))$ and $K \in \mathcal{K}_{\mathcal{D}}(P_y y - \rho w)$ such that

$$\begin{aligned} M &= \Theta[P_y^T(I_n - B_K^T(B_K B_K^T)^{-1} B_K) P_y] \\ &= \Theta P_y^T (I_n - B^T(\Sigma_K B B^T \Sigma_K)^\dagger B) P_y. \end{aligned}$$

Since $\Sigma_K \in \Re^{(n-1)\times(n-1)}$ is an N-block diagonal matrix with

$$\Sigma_K = \mathrm{Diag}\{\Lambda_1, \ldots, \Lambda_N\},$$

where for $i = 1, \ldots, N$, Λ_i is either $\mathbf{O}_{n_i}$ or I_{n_i}, and any two consecutive blocks are not of the same type. Define $J := \{j \mid \Lambda_j = I_{n_j},\ j = 1, \ldots, N\}$. It then follows from Proposition 2.8 that

$$M = \Theta P_y^T \Gamma P_y,$$

where $\Gamma = \mathrm{Diag}(\Gamma_1, \ldots, \Gamma_N)$ is defined as in Proposition 2.8. Define $\widetilde{\Theta} \in \Re^{n\times n}$ as

$$\widetilde{\Theta} = P_y \Theta P_y^T = \mathrm{Diag}(P_y \mathrm{diag}(\Theta)),$$

which is also a 0-1 diagonal matrix. Thus,

$$M = P_y^T \widetilde{\Theta} P_y P_y^T \Gamma P_y = P_y^T \widetilde{\Theta} \Gamma P_y = P_y^T (\widetilde{\Theta}\Gamma) P_y.$$

In order to prove that M is positive semidefinite, it suffices to show that $\widetilde{\Theta}\Gamma$ is positive semidefinite. Note that $\widetilde{\Theta}$ can be decomposed as $\widetilde{\Theta} = \mathrm{Diag}(\widetilde{\Theta}_1, \ldots, \widetilde{\Theta}_N)$ and hence $\widetilde{\Theta}\Gamma = \mathrm{Diag}(\widetilde{\Theta}_1\Gamma_1, \ldots, \widetilde{\Theta}_N\Gamma_N)$, so we only need to prove that for all $j = 1, \ldots, N$, $\widetilde{\Theta}_j\Gamma_j$ is positive semidefinite. When Γ_j is an identity matrix, it is obvious that $\widetilde{\Theta}_j\Gamma_j = \widetilde{\Theta}_j$ and hence $\widetilde{\Theta}_j\Gamma_j$ is positive semidefinite. When Γ_j is not an identity matrix but of the form $\Gamma_j = \frac{1}{n_j+1}\mathbf{E}_{n_j+1}$ from Proposition 2.8, then we have

$$\left\{ \sum_{t=1}^{j-1} n_t + 1, \sum_{t=1}^{j-1} n_t + 2, \ldots, \sum_{t=1}^{j} n_t \right\} \subseteq K \subseteq \mathcal{I}_{\mathcal{D}}(P_y y - \rho w),$$

which means that

$$\big(\Pi_{\mathcal{D}}(P_y y - \rho w)\big)_i = \big(\Pi_{\mathcal{D}}(P_y y - \rho w)\big)_{i+1} \ \forall i \in \left\{ \sum_{t=1}^{j-1} n_t + 1, \ldots, \sum_{t=1}^{j} n_t \right\}.$$

As one can see, no matter what value $|(\Pi_{\mathcal{D}}(P_y y - \rho w))_{\sum_{t=1}^{j-1} n_t + 1}|$ takes, $\mathrm{diag}(\widetilde{\Theta}_j)$ should be all ones or all zeros, otherwise it will contradict the fact that $\widetilde{\Theta}_j\Gamma_j$ is symmetric. That is to say,

$$\widetilde{\Theta}_j = \mathbf{O}_{n_j+1} \quad \text{or} \quad I_{n_j+1}.$$

Thus, $\widetilde{\Theta}_j\Gamma_j = \mathbf{O}_{n_j+1}$ or $\frac{1}{n_j+1}\mathbf{E}_{n_j+1}$, which is obviously positive semidefinite.

For the case of $I - M$, we have that

$$I - M = I - P_y^T(\widetilde{\Theta}\Gamma)P_y = P_y^T(I - \widetilde{\Theta}\Gamma)P_y.$$

From the previous derivation, we can see that $0 \preceq \widetilde{\Theta}\Gamma \preceq I$, which yields that $I - M$ is positive semidefinite. This completes the proof. ∎

For later purposes, we recall the concept of semismoothness introduced in [23, 25, 16, 31].

DEFINITION 2.11. *Let $f : \mathcal{O} \subseteq \Re^n \to \Re^m$ be a locally Lipschitz continuous function on the open set $\mathcal{O}$ and $\mathcal{K} : \mathcal{O} \rightrightarrows \Re^{m\times n}$ be a nonempty, compact-valued and upper-semicontinuous multifunction. We say that f is* semismooth *at $x \in \mathcal{O}$ with respect to the multifunction $\mathcal{K}$ if* (i) *f is directionally differentiable at x; and* (ii) *for any $\Delta x \in \Re^n$ and $V \in \mathcal{K}(x + \Delta x)$ with $\Delta x \to 0$,*

$$f(x + \Delta x) - f(x) - V(\Delta x) = o(\|\Delta x\|). \tag{2.6}$$

Furthermore, if (2.6) *is replaced by*

$$f(x + \Delta x) - f(x) - V(\Delta x) = O(\|\Delta x\|^{1+\gamma}), \tag{2.7}$$

where $\gamma > 0$ is a constant, then f is said to be γ-order (strongly if $\gamma = 1$) semismooth *at x with respect to $\mathcal{K}$. We say that f is a semismooth function on $\mathcal{O}$ with respect to $\mathcal{K}$ if it is semismooth everywhere in $\mathcal{O}$ with respect to $\mathcal{K}$.*

Remark 2.12. Since $\mathrm{Prox}_p(\cdot)$ is a Lipschitz continuous piecewise affine function, it follows from [8, Lemma 4.6.1] that it is directionally differentiable at any point. Combining with Theorem 2.10, we conclude that for any arbitrary constant $\gamma > 0$, $\mathrm{Prox}_p(\cdot)$ is γ-order semismooth on $\Re^n$ with respect to $\mathcal{M}$.

2.3. Finding a computable element in $\mathcal{M}(y)$. In order for the multifunction that we defined in (2.4) to be useful in designing algorithms for problem (1.1), we need to construct at least one computable element explicitly in $\mathcal{M}(y)$ for any given $y \in \Re^n$. Let $\Sigma = \mathrm{Diag}(\sigma) \in \Re^{(n-1)\times(n-1)}$ be defined as

$$\sigma_i = \begin{cases} 1 & \text{if } i \in \mathcal{I}_{\mathcal{D}}(P_y y - \rho w), \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 1, 2, \ldots, n-1, \tag{2.8}$$

where $\mathcal{I}_{\mathcal{D}}(\cdot)$ is defined in (2.3), and $\Theta = \mathrm{Diag}(\theta) \in \Re^{n\times n}$ be defined as

$$\theta_i = \begin{cases} 0 & \text{if } |S_\rho(y)|_i \le \beta, \\ 1 & \text{otherwise} \end{cases} \qquad \text{for } i = 1, 2, \ldots, n. \tag{2.9}$$

From Proposition 2.5 and Proposition 2.6, we have that $M \in \mathcal{M}(y)$, which is given by

$$\begin{aligned} M &= \Theta P_y^T (I_n - B_{\mathcal{I}_{\mathcal{D}}(P_y y-\rho w)}^T (B_{\mathcal{I}_{\mathcal{D}}(P_y y-\rho w)} B_{\mathcal{I}_{\mathcal{D}}(P_y y-\rho w)}^T)^\dagger B_{\mathcal{I}_{\mathcal{D}}(P_y y-\rho w)}) P_y \\ &= \Theta P_y^T (I_n - B^T (\Sigma B B^T \Sigma)^\dagger B) P_y. \end{aligned}$$

Then we can apply Proposition 2.8 to compute M explicitly.

3. A semismooth Newton augmented Lagrangian method for the dual problem. The primal form of our problem (1.1) can be written as

$$\min_{x\in\Re^n} \ \Big\{ f(x) := \frac{1}{2}\|Ax - b\|^2 + p(x) \Big\}, \tag{P}$$

and the dual of (P) admits the following equivalent minimization form:

$$\min_{\xi\in\Re^m, u\in\Re^n} \ \Big\{ \frac{1}{2}\|\xi\|^2 + \langle b, \xi\rangle + p^*(u) \ \Big|\ A^T\xi + u = 0 \Big\}. \tag{D}$$

The Lagrangian function associated with (D) is defined by

$$l(\xi, u; x) := \frac{1}{2}\|\xi\|^2 + \langle b, \xi\rangle + p^*(u) - \langle x, A^T\xi + u\rangle.$$

Let $\sigma > 0$ be given. Then, the corresponding augmented Lagrangian function is given by

$$\mathcal{L}_\sigma(\xi, u; x) := l(\xi, u; x) + \frac{\sigma}{2}\|A^T\xi + u\|^2.$$

3.1. A semismooth Newton augmented Lagrangian method for (D). We denote the whole algorithm by SSNAL since a semismooth Newton method (SSN) is used in solving the subproblem of the inexact augmented Lagrangian method (ALM) [28]. We briefly describe the SSNAL algorithm in Algorithm 3.1.

For the SSNAL algorithm, we use the following implementable stopping criteria as in [28, 29]:

$$\|\nabla\psi_k(\xi^{k+1})\| \le \epsilon_k/\sqrt{\sigma_k}, \ \sum_{k=0}^{\infty} \epsilon_k < \infty, \tag{A}$$

$$\|\nabla\psi_k(\xi^{k+1})\| \le \delta_k\sqrt{\sigma_k}\|A^T\xi^{k+1} + u^{k+1}\|, \ \sum_{k=0}^{\infty} \delta_k < \infty, \tag{B1}$$

$$\|\nabla\psi_k(\xi^{k+1})\| \le \delta_k'\|A^T\xi^{k+1} + u^{k+1}\|, \ 0 \le \delta_k' \to 0, \tag{B2}$$

where $\{\epsilon_k\}$, $\{\delta_k\}$, $\{\delta_k'\}$ are given nonnegative error tolerance sequences.

Algorithm 3.1 SSNAL: A semismooth Newton augmented Lagrangian method for (D).

Input: $\sigma_0 > 0$, $(\xi^0, u^0, x^0) \in \Re^m \times \Re^n \times \Re^n$ and $k = 0$.

1: Approximately compute

$$\xi^{k+1} \approx \operatorname*{argmin}_{\xi \in \Re^m} \Big\{ \psi_k(\xi) := \inf_u \mathcal{L}_{\sigma_k}(\xi, u; x^k) \Big\} \tag{3.1}$$

to satisfy the conditions (A), (B1), and (B2).

2: $u^{k+1} = (x^k/\sigma_k - A^T\xi^{k+1}) - \mathrm{Prox}_p(x^k/\sigma_k - A^T\xi^{k+1})$.

3: $x^{k+1} = x^k - \sigma_k(A^T\xi^{k+1} + u^{k+1}) = \sigma_k \mathrm{Prox}_p(x^k/\sigma_k - A^T\xi^{k+1})$.

4: Update $\sigma_{k+1} \uparrow \sigma_\infty \le \infty$, $k \leftarrow k+1$, and go to step 1.

Define the following maximal monotone operators [28]:

$$\mathcal{T}_f(x) := \partial f(x),\ \mathcal{T}_l(\xi, u; x) := \{(\xi', u', x') \mid (\xi', u', -x') \in \partial l(\xi, u; x)\}.$$

The piecewise linear-quadratic property of f leads to the fact that $\mathcal{T}_f$ and $\mathcal{T}_l$ satisfy the error bound condition [22] at point 0 with positive moduli a_f and a_l, respectively [26, 32]. That is to say, there exists $\varepsilon > 0$ such that if $\mathrm{dist}(0, \mathcal{T}_f(x)) \le \varepsilon$, then

$$\mathrm{dist}(x, \Omega_p) \le a_f \mathrm{dist}(0, \mathcal{T}_f(x)). \tag{3.2}$$

In addition, there exists $\varepsilon' > 0$ such that if $\mathrm{dist}(0, \mathcal{T}_l(\xi, u; x)) \le \varepsilon'$, then

$$\mathrm{dist}((\xi, u, x), (\xi^*, u^*) \times \Omega_p) \le a_l \mathrm{dist}(0, \mathcal{T}_l(\xi, u; x)), \tag{3.3}$$

where (ξ^*, u^*) is the unique optimal solution of (D).

The global and local convergence of the SSNAL algorithm have been studied in [28, 29, 22]. Here we simply state some relevant results.

THEOREM 3.1. (1) *Let $\{(\xi^k, u^k, x^k)\}$ be the infinite sequence generated by the* SSNAL *algorithm with stopping criterion* (A). *Then, the sequence $\{x^k\}$ converges to an optimal solution of* (P). *In addition, $\{(\xi^k, u^k)\}$ converges to the unique optimal solution (ξ^*, u^*) of* (D).

(2) *For the sequence $\{(\xi^k, u^k, x^k)\}$ generated by the* SSNAL *algorithm with stopping criteria* (A) *and* (B1), *one has that for all k sufficiently large,*

$$\mathrm{dist}(x^{k+1}, \Omega_p) \le \theta_k \mathrm{dist}(x^k, \Omega_p), \tag{3.4}$$

where $\theta_k = (a_f(a_f^2 + \sigma_k^2)^{-1/2} + 2\delta_k)(1 - \delta_k)^{-1} \to \theta_\infty = a_f(a_f^2 + \sigma_\infty^2)^{-1/2} < 1$ as $k \to +\infty$, and a_f is from (3.2). *If the stopping criterion* (B2) *is also satisfied, it holds that for k sufficiently large,*

$$\|(\xi^{k+1}, u^{k+1}) - (\xi^*, u^*)\| \le \theta_k' \|x^{k+1} - x^k\|, \tag{3.5}$$

where $\theta_k' = a_l(1 + \delta_k')/\sigma_k \to a_l/\sigma_\infty$ as $k \to +\infty$, and a_l is from (3.3).

Proof. The first part of this theorem can be obtained from [28, Theorem 4]. Since $\mathcal{T}_f$ and $\mathcal{T}_l$ satisfy the error bound condition, it follows from [22, Theorem 2.1] that (3.4) holds. If (A), (B1), and (B2) are all satisfied, combining [6] with [17, Remark 1], we get the desired result that (3.5) holds. This completes the proof. ∎

3.2. A semismooth Newton method for the subproblem. In this subsection, we present an efficient semismooth Newton method for solving the ALM subproblem (3.1). Given $\tilde{x} \in \Re^n$ and $\sigma > 0$, we consider the minimization problem

$$\min_{\xi \in \Re^m} \left\{ \psi(\xi) := \inf_u \mathcal{L}_\sigma(\xi, u; \tilde{x}) \right\}, \tag{3.6}$$

where using the Moreau identity, we get that

$$\begin{aligned}\psi(\xi) = \inf_u \mathcal{L}_\sigma(\xi, u; \tilde{x}) = \frac{1}{2}\|\xi\|^2 + \langle b, \xi\rangle + p^*(\mathrm{Prox}_{p^*/\sigma}(-A^T\xi + \tilde{x}/\sigma)) \\ + \frac{1}{2\sigma}\|\mathrm{Prox}_{\sigma p}(-\sigma A^T\xi + \tilde{x})\|^2 - \frac{1}{2\sigma}\|\tilde{x}\|^2.\end{aligned}$$

Since $\psi(\cdot)$ is strongly convex and continuously differentiable, the minimization problem (3.6) has a unique solution $\hat{\xi}$ which can be obtained via solving the following nonsmooth equation:

$$0 = \nabla\psi(\xi) = \xi + b - A\mathrm{Prox}_{\sigma p}(\tilde{x} - \sigma A^T\xi) = \xi + b - \sigma A\mathrm{Prox}_p(\tilde{x}/\sigma - A^T\xi). \tag{3.7}$$

Here we use the fact that $\mathrm{Prox}_{\sigma p}(z) = \sigma\mathrm{Prox}_p(z/\sigma)$ for any $z \in \Re^n$.

Define the multifunction $\mathcal{V} : \Re^m \rightrightarrows \Re^{m\times m}$ by

$$\mathcal{V}(\xi) := \left\{ V \in \Re^{m\times m} \;\middle|\; V = I_m + \sigma AMA^T,\ M \in \mathcal{M}(\tilde{x}/\sigma - A^T\xi) \right\},$$

where $\mathcal{M}(\cdot)$ is the multifunction defined in (2.4). By virtue of Theorem 2.10, we know that $\mathcal{V}$ is nonempty, compact, and upper-semicontinuous. It is obvious that for any $\xi \in \Re^m$, all elements of $\mathcal{V}(\xi)$ are symmetric and positive definite. In addition, $\nabla\psi$ is γ-order semismooth on $\Re^m$ with respect to $\mathcal{V}$ for any $\gamma > 0$.

We shall apply a semismooth Newton (SSN) method to solve (3.7) as follows, and could expect to get a fast superlinear or even quadratic convergence.

Algorithm 3.2 SSN: A semismooth Newton method for solving (3.7).

Input: $\mu \in (0, 1/2)$, $\bar{\eta} \in (0, 1)$, $\tau \in (0, 1]$, $\delta \in (0, 1)$, ξ^0, $\tilde{x}$, σ, and $j = 0$.

1: Choose $V_j \in \mathcal{V}(\xi^j)$. Solve the linear system

$$V_j h = -\nabla\psi(\xi^j) \tag{3.8}$$

exactly or by the conjugate gradient (CG) algorithm to find h^j such that

$$\|V_j h^j + \nabla\psi(\xi^j)\| \le \min(\bar{\eta}, \|\nabla\psi(\xi^j)\|^{1+\tau}).$$

2: Set $\alpha_j = \delta^{m_j}$, where m_j is the first nonnegative integer m for which

$$\psi(\xi^j + \delta^m h^j) \le \psi(\xi^j) + \mu\delta^m\langle\nabla\psi(\xi^j), h^j\rangle.$$

3: Set $\xi^{j+1} = \xi^j + \alpha_j h^j$, $j \leftarrow j + 1$, and go to step 1.

The convergence analysis for the SSN algorithm can be established as in [18, Theorem 3].

THEOREM 3.2. *Let $\{\xi^j\}$ be the infinite sequence generated by the* SSN *algorithm. Then, $\{\xi^j\}$ converges to the unique optimal solution $\hat{\xi}$ of problem* (3.6) *and*

$$\|\xi^{j+1} - \hat{\xi}\| = O(\|\xi^j - \hat{\xi}\|^{1+\tau}).$$

Proof. According to [43, Proposition 3.3 and Theorem 3.4] and the fact that $\psi(\cdot)$ is strongly convex, $\{\xi^j\}$ converges to the unique optimal solution $\hat{\xi}$ of problem (3.6). Since $\mathcal{V}(\cdot)$ is a nonempty, compact-valued, and upper-semicontinuous set mapping, and all elements of $\mathcal{V}(\hat{\xi})$ are nonsingular, it follows from [8, Lemma 7.5.2] that $\{\|V_j^{-1}\|\}$ is uniformly bounded for sufficiently large j. In addition, $\nabla\psi$ is strongly semismooth on $\Re^m$ with respect to $\mathcal{V}$. By mimicking the proofs in [43, Theorem 3.5], we know that there exists $\hat{\delta} > 0$ such that for all sufficiently large j, one has

$$\|\xi^j + h^j - \hat{\xi}\| = O(\|\xi^j - \hat{\xi}\|^{1+\tau}) \tag{3.9}$$

and

$$-\langle \nabla\psi(\xi^j), h^j \rangle \geq \hat{\delta}\|h^j\|^2.$$

By using (3.9), [18, Proposition 7], and [8, Proposition 8.3.18], we can derive that for $\mu \in (0, 1/2)$, there exists an integer j_0 such that for all $j \geq j_0$,

$$\psi(\xi^j + h^j) \leq \psi(\xi^j) + \mu\langle \nabla\psi(\xi^j), h^j \rangle,$$

which implies that $\xi^{j+1} = \xi^j + h^j$ for all $j \geq j_0$. Combining with (3.9), we complete the proof. ∎

3.3. On the implementation of the SSNAL algorithm for the dual problem. The most time-consuming step in our algorithm is in solving the Newton equation (3.8). In this subsection, we shall design an efficient procedure to solve it.

Given $y := \tilde{x}/\sigma - A^T\xi$, we already know that

$$M = \Theta Q \in \mathcal{M}(y),$$

where $Q = P_y^T(I_n - B^T(\Sigma BB^T\Sigma)^\dagger B)P_y$, and Σ and Θ are defined in (2.8) and (2.9), respectively. For the Newton equation (3.8), we need to deal with the matrix AMA^T. Thus it is important to analyze its structure in order to solve (3.8) efficiently.

Noting that $\Sigma = \mathrm{Diag}\{\Lambda_1, \ldots, \Lambda_N\}$ is an N-block diagonal matrix with each Λ_i being either a zero matrix or an identity matrix, and any two consecutive blocks are not of the same type, we can apply Proposition 2.8 to simplify our computation. Let $J := \{j \mid \Lambda_j = I_{n_j},\ j = 1, \ldots, N\}$. Then we have

$$Q = P_y^T(H + U_J U_J^T)P_y = P_y^T H P_y + P_y^T U_J U_J^T P_y,$$

where the N-block diagonal matrix $H = \mathrm{Diag}(\Upsilon_1, \ldots, \Upsilon_N) \in \Re^{n\times n}$ is defined by

$$\Upsilon_i = \begin{cases} \mathbf{O}_{n_i+1} & \text{if } i \in J, \\ I_{n_i} & \text{if } i \notin J \text{ and } i \in \{1, N\}, \\ I_{n_i-1} & \text{otherwise,} \end{cases}$$

and U_J is defined in Proposition 2.8.

Since $M = \Theta Q$ is symmetric, it holds that $\Theta Q = M = M^T = Q\Theta$. Due to the fact that Θ is a 0-1 diagonal matrix, we have that $\Theta = \Theta^2$ and hence

$$M = \Theta Q = \Theta(\Theta Q) = \Theta(Q\Theta).$$

Thus, after plugging in the derived formula for Q, we get that

$$M = \Theta\widetilde{H}\Theta + \Theta P_y^T U_J (P_y^T U_J)^T \Theta,$$

where the matrix

$$\widetilde{H} = P_y^T H P_y = \mathrm{Diag}(P_y^T \mathrm{diag}(H))$$

is also a 0-1 diagonal matrix. It follows that

$$AMA^T = A\Theta\widetilde{H}\Theta A^T + A\Theta P_y^T U_J (P_y^T U_J)^T \Theta A^T.$$

Define the index sets

$$\alpha := \Big\{ i \;\Big|\; \theta_i = 1,\ i \in \{1, \ldots, n\} \Big\},\ \gamma := \Big\{ i \;\Big|\; \widetilde{h}_i = 1,\ i \in \alpha \Big\},$$

where θ_i and $\widetilde{h}_i$ are the ith diagonal entries of Θ and $\widetilde{H}$, respectively. Then, we immediately get the formula

$$A\Theta\widetilde{H}\Theta A^T = A_\alpha \widetilde{H} A_\alpha^T = A_\gamma A_\gamma^T,$$

where $A_\alpha \in \Re^{m\times|\alpha|}$ and $A_\gamma \in \Re^{m\times|\gamma|}$ are two submatrices obtained from A by extracting those columns with indices in α and γ, respectively. Furthermore, we have that

$$A\Theta P_y^T U_J (P_y^T U_J)^T \Theta A^T = A_\alpha P_y^T U_J (P_y^T U_J)^T A_\alpha^T = A_\alpha \widetilde{U}\widetilde{U}^T A_\alpha^T,$$

where $\widetilde{U} \in \Re^{|\alpha|\times t}$ is a submatrix obtained from $\Theta(P_y^T U_J)$ by extracting those rows with indices in α and removing the zero columns from $\Theta(P_y^T U_J)$. Finally, we obtain that

$$AMA^T = A_\gamma A_\gamma^T + A_\alpha \widetilde{U}\widetilde{U}^T A_\alpha^T.$$

Li, Sun, and Toh [18] referred to the above structure of AMA^T and that of $I_m + \sigma AMA^T$ inherited from M as the second-order structured sparsity. They also gave a thorough analysis of computational cost, which is quite similar in our case. Without considering the cost of computing $P_y^T \mathrm{diag}(H)$ and $P_y^T U_J$, the arithmetic operations of computing AMA^T and $AMA^T d$ for a given vector d are $O(m|\alpha|(m+t))$ and $O(|\alpha|(m+t))$, respectively. With the use of the Sherman–Morrison–Woodbury formula [13], the computational cost can be further reduced. We omit the details here.

4. A semismooth Newton proximal augmented Lagrangian method for the primal problem. The augmented Lagrangian method (ALM) for the dual problem (D) is expected to be efficient for the case when $m \ll n$, since the semismooth Newton system (3.8) is of dimension m-by-m. But for the case when $m \gg n$, as we shall see later in the numerical experiments, it is naturally more efficient to apply the ALM on the primal problem to avoid having to deal with a large m-by-m linear system in each semismooth Newton iteration. In this section, we will derive a semismooth Newton proximal ALM for the primal problem.

First we rewrite the primal problem as

$$(\mathrm{P}') \qquad \min_{x\in\Re^n, z\in\Re^n} \Big\{ \frac{1}{2}\|Ax-b\|^2 + p(z) \;\Big|\; x - z = 0 \Big\}.$$

The dual of (P′) is given as

$$(\mathrm{D}') \qquad \max_{y\in\Re^n, v\in\Re^n} \Big\{ -\frac{1}{2}\|Av-b\|^2 - \langle b, Av-b\rangle - p^*(-y) \;\Big|\; A^T(Av-b) - y = 0 \Big\}.$$

Given $\sigma > 0$, the augmented Lagrangian function of problem (P′) is given by

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

4.1. A semismooth Newton proximal augmented Lagrangian method for (P′). The semismooth Newton proximal ALM for (P′) has a similar framework to the SSNAL algorithm for (D). For simplicity, we just state Algorithm 4.1 here without giving the detailed derivation.

Algorithm 4.1 p-SSNAL: A semismooth Newton augmented Lagrangian method for (P′).

Input: $\sigma_0 > 0$, $(x^0, z^0, y^0) \in \Re^n \times \Re^n \times \Re^n$, and $k = 0$.

1: Adapt the semismooth Newton method to approximately compute

$$(4.1)\quad x^{k+1} \approx \operatorname*{argmin}_{x\in\Re^n}\Big\{\phi_k(x) := \widetilde{\mathcal{L}}_{\sigma_k}(x, \mathrm{Prox}_{p/\sigma_k}(x - y^k/\sigma_k); y^k) + \frac{1}{2\sigma_k}\|x - x^k\|^2\Big\}$$

to satisfy condition (A2) below.

2: $z^{k+1} = \mathrm{Prox}_{p/\sigma_k}(x^{k+1} - y^k/\sigma_k)$.

3: $y^{k+1} = y^k - \sigma_k(x^{k+1} - z^{k+1})$.

4: Update $\sigma_{k+1} \uparrow \sigma_\infty \le \infty$, $k \leftarrow k + 1$, and go to step 1.

In the p-SSNAL algorithm, we apply a semismooth Newton method (SSN) to solve (4.1) with the following stopping criterion:

$$(\mathrm{A2})\quad \|\nabla\phi_k(x^{k+1})\| \le \frac{\epsilon_k}{\sigma_k}\min(1, \|(x^{k+1}, z^{k+1}, y^{k+1}) - (x^k, z^k, y^k)\|),\ \sum_{k=0}^{\infty}\epsilon_k < \infty.$$

The proximal ALM has been studied in [28, section 5], which is also called the proximal method of multipliers. We can get the global convergence and local linear convergence of the proximal ALM without any difficulty from [28, 29, 22].

4.2. A semismooth Newton method for solving (4.1). Similar to the case of the SSNAL algorithm, the most expensive step in each iteration of the p-SSNPAL algorithm is in solving the subproblem (4.1). Given $\sigma > 0$ and $(\tilde{x}, \tilde{y}) \in \Re^n \times \Re^n$, we adapt a semismooth Newton method to solve a typical subproblem of the following form:

$$\min_{x\in\Re^n}\ \phi(x) := \widetilde{\mathcal{L}}_\sigma(x, \mathrm{Prox}_{p/\sigma}(x - \tilde{y}/\sigma); \tilde{y}) + \frac{1}{2\sigma}\|x - \tilde{x}\|^2.$$

Since $\phi(\cdot)$ is continuously differentiable and strongly convex, the above optimization problem has a unique solution $\hat{x}$. Thus, it is equivalent to solving the following nonsmooth equation:

$$\begin{aligned}(4.2)\qquad 0 = \nabla\phi(x) &= A^T(Ax - b) + \sigma x - \tilde{y} - \sigma\mathrm{Prox}_{p/\sigma}(x - \tilde{y}/\sigma) + (x - \tilde{x})/\sigma \\ &= A^T(Ax - b) + (\sigma + 1/\sigma)x - (\tilde{y} + \tilde{x}/\sigma) - \mathrm{Prox}_p(\sigma x - \tilde{y}).\end{aligned}$$

Define the multifunction $\mathcal{U} : \Re^n \rightrightarrows \Re^{n\times n}$ by

$$\mathcal{U}(x) := \Big\{U \in \Re^{n\times n}\ \Big|\ U = A^TA + \sigma(I_n - M) + \frac{1}{\sigma}I_n,\ M \in \mathcal{M}(\sigma x - \tilde{y})\Big\},$$

where $\mathcal{M}(\cdot)$ is defined as in (2.4). From Theorem 2.10, we obtain that $\mathcal{U}$ is a nonempty, compact-valued, and upper-semicontinuous multifunction with its elements being symmetric and positive definite. In addition, $\nabla\phi$ is γ-order semismooth on $\Re^n$ with respect to $\mathcal{U}$ for all $\gamma > 0$. Thus we can apply an SSN method to solve (4.2). Similar to the results in section 3.2, the SSN method has a fast superlinear or even quadratic convergence.

The efficiency of the SSN method depends on the generalized Jacobian of $\nabla\phi(\hat{x})$. Next, we characterize the positive definiteness of the elements in $\mathcal{U}(\hat{x})$ in the following proposition.

PROPOSITION 4.1. *For any* $U = A^TA + \sigma(I_n - M) + \frac{1}{\sigma}I_n \in \mathcal{U}(\hat{x})$, *we have that*

$$\lambda_{\min}(U) \geq \lambda_{\min}(A^TA + \sigma(I_n - M)) + \frac{1}{\sigma} \geq \lambda_{\min}(A^TA) + \sigma\lambda_{\min}(I_n - M) + \frac{1}{\sigma} \geq \frac{1}{\sigma}.$$

Proof. From Theorem 2.10, we know that for any $M \in \mathcal{M}(\sigma\hat{x} - \tilde{y})$, $I_n - M$ is symmetric and positive semidefinite, which yields that the desired conclusion holds trivially. ∎

Remark 4.2. When the columns of A are linearly independent, for any $U \in \mathcal{U}(\hat{x})$, we have that

$$\lambda_{\min}(U) \geq \lambda_{\min}(A^TA) + \sigma\lambda_{\min}(I_n - M) + \frac{1}{\sigma} \geq \lambda_{\min}(A^TA) + \frac{1}{\sigma}.$$

In that case, U is positive definite if we do not add the proximal term $\frac{1}{2\sigma_k}\|x - x^k\|^2$ in (4.1). Since here we mainly focus on the case when $m \gg n$, the columns of A are very likely to be linearly independent.

5. Numerical experiments. In this section, we will evaluate the performance of our SSNAL algorithm for solving the clustered Lasso problems on the high-dimension–low-sample setting and the high-sample–low-dimension setting. For simplicity, we use the following abbreviations. SSNAL represents the semismooth Newton augmented Lagrangian method, ADMM represents the alternating direction method of multipliers, IADMM represents the inexact ADMM, LADMM represents the linearized ADMM, and APG represents the accelerated proximal gradient method. We implemented ADMM, IADMM, and LADMM in MATLAB with the step length set to be 1.618.

In our experiments, the regularization parameters β and ρ in the clustered Lasso problem (1.1) are chosen to have the form

$$\beta = \alpha_1\|A^Tb\|_\infty,\ \rho = \alpha_2\beta,$$

where $0 < \alpha_1 < 1$ and $\alpha_2 > 0$. To produce reasonable clustering results, we choose $\alpha_2 = O(1/n)$ to make sure that the two penalty terms have the same magnitude of influence.

We stop the tested algorithms according to some specified stopping criteria, which will be given in the following subsections. In addition, the algorithms will be stopped when they reach the maximum computation time of 3 hours or the pre-set maximum number of iterations (100 for SSNAL, and 20000 for ADMM, IADMM, LADMM, APG). All our computational results are obtained by running MATLAB (version 9.0) on a windows workstation (12-core, Intel Xeon E5-2680 @ 2.50 GHz, 128 GB RAM).

5.1. First-order methods. For comparison purposes, we summarize two types of first-order methods that are suitable for solving the clustered Lasso problem. An important point to mention here is that the proximal mapping given in section 2.1 plays a crucial role in the projection steps of these methods. Indeed, the new characteristic of the clustered Lasso regularizer vastly improves the performance of the first-order methods as the computation of the proximal mapping is now much cheaper.

Alternating direction method of multipliers for (D). We start by adapting the widely-used alternating direction method of multipliers (ADMM) [7, 11, 12] for solving (D), which can be described as Algorithm 5.1.

Algorithm 5.1 d-ADMM: An alternating direction method of multipliers for (D).

Input: $\kappa \in (0, (1+\sqrt{5})/2)$, $\sigma > 0$, $x^0 \in \Re^n$, $u^0 \in \Re^n$, and $k = 0$.

1: Compute

$$\xi^{k+1} \approx \operatorname*{argmin}_{\xi \in \Re^m} \mathcal{L}_\sigma(\xi, u^k; x^k). \tag{5.1}$$

2: $u^{k+1} = \operatorname{argmin}_{u \in \Re^n} \mathcal{L}_\sigma(\xi^{k+1}, u; x^k) = \operatorname{Prox}_{p^*/\sigma}(-A^T\xi^{k+1} + x^k/\sigma)$.

3: $x^{k+1} = x^k - \kappa\sigma(A^T\xi^{k+1} + u^{k+1})$.

4: $k \leftarrow k+1$, and go to step 1.

Note that in practice, κ should be chosen to be at least 1 for faster convergence. For the subproblem (5.1), the optimality condition that ξ^{k+1} must satisfy is given by

$$(I_m + \sigma AA^T)\xi = -b + A(x^k - \sigma u^k).$$

The linear system of equations of the form $(I_m + \sigma AA^T)\xi = h$ has to be solved repeatedly with a different right-hand side vector h. One can solve this linear system directly or use an iterative solver such as the preconditioned conjugate gradient (PCG) method.

The convergence results of the classical ADMM with the subproblems solved exactly have been discussed in [9], while the convergence analysis of the inexact ADMM can be found in [5]. The linearized ADMM algorithm [41] can also be used to solve this problem by linearizing the quadratic term in (5.1). It is worthwhile to mention that inexact ADMM and linearized ADMM are often used in the case when m is large.

Alternating direction method of multipliers for (P′). Next we present the ADMM algorithm for (P′), which is described as Algorithm 5.2.

Algorithm 5.2 p-ADMM: An alternating direction method of multipliers for (P′).

Input: $\kappa \in (0, (1+\sqrt{5})/2)$, $\sigma > 0$, $z^0 \in \Re^n$, $y^0 \in \Re^n$, and $k = 0$.

1: Compute

$$x^{k+1} \approx \operatorname*{argmin}_{x \in \Re^n} \widetilde{\mathcal{L}}_\sigma(x, z^k; y^k). \tag{5.2}$$

2: $z^{k+1} = \operatorname{argmin}_{z \in \Re^n} \widetilde{\mathcal{L}}_\sigma(x^{k+1}, z; y^k) = \operatorname{Prox}_{p/\sigma}(x^{k+1} - y^k/\sigma)$.

3: $y^{k+1} = y^k - \kappa\sigma(x^{k+1} - z^{k+1})$.

4: $k \leftarrow k+1$, and go to step 1.

Note that, for the subproblem (5.2), x^{k+1} is the solution of the following linear system:

$$(\sigma I_n + A^T A)x = A^T b + \sigma(z^k + y^k/\sigma).$$

Both direct solvers and iterative solvers can be used here.

An accelerated proximal gradient method of (P). Since the function $\|Ax-b\|^2/2$ in (P) has Lipschitz continuous gradient (with Lipschitz constant L, which is the largest eigenvalue of A^TA), one can attempt to use the accelerated proximal gradient (APG) method in [1] to solve (P). The basic template of the APG algorithm is given in Algorithm 5.3.

Algorithm 5.3 APG: An accelerated proximal gradient method for (P).

Input: $\varepsilon > 0$, $w^0 = x^0 \in \Re^n$, $t_0 = 1$, and $k = 0$.

1: Compute

$$x^{k+1} = \mathrm{Prox}_{p/L}(w^k - L^{-1}A^T(Aw^k - b)).$$

2: Set $t_{k+1} = (1 + \sqrt{1 + 4t_k^2})/2$.
3: Update $w^{k+1} = x^{k+1} + (t_k - 1)/t_{k+1}(x^{k+1} - x^k)$.
4: $k \leftarrow k + 1$, and go to step 1.

It is clear that the practical performance of the APG algorithm hinges crucially on whether one can compute the proximal mapping $\mathrm{Prox}_{\nu p}(y)$ for any $y \in \Re^n$ and $\nu > 0$ efficiently. Fortunately, we have provided an analytical solution to this problem in section 2.1.

5.2. Stopping criteria. Since the primal problem (P) is unconstrained, it is reasonable to measure the accuracy of an approximate optimal solution (ξ, u, x) of problem (D) and problem (P) by the relative duality gap and dual infeasibility. Specifically, let

$$\mathrm{pobj} := \frac{1}{2}\|Ax-b\|^2 + p(x), \quad \mathrm{dobj} := -\frac{1}{2}\|\xi\|^2 - \langle b, \xi\rangle$$

be the primal and dual objective function values. The relative duality gap and the relative dual infeasibility are given as

$$\eta_{gap} := \frac{|\mathrm{pobj} - \mathrm{dobj}|}{1 + |\mathrm{pobj}| + |\mathrm{dobj}|}, \quad \eta_D := \frac{\|A^T\xi + u\|}{1 + \|u\|}.$$

In addition, the relative KKT residual of the primal problem (P),

$$\eta_{kkt} = \frac{\|x - \mathrm{Prox}_p(x - A^T(Ax-b))\|}{1 + \|x\| + \|A^T(Ax-b)\|}, \tag{5.3}$$

can be adopted to measure the accuracy of an approximate optimal solution x.

5.3. Numerical results for UCI data sets. In this subsection, we conduct some experiments on the same large-scale UCI data sets (A, b) as in [18] that are

originally obtained from the LIBSVM data sets [4]. All instances are in the high-dimension–low-sample setting. According to what we have discussed in section 3, the dual approaches are better choices since we have $m \ll n$ in this setting.

For a given tolerance ϵ, we will terminate the SSNAL algorithm when

$$\max\{\eta_{gap}, \eta_D, \eta_{kkt}\} \le \epsilon. \tag{5.4}$$

Table 1 gives the numerical results for SSNAL when solving the clustered Lasso problem (1.1) on UCI data sets. In the table, m and n denote the number of samples and features, respectively. We use $\mathrm{nnz}(x)$ to denote the number of nonzeros in the solution x using the estimation

$$\mathrm{nnz}(x) := \min\left\{k \;\middle|\; \sum_{i=1}^{k} |\hat{x}_i| \ge 0.99999\|x\|_1\right\},$$

where $\hat{x}$ is obtained by sorting x such that $|\hat{x}_1| \ge |\hat{x}_2| \ge \cdots \ge |\hat{x}_n|$. We also use $\mathrm{gnnz}(x)$ to denote the number of groups in the solution, where the pairwise ratios among the sorted elements in each group are between 5/6 and 6/5. In order to get reasonable grouping results, we regard the elements with absolute value below 10^{-4} to be in the same group.

In order to get a reasonable number of nonzero elements in the optimal solution x, we choose $\alpha_1 \in \{10^{-6}, 10^{-7}\}$ for the problems E2006.train and E2006.test, $\alpha_1 \in \{10^{-2}, 10^{-3}\}$ for problem triazines4, $\alpha_1 \in \{10^{-5}, 10^{-6}\}$ for problem bodyfat and $\alpha_1 \in \{10^{-3}, 10^{-4}\}$ for the other instances. As we mentioned before, when $\alpha_2 = O(1/n)$, we can get reasonable clustering results. In total, we tested 54 instances.

From Table 1, we see that the SSNAL algorithm is efficient and robust against different parameter selections. It can be observed that all the 54 tested instances are successfully solved by SSNAL in about 5 minutes. In fact, for most of the cases, they are solved in less than one minute.

Table 1: The performance of the SSNAL algorithm on UCI data sets with different parameter selections. We terminate SSNAL when $\max\{\eta_{gap}, \eta_D, \eta_{kkt}\} \le 10^{-6}$. $\mathrm{nnz}(x)$ and $\mathrm{gnnz}(x)$ are obtained by SSNAL. Time is shown in the format of (hours:minutes:seconds).

Proname (m; n)	α_1; α_2	$\mathrm{nnz}(x)$; $\mathrm{gnnz}(x)$	pobj	η_{kkt}	$\max\{\eta_{gap}, \eta_D\}$	Time
E2006.train (16087; 150360) $\lambda_{\max}(AA^T)$ =1.91e+05	1e−6; 1e−5	4; 4	1.19083+3	2.9-7	8.7-7	05
	1e−6; 1e−6	22; 8	1.18031+3	8.2-9	7.2-8	07
	1e−6; 1e−7	27; 6	1.17744+3	6.3-8	7.6-7	06
	1e−7; 1e−4	8; 5	1.18600+3	1.2-8	5.3-8	06
	1e−7; 5e−5	36; 6	1.17237+3	4.4-8	4.7-8	09
	1e−7; 1e−5	380; 6	1.10710+3	1.4-8	4.8-7	01:20
E2006.test (3308; 150358) $\lambda_{\max}(AA^T)$ =4.79e+04	1e−6; 1e−5	10; 5	2.38906+2	2.9-8	4.5-7	04
	1e−6; 1e−6	35; 5	2.29669+2	2.7-9	9.9-8	04
	1e−6; 1e−7	53; 5	2.27308+2	2.3-9	1.2-7	04
	1e−7; 1e−4	20; 7	2.34499+2	7.8-10	3.4-8	04
	1e−7; 5e−5	76; 8	2.23445+2	9.0-9	2.3-7	10
	1e−7; 1e−5	550; 5	1.74748+2	2.7-10	4.5-8	04:00
log1p.train (16087; 4272227) $\lambda_{\max}(AA^T)$ =5.86e+07	1e−3; 1e−6	3; 3	2.80871+3	7.8-8	7.8-8	49
	1e−3; 1e−7	3; 3	1.58340+3	2.0-7	2.0-7	55
	1e−3; 1e−8	5; 5	1.45745+3	8.7-8	8.7-8	01:09
	1e−4; 1e−6	38; 11	1.27870+3	3.9-7	4.0-7	01:27
	1e−4; 5e−7	92; 5	1.18724+3	1.1-7	1.1-7	02:30
	1e−4; 1e−7	321; 5	1.08486+3	1.8-7	1.8-7	05:08

Continued on next page

Table 1 — continued from previous page

Proname (m; n)	α_1; α_2	nnz(x); gnnz(x)	pobj	η_{kkt}	max$\{\eta_{gap}, \eta_D\}$	Time
log1p.test (3308; 4272226) $\lambda_{\max}(AA^T)$ =1.46e+07	1e−3; 1e−6	3; 2	6.27631+2	4.3-7	4.3-7	40
	1e−3; 1e−7	4; 4	3.41971+2	4.0-8	4.0-8	01:08
	1e−3; 1e−8	8; 5	3.10745+2	5.5-8	5.5-8	01:06
	1e−4; 1e−6	50; 6	2.61434+2	2.4-7	2.4-7	01:47
	1e−4; 5e−7	172; 5	2.34526+2	2.0-7	2.0-7	03:28
	1e−4; 1e−7	726; 5	1.90645+2	1.5-7	1.5-7	04:26
pyrim5 (74; 201376) $\lambda_{\max}(AA^T)$ =1.22e+06	1e−3; 5e−5	48; 5	6.08424-1	2.1-7	2.2-7	36
	1e−3; 1e−5	65; 7	1.94647-1	4.1-7	4.4-7	28
	1e−3; 1e−6	91; 10	8.80020-2	2.2-7	2.3-7	21
	1e−4; 5e−5	102; 5	8.19911-2	6.2-7	6.6-7	52
	1e−4; 1e−5	88; 5	2.67953-2	2.1-7	2.3-7	01:07
	1e−4; 1e−6	83; 6	1.24353-2	4.7-7	6.2-7	28
triazines4 (186; 635376) $\lambda_{\max}(AA^T)$ =2.07e+07	1e−2; 1e−5	341; 3	7.81486+0	8.2-8	2.6-7	37
	1e−2; 1e−6	373; 6	3.00214+0	2.0-7	2.1-7	56
	1e−2; 1e−7	411; 8	2.31560+0	8.9-8	8.9-8	50
	1e−3; 1e−5	611; 6	1.86205+0	1.7-7	1.7-7	02:06
	1e−3; 1e−6	641; 6	7.53899-1	9.6-7	9.6-7	02:00
	1e−3; 5e−7	877; 7	6.55800-1	5.6-7	5.6-7	04:52
abalone (4177; 6435) $\lambda_{\max}(AA^T)$ =5.21e+05	1e−3; 1e−4	25; 8	1.24134+4	4.7-7	4.8-7	01
	1e−3; 5e−5	24; 8	1.19308+4	5.7-7	5.8-7	01
	1e−3; 1e−5	26; 9	1.15154+4	4.8-7	4.8-7	01
	1e−4; 1e−4	50; 8	9.54332+3	1.4-7	1.6-7	04
	1e−4; 5e−5	51; 5	9.42227+3	1.9-7	2.2-7	04
	1e−4; 1e−5	62; 7	9.31717+3	5.1-7	6.1-7	05
bodyfat (252; 116280) $\lambda_{\max}(AA^T)$ =5.29e+04	1e−5; 5e−5	10; 5	2.09723-2	2.0-8	2.8-8	04
	1e−5; 1e−5	20; 8	7.14784-3	3.7-7	6.4-7	05
	1e−5; 1e−6	27; 6	3.93005-3	1.3-7	2.3-7	06
	1e−6; 5e−5	38; 6	2.55045-3	7.4-8	1.6-7	07
	1e−6; 1e−5	78; 6	9.90203-4	6.5-8	1.3-7	13
	1e−6; 1e−6	108; 7	5.93863-4	6.6-8	1.3-7	11
housing (506; 77520) $\lambda_{\max}(AA^T)$ =3.28e+05	1e−3; 5e−5	106; 9	6.69490+3	3.5-7	4.6-7	07
	1e−3; 1e−5	139; 6	3.76003+3	3.7-8	3.9-8	09
	1e−3; 1e−6	158; 5	2.88365+3	5.3-8	5.4-8	08
	1e−4; 5e−5	207; 6	1.94260+3	1.8-7	1.9-7	42
	1e−4; 1e−5	255; 11	1.21114+3	4.2-7	5.8-7	28
	1e−4; 1e−6	292; 9	9.54315+2	8.1-8	1.0-7	22

For comparison, we also conduct numerical experiments on ADMM, iADMM, LADMM, and APG. We select two pairs of parameters for each data set when computing. Let $\text{pobj}_{\text{SSNAL}}$ be the optimal primal objective value obtained by SSNAL with stopping criterion (5.4). Since the minimization problem (P) is unconstrained, it is reasonable to terminate a first-order algorithm when

$$\eta_{rel} := \frac{\text{pobj} - \text{pobj}_{\text{SSNAL}}}{1 + |\text{pobj}_{\text{SSNAL}}|} \leq \epsilon_2, \tag{5.5}$$

where pobj is the primal objective value obtained by the first-order algorithm and ϵ_2 is a given tolerance. Here, we treat $\text{pobj}_{\text{SSNAL}}$ as an accurate approximate optimal objective value to (P) and stop the other algorithms by using the relative difference between the obtained primal objective value and $\text{pobj}_{\text{SSNAL}}$.

Tables 2 and 3 show the numerical results. In the tables, t_{SSNAL} represents the time needed by SSNAL when using the stopping criterion (5.4) with $\epsilon = 10^{-6}$. We test two different choices of ϵ_2. The results for $\epsilon_2 = 10^{-4}$ are shown in Table 2 and the results for $\epsilon_2 = 10^{-6}$ are shown in Table 3. The values highlighted in bold in these tables mean that the corresponding algorithms cannot achieve the required accuracy when the maximum iteration is attained.

When $\epsilon_2 = 10^{-4}$, we can see from Table 2 that ADMM is able to solve 18 instances and iADMM can solve 17 instances successfully, while LADMM and APG can solve 16 and 12 instances successfully, respectively. When $\epsilon_2 = 10^{-6}$, we can see from Table 3 that ADMM is able to solve 16 instances and iADMM can solve 15 instances, while LADMM and APG can only solve 11 and 3 instances, respectively. We note that iADMM and LADMM are computationally more advantageous than ADMM when solving instances with large m, and thus it is not surprising that ADMM is more efficient than iADMM and LADMM in solving the tested instances for which m is not too large.

By comparing the computation time between SSNAL and the first-order algorithms, we can see that SSNAL takes much less time than the first-order algorithms but gets much better results in almost all cases. If we require a high accuracy, then the first-order methods will take much longer than SSNAL and may not even achieve the required accuracy.

Table 2: The performance of various algorithms on UCI data sets. In the table, "b" = ADMM, "c" = iADMM, "d" = LADMM, "e" = APG. We terminate the first-order algorithms when $\eta_{rel} \leq 10^{-4}$. t_{SSNAL} represents the time needed by SSNAL when using stopping criterion $\max\{\eta_{gap}, \eta_D, \eta_{kkt}\} \leq 10^{-6}$. Time is shown in the format of (hours:minutes:seconds).

			η_{rel}	Time
Proname	α_1; α_2	t_{SSNAL}	b | c | d | e	b | c | d | e
E2006.train	1e−6; 1e−7	06	1.0-4 | 1.0-4 | 9.9-5 | 9.6-5	34:05 | 06:47 | 32 | 53:30
	1e−7; 1e−5	01:20	1.0-4 | 9.9-5 | 1.0-4 | **5.5-3**	41:51 | 09:54 | 01:19 | 59:38
E2006.test	1e−6; 1e−7	04	9.8-5 | 9.8-5 | 9.9-5 | **3.9-3**	03:41 | 03:09 | 32 | 23:57
	1e−7; 1e−5	04:00	1.0-4 | 9.9-5 | 1.0-4 | **6.6-2**	04:41 | 04:28 | 01:27 | 24:21
log1p.train	1e−3; 1e−8	01:09	9.9-5 | 9.5-5 | 9.4-5 | 8.2-5	29:09 | 09:19 | 05:14 | 01:03:59
	1e−4; 1e−7	05:08	1.0-4 | 9.9-5 | 9.9-5 | 5.4-5	35:50 | 14:44 | 11:13 | 01:47:40
log1p.test	1e−3; 1e−8	01:06	9.7-5 | 9.5-5 | 9.3-5 | 2.6-5	08:20 | 08:01 | 03:36 | 49:14
	1e−4; 1e−7	04:26	1.0-4 | 1.0-4 | 9.9-5 | 9.6-5	13:07 | 12:37 | 09:31 | 01:30:30
pyrim5	1e−3; 1e−6	21	1.0-4 | 1.0-4 | 1.0-4 | 9.7-5	06:17 | 14:19 | 14:52 | 34:08
	1e−4; 1e−6	28	1.0-4 | 1.0-4 | 1.0-4 | **2.1-4**	05:50 | 21:17 | 12:37 | 39:31
triazines4	1e−2; 1e−6	56	1.0-4 | 1.0-4 | 1.0-4 | 9.6-5	55:38 | 01:47:38 | 02:01:36 | 02:25:28
	1e−3; 1e−6	02:00	1.0-4 | **4.7-4** | **9.1-4** | **3.1-3**	01:12:35 | 03:00:01 | 01:34:51 | 03:00:00
abalone	1e−3; 1e−5	01	1.0-4 | 1.0-4 | 1.0-4 | 7.5-5	32 | 24 | 08 | 40
	1e−4; 1e−5	05	9.9-5 | 1.0-4 | 1.0-4 | 9.9-5	01:15 | 01:40 | 37 | 02:00
bodyfat	1e−5; 1e−6	06	9.9-5 | 9.9-5 | 9.9-5 | 9.5-5	53 | 02:59 | 29 | 07:57
	1e−6; 1e−6	11	1.0-4 | 1.0-4 | 1.0-4 | 9.8-5	01:18 | 05:41 | 01:11 | 11:12
housing	1e−3; 1e−6	08	1.0-4 | 1.0-4 | 1.0-4 | 9.5-5	01:08 | 04:27 | 03:53 | 06:26
	1e−4; 1e−6	22	1.0-4 | 1.0-4 | **9.9-3** | **3.8-4**	02:06 | 17:25 | 08:06 | 14:52

Table 3: As for Table 2 but we terminate the first-order algorithms when $\eta_{rel} \leq 10^{-6}$.

			η_{rel}	Time
Proname	α_1; α_2	t_{SSNAL}	b | c | d | e	b | c | d | e
E2006.train	1e−6; 1e−7	06	9.9-7 | 9.9-7 | 9.9-7 | **4.5-5**	37:47 | 08:01 | 48 | 01:00:44
	1e−7; 1e−5	01:20	9.8-7 | 9.9-7 | 9.9-7 | **5.5-3**	44:51 | 11:17 | 01:46 | 59:38
E2006.test	1e−6; 1e−7	04	9.8-7 | 9.8-7 | 9.9-7 | **3.9-3**	03:59 | 03:26 | 42 | 23:57
	1e−7; 1e−5	04:00	**1.3-5** | **2.6-5** | **2.1-5** | **6.6-2**	05:19 | 04:43 | 02:06 | 24:21
log1p.train	1e−3; 1e−8	01:09	9.7-7 | 9.6-7 | 1.0-6 | 7.5-7	32:57 | 12:34 | 10:21 | 02:07:44
	1e−4; 1e−7	05:08	9.9-7 | 1.0-6 | 1.0-6 | **8.0-6**	57:11 | 33:23 | 34:48 | 03:00:01

Continued on next page

Table 3 — continued from previous page

			η_{rel}	Time
Proname	α_1; α_2	t_{SSNAL}	b | c | d | e	b | c | d | e
log1p.test	1e−3; 1e−8	01:06	9.9-7 | 1.0-6 | 7.8-7 | 7.9-7	12:20 | 11:48 | 05:11 | 02:04:11
	1e−4; 1e−7	04:26	9.9-7 | 9.9-7 | 9.8-7 | **7.3-6**	21:21 | 20:11 | 20:41 | 03:00:01
pyrim5	1e−3; 1e−6	21	1.0-6 | 1.0-6 | **3.5-5** | **8.2-5**	30:08 | 55:12 | 27:44 | 35:11
	1e−4; 1e−6	28	1.0-6 | 1.0-6 | **7.2-5** | **2.1-4**	36:43 | 01:51:44 | 19:41 | 39:31
triazines4	1e−2; 1e−6	56	1.0-6 | **1.1-5** | **8.0-5** | **1.8-5**	02:14:05 | 03:00:01 | 02:12:43 | 03:00:01
	1e−3; 1e−6	02:00	**6.6-6** | **4.7-4** | **9.1-4** | **3.1-3**	03:00:00 | 03:00:01 | 01:34:51 | 03:00:00
abalone	1e−3; 1e−5	01	9.6-7 | 9.5-7 | 1.0-6 | 5.1-7	01:02 | 01:05 | 33 | 02:08
	1e−4; 1e−5	05	9.8-7 | 9.7-7 | **1.0-4** | **2.7-6**	01:37 | 02:27 | 03:19 | 05:18
bodyfat	1e−5; 1e−6	06	1.0-6 | 1.0-6 | 1.0-6 | **4.3-6**	01:10 | 04:00 | 01:39 | 15:25
	1e−6; 1e−6	11	9.8-7 | 9.7-7 | 1.0-6 | **4.3-5**	01:43 | 09:07 | 04:11 | 15:18
housing	1e−3; 1e−6	08	9.9-7 | 9.9-7 | 1.0-6 | **1.7-6**	02:33 | 10:37 | 11:20 | 14:50
	1e−4; 1e−6	22	1.0-6 | 1.0-6 | **9.9-3** | **3.8-4**	03:39 | 31:41 | 08:06 | 14:52

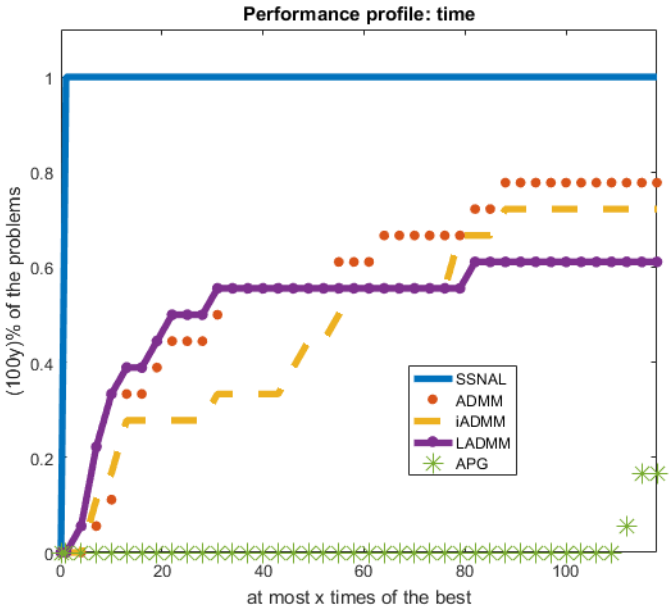

FIG. 1. *Performance profiles for* SSNAL, ADMM, IADMM, LADMM, *and* APG *on UCI data sets. Results of* SSNAL *are obtained by setting* $\max\{\eta_{gap}, \eta_D, \eta_{kkt}\} \le 10^{-6}$, *and results of* ADMM, IADMM, LADMM, *and* APG *are obtained by setting* $\eta_{rel} \le 10^{-6}$.

We also present in Figure 1 the performance profiles of SSNAL, ADMM, IADMM, LADMM, and APG for all the tested problems.[1] In the figure, the results for SSNAL are obtained by setting $\max\{\eta_{gap}, \eta_D, \eta_{kkt}\} \le 10^{-6}$, and the results for ADMM, IADMM, LADMM, and APG are obtained for $\eta_{rel} \le 10^{-6}$. Thus the accuracy of SSNAL is higher than the other algorithms in this sense. Recall that a point (x, y) is in the performance profile curve of a method if and only if it can solve $(100y\%)$ of all tested instances successfully in at most x times of the best methods for each instance. It can be seen that SSNAL outperforms all the other methods by a very large margin.

[1] See online version for color figures.

In terms of efficiency and robustness, we can see that SSNAL performs much better than all the other first-order methods on these difficult large-scale problem. For example, SSNAL only needs about 2 minutes to produce a solution with the required accuracy such that $\max\{\eta_{gap}, \eta_D, \eta_{kkt}\} \leq 10^{-6}$ for the problem triazines4, while all first-order algorithms spend over 1 hour (3 hours for IADMM) and only produce poor accuracy solutions (with $\eta_{rel} \approx 10^{-4}$) that are much less accurate than SSNAL. One can see from Tables 2 and 3 that SSNAL can easily be 5 to 20 times faster than the best first-order method on different instances such as triazines4 and pyrim5.

5.4. Numerical results for synthetic data. Next we test our algorithms in the high-sample–low-dimension setting. The data used in this subsection are generated randomly from the following true model:

$$b = Ax + \varsigma\epsilon, \ \epsilon \sim N(\mathbf{0}, I).$$

In the experiments, the rows of $A \in \Re^{m\times n}$ are generated randomly from the multivariate normal distribution $N(\mathbf{0}, \Sigma)$. Here $\Sigma \in \Re^{n\times n}$ is a given symmetric matrix such that $\Sigma_{ij} = \hat{\gamma}^{|i-j|}$ for $i, j = 1, \ldots, p$ and $\hat{\gamma}$ is a given parameter. The tuning parameters β and ρ in (1.1) are chosen based on numerical experience.

The examples of x_0 presented below were mainly constructed based on the simulation scenarios used in [45, 30, 24]. As we want to focus on large-scale problems, we introduce a parameter k. In the first six scenarios, we use k to repeat every component of $x_0 \in \Re^{n_0}$ by k times consecutively to construct the actual $x \in \Re^n$, where $n = n_0 k$, while in the last case we use k in another strategy which will be explained later. The corresponding number of observations is chosen to be $\max\{80000, 0.5nk\}$. We use 80% of the observations to do the training. Instead of using a specified noise level ς for each case, we set $\varsigma = 0.1\|Ax\|/\|\epsilon\|$ for all examples.

1. The first setting is specified by the parameter vector

$$x^0 = (3, 1.5, 0, 0, 0, 2, 0, 0)^T.$$

   The correlation between the ith and jth predictor is

$$\text{corr}(i, j) = 0.9^{|i-j|} \ \forall \ i, j \in \{1, \ldots, 8\}.$$

2. In this setting, we have $n_0 = 20$ predictors. The parameter vector is structured into blocks:

$$x^0 = (\underbrace{0, \ldots, 0}_{5}, \underbrace{2, \ldots, 2}_{5}, \underbrace{0, \ldots, 0}_{5}, \underbrace{2, \ldots, 2}_{5})^T.$$

   The correlation between ith and jth predictor is given by $\text{corr}(i, j) = 0.3$.

3. This setting consists of $n_0 = 20$ predictors. The parameter vector is given by

$$x^0 = (5, 5, 5, 2, 2, 2, 10, 10, 10, \underbrace{0, \ldots, 0}_{11})^T.$$

   Within each of the first three blocks of 3 variables, the correlation between the two predictors is 0.9, but there is no correlation among different blocks.

4. The fourth setting consists of $n_0 = 13$ predictors. The parameter vector is structured into many small clusters:

$$x^0 = (0, 0, -1.5, -1.5, -2, -2, 0, 0, 1, 1, 4, 4, 4)^T.$$

The correlation between the ith and jth predictor is

$$\text{corr}(i,j) = 0.5^{|i-j|} \ \forall\ i,j \in \{1,\dots,13\}.$$

5. The fifth setting is the same as the fourth one, but with a higher correlation between the predictors where $\text{corr}(i,j) = 0.9^{|i-j|} \ \forall\ i,j \in \{1,\dots,13\}$.
6. In the sixth setting, we have $n_0 = 16$ predictors. The parameter vectors is structured such that big clusters coexist with small ones:

$$x^0 = (\underbrace{0,\dots,0}_{3},\underbrace{4,\dots,4}_{5},\underbrace{-4,\dots,-4}_{5},2,2,-1)^T.$$

The predictors are possibly negatively correlated: $\text{corr}(i,j) = (-1)^{|i-j|}0.8$.

7. (Another strategy to use k.) In the last setting, we use another strategy to construct the example. First we generate a 100-by-1 vector $\nu \sim N(\mathbf{0}, I_{100})$, then we create a histogram of the elements of the vector ν. To be specific, we bin the elements of ν into 20 equally spaced containers and return a 20-by-1 vector x_0 as the number of elements in each container. Let

$$x = (\underbrace{x_0,\dots,x_0}_{2k})^T.$$

The correlation between the ith and jth predictors is given by $\text{corr}(i,j) = 0.5$.

For all seven examples, we tested large-scale problems by setting $k = 100$. Note that all instances in this subsection are in the high-sample–low-dimension setting, that is $m \gg n$, just as discussed in section 4, so the primal approach is a better choice. In the following, we use "p-" to represent the primal approach and "d-" to represent the dual approach. For comparison, we terminate all the algorithms when the relative KKT residual satisfies $\eta_{kkt} \le 10^{-6}$.

Figure 2 shows the recovery results for the seven examples, where the red dots represent the actual x, and the blue dots represent the solution we obtained via p-Ssnal. As we can see from the figure, the clustered Lasso model can recover the group structure of the true regression parameter vector successfully.

As for the purpose of comparing the computational time, we can refer to Table 4 for the details. Since d-iAdmm and d-LAdmm can deal with the case when m is large, we also apply d-iAdmm and d-LAdmm here. In the table, we can see that p-Ssnal, p-Admm, and d-LAdmm can solve all the instances efficiently and accurately. In addition, d-iAdmm also gives a good performance except for eg6. The slightly poorer performance of d-iAdmm is reasonable since in these cases, m is large enough that the linear systems needed to be solved in the algorithm are huge. As for p-Apg, the numerical results of eg2 and eg6 are not particularly good since the corresponding Lipschitz constants in the problem are large.

As we can see, some of the first-order methods are comparable to p-Ssnal in these cases because our new formulation of the clustered Lasso regularizer vastly improves the performance of the first-order methods in the projection steps.

6. Conclusion. In this paper, we reformulate the clustered Lasso regularizer as a weighted ordered-Lasso regularizer. Based on the new formulation, we are able to derive a highly efficient algorithm for computing the proximal mapping in $O(n\log(n))$ operations that is crucial for designing efficient first-order and second-order algorithms for solving the clustered Lasso problem. Based on efficiently computing the generalized Jacobian of the proximal mapping, we design extremely fast semismooth Newton

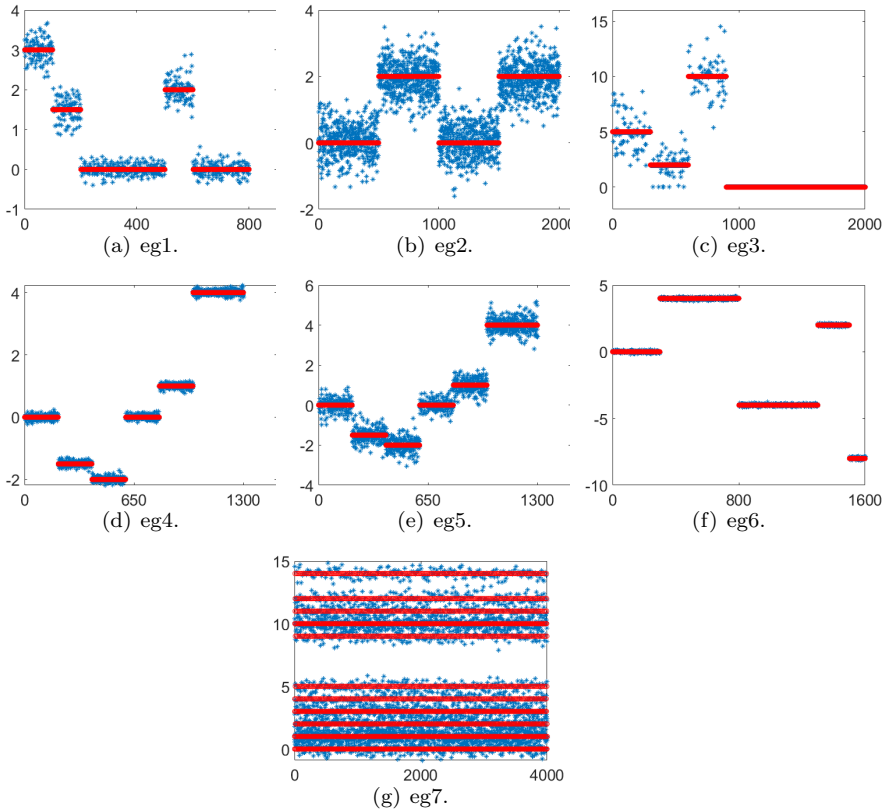

FIG. 2. *Recovery results obtained by the p-*SSNAL *algorithm when solving the clustered Lasso model on seven synthetic data sets. The algorithm is terminated by setting* $\eta_{kkt} \leq 10^{-6}$. *In the figures, the red dots represent the actual* x, *and the blue dots represent the solution we obtained via p-*SSNAL.

augmented Lagrangian algorithms, i.e., SSNAL, for solving the clustered Lasso problem or its dual. Our efficient implementation of the SSNAL algorithm heavily relies on the special structure that we have uncovered for the clustered Lasso regularizer. The numerical experiments on large-scale real data and synthetic data show the great advantages of our algorithms in comparison with other well-designed first-order methods for the clustered Lasso problem.

Table 4: The performance of various algorithms on the synthetic data sets. In the table, "a" = p-SSNAL, "b" = p-ADMM, "c" = p-APG, "d" = d-IADMM, "e" = d-LADMM. We terminate the algorithms when $\eta_{kkt} \leq 10^{-6}$. MSE denotes the MSE obtained by p-SSNAL. nnz(x) and gnnz(x) are obtained by p-SSNAL. Time is shown in the format of (hours:minutes:seconds).

	η_{kkt}	Time
Proname (m; n)	a | b | c | d | e	a | b | c | d | e
eg1 (32000; 800)	MSE = 1.4-1, nnz(x) = 714, gnnz(x) = 5	
$\lambda_{\max}(A^TA) = 6.21e{+}05$	7.4-7 | 9.7-7 | 9.9-7 | 9.9-7 | 9.9-7	02 | 01 | 02 | 20 | 44
eg2 (64000; 2000)	MSE = 3.4-1, nnz(x) = 1925, gnnz(x) = 7	
$\lambda_{\max}(A^TA) = 3.84e{+}07$	5.2-7 | 9.9-7 | 1.0-6 | 9.0-7 | 4.3-7	22 | 01:37 | 16:53 | 42 | 12

Continued on next page

Table 4 — continued from previous page

	η_{kkt}	Time
Proname (m; n)	a | b | c | d | e	a | b | c | d | e
eg3 (64000; 2000)	MSE = 1.1-1, nnz(x) = 898, gnnz(x) = 5	
$\lambda_{\max}(A^TA) = 1.76e{+}07$	8.0-7 | 9.8-7 | 9.8-7 | 9.2-7 | 9.9-7	01:40 | 02:50 | 03:18 | 25 | 01:59
eg4 (52000; 1300)	MSE = 3.4-2, nnz(x) = 1285, gnnz(x) = 6	
$\lambda_{\max}(A^TA) = 1.70e{+}05$	6.5-7 | 1.0-6 | 9.6-7 | 9.9-7 | 9.9-7	17 | 52 | 09 | 08 | 10
eg5 (52000; 1300)	MSE = 1.5-1, nnz(x) = 1231, gnnz(x) = 5	
$\lambda_{\max}(A^TA) = 1.00e{+}06$	8.4-7 | 9.8-7 | 1.0-6 | 9.9-7 | 1.0-6	01:26 | 01:37 | 02:07 | 41 | 01:37
eg6 (64000; 1600)	MSE = 1.5-2, nnz(x) = 1575, gnnz(x) = 6	
$\lambda_{\max}(A^TA) = 8.19e{+}07$	3.0-7 | 9.9-7 | **9.1-3** | **9.1-5** | 7.8-7	11 | 38 | 21:31 | 35:23 | 11
eg7 (64000; 4000)	MSE = 5.5-2, nnz(x) = 3998, gnnz(x) = 3	
$\lambda_{\max}(A^TA) = 2.21e{+}05$	8.1-7 | 9.0-7 | 9.4-7 | 1.0-6 | 9.3-7	58 | 01:16 | 50 | 03:22 | 01:00

REFERENCES

[1] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.

[2] M. J. Best and N. Chakravarti, *Active set algorithms for isotonic regression; a unifying framework*, Math. Program., 47 (1990), pp. 425–439.

[3] H. D. Bondell and B. J. Reich, *Simultaneous regression shrinkage, variable selection and clustering of predictors with OSCAR*, Biometrics, 64 (2008), pp. 115–123.

[4] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM Trans. Intell. Syst. Tech., 2 (2011), No. 27.

[5] 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*, Math. Program., 161 (2017), pp. 237–270.

[6] Y. Cui, D. F. Sun, and K.-C. Toh, *On the R-superlinear convergence of the KKT residuals generated by the augmented Lagrangian method for convex composite conic programming*, Math. Program. (2018), https://doi.org/10.1007/s10107-018-1300-6.

[7] J. Eckstein and D. P. Bertsekas, *On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program., 55 (1992), pp. 293–318.

[8] F. Facchinei and J.-S. Pang, *Finite-dimensional Variational Inequalities and Complementarity Problems*, Springer Ser. Oper. Res. Financ. Eng., Springer, New York, 2007.

[9] M. Fazel, T. K. Pong, D. F. Sun, and P. Tseng, *Hankel matrix rank minimization with applications to system identification and realization*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 946–977.

[10] J. Friedman, T. Hastie, and R. Tibshirani, *A Note on the Group Lasso and a Sparse Group Lasso*, preprint, https://arxiv.org/abs/1001.0736, 2010.

[11] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Comput. Math. Appl., 2 (1976), pp. 17–40.

[12] 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*, Rev. Fr. Autom. Inform. Rech. Opér. Anal. Numér., 9 (1975), pp. 41–76.

[13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed., Johns Hopkins Stud. Math. Sci., Johns Hopkins University Press, Baltimore, MD, 2013.

[14] J. Han and D. F. Sun, *Newton and quasi-Newton methods for normal maps with polyhedral sets*, J. Optim. Theory Appl., 94 (1997), pp. 659–676.

[15] 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, Association for Computing Machinery, New York, 2009, pp. 433–440.

[16] B. Kummer, *Newton's method for non-differentiable functions*, Adv. Math. Optim., 45 (1988), pp. 114–125.

[17] X. Li, D. F. Sun, and K.-C. Toh, *A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems*, SIAM J. Optim., 28 (2018), pp. 433–458.

[18] X. Li, D. F. Sun, and K.-C. Toh, *On efficiently solving the subproblems of a level-set method for fused Lasso problems*, SIAM J. Optim., 28 (2018), pp. 1842–1866.
[19] X. Li, D. F. Sun, and K.-C. Toh, *On the efficient computation of a generalized Jacobian of the projector over the Birkhoff polytope*, Math. Program. (2019), https://doi.org/10.1007/s10107-018-1342-9.
[20] J. Liu, L. Yuan, and J. Ye, *An efficient algorithm for a class of fused Lasso problems*, in Proceedings of the 16th ACM SIGKDD International conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, 2010, pp. 323–332.
[21] Z. Luo, D. F. Sun, K.-C. Toh, and N. Xiu, *Solving the OSCAR and SLOPE Models Using a Semismooth Newton-based Augmented Lagrangian Method*, preprint, https://arxiv.org/abs/1803.10740, 2018.
[22] F. J. Luque, *Asymptotic convergence analysis of the proximal point algorithm*, SIAM J. Control Optim., 22 (1984), pp. 277–293.
[23] R. Mifflin, *Semismooth and semiconvex functions in constrained optimization*, SIAM J. Control Optim., 15 (1977), pp. 959–972.
[24] S. Petry, C. Flexeder, and G. Tutz, *Pairwise Fused Lasso*, Technical report 102, Department of Statistics, University of Munich, Munich, 2011.
[25] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Math. Program., 58 (1993), pp. 353–367.
[26] S. M. Robinson, *Some continuity properties of polyhedral multifunctions*, Math. Program. Stud., 14 (1981), pp. 206–214.
[27] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
[28] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res., 1 (1976), pp. 97–116.
[29] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optim., 14 (1976), pp. 877–898.
[30] Y. She, *Sparse regression with exact clustering*, Electron. J. Stat., 4 (2010), pp. 1055–1096.
[31] D. F. Sun and J. Sun, *Semismooth matrix-valued functions*, Math. Oper. Res., 27 (2002), pp. 150–169.
[32] J. Sun, *On Monotropic Piecewise Quadratic Programming*, PhD thesis, University of Washington, Seattle, WA, 1986.
[33] L. Tang and P. X. Song, *Fused Lasso approach in regression coefficients clustering: Learning parameter heterogeneity in data integration*, J. Mach. Learn. Res., 17 (2016), pp. 3915–3937.
[34] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. R. Stat. Soc. Ser. B. Stat. Methodol., 58 (1996), pp. 267–288.
[35] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, *Sparsity and smoothness via the fused Lasso*, J. R. Stat. Soc. Ser. B. Stat. Methodol., 67 (2005), pp. 91–108.
[36] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, *A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation*, SIAM J. Sci. Comput., 32 (2010), pp. 1832–1857.
[37] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process., 57 (2009), pp. 2479–2493.
[38] G.-B. Ye and X. Xie, *Split Bregman method for large scale fused Lasso*, Comput. Statist. Data Anal., 55 (2011), pp. 1552–1569.
[39] Y.-L. Yu, *On decomposing the proximal map*, in Adv. Neural Inf. Process. Syst. 26, Curran Associates, Red Hook, NY, 2013, pp. 91–99.
[40] M. Yuan and Y. Lin, *Model selection and estimation in regression with grouped variables*, J. R. Stat. Soc. Ser. B. Stat. Methodol., 68 (2006), pp. 49–67.
[41] X. Zhang, M. Burger, and S. Osher, *A unified primal-dual algorithm framework based on Bregman iteration*, J. Sci. Comput., 46 (2011), pp. 20–46.
[42] Y. Zhang, N. Zhang, D. F. Sun, and K.-C. Toh, *An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems*, Math. Program. (2018), https://doi.org/10.1007/s10107-018-1329-6.
[43] X.-Y. Zhao, D. F. Sun, and K.-C. Toh, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM J. Optim., 20 (2010), pp. 1737–1765.
[44] L. W. Zhong and J. T. Kwok, *Efficient sparse modeling with automatic feature grouping*, IEEE Trans. Neural Netw. Learn. Syst., 23 (2012), pp. 1436–1447.
[45] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, J. R. Stat. Soc. Ser. B. Stat. Methodol., 67 (2005), pp. 301–320.