EFFICIENT HESSIAN BASED ALGORITHMS FOR SOLVING SPARSE GROUP LASSO AND MULTIPLE GRAPHICAL LASSO PROBLEMS

ZHANG YANGJING

(B.Sc., Tsinghua University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MATHEMATICS

NATIONAL UNIVERSITY OF SINGAPORE

2019

Supervisor:

Professor Toh Kim-Chuan, Main Supervisor

Examiners:

Professor Zhao Gong Yun

Associate Professor Chua Chek Beng, Nanyang Technological University

Professor Qi Hou-Duo, University of Southampton

To my parents

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

张羊晶

Zhang Yangjing

14 January 2019

Acknowledgements

I would like to take this opportunity to express my heartfelt thanks to all those who have contributed in one way or another to the completion of this thesis. Firstly, I would like to express my sincere gratitude to my supervisor Professor Toh Kim Chuan for the professional supervision and continuous support of my Ph.D study. I have learned a lot from his insight on numerical optimization and knowledge on algorithmic implementation. Additionally, the financial support from his research grant sustains my fifth year's research. This thesis would not have been possible without the mentorship and support of him.

My deepest thanks also go to Professor Sun defeng. He has led me to the flourishing area of optimization. His rigorous attitude and inexhaustible enthusiasm for research have been influencing me profoundly and continuously. I owe him a great debt of gratitude.

I would also like to express my thanks to all the members in our optimization group, for the fruitful discussions we made and the Saturday optimization seminars we attended. Particularly, I would like to convey my special thanks to my collaborator Dr. Zhang Ning for the helpful discussions and valuable cooperations, to my senior fellows Dr. Cui Ying and Dr. Li Xudong for their help and suggestions.

I am also grateful to my fellow graduate students at the Department of Mathematics, National University of Singapore, in particular, Guan Yu, Huang Ruizhi, Lam Xin Yee, Li Ning, Pang Tongyao, Wang Liuquan, Wu Bin. We shared the same office or the same hostel in the past few years and strived for the same goal together. The valuable and enjoyable time with them will always remain in my memory.

I am thankful to the National University of Singapore for providing four-year research scholarship towards the completion of the present dissertation and the financial support for the conference trip.

I am indebted to my father Zhang Wenhong and my mother Yang Huaxiang for their unconditional love, devotion, and support. Last but not least, my profound love and gratitude go to my husband Yao Jie. Our first meeting is in a math class.

"What does this symbol mean?"

"You sure you are in the right classroom? Haha, let me help you."
It is his understanding, encouragement, and love that accompany me along the long journey of Ph.D study. I would not have been here today if it were not for the love and support from my family.

Contents

Summary

This thesis focuses on designing efficient algorithms for solving large-scale statistical problems: the sparse group Lasso and multiple graphical Lasso problems.

The first part of the thesis has proposed an augmented Lagrangian method for large-scale non-overlapping sparse group Lasso problems with each subproblem being solved by a superlinearly convergent semismooth Newton method. The sparse group Lasso is a widely used statistical model which encourages the sparsity both on a group and within the group level. 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. Numerical experiments on both the synthetic and real data sets demonstrate that our proposed algorithm not only can solve the problems to high accuracy, but it is also far more superior than existing state-of-the-art first order methods for solving the sparse group Lasso problems.

The second part of the thesis is devoted to the computation of two multiple graphical models: the group graphical Lasso model and the fused graphical Lasso model, which can be applied to analysing data from different classes or over a temporal grid. We develop an efficient proximal point algorithm for solving these two models, where the subproblem in each iteration of the algorithm is solved by a superlinearly convergent semismooth Newton method. To implement the semismooth Newton method, we derived explicit expressions for the generalized Jacobian of the proximal mapping of the group graphical Lasso regularizer and that of the fused graphical Lasso regularizer. Unlike those widely used first order methods, our method has fully exploited the underlying second order information through the semismooth Newton method. This has not only accelerated the convergence of the algorithm, but also improves its robustness. The efficiency and robustness of our proposed algorithm are demonstrated by comparing with some state-of-the-art methods on both synthetic and real data sets.

Chapter 1

Introduction

In this thesis, we focus on designing efficient algorithms for solving some large-scale convex composite statistical problems. Of particular interest is the non-overlapping sparse group Lasso model, which has been widely applied to different fields, such as text processing, bioinformatics, signal interpretation, and object tracking. Besides, another aim of this thesis is to solve two multiple graphical models, which have been especially appealing for learning the conditional independence structures among a large number of variables from different classes or over a temporal grid.

1.1 Literature review

1.1.1 Sparse group Lasso models

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 [81]:

$$\min_{x\in\mathbb{R}^n} \ \frac{1}{2}\|\mathcal{A}x-b\|^2+\lambda_1\|x\|_1, \tag{1.1}$$

where $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map, $b \in \mathbb{R}^m$ is the given response vector, λ_1 is a nonnegative regularization parameter, $\|\cdot\|$ and $\|\cdot\|_1$ denote the ℓ_2 norm and ℓ_1 norm, respectively. Furthermore, by assuming that some prior information about the group structure of the underlying solution x is known, Yuan and Lin [92] proposed the group Lasso model:

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|\mathcal{A}x - b\|^2 + \lambda_2 \sum_{l=1}^{g} w_l \|x_{G_l}\|, \tag{1.2}$$

where λ_2 is a nonnegative regularization parameter, for $l = 1, 2, \ldots, g$, $w_l > 0$, and $G_l \subseteq \{1, 2, \ldots, n\}$ is the set of 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}. The group Lasso model (1.2) 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. [31] proposed the sparse group Lasso (SGLasso) model (1.3), potentially with overlaps between groups:

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|\mathcal{A}x - b\|^2 + \lambda_1\|x\|_1 + \lambda_2 \sum_{l=1}^{g} w_l \|x_{G_l}\|. \tag{1.3}$$

For convenience, we denote the SGLasso regularizer by the proper closed convex function

$$p(x) := \lambda_1\|x\|_1 + \lambda_2 \sum_{l=1}^{g} w_l \|x_{G_l}\|, \ \forall\, x \in \mathbb{R}^n. \tag{1.4}$$

One can observe that the SGLasso model (1.3) is a combination of the Lasso model (1.1) and the group Lasso model (1.2). The SGLasso problem (1.3) contains the Lasso problem (1.1) as a special case if we take the parameter $\lambda_2 = 0$. Besides, the SGLasso problem (1.3) reduces to the group Lasso problem (1.2) if λ_1 is zero. Apart from the above penalized regression models, there exist a number of variants with different regularizers, such as the fused Lasso [82] and the network Lasso [36].

The SGLasso model has been widely applied to different fields, such as text processing, bioinformatics, signal interpretation, and object tracking (e.g., [23, 43, 44, 47, 70, 99]). 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., [3, 92]), (randomized) block coordinate descent algorithm (see e.g., [72, 73, 78]), and alternating direction method of multipliers (see e.g., [8]). To the best of our knowledge, these existing algorithms are first order methods that are applied directly to the primal problem (1.3) and they hardly utilize any second order information. We note that even though there exist a number of algorithms for the Lasso problem (1.1) with second order information being incorporated, such as block active set methods [9, 46], orthant based methods [2, 9, 19], and the semismooth Newton augmented Lagrangian method (Ssnal) [54], there is currently no second order method designed for the SGLasso problem.

To design an efficient second order information based algorithm for solving the dual problem of the SGLasso problem (1.3), this thesis extends the Ssnal for the Lasso problem established in [54] with three major reasons. First of all, unlike the other methods, the Ssnal does not require the uniqueness of solutions for the primal problem. 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 an 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 S-GLasso 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 S-GLasso problem. The non-overlapping case can be treated as a preliminary study towards the final goal of designing a second order information based algorithm for solving the overlapping SGLasso problem. For the rest of this thesis, we make the following blanket assumption.

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

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 (1.3). This approach is motivated by the success of the Ssnal when applied to the dual of the Lasso problem [54] and that of the fused Lasso problem [55]. We note that the objective functions of the Lasso and fused Lasso problems are piecewise linear-quadratic, and therefore as proven in [54, 55], 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 performances of the Ssnal for solving the Lasso and fused Lasso problems. However, the regularizer p in the objective function of the SGLasso problem (1.3) 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 existing theoretical results. Fortunately, by leveraging on the recent advances made by Cui, Sun, and Toh [16] 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 (1.3) is in solving the subproblem in each iteration. As inspired by the success in [54, 55], 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 thesis 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 thesis aims to derive an explicit formula for them. We note that the SGLasso regularizer p enjoys the "prox-decomposition" property [90], similar to the fused Lasso regularizer (see [55]). 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) in 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.

In the numerical experiments, our ALM will be compared with existing state-of-the-art first order methods: (a) the sPADMM (semi-proximal alternating direction method of multipliers), (b) the APG (accelerated proximal gradient) method, and (c) the BCD (block coordinate descent) method. For one instance in the data set *E2006.train*, the dimension is $(16087, 150360)$, and Ssnal solved it to the desired accuracy in 3 seconds, sPADMM took more than 8 minutes, while APG failed to solve it within 10000 steps. And our algorithm is around 30 times faster compared to sPADMM and APG for over 60% of the tested instances on UCI data sets with random groups.

1.1.2 Multiple graphical Lasso models

In modern multivariate data analysis, one of the most important problems is the estimation of the precision matrix (or the inverse covariance matrix) via an undirected graphical model. A Gaussian graphical model for a Gaussian random vector $\Delta \sim \mathcal{N}_n(\mu, \Sigma)$ is represented by a graph $\mathcal{G} = (V, E)$, where V is a collection of n vertices corresponding to the n random variables and an edge E_{ij} is absent if and only if the i-th and j-th random variables are conditionally independent of each other, given all other variables. Furthermore, the i-th and j-th random variables are independent conditionally on the other variables if and only if the (i, j)-th entry in the precision matrix $(\Sigma^{-1})_{ij}$ is zero, as proven by Dempster [20]. Thus, finding the graph structure of a Gaussian graphical model is equivalent to the estimation of the corresponding precision matrix. In the high-dimensional and low-sample-size setting, it is always assumed that the conditional independence structure or the precision matrix is sparse in some sense. In other words, its corresponding undirected graph $\mathcal{G}$ is expected to to be sparse. To promote sparsity, there has been a great deal of interest in using the ℓ_1 norm penalties in statistical applications, for example, the famous Lasso model (1.1). In particular, various researchers (e.g., [4, 30, 93]) have adopted the following ℓ_1-norm penalized maximum likelihood approach to estimate the precision matrix:

$$\min_{\Theta \succ 0} \quad -\log\det\Theta + \langle S, \Theta \rangle + \lambda_1 \textstyle\sum_{i \neq j} |\Theta_{ij}| \tag{1.5}$$

where λ_1 is again a nonnegative penalty parameter, $S = \frac{1}{N-1}\sum_{i=1}^{N}(\delta_i - \bar{\mu})(\delta_i - \bar{\mu})^T$ is the sample covariance matrix with $\delta_1, \delta_2, \ldots, \delta_N$ being N samples drawn independently from the multivariate Gaussian distribution $\mathcal{N}_n(\mu, \Sigma)$ and $\bar{\mu}$ being the sample mean, i.e., $\bar{\mu} = \frac{1}{N}\sum_{i=1}^{N}\delta_i$.

The single Gaussian graphical model has been widely used in many applications. For solving the ℓ_1 norm penalized single Gaussian graphical model (1.5), numerous algorithms have been designed in the literature, e.g., block coordinate descent method [4, 5, 18, 30, 95], alternating direction method of multipliers [94], Nesterov's first order methods and their variants [4, 5, 59, 60], projected gradient method [22], interior point method [53], and proximal point algorithm [87]. Among them, the first four types of methods are generally referred to as first order methods. First order methods are often believed to be easily implementable and efficient in computing solutions of low or moderate accuracy in many cases. However, they generally have difficulty in obtaining high accuracy solutions. Unlike the first order methods, the last two types of methods can utilize second order information and therefore are always efficient in finding solutions of high accuracy.

In many applications, a single Gaussian graphical model is typically enough to capture the conditional independence structure of the random variables. However, in some situations it is more reasonable to fit a collection of such models jointly, due to the similarity or heterogeneity of the data involved. We refer to these models for estimating multiple precision matrices simultaneously as multiple graphical models. A scenario where multiple graphical models are more suitable than a single graphical model is when the data comes from several distinct but closely related classes, which share the same collection of variables but differ in terms of the dependency structure. Their dependency graphs can have common edges across a portion of all classes and unique edges restricted to only certain classes. In this case, fitting separate graphical models for distinct classes does not exploit the similarity among the dependency graphs. In contrast, joint estimation of these models could exploit information across different but related classes. An example is the inference of words relationships from webpages, which is used in our numerical experiments in section 4.4.4. In this example, webpages from the computer science departments of various universities are classified into several classes: Student, Faculty, Staff, Department, etc. In addition to the data from different classes, another scenario that would favor multiple graphical models over a single graphical model is when the data contains sequences of multivariate time-stamped observations. Such data might correspond to a series of dependency graphs over time. To visualize these dependency graphs and understand how they evolve over time, one might estimate the precision matrices jointly instead of separately. For example, the Standard & Poor's 500 stock price data, tested in section 4.4.3, contains the daily returns of 500 stocks over several years. The relationships among stocks might change smoothly over time. In summary, there are two major applications of multiple graphical models: (i) estimating multiple precision matrices simultaneously for a collection of variables across distinct classes; (ii) inferring the time-varying networks and finding the change-points.

The following paragraph formulates the multiple graphical model explicitly. Suppose that there are L random n-vectors $\Delta^{(l)}$ (from different classes or over a temporal grid) drawn independently from different distributions $\mathcal{N}_n(\mu^{(l)}, \Sigma^{(l)})$, $l = 1, 2, \ldots, L$. Assume that the vector $\Delta^{(l)}$ has N_l observations $\delta_1^{(l)}, \delta_2^{(l)}, \ldots, \delta_{N_l}^{(l)}$ for each $l \in \{1, 2, \ldots, L\}$. Then, the sample means are $\bar{\mu}^{(l)} := \frac{1}{N_l}\sum_{i=1}^{N_l} \delta_i^{(l)}$ and the sample covariance matrices are $S^{(l)} := \frac{1}{N_l-1}\sum_{i=1}^{N_l}(\delta_i^{(l)} - \bar{\mu}^{(l)})(\delta_i^{(l)} - \bar{\mu}^{(l)})^T$, $l = 1, 2, \ldots, L$. In this setting, a single Gaussian graphical model is usually not applicable since it is limited to handle observations from the same distribution. A multiple graphical model for estimating precision matrices $(\Sigma^{(l)})^{-1}$, $l = 1, 2, \ldots, L$ jointly is the optimization model with arguments $\Theta := (\Theta^{(1)}, \ldots, \Theta^{(L)}) \in \mathbb{S}^n \times \cdots \times \mathbb{S}^n$:

$$\min_{\Theta} \quad \sum_{l=1}^{L} \big(-\log\det\,\Theta^{(l)} + \langle S^{(l)}, \Theta^{(l)}\rangle\big) + \mathcal{P}(\Theta), \tag{1.6}$$

where $\mathcal{P}$ is a penalty function, which usually promotes sparsity in each $\Theta^{(l)}$ and encourages different $\Theta^{(l)}$'s to share certain characteristics. The solution $(\Theta^{(1)}, \ldots, \Theta^{(L)})$ of problem (1.6) will constitute an estimate of $((\Sigma^{(1)})^{-1}, \ldots, (\Sigma^{(L)})^{-1})$. One can observe that the multiple graphical model (1.6) reduces to the single graphical model (1.5) when $L = 1$ and $\mathcal{P}(\Theta) = \lambda_1 \sum_{i\neq j} |\Theta_{ij}^{(1)}|$. In the literature, various forms of the penalty function $\mathcal{P}$ that can encourage similarity among these graphical models have been considered to link the estimation of separate graphical models [1, 17, 33, 35, 38, 40, 66, 84]. The following notations of various norms are defined in section 2.1 which summarizes all the used notations. One typical type of penalties is a mixture of the ℓ_1 norm and the (sequential or pairwise) fused penalty: Ahmed and Xing [1] and Yang et al. [88] dealt with the following penalty of this type

$$\mathcal{P}(\Theta) = \lambda_1 \|\Theta\|_{1,1,*} + \lambda_2 \sum_{l=2}^{L} \|\Theta^{(l)} - \Theta^{(l-1)}\|_{1,*}.$$

A slightly different penalty, which involves diagonal elements, was used by Monti et al. [66]

$$\mathcal{P}(\Theta) = \lambda_1 \|\Theta\|_{1,1} + \lambda_2 \sum_{l=2}^{L} \|\Theta^{(l)} - \Theta^{(l-1)}\|_{1}.$$

Applying the Frobenius norm to the sequential fused terms, Gibberd and Nelson [33] presented another penalty

$$\mathcal{P}(\Theta) = \lambda_1 \|\Theta\|_{1,1,*} + \lambda_2 \sum_{l=2}^{L} \|\Theta^{(l)} - \Theta^{(l-1)}\|_{F,*}.$$

In addition to the sequential fused penalty used above, the pairwise fused penalty was also involved, for example, in the penalty proposed by Danaher, Wang, and Witten [17]

$$\mathcal{P}(\Theta) = \lambda_1 \|\Theta\|_{1,1,*} + \lambda_2 \sum_{l<l'} \|\Theta^{(l)} - \Theta^{(l')}\|_{1}.$$

Another type of penalties makes use of the $\ell_{1,q}$ regularizers: such as $\mathcal{P}(\Theta) = \lambda_2 \|\Theta\|_{1,\infty}$ by Honorio and Samaras [40], $\mathcal{P}(\Theta) = \lambda_2 \|\Theta\|_{1,2,*}$ by Varoquaux et al. [84], and $\mathcal{P}(\Theta) = \lambda_1 \|\Theta\|_{1,1,*} + \lambda_2 \|\Theta\|_{1,2,*}$ by Danaher, Wang, and Witten [17]. Apart from the above two types, another type of penalties is based on the smart decompositions of $\Theta^{(l)}$ into a common substructure A and an individual substructure $B^{(l)}$. Guo et al. [35] utilized the decompositions $\Theta^{(l)} = A \odot B^{(l)}$, where $\odot$ denotes the Hadamard product, and proposed the following penalty via a nonconvex optimization problem:

$$\begin{aligned} \mathcal{P}(\Theta) = \min_{A,B} \quad & \lambda_1 \|A\|_{1,*} + \lambda_2 \|B\|_{1,1,*} \\ \text{s.t.} \quad & \Theta^{(l)} = A \odot B^{(l)},\ l = 1, 2, \ldots, L, \\ & A_{ii} = 1,\ A_{ij} \geq 0,\ i, j = 1, 2, \ldots, p, \\ & A \in \mathbb{S}^p,\ B = (B^{(1)}, \ldots, B^{(L)}) \in \mathbb{S}^p \times \cdots \times \mathbb{S}^p. \end{aligned}$$

While Hara and Washio [38] adopted different decompositions $\Theta^{(l)} = A + B^{(l)}$ and proposed the following penalty based on a convex optimization problem:

$$\begin{aligned} \mathcal{P}(\Theta) = \min_{A,\,B} \quad & \lambda_1 \|A\|_1 + \lambda_2 \|B\|_{1,\,q} \\ \text{s.t.} \quad & \Theta^{(l)} = A + B^{(l)}, \\ & A + B^{(l)} \succ 0,\ l = 1, 2, \dots, L, \end{aligned}$$

In this thesis, we focus on the following two particular choices of regularizers:

(i) the group graphical Lasso (GGL) regularizer which was initially presented in [17]:

$$\begin{aligned} \mathcal{P}(\Theta) \ &= \lambda_1 \|\Theta\|_{1,\,1,\,*} + \lambda_2 \|\Theta\|_{1,\,2,\,*} \\ &= \lambda_1 \sum_{l=1}^{L} \sum_{i \neq j} |\Theta_{ij}^{(l)}| + \lambda_2 \sum_{i \neq j} \left(\sum_{l=1}^{L} |\Theta_{ij}^{(l)}|^2 \right)^{1/2}; \end{aligned} \tag{1.7}$$

(ii) the fused graphical Lasso (FGL) regularizer which was used in [1, 88]:

$$\begin{aligned} \mathcal{P}(\Theta) \ &= \lambda_1 \|\Theta\|_{1,\,1,\,*} + \lambda_2 \sum_{l=2}^{L} \|\Theta^{(l)} - \Theta^{(l-1)}\|_{1,\,*} \\ &= \lambda_1 \sum_{l=1}^{L} \sum_{i \neq j} |\Theta_{ij}^{(l)}| + \lambda_2 \sum_{l=2}^{L} \sum_{i \neq j} |\Theta_{ij}^{(l)} - \Theta_{ij}^{(l-1)}|. \end{aligned} \tag{1.8}$$

We refer to the problem (1.6) where the regularizer $\mathcal{P}$ is defined by (1.7) as the GGL problem and the problem (1.6) where the regularizer $\mathcal{P}$ is defined by (1.8) as the FGL problem. For both GGL and FGL regularizers, the first terms are exactly the same, which promote sparsity in the estimated precision matrices. When $\lambda_2 = 0$, the GGL and FGL problems amount to performing L uncoupled precision matrix estimation problems (1.5). Without loss of generality, we assume that $\lambda_1 + \lambda_2 > 0$ throughout this thesis. Next, we give further explanations and comparisons of the two regularizers.

(a) The GGL regularizer defined in (1.7), acting on a collection of matrices, can be viewed as an extension of the sparse group Lasso regularizer acting on a vector proposed in [31]. More precisely, the GGL regularizer can be viewed as the sparse group Lasso regularizer applied to the (i, j)-th elements across all L precision matrices (see (4.5) for details). Due to the second term of the GGL regularizer, the zeros in the L estimated precision matrices $\Theta^{(l)}$'s tend to occur at the same indices. Specifically, each $\Theta^{(l)}$ in the solution to the GGL problem will have an identical pattern of nonzero elements when $\lambda_1 = 0$ and $\lambda_2 > 0$, as stated in [17].

(b) The FGL regularizer in (1.8) is in some sense a generalized fused Lasso regularizer [82]. It applies the ℓ_1 penalty to all the off-diagonal elements of the L precision matrices and the differences of the elements of successive precision matrices. Many elements with the same indices in the estimated matrices $\Theta^{(1)}, \ldots, \Theta^{(L)}$ will be close or even identical when the parameter λ_2 is large enough. Therefore, the FGL regularizer encourages not only shared pattern of sparsity, but also shared values across different graphs, whereas the GGL regularizer encourages a weaker form of regularity that merely promotes the similarity of the network structure.

The regularized multiple Gaussian graphical models have been widely used in various applications. However, the existing algorithms for solving the GGL or FGL problem are quite limited in the literature. One of the most extensively used algorithms for solving this class of problems is the alternating direction method of multipliers (ADMM), see e.g., [17, 33, 37]. Besides, a proximal Newton-type method (see e.g., [41, 51, 83]) was implemented by Yang et al. [88] for solving the FGL problems. In particular, the method in [88] is a hybrid of the proximal gradient method with Armijo backtracking line search and the non-monotone spectral projected gradient (NSPG) method (see e.g., [61]). As we know, ADMM could be a fast first order method for finding approximate solutions of low or moderate accuracy. However, ADMM hardly utilize any second order information, which must be used in order to obtain highly accurate solutions. Although the proximal Newton-type method does incorporate some form of second order information, a complicated quadratic approximation problem has to be solved in each iteration, and this computation is usually time-consuming. It is worth mentioning that the regularizers are often introduced to promote certain structures in the estimated precision matrices, and the trade-off between the biases and variances in the resulting estimates is controlled by the choice of the regularization parameters [26]. But in practice, it is extremely hard to find the optimal regularization parameters. Therefore, a sequence of regularization parameters is usually applied in the practical implementations, and consequently, a sequence of corresponding optimization problems need to be solved [27]. Under such a circumstance, a highly efficient and robust algorithm for finding the optimal solutions becomes particularly important.

In this thesis, we apply a semismooth Newton (SSN) based proximal point algorithm (PPA) for solving the GGL and FGL models. It is well known that the PPA can be treated as a dual application of the augmented Lagrangian method (ALM) for solving convex optimization problems [75]. Our approach is greatly inspired by and based on the works [97] (details are shown in Chapter 3) and [55], which have presented superior numerical performance of the SSN based ALM, known as Ssnal, for solving the SGLasso and fused Lasso problems, respectively. Thanks to the fact that the GGL and FGL problems have close connections to the SGLasso and fused Lasso problems, respectively, many of the virtues and theoretical insights of the Ssnal for solving the SGLasso and fused Lasso problems can be observed when we apply the SSN based PPA to solve the GGL and FGL problems. However, we should emphasize that solving the GGL and FGL problems is much more challenging than solving the SGLasso and fused Lasso problems. Our current problems are matrix optimization problems over the Cartesian product of a collection of cones of symmetric positive definite matrices, whereas each of the latter problems is an optimization problem over a single vector variable. Specifically, the difficulties are mainly due to the log-determinant function $\log\det(\cdot)$ and the matrix arguments, as described below.

(a) Unlike the simple quadratic functions in the SGLasso and fused Lasso problems, which are essentially regularized least square problems, the function $\log\det(\cdot)$ is strictly concave and twice differentiable on the space of positive definite matrices. Therefore, the GGL and FGL models contain the underlying constraints: their solutions should be positive definite. The positive definite constraints increase greatly the difficulty and complexity of theoretical analysis and numerical implementation.

(b) The construction of an efficiently computable element in the generalized Jacobian of the proximal mapping of the SGLasso (or fused Lasso) regularizer is an essential step in [97] (or [55]) for solving the SGLasso (or fused Lasso) problem. Based on the well-established constructions, we could obtain an efficiently computable generalized Jacobian of the proximal mapping of the GGL (or FGL) regularizer. However, this process needs more complicated manipulations of coordinates for a collection of matrix variables, unlike the vector case of the SGLasso (or fused Lasso) problem.

The key issue in the implementation of the PPA for solving the GGL (or FGL) model is the computation of the solution of the subproblem in each PPA iteration. For this purpose, we will design an SSN method to solve those subproblems. We note that the numerical performance of the SSN method relies critically on the efficient calculation of the generalized Jacobian of the proximal mapping of the GGL (or FGL) regularizer and that of the log-determinant function. Fortunately, the generalized Jacobian of the proximal mapping of the GGL regularizer and that of the FGL regularizer can be constructed efficiently based on those of the proximal mapping of the SGLasso regularizer given in [97] and that of the fused Lasso regularizer given in [55], respectively. As a result, the generalized Jacobian of the proximal mapping of the GGL or FGL regularizer would inherit the structured sparsity (referred to as second order sparsity) from that of the SGLasso or fused Lasso regularizer. Due to the structured sparsity, the computation of a matrix-vector product in the SSN method is reasonably cheap and thus the SSN method is quite efficient for solving each subproblem. To summarize, our SSN based PPA for solving the GGL and FGL problems has a linear convergent guarantee and the convergence rate can be as fast as one wish by choosing a sufficiently large proximal penalty parameter. Moreover, the SSN method for solving each of the PPA subproblems can be shown to be superlinearly convergent. Thus, based on these excellent convergent properties and the novel exploitation of second order sparsity, we can expect the SSN based PPA for solving the GGL and FGL problems to be highly efficient numerically.

We illustrate the performance on a number of data sets: (a) recovering the structure of the simulated nearest-neighbour networks, (b) analysing the correlations among Standard & Poor 500 stocks, (c) learning semantic connections among terms from webpages or 20 newsgroups. The multiple graphical Lasso model can fully recover the true structure of the nearest-neighbour networks. On the Standard & Poor 500 stock price data sets, our algorithm solved all instances within 2 minutes and is faster than the alternating direction method of multipliers and the proximal Newton-type method except for one instance. When applied to analysing the dependency structure of terms in the newsgroups data, our algorithm is $8 \sim 10$ times faster in comparison with the other two methods for over 40% of the tested instances.

1.2 Contributions

One important contribution of the thesis is the design of 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. Meanwhile, the explicit formula for the generalized Jacobian of the proximal mapping of the SGLasso regularizer has been established. As far as we know, the elements in the generalized Jacobian of the proximal mapping of the SGLasso regularizer have not been derived before. Theoretically, it is shown 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. Our proposed algorithm is also shown numerically to be far more efficient than existing state-of-the-art first order methods for solving sparse group Lasso problems. The first part of the thesis has been submitted to and accepted by the journal *Mathematical Programming* [97].

The thesis also contributes to the efficient computations of the group graphical Lasso problem and the fused graphical Lasso problem. Specifically, it has designed a semismooth Newton based proximal point algorithm for both problems. The proposed algorithm is proven to be superlinearly convergent for solving the group and fused graphical Lasso problems. As indispensable parts for the semismooth Newton method for solving each subproblem, the generalized Jacobian of the proximal mapping of the group graphical Lasso regularizer and that of the fused graphical Lasso regularizer have been derived for the first time in the literature. Extensive numerical experiments have been conducted to show the efficiency of our algorithm on the data sets of simulated nearest-neighbour graphs, Standard & Poor's 500 stock price, university webpages, and 20 newsgroups.

1.3 Thesis organization

The rest of the thesis is organized as follows. Chapter 2 presents some preliminaries that are critical for subsequent discussions. In chapter 3, we design the semismooth Newton based augmented Lagrangian method (Ssnal) for solving the dual of the SGLasso problem and derive the superlinear convergence results. A semismooth Newton method based proximal point algorithm is implemented for solving the group graphical Lasso problem and the fused graphical Lasso problem in chapter 4. Finally, concluding remarks are given in chapter 5.

Chapter 2

Preliminaries

2.1 Notations

- For a linear map $\mathcal{A}$, we denote its adjoint by $\mathcal{A}^*$. For a matrix A, we denote its transpose by A^T.
- For any proper convex function f, we denote its conjugate function by f^*, i.e., $f^*(x) = \sup_z\{\langle x, z\rangle - f(z)\}$. We denote its effective domain by $\operatorname{dom} f$.
- For a set C, we denote its convex hull by $\operatorname{conv}\{C\}$ and its relative interior by $\operatorname{ri}\{C\}$.
- For a given closed convex set C and a vector x, we denote the distance of x to C by $\operatorname{dist}(x, C) := \inf_{x'\in C}\{\|x - x'\|\}$ and the Euclidean projection of x onto C by $\Pi_C(x) := \arg\min_{x'\in C}\{\|x - x'\|\}$.
- We define $\operatorname{sign}(\cdot)$ in a component-wise fashion such that $\operatorname{sign}(t) = 1$ if $t > 0$, $\operatorname{sign}(t) = 0$ if $t = 0$, and $\operatorname{sign}(t) = -1$ if $t < 0$. The function $\max(\cdot\,, 0) = (\cdot)_+$ is also defined in a component-wise fashion, where $(t)_+ = t$ if $t > 0$ and $(t)_+ = 0$ if $t \leq 0$.
- The function composition is denoted by $\circ$, that is, for any functions f and g, $(f \circ g)(\cdot) := f(g(\cdot))$. The Hadamard product is denoted by $\odot$.

- For a given vector x, $\mathrm{supp}(x)$ denotes the support of x, i.e., the set of indices such that $x_i \neq 0$.
- For a given vector $x \in \mathbb{R}^n$ and an index set $G \subseteq \{1, 2, \ldots, n\}$, x_G denotes the restriction of the vector x to the index set G.
- e denotes the vector of all ones.
- I_n denotes the $n \times n$ identity matrix, and I denotes an identity matrix or map when the dimension is clear from the context.
- $\mathrm{Diag}(A_1, A_2, \ldots, A_n)$ denotes the block diagonal matrix whose i-th diagonal block is the matrix A_i, $i = 1, 2, \ldots, n$. $\mathrm{diag}(A)$ denotes the diagonal vector of a matrix A.
- We use the MATLAB notation $[A; B]$ to denote the matrix obtained by appending B below the last row of A, when the number of columns of A and B is identical.
- Given $\{G_l \,|\, l = 1, 2, \ldots, g\}$ as a partition of the set $\{1, 2, \ldots, n\}$, define the linear operator $\mathcal{P}_l : \mathbb{R}^n \to \mathbb{R}^{|G_l|}$ by $\mathcal{P}_l x = x_{G_l}$. Define $\mathcal{P} : \mathbb{R}^n \to \mathbb{R}^n$ by $\mathcal{P}x = [\mathcal{P}_1 x; \mathcal{P}_2 x; \ldots; \mathcal{P}_g x] = [x_{G_1}; x_{G_2}; \ldots; x_{G_g}]$, $\forall x \in \mathbb{R}^n$. Define $\mathcal{B}_2 := \mathcal{B}_2^{\lambda_{2,1}} \times \cdots \times \mathcal{B}_2^{\lambda_{2,g}}$, where $\mathcal{B}_2^{\lambda_{2,l}} := \{u_l \in \mathbb{R}^{|G_l|} \,|\, \|u_l\| \leq \lambda_{2,l}\}$ and $\lambda_{2,l} := \lambda_2 w_l$.
- $\mathbb{S}^n_+$ ($\mathbb{S}^n_{++}$) denotes the cone of positive semidefinite (definite) matrices in the space of $n \times n$ real symmetric matrices $\mathbb{S}^n$. For any A, $B \in \mathbb{S}^n$, we denote $A \succeq B$ if $A - B \in \mathbb{S}^n_+$ and $A \succ B$ if $A - B \in \mathbb{S}^n_{++}$. In particular, $A \succeq 0$ indicates $A \in \mathbb{S}^n_+$, and $A \succ 0$ indicates $A \in \mathbb{S}^n_{++}$.
- We let $\mathcal{X} := \mathbb{S}^n_+ \times \cdots \times \mathbb{S}^n_+$ and $\mathcal{Y} =: \mathbb{S}^n \times \cdots \times \mathbb{S}^n$ be the Cartesian product of L positive semidefinite cones $\mathbb{S}^n_+$ and that of L spaces of symmetric matrices $\mathbb{S}^n$ respectively.
- For any matrix $A \in \mathbb{R}^{m \times n}$, A_{ij} denotes the (i, j)-th element of A. For any $X := (X^{(1)}, \ldots, X^{(L)}) \in \mathcal{Y}$, $X_{[ij]} := [X^{(1)}_{ij}; \ldots; X^{(L)}_{ij}] \in \mathbb{R}^L$ denotes the column

vector obtained by taking out the (i, j)-th elements across all L matrices $X^{(l)}$, $l = 1, 2, \ldots, L$.

- We define the vectorization operator $vec : \mathcal{Y} \to \mathbb{R}^{Ln(n+1)/2}$ as follows:

  $$vec(X) = [X_{[11]}; X_{[22]}; \ldots; X_{[nn]}; X_{[12]}; X_{[13]}; X_{[23]}; \ldots; X_{[1n]}; \ldots; X_{[(n-1)n]}], \quad X \in \mathcal{Y}.$$

  Besides, we let $mat : \mathbb{R}^{Ln(n+1)/2} \to \mathcal{Y}$ be the inverse of vec.

- $\text{cov}(A) \in \mathbb{R}^{L\times L}$ denotes the sample covariance if $A \in \mathbb{R}^{N\times L}$ is a matrix whose columns represent L random variables and whose rows represent N observations. Suppose a_i^T is the i-th row of the matrix A, $i = 1, 2, \ldots, N$, the sample mean is $\bar{\mu} := \frac{1}{N}\sum_{i=1}^N a_i$, and the sample covariance is $\text{cov}(A) := \frac{1}{N-1}\sum_{i=1}^N (a_i - \bar{\mu})(a_i - \bar{\mu})^T$.

- We denote the Moore-Penrose pseudo inverse of any matrix $A \in \mathbb{R}^{m\times n}$ by $A^\dagger$.

- For any matrix A, we denote $\|A\|_1 := \sum_{i,j}|A_{ij}|$, $\|A\|_{1,*} := \sum_{i\neq j}|A_{ij}|$, $\|A\|_F := \sqrt{\sum_{i,j}|A_{ij}|^2}$, $\|A\|_{F,*} := \sqrt{\sum_{i\neq j}|A_{ij}|^2}$.

- For a collection of matrices $\Theta := (\Theta^{(1)}, \ldots, \Theta^{(L)}) \in \mathbb{S}^n \times \cdots \times \mathbb{S}^n$, we denote $\|\Theta\|_{1,q} := \sum_{i,j}\left(\sum_{l=1}^L |\Theta_{ij}^{(l)}|^q\right)^{1/q}$, $\|\Theta\|_{1,q,*} := \sum_{i\neq j}\left(\sum_{l=1}^L |\Theta_{ij}^{(l)}|^q\right)^{1/q}$, for $q \in [1, \infty)$; and $\|\Theta\|_{1,\infty} := \sum_{i,j}\max_l |\Theta_{ij}^{(l)}|$, $\|\Theta\|_{1,\infty,*} := \sum_{i\neq j}\max_l |\Theta_{ij}^{(l)}|$.

2.2 Moreau-Yosida regularization

In this section, an important tool, the Moreau-Yosida regularization is introduced, which is in some sense one way to smooth a nonsmooth function. Let $\mathcal{E}$ be a real finite dimensional Euclidean space equipped with an inner product $\langle\cdot,\cdot\rangle$ and its induced norm $\|\cdot\|$. Let $f : \mathcal{E} \to (-\infty, +\infty]$ be a closed proper convex function. The Moreau-Yosida regularization of f at $x \in \mathcal{E}$ is defined by

$$\Phi_f(x) := \min_{y\in\mathcal{E}}\left\{f(y) + \frac{1}{2}\|y - x\|^2\right\}, \ \forall\, x \in \mathcal{E}. \tag{2.1}$$

The function Φ is also known as the Moreau envelope of f. From [67, 89], we have the following proposition which shows that (2.1) is well-defined.

Proposition 2.1. *For any $x \in \mathcal{E}$, problem* (2.1) *has a unique optimal solution.*

Definition 2.1 (Proximal point mapping). The unique optimal solution of (2.1), denoted by $\text{Prox}_f(x)$, is called the proximal mapping of x associated with f.

It is well known that the proximal mappings associated with ℓ_1 norm and ℓ_2 norm can be expressed as follows: for any given $c > 0$ and $u \in \mathbb{R}^n$,

$$\text{Prox}_{c\|\cdot\|_1}(u) = \text{sign}(u) \odot \max\{|u| - ce, 0\},$$

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

The following propositions provide some important properties of the Moreau-Yosida regularizarion.

Proposition 2.2. *[39, Theorem XV.4.1.4 and Theorem XV.4.1.7] Let $f : \mathcal{E} \to (-\infty, +\infty]$ be a closed proper convex function, Φ_f be the Moreau-Yosida regularization of f, and Prox_f be the associated proximal point mapping. Then the following properties hold.*

(i) $\arg\min_{x\in\mathcal{E}} f(x) = \arg\min_{x\in\mathcal{E}} \Phi_f(x)$.

(ii) *Both Prox_f and $I - \text{Prox}_f$ are firmly non-expansive, i.e., for any x, $y \in \mathcal{E}$,*

$$\begin{aligned} \|\text{Prox}_f(x) - \text{Prox}_f(y)\|^2 &\leq \langle \text{Prox}_f(x) - \text{Prox}_f(y), x - y\rangle, \\ \|(x - \text{Prox}_f(x)) - (y - \text{Prox}_f(y))\|^2 &\leq \langle (x - \text{Prox}_f(x)) - (y - \text{Prox}_f(y)), x - y\rangle. \end{aligned}$$

(iii) *The Moreau envelope Φ_f is continuously differentiable, and its gradient can be computed via*

$$\nabla\Phi_f(x) = x - \text{Prox}_f(x),\ \forall\, x \in \mathcal{E}.$$

Proposition 2.3 (**Moreau decomposition**). *Let $f : \mathcal{E} \to (-\infty, +\infty]$ be a closed proper convex function and f^* be its conjugate function. Let σ be a positive scalar. Then the following Moreau identity holds:*

$$\mathrm{Prox}_{\sigma f}(x) + \sigma \mathrm{Prox}_{\sigma^{-1} f^*}(\sigma^{-1} x) = x, \ \forall\, x \in \mathcal{E}.$$

Proof. For any $u \in \mathcal{E}$, we have that $u = \mathrm{Prox}_{\sigma f}(x) \iff x - u \in \sigma \partial f(u) \iff u \in \partial f^*(\sigma^{-1}x - \sigma^{-1}u) \iff \mathrm{Prox}_{\sigma^{-1} f^*}(\sigma^{-1}x) = \sigma^{-1}x - \sigma^{-1}u \iff u + \sigma \mathrm{Prox}_{\sigma^{-1} f^*}(\sigma^{-1}x) = x$. This completes the proof. □

Next we present some properties about the proximal mapping associated with a log-determinant function and its Jacobian, which are mainly adopted from [85, 87]. The results shall be used in the sequel since the log-determinant function is involved in the objective function of the multiple graphical Lasso model (1.6). The log-determinant function is defined as follows:

$$\vartheta(A) := -\log\det(A), \ \forall\, A \in \mathbb{S}^n.$$

Given $\beta > 0$, two scalar functions are defined as follow:

$$\phi_\beta^+(x) := \frac{\sqrt{x^2 + 4\beta} + x}{2}, \ \phi_\beta^-(x) := \frac{\sqrt{x^2 + 4\beta} - x}{2}, \ \forall\, x \in \mathbb{R}.$$

In addition, their matrix counterparts are defined by

$$\phi_\beta^+(A) := Q\mathrm{Diag}(\phi_\beta^+(d_1), \ldots, \phi_\beta^+(d_n))Q^T,$$
$$\phi_\beta^-(A) := Q\mathrm{Diag}(\phi_\beta^-(d_1), \ldots, \phi_\beta^-(d_n))Q^T,$$

for any $A \in \mathbb{S}^n$ with its eigenvalue decomposition $A = Q\mathrm{Diag}(d_1, d_2, \ldots, d_n)Q^T$, where $d_1 \geq d_2 \geq \cdots \geq d_n$. It is easy to show that ϕ_β^+ and ϕ_β^- are well-defined. Moreover, $\phi_\beta^+(A)$ and $\phi_\beta^-(A)$ are positive definite for any $A \in \mathbb{S}^n$.

Utilizing the functions defined above, the following proposition gives the proximal mapping of the log-determinant function ϑ.

Proposition 2.4. *Let $\vartheta(\cdot) := -\log\det(\cdot)$ be defined on $\mathbb{S}^n_{++}$ and $\beta > 0$. Then it holds that*

$$\begin{aligned} &\phi^+_\beta(A) = \mathrm{Prox}_{\beta\vartheta}(A) = \arg\min_{B\succ 0}\left\{\vartheta(B) + \tfrac{1}{2\beta}\|B-A\|^2\right\},\ \forall\, A\in\mathbb{S}^n, \\ &\Phi_{\beta\vartheta}(A) = -\log\det(\phi^+_\beta(A)) + \tfrac{1}{2\beta}\|\phi^-_\beta(A)\|^2. \end{aligned} \tag{2.2}$$

Furthermore, the function $\Phi_{\beta\vartheta}(\cdot)$ is strictly convex on $\mathbb{S}^n_{++}$.

Proof. The relationships in (2.2) can be obtained directly from [87, Proposition 2.3]. From the fact the $\vartheta(\cdot)$ is strictly convex on $\mathbb{S}^n_{++}$, and using the definitions of the Moreau-Yosida envelope and strict convexity, we can deduce that the function $\Phi_{\beta\vartheta}(\cdot)$ is strictly convex on $\mathbb{S}^n_{++}$. The proof is completed. $\square$

Proposition 2.5. *[85, Lemma 2.1(b)] Let β be a given positive scalar. The function $\phi^+_\beta : \mathbb{S}^n \to \mathbb{S}^n$ is continuously differentiable, and its directional derivative $(\phi^+_\beta)'(A)[B]$ at A for any $B \in \mathbb{S}^n$ is given by*

$$(\phi^+_\beta)'(A)[B] = Q(\Gamma \odot (Q^TBQ))Q^T,$$

where A admits the eigenvalue decomposition $A = Q\mathrm{Diag}(d_1, d_2, \dots, d_n)Q^T$, $d_1 \geq d_2 \geq \cdots \geq d_n$, and $\Gamma \in \mathbb{S}^n$ is defined by

$$\Gamma_{ij} = \frac{\phi^+_\beta(d_i) + \phi^+_\beta(d_j)}{\sqrt{d_i^2 + 4\beta} + \sqrt{d_j^2 + 4\beta}},\ \ i, j = 1, 2, \dots, n.$$

2.3 Semismooth Newton methods

This section introduces the concept of semismoothness and semismooth Newton methods. We begin with the definitions of directional differentiability. Various definitions of directional derivatives and their relationships can be found in, for example, [6, 77].

Definition 2.2 (**Directional differentiability**). *[6, Definition 2.44]* We say that $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$ is directionally differentiable at a point $x \in \mathbb{R}^n$ in a direction $h \in \mathbb{R}^n$ if the limit

$$\mathcal{F}'(x, h) := \lim_{t\downarrow 0} \frac{\mathcal{F}(x + th) - \mathcal{F}(x)}{t}$$

exists. If $\mathcal{F}$ is differentiable at x in every direction $h \in \mathbb{R}^n$, we say that $\mathcal{F}$ is directionally differentiable at x.

One, often used, concept of differentiability is differentiability in the sense of Fréchet.

Definition 2.3 (**Fréchet differentiability**). *[6, Definition 2.48]* We say that $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$ is directionally differentiable at a point $x \in \mathbb{R}^n$ in the Fréchet sense if $\mathcal{F}$ is directionally differentiable at x and

$$\mathcal{F}(x+h) = \mathcal{F}(x) + \mathcal{F}'(x,h) + o(\|h\|),\ h \in \mathbb{R}^n.$$

If, in addition, $\mathcal{F}'(x,\cdot)$ is linear and continuous, it is said that $\mathcal{F}$ is Fréchet differentiable at x.

The following theorem will lead to the definition of the generalized Jacobian in the sense of Clarke.

Theorem 2.6 (**Rademacher's theorem**). *Suppose that $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$ is locally Lipschitz continuous on an open set $\mathcal{O} \subseteq \mathbb{R}^n$. Then $\mathcal{F}$ is almost everywhere (Fréchet) differentiable in $\mathcal{O}$.*

Let $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$ be a locally Lipschitz continuous function. Then by Rademacher's theorem, $\mathcal{F}$ is (Fréchet) differentiable almost everywhere. Let $D_{\mathcal{F}}$ be the set of points in $\mathbb{R}^n$ where $\mathcal{F}$ is differentiable and $\mathcal{F}'(x)$ be the Jacobian of $\mathcal{F}$ at $x \in D_{\mathcal{F}}$. The Bouligand subdifferential (B-subdifferential) of $\mathcal{F}$ at any $x \in \mathbb{R}^n$ is defined as

$$\partial_B \mathcal{F}(x) = \left\{ \lim_{x^k \to x} \mathcal{F}'(x^k) \,|\, x^k \in D_{\mathcal{F}} \right\},$$

and the Clarke generalized Jacobian (Hessian) of $\mathcal{F}$ at $x \in \mathbb{R}^n$ is defined as

$$\partial \mathcal{F}(x) = \operatorname{conv}\{\partial_B \mathcal{F}(x)\}.$$

From [14], we have the following proposition about the properties of the B-subdifferential and the Clarke generalized Jacobian.

Proposition 2.7. *Let $\mathcal{O} \subseteq \mathbb{R}^n$ be an open set and $\mathcal{F} : \mathcal{O} \to \mathbb{R}^m$ be a locally Lipschitz continuous function. Then the following properties hold:*

(i) *$\partial_B \mathcal{F}(x)$ is a nonempty compact subset of $\mathbb{R}^{m\times n}$, for any $x \in \mathcal{O}$.*

(ii) *$\partial_B \mathcal{F}(x)$ is upper semicontinuous at $x \in \mathcal{O}$, i.e., for any $\varepsilon > 0$ there exists $\delta > 0$ such that*

$$\partial_B \mathcal{F}(y) \subseteq \partial_B \mathcal{F}(x) + \varepsilon \mathcal{B}_{m\times n}, \ \forall \, y \ satisfying \ \|y - x\| < \delta,$$

where $\mathcal{B}_{m\times n} \subseteq \mathbb{R}^{m\times n}$ is the open unit ball centered at the origin.

The properties above are also true for $\partial \mathcal{F}(\cdot)$.

The following definitions of semismoothness and "semismoothness with respect to a multifunction", which are mainly adopted from [48, 55, 65, 71], will play an important role in the semismooth Newton methods.

Definition 2.4. Let $\mathcal{O} \subseteq \mathbb{R}^n$ be an open set and $\mathcal{F} : \mathcal{O} \to \mathbb{R}^m$ be a locally Lipschitz continuous function. $\mathcal{F}$ is said to be G-semismooth at $x \in \mathcal{O}$ if for any $V \in \partial \mathcal{F}(x + \Delta x)$ with $\Delta x \to 0$,

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

$\mathcal{F}$ is said to be strongly G-semismooth at $x \in \mathcal{O}$ if for any $V \in \partial \mathcal{F}(x + \Delta x)$ with $\Delta x \to 0$,

$$\mathcal{F}(x + \Delta x) - \mathcal{F}(x) - V\Delta x = o(\|\Delta x\|^2).$$

If, in addition, $\mathcal{F}$ is directionally differentiable at x, then it is said that $\mathcal{F}$ is semismooth and strongly semismooth at x respectively.

Definition 2.5. Let $\mathcal{O} \subseteq \mathbb{R}^n$ be an open set, $\mathcal{K} : \mathcal{O} \rightrightarrows \mathbb{R}^{m\times n}$ be a nonempty compact valued, upper semicontinuous multifunction, 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 α be a positive constant. $\mathcal{F}$ is said to be α-order (strongly, if $\alpha = 1$) semismooth at $x \in \mathcal{O}$ with respect to $\mathcal{K}$ if $\mathcal{F}$ is directionally differentiable at x and for any $V \in \mathcal{K}(x+\Delta x)$ with $\Delta x \to 0$,

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

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

We usually regard Definition 2.4 as the classic and standard definition of semismoothness, whereas Definition 2.5 is more general as it involves a multifunction which could be but not limited to the Clarke generalized Jacobian. By definition, if $\mathcal{F}$ is semismooth (strongly semismooth) at x under Definition 2.4, then $\mathcal{F}$ is semismooth (strongly semismooth) at x with respect to the multifunction $\partial\mathcal{F}$ under Definition 2.5. We will see in the next lemma that the proximal mappings of the ℓ_1 norm and ℓ_2 norm are strongly semismooth, which is an important result for solving the SGLasso problem later.

Lemma 2.8. *For any $c > 0$, the proximal mappings $\mathrm{Prox}_{c\|\cdot\|_1}(\cdot)$ and $\mathrm{Prox}_{c\|\cdot\|}(\cdot)$ are strongly semismooth.*

Proof. Since $\mathrm{Prox}_{c\|\cdot\|_1}(\cdot)$ is a Lipschitz continuous piecewise affine function, it follows from [25, 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 [13, Proposition 4.3], we can obtain the conclusion directly from [64, Theorem 4]. $\square$

Next, the semismooth Newton (SSN) method is briefly introduced. Generally, to solve

$$\mathcal{F}(x) = 0,$$

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

$$x^{k+1} = x^k - V_k^{-1}\mathcal{F}(x^k),\ V_k \in \partial\mathcal{F}(x^k),\ k = 0, 1, \ldots.$$

The SSN method is extremely appealing and has been extensively investigated in the literature (e.g., [48, 49, 71, 80, 98]). We find that most of the existing studies (e.g., [71, 80]) used the Clarke generalized Jacobian $V_k \in \partial\mathcal{F}(x^k)$ in the updating scheme and established correspondingly the convergence results of the SSN method. However, using elements of the Clarke generalized Jacobian $\partial\mathcal{F}(\cdot)$ to define the method is not without drawbacks. An important point to consider is that the calculation of the Clarke generalized Jacobian is too difficult a task to accomplish in some cases. To address this burden, some researchers considered to find cheaper substitutes for the Clarke generalized Jacobian. Roughly speaking, the SSN method and its convergence analysis in the literature mainly relied on two properties of the Clarke generalized Jacobian of a semismooth function: (i) it is a nonempty compact valued, upper semicontinuous multifunction; (ii) the linear approximation equation holds: for any $V \in \partial\mathcal{F}(x + \Delta x)$ with $\Delta x \to 0$, $\mathcal{F}(x + \Delta x) - \mathcal{F}(x) - V\Delta x = o(\|\Delta x\|)$. In view of the above observations, one might consider some surrogate generalized Jacobians possessing the two special properties of the Clarke generalized Jacobian. For example, based on the two properties, the concept of "linear Newton approximations" was proposed in [25]. This concept is claimed in [25] to be more general in terms of three aspects: (i) the Clarke generalized Jacobian of a semismooth function defines a linear Newton approximation; (ii) semismooth functions can have linear Newton approximations other than the Clarke generalized Jacobian; (iii) functions that are not semismooth can admit linear Newton approximations also. Apart from the concept of "linear Newton approximations", the paper [55] proposed to use such a multifunction in Definition 2.5, with respect to which $\mathcal{F}$ is semismooth, in place of the Clarke generalized Jacobian. Besides, the superlinear convergence of the SSN method was established in [55] where a surrogate multifunction is used in place of the Clarke generalized Jacobian. Owing to the well established convergence results of the SSN methods and a wide range of choices of a multifunction in place of the Clarke generalized Jacobian, the thesis aims to construct qualified multifunctions as surrogate Clarke generalized Jacobians and then implement the SSN method for solving nonsmooth equations.

Since the SSN method is applied for solving subproblems in our algorithms, and these subproblems are minimizations of SC^1 functions, we consider the SSN method for solving the following optimization problem

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex LC^1 function. An LC^1 function is a continuously differentiable function whose gradient is locally Lipschitz continuous. Moreover, an LC^1 function is said to be SC^1 if its gradient is semismooth. Or, equivalently, the nonsmooth equation is considered

$$\nabla f(x) = 0. \tag{2.3}$$

In fact, there exist a number of works investigating the SSN method for minimizing SC^1 problems, such as [24, 69]. However, they used the B-subdifferential or the Clarke generalized Jacobian in the SSN method. In some cases, for instance, the problems that this thesis aims to solve, the B-subdifferential or the Clarke generalized Jacobian is not easy to find. As a result, the existing SSN method and theoretical results can not be applied directly to our problems. Instead, an extended SSN method and its convergence results could be used. For the ease of reading, we summarize the framework of the SSN method and its superlinear convergence here.

Algorithm 1 A semismooth Newton method for solving (2.3)

```
Given $\mu \in (0, 1/2)$, $\bar{\eta} \in (0, 1)$, $\tau \in (0, 1]$, and $\delta \in (0, 1)$. Let $\mathcal{K}$ be a nonempty compact valued, upper semicontinuous multifunction, with respect to which $\nabla f$ is semismooth. Choose $x^0 \in \mathbb{R}^n$. Iterate the following steps for $j = 0, 1, \ldots$

Step 1. (Newton direction) Choose $V_j \in \mathcal{K}(x^j)$. Solve the following linear system
        $V_j d = -\nabla f(x^j)$        (2.4)
    by a direct method or by the conjugate gradient (CG) algorithm to find $d^j$ such that $\|V_j d^j + \nabla f(x^j)\| \leq \min(\bar{\eta}, \|\nabla f(x^j)\|^{1+\tau})$.

Step 2. (Line search) Set $\alpha_j = \delta^{m_j}$, where $m_j$ is the smallest nonnegative integer $m$ for which
        $f(y^j + \delta^m d^j) \leq f(y^j) + \mu \delta^m \langle \nabla f(y^j), d^j \rangle$.

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

The convergence result for the above SSN method can be obtained from [55, 98].

Theorem 2.9. *Suppose that the equation* (2.3) *admits an unique solution* $\bar{x}$, $\mathcal{K}$ *is a nonempty compact valued, upper semicontinuous multifunction, with respect to which* ∇f *is semismooth, and every* $V \in \mathcal{K}(\bar{x})$ *is nonsingular. Let* $\{x^j\}$ *be the infinite sequence generated by Algorithm 1. Then* $\{x^j\}$ *converges to the unique solution* $\bar{x}$ *of equation* (2.3). *Moreover, the convergence rate is at least superlinear:*

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

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

Chapter 3

Augmented Lagrangian method for solving sparse group Lasso problems

This chapter addresses the issue of solving the SGLasso problem (1.3). The SGLasso problem (1.3) 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 (see [7, 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

$$\mathcal{A}x - y - b = 0,\ \mathrm{Prox}_p(x+z) - x = 0,\ \mathcal{A}^*y + z = 0, \tag{3.1}$$

Next, we analyse the vital decomposition property, which is termed as "prox-decomposition" in [90], 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 $\operatorname{Prox}_{\varphi}(\cdot)$ and $\operatorname{Prox}_{\phi}(\cdot)$. With this decomposition property, we are able to compute $\operatorname{Prox}_{p}(\cdot)$ in a closed form. This decomposition result was proven in [91, Theorem 1], which is mainly an extension of that for the fused Lasso regularizer in [29]. Here, we give another short proof based on the systematic investigation in [90].

Proposition 3.1. *Under Assumption 1, it holds that*

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

Proof. Under Assumption 1, the function p has a separable structure. Hence, the optimization problem in the definition of Prox_p is separable for each group. Therefore, it is sufficient to prove that

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

By [90, Theorem 1], for each $l \in \{1, 2, \ldots, g\}$, it suffices to show that

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

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 $\operatorname{sign}(v_l) = \operatorname{sign}(u_l)$. 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 $\operatorname{Prox}_p(u)$ explicitly. Let $v := \operatorname{Prox}_{\varphi}(u)$. 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

$$\mathrm{Prox}_p(u) = \mathrm{Prox}_\phi(v) = v - \mathcal{P}^* \Pi_{\mathcal{B}_2}(\mathcal{P} v). \tag{3.2}$$

In the paragraph that follows, we introduce some error bound results which will be used later in the convergence rate analysis. Define the proximal residual function $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}^n$ by

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

Since $\mathrm{ri}\{\mathrm{dom}\, f\} \cap \mathrm{ri}\{\mathrm{dom}\, p\} \neq \emptyset$, we know from [74, Theorem 23.8] that the Clarke generalized Jacobian $\partial h : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ takes the following form:

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

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 was proven in [96, Theorem 1] that the local error bound condition (in the sense of Luo and Tseng [62]) holds around the optimal solution set Ω_P, i.e., for every $\xi \geq \inf_x h(x)$, there exist positive scalars κ_0 and δ_0 such that

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

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

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

Furthermore, by mimicking the proofs in [21, Theorem 3.1] or [15, Proposition 2.4] and noting that Ω_P is a compact set, we can obtain the following result with no difficulty.

Proposition 3.2. *For any $r > 0$, there exists $\kappa > 0$ such that*

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

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

This section shall analyse the generalized Jacobian of the proximal mapping $\mathrm{Prox}_p(\cdot)$ of the SGLasso regularizer p. From Proposition 3.1, 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 [79] 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 [79, Lemma 2.1] hold, then we could have obtained by the chain rule the following B-subdifferential, 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 [79, 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 nontrivial to obtain, and we have to find an alternative surrogate to bypass this difficulty. The challenge just highlighted also appeared in [55] 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" (Definiton 2.5) 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 (3.2) 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 $u \in \mathbb{R}^n$:

$$\mathcal{M}(u) := \left\{ (I - \mathcal{P}^*\Sigma\mathcal{P})\Theta \;\middle|\; \begin{array}{l} \Sigma = \mathrm{Diag}(\Sigma_1, \ldots, \Sigma_g), \Sigma_l \in \partial\Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_{G_l}), l = 1, 2, \ldots, g, \\ v = \mathrm{Prox}_\varphi(u),\ \Theta \in \partial\mathrm{Prox}_\varphi(u) \end{array} \right\}. \tag{3.3}$$

Remark 3.3. 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:

$$\Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_l) = \begin{cases} \lambda_{2,l}\frac{v_l}{\|v_l\|}, & \text{if } \|v_l\| > \lambda_{2,l}, \\ v_l, & \text{otherwise}, \end{cases} \tag{3.4}$$

$$\partial\Pi_{\mathcal{B}_2^{\lambda_{2,l}}}(v_l) = \begin{cases} \{\frac{\lambda_{2,l}}{\|v_l\|}(I - \frac{v_l v_l^T}{\|v_l\|^2})\}, & \text{if } \|v_l\| > \lambda_{2,l}, \\ \{I - t\frac{v_l v_l^T}{(\lambda_{2,l})^2} \,|\, 0 \le t \le 1\}, & \text{if } \|v_l\| = \lambda_{2,l}, \\ \{I\}, & \text{if } \|v_l\| < \lambda_{2,l}. \end{cases} \tag{3.5}$$

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

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

In addition, the Clarke generalized Jacobian of Prox_φ are given as follows:

$$\partial\mathrm{Prox}_\varphi(u) = \left\{ \mathrm{Diag}(\theta) \;\middle|\; \theta \in \mathbb{R}^n, \theta_i \in \begin{cases} \{1\}, & \text{if } |u_i| > \lambda_1, \\ \{t \,|\, 0 \le t \le 1\}, & \text{if } |u_i| = \lambda_1, \\ \{0\}, & \text{if } |u_i| < \lambda_1, \end{cases} i = 1, \ldots, n \right\}. \tag{3.6}$$

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

$$\theta_i = \begin{cases} 0, & \text{if } |u_i| \le \lambda_1, \\ 1, & \text{otherwise}, i = 1, \ldots, n. \end{cases} \tag{3.7}$$

We also construct one element for numerical implementations:

$$\Theta = \mathrm{Diag}(\theta) \in \partial \mathrm{Prox}_{\varphi}(u). \tag{3.8}$$

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 (3.3) 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 (3.3).

Theorem 3.4. *Assume that Assumption 1 holds. Let $u \in \mathbb{R}^n$. Then the multifunction $\mathcal{M}$, defined in* (3.3)*, 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 $w \to u$,*

$$\mathrm{Prox}_p(w) - \mathrm{Prox}_p(u) - M(w-u) = O(\|w-u\|^2). \tag{3.9}$$

Proof. By Lemma 2.8, Proposition 3.1, and [25, 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 (3.9) 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 (3.5) and (3.6), 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

$$\Xi_l := \{i \in G_l \,|\, \theta_i = 1\},\ l = 1, 2, \ldots, g. \tag{3.10}$$

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

Case 1: $\|v_l\| < \lambda_{2,l}$. By (3.5), $I - \Sigma_l = 0$.

Case 2: $\|v_l\| = \lambda_{2,l}$. By (3.5), 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 $\text{supp}(\mathcal{P}_l^* v_l) \subseteq \Xi_l$. It follows from (3.10) 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 (3.5) 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. □

3.2 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 (1.3). Compared with the previous work [54], this work adopts the same algorithmic framework of ALM and SSN. 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 [54] 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 consist of two parts, since the SGLasso 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 [54].

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

$$\begin{aligned} \min \quad & \langle b, y\rangle + \tfrac{1}{2}\|y\|^2 + p^*(z) \\ \text{s.t.} \quad & \mathcal{A}^* y + z = 0. \end{aligned} \tag{3.11}$$

For $\sigma > 0$, the augmented Lagrangian function associated with (3.11) is given by

$$\mathcal{L}_\sigma(y, z; x) = \langle b, y\rangle + \frac{1}{2}\|y\|^2 + p^*(z) + \frac{\sigma}{2}\|\mathcal{A}^* y + z - \sigma^{-1}x\|^2 - \frac{1}{2\sigma}\|x\|^2. \tag{3.12}$$

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:

$$\min_{y,z}\{\mathcal{L}_{\sigma_k}(y, z; x^k)\}. \tag{3.13}$$

Since for any given $x^k \in \mathbb{R}^n$ and $\sigma_k > 0$, $\mathcal{L}_{\sigma_k}(\cdot, \cdot; x^k)$ is a strongly convex function, the subproblem (3.13) admits a unique optimal solution. For any $y \in \mathbb{R}^m$, define

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

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:

$$y^{k+1} \approx \arg\min_y \psi_k(y) \quad \text{and} \quad z^{k+1} = \mathrm{Prox}_{p^*/\sigma_k}(\sigma_k^{-1}x^k - \mathcal{A}^* y^{k+1}). \tag{3.14}$$

Now, we propose an inexact augmented Lagrangian method for solving (3.11).

Algorithm 2 An inexact augmented Lagrangian method for solving (3.11)

```
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 the following via (3.14):
    $$(y^{k+1}, z^{k+1}) \approx \arg\min_{y,z}\{\mathcal{L}_{\sigma_k}(y, z; x^k)\}. \tag{3.15}$$

Step 2. Compute
    $$x^{k+1} = x^k - \sigma_k(\mathcal{A}^*y^{k+1} + z^{k+1}) \;=\; \sigma_k \mathrm{Prox}_p(\sigma_k^{-1}x^k - \mathcal{A}^*y^{k+1}). \tag{3.16}$$

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 (3.15) via the standard stopping criteria studied in [75]:

(A1) $\mathcal{L}_{\sigma_k}(y^{k+1}, z^{k+1}; x^k) - \inf_{y,z} \mathcal{L}_{\sigma_k}(y, z; x^k) \;\leq\; \varepsilon_k^2/2\sigma_k,$

(B1) $\mathcal{L}_{\sigma_k}(y^{k+1}, z^{k+1}; x^k) - \inf_{y,z} \mathcal{L}_{\sigma_k}(y, z; x^k) \;\leq\; (\delta_k^2/2\sigma_k)\|x^{k+1} - x^k\|^2.$

Since $\psi_k(\cdot)$ 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 (A1) and (B1) can be replaced by the following easy-to-check criteria, respectively,

(A1′) $\|\nabla\psi_k(y^{k+1})\| \leq \varepsilon_k/\sqrt{\sigma_k},$

(B1′) $\|\nabla\psi_k(y^{k+1})\| \leq (\delta_k/\sqrt{\sigma_k})\|x^{k+1} - x^k\|.$

3.2.1 Convergence rates for ALM (Algorithm 2)

Note that the superlinear convergence of the primal and dual sequences generated by the semismooth Newton ALM for solving the Lasso and fused Lasso problems ([54, 55]) heavily relies on the polyhedral properties of the Lasso and fused Lasso regularizers. However, the SGLasso regularizer is non-polyhedral. Therefore, one has to modify the convergence analysis in ([54, 55]) to obtain the fast linear convergence property under a suitable error bound assumption. In the following, we shall analyse the global linear convergence at an arbitrarily fast rate of the inexact augmented Lagrangian method for solving problem (3.11).

For the nonnegative summable sequence $\{\varepsilon_k\}$ in the stopping criterion $(A1')$, we introduce a scalar α such that

$$\sum_{k=0}^{\infty} \varepsilon_k \leq \alpha. \tag{3.17}$$

Let r be any given positive scalar satisfying $r > \alpha$. It follows from Proposition 3.2 that there exists a positive scalar κ such that

$$\operatorname{dist}(x, \Omega_P) \leq \kappa \operatorname{dist}(0, \partial h(x)), \ \forall\, x \in \mathbb{R}^n \textit{ satisfying } \operatorname{dist}(x, \Omega_P) \leq r. \tag{3.18}$$

The next lemma measures the distance of each primal iterate generated by Algorithm 2 to the optimal solution set Ω_P. The proof of Lemma 3.5 is mainly based on [16, Proposition 1(c)], which itself is an extension of [76, Theorem 2] and [63, Theorem 2.1]. Compared to the proof in [16, Proposition 1(c)], the following lemma uses (3.18) instead of the calmness condition of $(\partial h)^{-1}$ at the origin for some $\bar{x} \in \Omega_P$.

Lemma 3.5. *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 in* (3.17)*. Let $\{x^k\}$ be any infinite sequence generated by Algorithm 2 under criteria $(A1')$ and $(B1')$ 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 (3.18).

Proof. Denote the proximal point mapping by $P_k := (\mathcal{I} + \sigma_k \partial h)^{-1}$. Then, it follows from [75, Proposition 6] and criterion $(A1')$ that

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

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

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

Therefore, one has

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

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

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

which implies that

$$\operatorname{dist}(P_k(x^k), \Omega_P) \le r, \;\forall\, k \ge 0.$$

Additionally, it was shown in [76, Proposition 1(a)] that

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

Then it follows from (3.18) that

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

Therefore, from the proof in [16, Proposition 1 (c)], for all $k \ge 0$, we obtain that

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

and that

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

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

While the global convergence of Algorihm 2 follows from [63, 75] directly, the conditions required in [63, 75] 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 [16] 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 3.6. *Let $\{(y^k, z^k, x^k)\}$ be an infinite sequence generated by Algorithm 2 under stopping criterion $(A1')$. 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 $(B1')$ is also executed in Algorithm 2 and the initial point $x^0 \in \mathbb{R}^n$ satisfies $\mathrm{dist}(x^0,\, \Omega_P) \leq r - \alpha$, then for all $k \geq 0$, we have

$$\mathrm{dist}(x^{k+1}, \Omega_P) \leq \mu_k \, \mathrm{dist}(x^k, \Omega_P), \tag{3.19a}$$

$$\|\mathcal{A}^* y^{k+1} + z^{k+1}\| \leq \mu'_k \, \mathrm{dist}(x^k, \Omega_P), \tag{3.19b}$$

$$\sup(D) - g(y^{k+1}, z^{k+1}) \leq \mu''_k \, \mathrm{dist}(x^k, \Omega_P), \tag{3.19c}$$

where

$$\mu_k := \left[\delta_k + (1+\delta_k)\kappa/\sqrt{\kappa^2 + \sigma_k^2}\right]/(1-\delta_k),$$

$$\mu'_k := 1/[(1-\delta_k)\sigma_k],$$

$$\mu''_k := [\delta_k^2\|x^{k+1} - x^k\| + \|x^{k+1}\| + \|x^k\|]/[2(1-\delta_k)\sigma_k],$$

and κ is from (3.18). 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 [75, Theorem 5] or [16, Proposition 2]. Inequality (3.19a) is a direct consequence of Lemma 3.5. From the updating formula (3.16) 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 [16, Lemma 3], i.e.

$$\|x^{k+1} - x^k\| \leq (1-\delta_k)^{-1}\text{dist}(x^k, \Omega_p), \tag{3.20}$$

implies that (3.19b) holds. Finally, it follows from [16, Proposition 2 (5b)] that

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

This, together with criterion $(B1)$ and (3.20), shows that (3.19c) holds. The proof of this theorem is completed. □

Remark 3.7. Assume that all the conditions in Theorem 3.6 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 $\{x \in \mathbb{R}^n \mid \text{dist}(x, \Omega_P) \leq r\}$ with modulus L. Therefore, one can obtain from Theorem 3.6 that for all $k \geq 0$,

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

This inequality, together with (3.19c) and the strong duality theorem, implies that

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

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 $\sigma_k \uparrow \sigma_\infty = +\infty$.

3.2.2 Semismooth Newton method for solving subproblem (3.14)

In this subsection, we propose an efficient semismooth Newton (SSN) method for solving the subproblem (3.14). As already mentioned earlier, having an efficient method for solving (3.14) is critical to the efficiency of Algorithm 2. In each iteration, we have to solve the following problem, for any given $\sigma > 0$ and fixed $\tilde{x}$,

$$\min_y \psi(y), \tag{3.21}$$

where $\psi(y) := \langle b, y\rangle + \frac{1}{2}\|y\|^2 + p^*(\text{Prox}_{p^*/\sigma}(\sigma^{-1}\tilde{x} - \mathcal{A}^*y)) + \frac{\sigma}{2}\|\text{Prox}_p(\sigma^{-1}\tilde{x} - \mathcal{A}^*y)\|^2$. Note that $\psi(\cdot)$ is strongly convex and continuously differentiable with

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

Thus, the unique solution $\bar{y}$ of (3.21) can be obtained by solving the following nonsmooth equation

$$\nabla\psi(y) = 0. \tag{3.22}$$

We should point out again that characterizing $\partial(\nabla\psi)(\cdot)$ is a difficult task to accomplish. In section 3.1, we have constructed a multifunction $\mathcal{M}$, which is used as a surrogate of the generalized Jacobian ∂Prox_p. Besides, it is illustrated in Theorem 3.4 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 (3.3). It follows from Theorem 3.4 and [25, 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 (3.22).

Algorithm 3 A semismooth Newton method for solving (3.22)

```
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. (Newton direction) Choose one $M_j \in \mathcal{M}(\sigma^{-1}\tilde{x} - \mathcal{A}^* y^j)$. Let $V_j = I + \sigma \mathcal{A} M_j \mathcal{A}^*$. Solve the following linear system
        $V_j d = -\nabla \psi(y^j)$        (3.23)
    by a direct method or by the conjugate gradient (CG) algorithm to find $d^j$
    such that $\|V_j d^j + \nabla \psi(y^j)\| \leq \min(\bar{\eta}, \|\nabla \psi(y^j)\|^{1+\tau})$.
Step 2. (Line search) Set $\alpha_j = \beta^{m_j}$, where $m_j$ is the smallest nonnegative integer
    $m$ for which
        $\psi(y^j + \beta^m d^j) \leq \psi(y^j) + \mu \beta^m \langle \nabla \psi(y^j), d^j \rangle$.
Step 3. Set $y^{j+1} = y^j + \alpha_j d^j$.
```

The following convergence theorem for Algorithm 3 can be obtained directly from [55, Theorem 3].

Theorem 3.8. *Let $\{y^j\}$ be the sequence generated by Algorithm 3. Then $\{y^j\}$ is well-defined and converges to the unique solution $\bar{y}$ of* (3.21). *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 3.

3.2.3 Efficient techniques for solving the linear system (3.23)

In this section, we analyse the sparsity structure of the matrix in the linear system (3.23) and design sophisticated numerical techniques for solving the large-scale linear systems involved in the SSN method. These techniques were first applied in [54] 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 3 is in solving the linear system (3.23). Let $(\tilde{x}, y) \in \mathbb{R}^n \times \mathbb{R}^m$ and $\sigma > 0$ be given. The linear system (3.23) has the following form:

$$(I + \sigma AMA^T)d = -\nabla\psi(y), \tag{3.24}$$

where A denotes the matrix representation of the linear operator $\mathcal{A}$, and $M \in \mathcal{M}(u)$ with $u = \sigma^{-1}\tilde{x} - A^Ty$. 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^Td 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 (3.24) 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 (3.24).

Next, we analyse the proof in Theorem 3.4 in detail in order to find the special structure of AMA^T, thereby reducing the computational cost mentioned above. Let $v := \text{Prox}_\varphi(u)$. From the proof in Theorem 3.4, 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

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

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 $\mathcal{P}_l$, (3.7), and (3.8). Therefore,

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

where Ξ_l is the index set defined by (3.10) that corresponds to the nonzero 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 $A(\mathcal{P}_l^*v_l)(\mathcal{P}_l^*v_l)^TA^T$. The following notations are introduced to express these techniques clearly. Denote the index set $\Xi_> := \{l \mid \|v_l\| > \lambda_{2,l}, l = 1, 2, \ldots, g\}$, which corresponds to case 3 in Theorem 3.4. 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

$$\begin{aligned} AMA^T &= \sum_{l\in\Xi_>} \Big(1 - \frac{\lambda_{2,l}}{\|v_l\|}\Big) A\mathcal{P}_l^*\mathcal{P}_l\Theta A^T + \frac{\lambda_{2,l}}{\|v_l\|^3} A(\mathcal{P}_l^*v_l)(\mathcal{P}_l^*v_l)^TA^T \\ &= \sum_{l\in\Xi_>} \Big(1 - \frac{\lambda_{2,l}}{\|v_l\|}\Big) A_lA_l^T + \frac{\lambda_{2,l}}{\|v_l\|^3}(A_ls_l)(A_ls_l)^T. \end{aligned} \tag{3.25}$$

Therefore, the cost of computing AMA^T and that of the matrix-vector product AMA^Td for any $d \in \mathbb{R}^m$ are $O(m^2(r + r_2))$ and $O(m(r + r_2))$, respectively, where $r := \sum_{l\in\Xi_>} |\Xi_l| \leq |\mathrm{supp}(v)|$ and $r_2 := |\Xi_>| \leq g$. 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 (3.24) 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 (3.24) 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 (3.23) 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 (3.24) efficiently through exploiting the fast computation of the matrix-vector product $AMA^T d$ for any given vector d.

3.3 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*[57] 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 [58]. ADMM was first proposed in [32, 34], and the implementation will be illustrated in section 3.3.1. In addition, we also compare with the block coordinate descent (BCD) algorithm when testing on the climate data set in section 3.3.5. The BCD method we used is efficiently implemented in [68] with a gap safe screening rule, and a PYTHON implementation is available as gl_path.py†. Therefore, we can test the performance of the BCD algorithm by running the PYTHON codes sgl_path.py. For a fair comparison, we directly run the PYTHON codes instead of translating them into MATLAB codes.

Since the primal problem (1.3) is unconstrained, it is reasonable to measure the accuracy of an approximate optimal solution (y, z, x) for problem (3.11) and problem (1.3) by the relative duality gap and dual infeasibility. Specifically, let

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

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

$$\eta_D < \varepsilon_D \quad \text{and} \quad \eta_G < \varepsilon_G, \tag{3.26}$$

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

*http://www.public.asu.edu/~jye02/Software/SLEP

†The source codes can be found in https://github.com/EugeneNdiaye/GAPSAFE_SGL

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. The BCD is terminated by its default stopping condition.

In our numerical experiments, we choose $\varepsilon_D = \varepsilon_G = 10^{-6}$ unless otherwise specified. That is, the condition (3.26) 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 (1.3). In the following tables, "S" stands for SSNAL; "P" for SLEP; "A" for sPADMM; "nnz" denotes the number of nonzero 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 2) and the total number of inner SSN iterations (in Algorithm 3) 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 code spl_path.py is implemented in Anaconda 2.

3.3.1 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 [32, 34]. This method is one of the most natural methods for solving (3.11) due to its separable structure. Generally, the framework of the sPADMM consists of the following iterations:

$$\begin{cases} y^{k+1} \approx \arg\min_y \mathcal{L}_\sigma(y, z^k; x^k) + \frac{1}{2}\|y - y^k\|_{\mathbb{S}_1}^2, \\ z^{k+1} \approx \arg\min_z \mathcal{L}_\sigma(y^{k+1}, z; x^k) + \frac{1}{2}\|z - z^k\|_{\mathbb{S}_2}^2, \\ x^{k+1} = x^k - \tau\sigma(\mathcal{A}^* y^{k+1} + z^{k+1}), \end{cases} \tag{3.27}$$

where $\tau \in (0, (1+\sqrt{5})/2)$, $\mathbb{S}_1$ and $\mathbb{S}_2$ are self-adjoint positive semidefinite linear operators, and $\mathcal{L}_\sigma$ is the augmented Lagrangian function defined in (3.12). The sPADMM is convergent under some mild conditions, and we refer the reader to [12, 28] for the convergence results. However, due to the lack of error bound conditions for the KKT system (3.1), the linear convergence rate of the sPADMM cannot be established from existing results.

In each iteration of (3.27), 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:

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

As the dimension m is a moderate number in many statistical applications. Thus, in our implementation, equation (3.3.1) was solved via the Cholesky factorization, and the proximal term $\mathbb{S}_1$ 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 [12, section 7.1] for the details on how to choose $\mathcal{S}_1$ appropriately.

The second step in (3.27) is to minimize a function of z. For the SGLasso problem, one would simply choose $\mathcal{S}_2 = 0$. In this case, z^{k+1} is updated by the following scheme:

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

where Prox_p is computable by Proposition 3.1. In summary, two subproblems of (3.27) are solvable and consequently the framework (3.27) is easily implementable. Moreover, in order to improve the convergence speed numerically, we set the step-length τ in (3.27) to be 1.618 and tune the parameter σ according to the progress between primal feasibility and dual feasibility in the implementation.

3.3.2 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 [78]. 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 nonzero elements of the underlying solution x is 100. The regularization parameters $\lambda_1 = \lambda_2$ are chosen to make the number of nonzero elements of the resulting solution close to the true number of 100.

Table 3.1 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 nonzero 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 3.2.3.

Table 3.1: 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	g	λ_1	nnz	iteration	time
(m, n)				S|A|P	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

3.3.3 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 [56] $(\mathcal{A}, b)$ that are originally obtained from the LIBSVM data sets [11]. In our numerical experiments, we follow [54] and apply the method in [42] 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 [54, 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 nonzero elements in the resulting solution x. Three values of γ are used for each UCI data set in our experiments.

Table 3.2 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 γ = 1e-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 3.3 is the same as Table 3.2 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 nonzero 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.

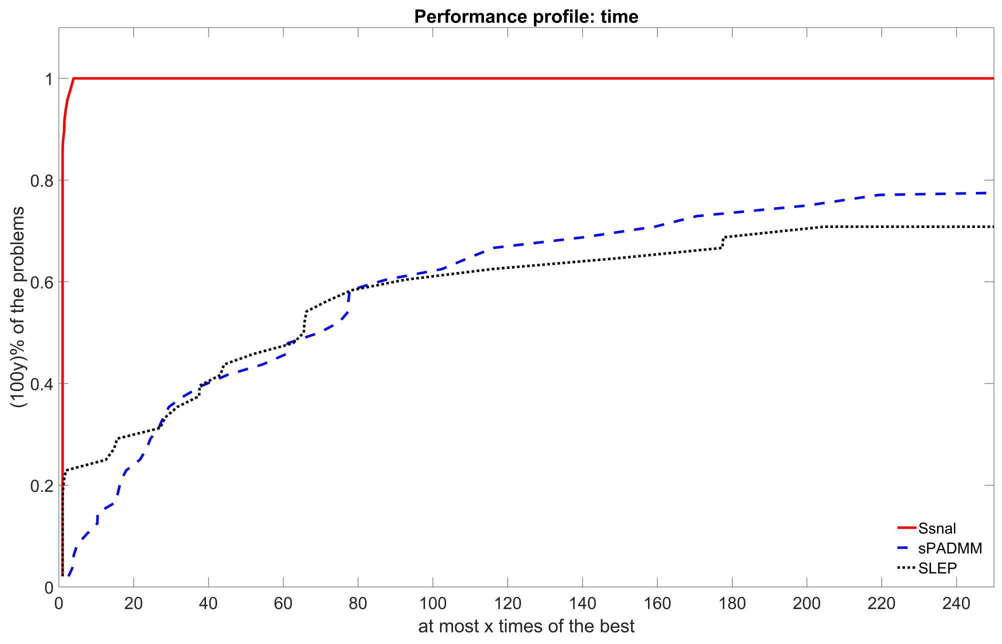


Figure 3.1: Performance profiles of SSNAL, sPADMM, and SLEP on UCI data sets with randomly generated groups.

Figure 3.1 presents the performance profiles of Ssnal, sPADMM, and SLEP for all 48 tested problems, which are presented in Table 3.2 and Table 3.3. 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 3.1 that Ssnal is around 30 times faster compared to sPADMM and SLEP for over 60% of the tested instances.

Table 3.2: 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	γ	nnz	iteration	time	error
(m, n); g			S|A|P	S|A|P	$\eta_S\|\eta_A\|\eta_P$
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 3.3: 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	γ	nnz	iteration	time	error
(m,n);g			S|A|P	S|A|P	$\eta_S\|\eta_A\|\eta_P$
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

3.3.4 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 (1.3) 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 3.4, 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

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

The comparison results with parameter sets (S1), (S2), and (S3) are presented in Table 3.4, Table 3.5, and Table 3.6, 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 3.2 presents the performance profiles of Ssnal, sPADMM, and SLEP for all 72 tested problems, which are presented in Table 3.4, Table 3.5, and Table

3.6. 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.

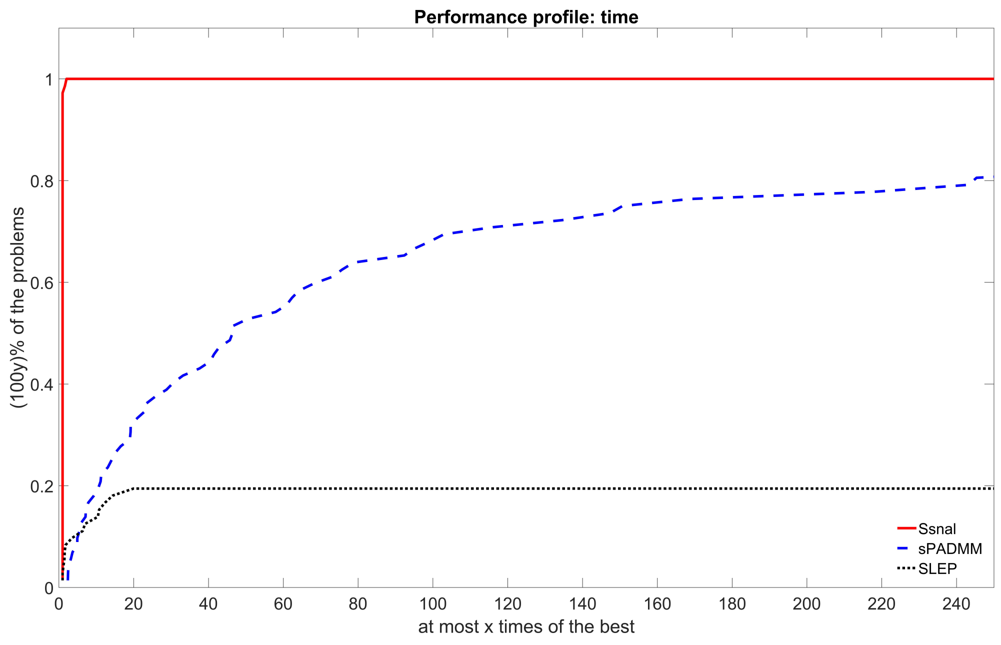


Figure 3.2: Performance profiles of Ssnal, sPADMM, and SLEP on UCI data sets with simulated groups.

Table 3.4: 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	γ	nnz	iteration	time	error
(m,n);g			S|A|P	S|A|P	$\eta_S\|\eta_A\|\eta_P$
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 3.5: 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	γ	nnz	iteration	time	error
(m,n);g			S|A|P	S|A|P	$\eta_S\|\eta_A\|\eta_P$
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 3.6: 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	γ	nnz	iteration	time	error
(m, n);g			S|A|P	S|A|P	$\eta_S\|\eta_A\|\eta_P$
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

3.3.5 NCEP/NCAR reanalysis 1 dataset

This section evaluates the performance of Ssnal, sPADMM, SLEP and BCD on the NCEP/NCAR reanalysis 1 dataset [45]. 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 [68], 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:

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

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.

The BCD used in [68] is terminated if

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

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

$$\frac{\|\mathcal{A}^* y + z\|}{1 + \|z\|} < \varepsilon,\ \mathrm{pobj} - \mathrm{dobj} < \varepsilon \|b\|^2. \tag{3.28}$$

We terminate sPADMM if (3.28) 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.,

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

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

Table 3.7 presents the comparison of Ssnal, sPADMM, SLEP, and BCD on the climate data along a solution path. As revealed by Table 3.7, 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 [68] 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 3.7: 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	time	success
ε	S|A|P|B	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

Chapter 4

Proximal point algorithm for solving multiple graphical Lasso problems

This chapter focuses on solving problem (1.6) with two particular choices of regularizer: the GGL regularizer defined by (1.7) and the FGL regularizer defined by (1.8). By introducing an auxiliary variable $\Omega = (\Omega^{(1)}, \ldots, \Omega^{(L)}) \in \mathcal{Y}$, we can rewrite problem (1.6) equivalently as

$$\begin{aligned} \min_{\Theta,\,\Omega} \quad & \sum_{l=1}^{L} \big(-\log\det\, \Omega^{(l)} + \langle S^{(l)}, \Theta^{(l)} \rangle \big) + \mathcal{P}(\Theta), \\ \text{s.t.} \quad & \Theta - \Omega = 0. \end{aligned} \tag{4.1}$$

The Lagrangian function of the above problem is given by

$$\mathcal{L}(\Theta, \Omega, X) = \sum_{l=1}^{L} \big(-\log\det\, \Omega^{(l)} + \langle S^{(l)}, \Theta^{(l)} \rangle \big) + \mathcal{P}(\Theta) - \langle \Theta - \Omega,\, X \rangle, \tag{4.2}$$

$(\Theta, \Omega, X) \in \mathcal{X} \times \mathcal{X} \times \mathcal{Y}$. It is easy to write the dual problem of (4.1), which takes the following form:

$$\max_{X} \quad \sum_{l=1}^{L} \big(\log\det\, X^{(l)} + n \big) - \mathcal{P}^*(X - S). \tag{4.3}$$

Besides, the Karush-Kuhn-Tucker (KKT) optimality conditions associate with (4.1) and (4.3) are given as follows:

$$\begin{cases} \Theta - \operatorname{Prox}_{\mathcal{P}}(\Theta + X - S) = 0, \\ \Theta - \Omega = 0, \\ \Omega^{(l)} X^{(l)} = I, \ \Omega^{(l)} \succ 0, \ X^{(l)} \succ 0, \ l = 1, 2, \ldots, L. \end{cases} \tag{4.4}$$

We make the following assumption on the KKT system throughout the thesis.

Assumption 2. *The KKT system* (4.4) *has at least one solution.*

4.1 Proximal mapping of the GGL regularizer and its generalized Jacobian

In this section, we analyse the proximal mapping of the GGL regularizer $\mathcal{P}$ defined by (1.7). For any $\Theta \in \mathcal{Y}$, one might observe that the GGL penalty term $\mathcal{P}(\Theta)$ merely penalizes the off-diagonal elements, and it is the same regularizer (the SGLasso regularizer) that acts on each vector $\Theta_{[ij]} \in \mathbb{R}^L$, $i \neq j$. More precisely, it holds that

$$\mathcal{P}(\Theta) = \sum_{i \neq j} p(\Theta_{[ij]}), \tag{4.5}$$

where

$$p(x) = \lambda_1 \|x\|_1 + \lambda_2 \|x\|, \ \forall\, x \in \mathbb{R}^L.$$

Note that the function p is actually a special SGLasso regularizer defined in (1.4) with the number of groups $g = 1$, $G_1 = \{1, 2, \ldots, n\}$, and the weight $w_1 = 1$. From this point of view, the GGL regularizer can be regarded as an extension of the SGLasso regularizer, and all entries in the same position belong to the same group.

With the representation of the GGL regularizer in (4.5), we can study its proximal mapping. By definition, the proximal mapping of $\mathcal{P}$ is given as follows: for any

$X \in \mathcal{Y}$,

$$\begin{aligned} & \mathrm{Prox}_{\mathcal{P}}(X) \\ = \ & \arg\min_{\Theta\in\mathcal{Y}} \left\{ \mathcal{P}(\Theta) + \frac{1}{2}\|\Theta - X\|^2 \right\} \\ = \ & \arg\min_{\Theta\in\mathcal{Y}} \left\{ \sum_{i\neq j} \left\{ p(\Theta_{[ij]}) + \frac{1}{2}\|\Theta_{[ij]} - X_{[ij]}\|^2 \right\} + \frac{1}{2}\sum_{i=1}^{L} \|\Theta_{[ii]} - X_{[ii]}\|^2 \right\}. \end{aligned} \tag{4.6}$$

It is obvious that the problem (4.6) is separable for each vector $\Theta_{[ij]} \in \mathbb{R}^L$. Therefore, for any $i, j \in \{1, 2, \dots, n\}$, the vector $(\mathrm{Prox}_{\mathcal{P}}(X))_{[ij]}$, consisting of all entries of $\mathrm{Prox}_{\mathcal{P}}(X)$ in the (i,j)-th position, is given explicitly by

$$(\mathrm{Prox}_{\mathcal{P}}(X))_{[ij]} = \begin{cases} \mathrm{Prox}_p(X_{[ij]}), & \textit{if} \ \ i \neq j, \\ X_{[ii]}, & \textit{if} \ \ i = j. \end{cases}$$

By this equation, one can compute $\mathrm{Prox}_{\mathcal{P}}$ via performing $n(n-1)/2$ computations of Prox_p, which can be done in parallel. The generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ contains the important second order information of the underlying problem. Here, we can characterise it by using the surrogate generalized Jacobian of Prox_p, based on the relationship between $\mathrm{Prox}_{\mathcal{P}}$ and Prox_p. Fortunately, the surrogate generalized Jacobian of Prox_p has been carefully investigated in section 3.1. Specifically, let the multifunction $\mathcal{M}_p : \mathbb{R}^L \rightrightarrows \mathbb{R}^{L\times L}$ be the surrogate generalized Jacobian of Prox_p. Directly from the formula (3.3), the multifunction $\mathcal{M}_p$ in this case can be described as follows: for any $u \in \mathbb{R}^L$,

$$\mathcal{M}_p(u) := \left\{ (I - \Sigma)\Upsilon \in \mathbb{S}^L_+ \,\middle|\, \Sigma \in \partial\Pi_{\mathcal{B}_2^{\lambda_2}}(v),\ v = \mathrm{Prox}_{\lambda_1\|\cdot\|_1}(u),\ \Upsilon \in \partial\mathrm{Prox}_{\lambda_1\|\cdot\|_1}(u) \right\},$$

where $\mathcal{B}_2^{\lambda_2} := \{v \in \mathbb{R}^L \,|\, \|v\| \leq \lambda_2\}$. Next, for any given $X \in \mathcal{Y}$, we define a multifunction $\mathcal{G}_p(X) : \mathcal{Y} \rightrightarrows \mathcal{Y}$, and show that it can be regarded as the surrogate generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ at X: for any $Y \in \mathcal{Y}$,

$$\mathcal{G}_p(X)[Y] := \left\{ Z \in \mathcal{Y} \,\middle|\, \begin{array}{l} Z_{[ij]} = M^{(ij)}Y_{[ij]},\ M^{(ij)} \in \mathcal{M}_p(X_{[ij]}) \text{ for } 1 \leq i < j \leq n; \\ Z_{[ii]} = Y_{[ii]} \text{ for } i = 1, 2, \dots, n \end{array} \right\}. \tag{4.7}$$

The generalized Jacobian is presented in the form of operators in (4.7). For the ease of analysis, we shall find its equivalent matrix form next. Actually, in view of the vectorization operator *vec*, the equivalent matrix formula of the generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ at X is given by

$$\mathbb{G}_p(X) := \left\{ M \in \mathbb{S}_+^{Ln(n+1)/2} \;\middle|\; \begin{array}{l} M = \mathrm{Diag}(I_{nL}, M^{(12)}, M^{(13)}, M^{(23)}, \ldots, M^{(1n)}, \ldots, M^{((n-1)n)}), \\ M^{(ij)} \in \mathcal{M}_p(X_{[ij]}) \text{ for } 1 \le i < j \le n \end{array} \right\}. \tag{4.8}$$

Any matrix in the set $\mathbb{G}_p(\cdot)$ is an $Ln(n+1)/2$ by $Ln(n+1)/2$ symmetric positive semidefinite matrix, and it can be regarded as a linear operator defined on the vector space $\mathbb{R}^{Ln(n+1)/2}$. Obviously, the equivalent relationship between $\mathcal{G}_p(\cdot)$ and $\mathbb{G}_p(\cdot)$ is given by the following expression:

$$\mathcal{G}_p(X) = mat \circ \mathbb{G}_p(X) \circ vec, \; \forall \; X \in \mathcal{Y}. \tag{4.9}$$

From Theorem 3.4, one can easily obtain the following theorem, which justifies why $\mathcal{G}_p(X)$ in (4.7) can be used as the surrogate generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ at X.

Theorem 4.1. *Given any $X \in \mathcal{Y}$ and the GGL regularizer $\mathcal{P}$ defined by* (1.7). *The multifunction $\mathcal{G}_p$ defined in* (4.7) *is a nonempty compact valued, upper semicontinuous multifunction, and the matrix form of $\mathcal{G}_p$ is $\mathbb{G}_p$ defined in* (4.8) *in the sense of the relationship* (4.9). *Besides, any matrix belonging to the set $\mathbb{G}_p(X)$ is symmetric and positive semidefinite. Moreover, for any $M \in \mathbb{G}_p(Y)$ with $Y \to X$, we have that*

$$\mathrm{Prox}_{\mathcal{P}}(Y) - \mathrm{Prox}_{\mathcal{P}}(X) - mat(M(vec(Y - X))) = O(\|Y - X\|^2).$$

4.2 Proximal mapping of the FGL regularizer and its generalized Jacobian

This section follows the analysis in the previous section and investigates the proximal mapping of the FGL regularizer $\mathcal{P}$ defined in (1.8). First of all, one might again observe that the FGL penalty term $\mathcal{P}(\Theta)$ penalizes the off-diagonal elements, and it is the same fused Lasso regularizer that acts on each vector $\Theta_{[ij]} \in \mathbb{R}^L$, $i \neq j$. It holds that

$$\mathcal{P}(\Theta) = \sum_{i \neq j} \gamma(\Theta_{[ij]}).$$

Here the fused Lasso regularizer is given by

$$\gamma(x) = \lambda_1 \|x\|_1 + \lambda_2 \|Bx\|_1, \, \forall\, x \in \mathbb{R}^L,$$

where the matrix $B \in \mathbb{R}^{(L-1)\times L}$ is defined by $Bx = [x_1 - x_2; x_2 - x_3; \ldots; x_{L-1} - x_L]$, $\forall\, x \in \mathbb{R}^L$. The fused Lasso regularizer was first proposed in [82] and is designed to encourage sparsity in all elements and their successive differences. The formula for the surrogate generalized Jacobian of Prox_γ has been derived in [55] and will be used in our subsequent algorithmic design. In the following context, we recall some notations and results that are used in the characterization of the surrogate generalized Jacobian of Prox_γ.

Denote the proximal mapping of $\lambda_2 \|B \cdot\|_1$ by

$$x_{\lambda_2}(v) := \arg\min_x \left\{ \lambda_2 \|Bx\|_1 + \frac{1}{2}\|x - v\|^2 \right\}, \; \forall\; v \in \mathbb{R}^L.$$

The following lemma provides the "prox-decomposition" property of Prox_γ.

Lemma 4.2. *[29, Proposition 1] Given $\lambda_1, \lambda_2 \geq 0$, it holds that*

$$\mathrm{Prox}_\gamma(v) = \mathrm{Prox}_{\lambda_1\|\cdot\|_1}(x_{\lambda_2}(v)) = \mathrm{sign}(x_{\lambda_2}(v)) \odot \max(|x_{\lambda_2}(v)| - \lambda_1, 0), \;\forall\, v \in \mathbb{R}^L.$$

The next lemma provides an alternative way of computing $x_{\lambda_2}(\cdot)$ through the dual solution $z_{\lambda_2}(\cdot)$,

Lemma 4.3. *[55, Lemma 1] Given $\lambda_2 \geq 0$, it holds that*

$$x_{\lambda_2}(v) = v - B^T z_{\lambda_2}(Bv), \;\forall\, v \in \mathbb{R}^L,$$

where $z_{\lambda_2}(u) := \arg\min_z \left\{ \frac{1}{2}\|B^T z\|^2 - \langle z,\, u\rangle \mid \|z\|_\infty \leq \lambda_2 \right\}, \;\forall\, u \in \mathbb{R}^{L-1}.$

Given $v \in \mathbb{R}^L$, consider the following sets:

$$\begin{aligned} \mathcal{I}_z(v) &:= \{ i \,|\, |(z_{\lambda_2}(Bv))_i| = \lambda_2,\ i = 1, 2, \ldots, L-1 \}, \\ \mathcal{K}_z(v) &:= \{ K \subseteq \{1, 2, \ldots, L-1\} \,|\, \mathrm{supp}(Bx_{\lambda_2}(v)) \subseteq K \subseteq \mathcal{I}_z(v) \}. \end{aligned}$$

Define the multifunction $\mathcal{Q}_z : \mathbb{R}^L \rightrightarrows \mathbb{R}^{(L-1)\times(L-1)}$ by

$$\mathcal{Q}_z(v) := \left\{ \widehat{Q} \in \mathbb{R}^{(L-1)\times(L-1)} \,|\, \widehat{Q} = (\Sigma_K B B^T \Sigma_K)^\dagger,\ K \in \mathcal{K}_z(v) \right\},$$

where $\Sigma_K = \mathrm{Diag}(\sigma_K) \in \mathbb{R}^{(L-1)\times(L-1)}$ with

$$(\sigma_K)_i = \begin{cases} 0, & \text{if } i \in K, \\ 1, & \text{otherwise}, \end{cases}$$

Define also the multifunction $\mathcal{Q}_x : \mathbb{R}^L \rightrightarrows \mathbb{R}^{L\times L}$ by

$$\mathcal{Q}_x(v) := \left\{ Q \in \mathbb{R}^{L\times L} \,|\, Q = I - B^T \widehat{Q} B,\ \widehat{Q} \in \mathcal{Q}_z(v) \right\}.$$

In [55], it has been shown that the surrogate generalized Jacobian of Prox_γ at v is the multifunction $\mathcal{M}_\gamma : \mathbb{R}^L \rightrightarrows \mathbb{R}^{L\times L}$ defined by

$$\mathcal{M}_\gamma(v) := \left\{ M \in \mathbb{S}^L_+ \,|\, M = \Upsilon Q,\ \Upsilon \in \partial_B \mathrm{Prox}_{\lambda_1\|\cdot\|_1}(x_{\lambda_2}(v)),\ Q \in \mathcal{Q}_x(v) \right\}.$$

With the above preparation, for any given $X \in \mathcal{Y}$, we can define a multifunction $\mathcal{G}_\gamma(X) : \mathcal{Y} \rightrightarrows \mathcal{Y}$, and show that it can be regarded as the surrogate generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ at X, i.e.,

$$\mathcal{G}_\gamma(X)[Y] := \left\{ Z \in \mathcal{Y} \,\middle|\, \begin{array}{l} Z_{[ij]} = M^{(ij)} Y_{[ij]},\ M^{(ij)} \in \mathcal{M}_\gamma(X_{[ij]}) \text{ for } 1 \le i < j \le n; \\ Z_{[ii]} = Y_{[ii]} \text{ for } i = 1, 2, \ldots, n \end{array} \right\}. \tag{4.10}$$

Besides, the equivalent matrix formula of the generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ at X is

$$\mathbb{G}_\gamma(X) := \left\{ M \in \mathbb{S}^{Ln(n+1)/2}_+ \,\middle|\, \begin{array}{l} M = \mathrm{Diag}(I_{nL}, M^{(12)}, M^{(13)}, M^{(23)}, \ldots, M^{(1n)}, \ldots, M^{((n-1)n)}), \\ M^{(ij)} \in \mathcal{M}_\gamma(X_{[ij]}) \text{ for } 1 \le i < j \le n \end{array} \right\}, \tag{4.11}$$

and the equivalent relationship between $\mathcal{G}_\gamma(\cdot)$ and $\mathbb{G}_\gamma(\cdot)$ is given by the expression:

$$\mathcal{G}_\gamma(X) = mat \circ \mathbb{G}_\gamma(X) \circ vec, \; \forall\, X \in \mathcal{Y}. \tag{4.12}$$

From [55, Theorem 1], one can easily obtain the following theorem, which justifies why $\mathcal{G}_\gamma(X)$ in (4.10) can be used as the surrogate generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ at X.

Theorem 4.4. *Given any $X \in \mathcal{Y}$ and the FGL regularizer $\mathcal{P}$ defined by* (1.8). *The multifunction $\mathcal{G}_\gamma$ defined in* (4.10) *is a nonempty compact valued, upper semicontinuous multifunction, and the matrix form of $\mathcal{G}_\gamma$ is $\mathbb{G}_\gamma$ defined in* (4.11) *in the sense of the relationship* (4.12). *Besides, any matrix belonging to the set $\mathbb{G}_\gamma(X)$ is symmetric and positive semidefinite. Moreover, there exists a neighborhood $\mathcal{U}_X$ of X such that for all $Y \in \mathcal{U}_X$,*

$$\mathrm{Prox}_{\mathcal{P}}(Y) - \mathrm{Prox}_{\mathcal{P}}(X) - mat(M(vec(Y - X))) = 0, \; \forall\; M \in \mathbb{G}_\gamma(Y).$$

4.3 Inexact semismooth Newton based proximal point algorithm

In this section, we present a proximal point algorithm (PPA) for solving the problem (4.1) with the GGL regularizer defined in (1.7) and the FGL regularizer defined in (1.8). The k-th iteration of the PPA is

$$(\Theta^{k+1}, \Omega^{k+1}) \approx \arg\min_{\Theta,\Omega} \left\{ \sup_X \mathcal{L}(\Theta, \Omega, X) + \frac{1}{2\sigma_k}(\|\Theta - \Theta^k\|^2 + \|\Omega - \Omega^k\|^2) \right\}, \; k \geq 0,$$

where the Lagrangian function $\mathcal{L}$ is defined in (4.2). As we can see, the key step is analysing the Moreau-Yosida envelope of the objective function or the essential objective function $\sup_X \mathcal{L}(\Theta, \Omega, X)$ of the problem (4.1). For any $\sigma > 0$, the Moreau-Yosida envelope of the essential objective function $\sup_X \mathcal{L}(\Theta, \Omega, X)$ of the problem (4.1) is defined by

$$F_\sigma(\overline{\Theta}, \overline{\Omega}) := \min_{\Theta,\Omega} \left\{ \sup_X \mathcal{L}(\Theta, \Omega, X) + \frac{1}{2\sigma}(\|\Theta - \overline{\Theta}\|^2 + \|\Omega - \overline{\Omega}\|^2) \right\}, \forall \overline{\Theta} \in \mathcal{Y}, \, \overline{\Omega} \in \mathcal{Y}.$$

From [74, Theorem 37.3], one can change the order of min and sup. As a result, $F_\sigma(\overline{\Theta}, \overline{\Omega})$ can be rewritten as

$$\begin{aligned} F_\sigma(\overline{\Theta}, \overline{\Omega}) &= \max_X \Big\{ \sum_{l=1}^L \min_{\Omega^{(l)}} \big(-\log\det \Omega^{(l)} + \langle \Omega^{(l)}, X^{(l)} \rangle + \frac{1}{2\sigma}\|\Omega^{(l)} - \overline{\Omega}^{(l)}\|^2 \big) \\ &\qquad + \min_\Theta \big(\mathcal{P}(\Theta) + \langle \Theta, S - X \rangle + \frac{1}{2\sigma}\|\Theta - \overline{\Theta}\|^2 \big) \Big\} \\ &= \max_X \Big\{ \sum_{l=1}^L \min_{\Omega^{(l)}} \big(-\log\det \Omega^{(l)} + \frac{1}{2\sigma}\|\Omega^{(l)} - W_\sigma^{(l)}\|^2 - \frac{1}{2\sigma}\|W_\sigma^{(l)}\|^2 + \frac{1}{2\sigma}\|\overline{\Omega}^{(l)}\|^2 \big) \\ &\qquad + \min_\Theta \big(\mathcal{P}(\Theta) + \frac{1}{2\sigma}\|\Theta - V_\sigma\|^2 - \frac{1}{2\sigma}\|V_\sigma\|^2 + \frac{1}{2\sigma}\|\overline{\Theta}\|^2 \big) \Big\}, \end{aligned}$$

where $W_\sigma^{(l)} = \overline{\Omega}^{(l)} - \sigma X^{(l)}$, $l = 1, 2, \ldots, L$, and $V_\sigma = \overline{\Theta} + \sigma(X - S)$. For simplicity, we express the Moreau envelope $F_\sigma(\overline{\Theta}, \overline{\Omega})$ in the following simple form:

$$F_\sigma(\overline{\Theta}, \overline{\Omega}) = \max_X \big\{ -\psi(X, \overline{\Theta}, \overline{\Omega}) \big\},$$

by defining the following functions, for $l = 1, 2, \ldots, L$,

$$\begin{aligned} h_l(X^{(l)}, \overline{\Omega}^{(l)}) &= \min_{\Omega^{(l)}} \big\{ -\log\det \Omega^{(l)} + \frac{1}{2\sigma}\|\Omega^{(l)} - W_\sigma^{(l)}\|^2 - \frac{1}{2\sigma}\|W_\sigma^{(l)}\|^2 + \frac{1}{2\sigma}\|\overline{\Omega}^{(l)}\|^2 \big\}, \\ g(X, \overline{\Theta}) &= \min_\Theta \big\{ \mathcal{P}(\Theta) + \frac{1}{2\sigma}\|\Theta - V_\sigma\|^2 - \frac{1}{2\sigma}\|V_\sigma\|^2 + \frac{1}{2\sigma}\|\overline{\Theta}\|^2 \big\}, \\ \psi(X, \overline{\Theta}, \overline{\Omega}) &= -\sum_{l=1}^L h_l(X^{(l)}, \overline{\Omega}^{(l)}) - g(X, \overline{\Theta}). \end{aligned} \tag{4.13}$$

By Proposition 2.4, the minimal values of the minimization problems involved in the functions h_l, $l = 1, 2, \ldots, L$ and g are attained at

$$\Omega^{(l)} = \phi_\sigma^+(W_\sigma^{(l)}),\ l = 1, 2, \ldots, L, \text{ and } \Theta = \mathrm{Prox}_{\sigma\mathcal{P}}(V_\sigma).$$

The PPA for solving (4.1) can be described as below.

Algorithm 4 A proximal point algorithm for solving (4.1)

```
Choose $\Theta^0 \in \mathcal{X}$, $\Omega^0 \in \mathcal{X}$. Iterate the following steps for $k = 0, 1, 2, \ldots$.

Step 1. Compute
    $$X^{k+1} \approx \arg\max_{X} \; -\psi(X, \Theta^k, \Omega^k), \qquad (4.14)$$
    where $\psi$ is defined by (4.13).

Step 2. Compute
    $$(\Omega^{(l)})^{k+1} = \phi^{+}_{\sigma_k}((\Omega^{(l)})^k - \sigma_k (X^{(l)})^{k+1}),\ l = 1, 2, \ldots, L,$$
    $$\Theta^{k+1} = \mathrm{Prox}_{\sigma_k \mathcal{P}}(\Theta^k + \sigma_k (X^{k+1} - S)).$$

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

Since in practice the inner subproblem (4.14) is usually not solved exactly, we will use the following standard stopping criteria studied in [75, 76] for estimating the accuracy of the approximate solution X^{k+1} of (4.14):

(A2) $\psi(X^{k+1}, \Theta^k, \Omega^k) - \inf_X \psi(X, \Theta^k, \Omega^k) \le \varepsilon_k^2 / 2\sigma_k$, $\varepsilon_k \ge 0$, $\sum_{k=0}^{\infty} \varepsilon_k < \infty$;

(B2) $\psi(X^{k+1}, \Theta^k, \Omega^k) - \inf_X \psi(X, \Theta^k, \Omega^k) \le (\delta_k^2 / 2\sigma_k) \|(\Theta^{k+1}, \Omega^{k+1}) - (\Theta^k, \Omega^k)\|^2$,

$\delta_k \ge 0$, $\sum_{k=0}^{\infty} \delta_k < \infty$;

Following the analysis of the convergence rate of the ALM for solving the SGLasso problem, i.e., Theorem 3.6, we can obtain a similar theorem from [16, 96, 100].

Theorem 4.5. *Let* $\{(\Theta^k, \Omega^k, X^k)\}$ *be an infinite sequence generated by Algorithm 4 under stopping criterion* (A2). *Then the sequence* $\{(\Theta^k, \Omega^k)\}$ *converges to the unique solution* (Θ^*, Ω^*) *of* (4.1)*, and the sequence* $\{X^k\}$ *converges to the unique solution of* (4.3)*. Furthermore, if the criterion* (B2) *is also executed in Algorithm 4, then there exist a positive integer* $\bar{k}$ *and a positive scalar* κ *such that for all* $k \ge \bar{k}$, *we have*

$$\|(\Theta^{k+1},\Omega^{k+1})-(\Theta^*,\Omega^*)\| \leq \mu_k\|\Theta^k,\Omega^k)-(\Theta^*,\Omega^*)\|,$$

$$\sup_X f_d(X) - f_d(X^{k+1}) \leq \mu_k'\|\Theta^k,\Omega^k)-(\Theta^*,\Omega^*)\|,$$

where $f_d(X)$ is the objective function of the dual problem (4.3)*,*

$$\mu_k := \left[\delta_k + (1+\delta_k)\kappa/\sqrt{\kappa^2+\sigma_k^2}\right]/(1-\delta_k),$$

$$\mu_k' := [\delta_k^2\|\Theta^{k+1}-\Theta^k\| + \|\Theta^{k+1}\| + \|\Theta^k\|]/[2(1-\delta_k)\sigma_k].$$

Moreover, μ_k and μ_k' go to 0 *if $\sigma_k \uparrow \sigma_\infty = +\infty$.*

4.3.1 Semismooth Newton method for solving subproblem (4.14)

In what follows, we design a semismooth Newton method (SSN) for solving the inner subproblem (4.14). Basically, we aim at solving the following problem for any given $\widetilde{\Theta} \in \mathcal{Y}$, $\widetilde{\Omega} \in \mathcal{Y}$,

$$\min_X \Psi(X) := \psi(X, \widetilde{\Theta}, \widetilde{\Omega}). \tag{4.15}$$

Since Ψ is continuously differentiable and strictly convex, the unique optimal solution of the problem (4.15) is the solution of the following linear system

$$\nabla\Psi(X) = 0. \tag{4.16}$$

Note that h_l, $l = 1,2,\ldots,L$ and g are smooth in the variable X with gradients:

$$\nabla_{X^{(l)}} h_l(X^{(l)}, \widetilde{\Omega}^{(l)}) = \phi_\sigma^+(W_\sigma^{(l)}),\ l=1,2,\ldots,L, \quad \nabla_X g(X,\widetilde{\Theta}) = -\mathrm{Prox}_{\sigma P}(V_\sigma).$$

Together with (4.13), one has

$$\nabla\Psi(X) = -\left(\phi_\sigma^+(W_\sigma^{(1)}),\ldots,\phi_\sigma^+(W_\sigma^{(L)})\right) + \mathrm{Prox}_{\sigma\mathcal{P}}(V_\sigma).$$

Recall that ϕ_σ^+ is differentiable and its Jacobian $(\phi_\sigma^+)'$ is given by Proposition 2.5. When $\mathcal{P}$ is the GGL regularizer given by (1.7), the surrogate generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ is the multifunction $\mathcal{G}_p$ defined in (4.7). When $\mathcal{P}$ is the FGL regularizer given by (1.8), the surrogate generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$ is the multifunction $\mathcal{G}_\gamma$ defined in (4.10). Therefore, the surrogate generalized Jacobian of $\nabla\Psi$ at X is the multifunction $\mathcal{V}(X) : \mathcal{Y} \rightrightarrows \mathcal{Y}$ such that for any $D \in \mathcal{Y}$,

$$\mathcal{V}(X)[D] = \sigma\left((\phi_\sigma^+)'(W_\sigma^{(1)})[D^{(1)}], \dots, (\phi_\sigma^+)'(W_\sigma^{(L)})[D^{(L)}]\right) + \sigma\mathcal{G}(V_\sigma/\sigma)[D],$$

with $\mathcal{G} = \mathcal{G}_p$ for the GGL regularizer $\mathcal{P}$ and $\mathcal{G} = \mathcal{G}_\gamma$ for the FGL regularizer $\mathcal{P}$. With the generalized Jacobian of $\nabla\Psi$, we are ready to solve the equation (4.16) by the following SSN method, where the Newton systems are solved inexactly by the conjugate gradient (CG) method.

Algorithm 5 A semismooth Newton method for solving (4.16)

Given $\mu \in (0, 1/2)$, $\bar{\eta} \in (0, 1)$, $\tau \in (0, 1]$, and $\delta \in (0, 1)$. Choose $X^0 \in \mathbb{S}^n_{++} \times \cdots \times \mathbb{S}^n_{++}$. Iterate the following steps for $j = 0, 1, \dots$.

Step 1. (Newton direction) Choose one specific map $V_j \in \mathcal{V}(X^j)$. Apply the CG method to find an approximate solution D^j to

$$V_j[D] = -\nabla\Psi(X^j)$$

such that

$$\|V_j[D^j] + \nabla\Psi(X^j)\| \le \min(\bar{\eta}, \|\nabla\Psi(X^j)\|^{1+\tau}).$$

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

$$\Psi(X^j + \delta^m D^j) \le \Psi(X^j) + \mu\delta^m \langle \nabla\Psi(X^j), D^j \rangle.$$

Step 3. Set $X^{j+1} = X^j + \alpha_j D^j$.

4.4 Numerical experiments

In this section, we compare the performance of our algorithm PPA with the alternating direction method of multipliers (ADMM) for which the implementation will be presented in section 4.4.1. The performance of PPA is also compared with the proximal Newton-type method implemented in the work [88] (referred to as MGL here) for which the solver can be downloaded from the site `http://senyang.info/`.

The following paragraph describes the measurement of the accuracy of an approximate optimal solution and the stopping criteria of the three methods. Since both PPA and ADMM can generate primal and dual approximate solutions, we can assess the accuracy of their solutions by the relative KKT residuals. Unlike the primal-dual method, MGL merely gives the primal solution and the KKT residual of a solution generated by MGL is not available. Instead, we measure the relative error of the objective value obtained by MGL with respect to that computed by P-PA. The accuracy of an approximate optimal solution (Θ, Ω, X) generated by PPA (Algorithm 4) is measured by the relative KKT residual:

$$\eta_P := \max\left\{ \frac{\|\Theta - \mathrm{Prox}_{\mathcal{P}}(\Theta + X - S)\|}{1+\|\Theta\|}, \frac{\|\Theta - \Omega\|}{1+\|\Theta\|}, \max_{1\le l\le L}\left\{ \frac{\|\Omega^{(l)}X^{(l)} - I\|}{1+\sqrt{n}} \right\} \right\}.$$

Likewise, the accuracy of an approximate optimal solution (Θ, X, Z) generated by ADMM (section (4.4.1)) is measured by the relative KKT residual:

$$\eta_A := \max\left\{ \frac{\|\Theta - \mathrm{Prox}_{\mathcal{P}}(\Theta + Z)\|}{1+\|\Theta\|}, \frac{\|X - Z - S\|}{1+\|S\|}, \max_{1\le l\le L}\left\{ \frac{\|\Theta^{(l)}X^{(l)} - I\|}{1+\sqrt{n}} \right\} \right\}.$$

In our numerical experiments, we terminate the PPA if is satisfies the condition $\eta_P < \varepsilon$ for a given accuracy tolerance ε; similarly for ADMM with the stopping condition $\eta_A < \varepsilon$. Note that the terminating condition for MGL is different. As mentioned before, MGL only generates a primal solution, and thus we evaluate its accuracy by the objective function value. Let

$$\begin{aligned} \mathrm{pobj}_P &:= \textstyle\sum_{l=1}^{L} \left(-\log\det\, \Theta^{(l)} + \langle S^{(l)}, \Theta^{(l)} \rangle \right) + \mathcal{P}(\Theta) \text{ and} \\ \mathrm{dobj}_P &:= \textstyle\sum_{l=1}^{L} \left(\log\det\, X^{(l)} + n \right) \end{aligned}$$

be the primal and dual objective function values computed by PPA. Note that we omit the term $\mathcal{P}^*(X-S)$ in the dual objective value since $\mathcal{P}^*(X-S)$ is an indicator function when $\mathcal{P}$ is a positive homogeneous function [74]. MGL will be terminated when the relative difference of its objective value with respect to the primal objective value obtained by PPA is smaller than the relative duality gap achieved by PPA or the given tolerance ε, i.e.,

$$\Delta_M := \frac{\mathrm{pobj}_M - \mathrm{pobj}_P}{1 + |\mathrm{pobj}_M| + |\mathrm{pobj}_P|} < \max\left\{\mathrm{relgap}_P, \varepsilon\right\},$$

where pobj_M is the primal objective value obtained by MGL, and $\mathrm{relgap}_P = |\mathrm{pobj}_P - \mathrm{dobj}_P|/(1 + |\mathrm{pobj}_P| + |\mathrm{dobj}_P|)$.

In our implementation of the PPA, we adopt a warm-start strategy to initialize the algorithm. That is, we first run ADMM (with identity matrices as starting point) for a fixed number of iterations to generate a good initial point to warm-start the PPA. We also stop the ADMM as soon as the relative KKT residual of the computed iterate is less than 100ε. Note that such a warm-starting strategy is sound since in the initial phase of the PPA where the iterates are not close to the optimal solution (as measured by the associated relative KKT residual), it is computationally wasteful to use the more expensive PPA iteration when the fast local linear convergence behavior of the algorithm has yet to kick in. Under such a scenario, naturally one would use cheaper iterations such as those of the ADMM to generate the approximate solution points until the relative KKT residual has been sufficiently reduced.

For the tuning parameters λ_1 and λ_2, we select three pairs of them for each instance that produce reasonable sparsity. In the following tables, "P" stands for PPA; "A" stands for ADMM; "M" stands for MGL; "nnz" denotes the number of nonzero entries in the solution Θ obtained by PPA using the following estimation: $\mathrm{nnz} := \min\{k \mid \sum_{i=1}^{k} |\hat{x}_i| \geq 0.999\|\hat{x}\|_1\}$, where $\hat{x} \in \mathbb{R}^{n^2 L}$ is the vector obtained via sorting all elements in Θ by magnitude in a descending order; "density" denotes the quantity $\mathrm{nnz}/(n^2 L)$. The time is displayed in the format of "hours:minutes:seconds", and the fastest method in terms of running time is highlighted in red. Since MGL is measured by the relative gap while PPA and ADMM are measured by the KKT residual, we present errors involving both the KKT residual and the relative duality gap. The errors of PPA and ADMM presented in the tables are the relative KKT residual or the relative duality gap, whichever is larger, i.e., $\max\{\eta_P, \mathrm{relgap}_P\}$ for PPA and $\max\{\eta_A, \mathrm{relgap}_A\}$ for ADMM; while the error of MGL is Δ_M, the relative difference of its objective value with respect to the primal objective value obtained by PPA.

4.4.1 Dual based ADMM

In this section, we briefly describe the ADMM for solving the problem (4.3), which is written equivalently as follows:

$$\min_{X,\,Z} \left\{ \sum_{l=1}^{L} \left(-\log\det X^{(l)}\right) + \mathcal{P}^*(Z) \,\middle|\, X - Z = S \right\}. \tag{4.17}$$

Due to its separable structure in terms of the variables X and Z, ADMM is always considered as a natural choice for solving (4.17). The classic ADMM was first proposed in [32, 34], and later extended in [12, 28]. We note that the paper [17] has implemented an ADMM for solving the primal problem (1.6). Here we adopt the ADMM for solving the dual problem (4.17). There are two reasons for us to implement our own ADMM. First, we failed to obtain the Matlab codes of the algorithm in [17]. Second, in our implementation, we tune the parameter σ wisely according to the progress of primal and dual feasibilities (see e.g. [50, Section 4.4]). We also use a larger step-length τ of 1.618, which has been demonstrated in various works to improve the performance of ADMM. The augmented Lagrangian function associated with (4.17), given parameter $\sigma > 0$, is defined by

$$\widehat{\mathcal{L}}_\sigma(X, Z, \Theta) = \textstyle\sum_{l=1}^{L} \left(-\log\det X^{(l)}\right) + \mathcal{P}^*(Z) + \langle X - Z - S,\, \Theta\rangle + \dfrac{\sigma}{2}\|X - Z - S\|^2.$$

The iteration scheme of ADMM for (4.17) can be described as follows: given $\tau \in (0, (1+\sqrt{5})/2)$, and initial point $(X, Z, \Theta) = (X^0, Z^0, \Theta^0)$, the $(k+1)$-th iteration is given by

$$\begin{cases} X^{k+1} = \arg\min_X \ \widehat{\mathcal{L}}_\sigma(X, Z^k, \Theta^k), \\ Z^{k+1} = \arg\min_Z \ \widehat{\mathcal{L}}_\sigma(X^{k+1}, Z, \Theta^k), \\ \Theta^{k+1} = \Theta^k + \tau\sigma(X^{k+1} - Z^{k+1} - S). \end{cases}$$

In particular, $X^{k+1} = ((X^{(1)})^{k+1}, \dots, (X^{(L)})^{k+1})$ can be updated by

$$\begin{aligned} (X^{(l)})^{k+1} &= \arg\min_{X^{(l)} \succ 0} \left\{ -\log\det X^{(l)} + \tfrac{\sigma}{2}\|X^{(l)} - (Z^{(l)})^k - S^{(l)} + \tfrac{1}{\sigma}(\Theta^{(l)})^k\|^2 \right\} \\ &= \phi^+_{\sigma^{-1}}\left((Z^{(l)})^k - \tfrac{1}{\sigma}(\Theta^{(l)})^k + S^{(l)}\right), \ l = 1, 2, \dots, L. \end{aligned}$$

And Z^{k+1} can be updated by

$$\begin{aligned} Z^{k+1} &= \arg\min_{Z \in \mathcal{Y}} \left\{ \mathcal{P}^*(Z) + \frac{\sigma}{2}\|Z + S - X^{k+1} - \frac{1}{\sigma}\Theta^k\|^2 \right\} \\ &= \mathrm{Prox}_{\mathcal{P}^*/\sigma}(X^{k+1} + \frac{1}{\sigma}\Theta^k - S) \\ &= (X^{k+1} + \frac{1}{\sigma}\Theta^k - S) - \mathrm{Prox}_{\mathcal{P}}(X^{k+1} + \frac{1}{\sigma}\Theta^k - S). \end{aligned}$$

Besides, the accuracy of an approximate solution generated by ADMM can be evaluated by the following KKT optimality conditions, which is slightly different from but essentially the same as the KKT system (4.4):

$$\Theta - \mathrm{Prox}_{\mathcal{P}}(\Theta + Z) = 0, \ X - Z - S = 0, \ \Theta^{(l)}X^{(l)} = I, \ \Theta^{(l)} \succ 0, \ X^{(l)} \succ 0, \ l = 1, 2, \dots, L.$$

Thus far, we have provided an easily implementable framework of ADMM for which each iteration requires the computation of the proximal mapping of the log-determinant function and that of the GGL or FGL regularizer $\mathcal{P}$.

4.4.2 Nearest-neighbour networks

In this section, we assess the performance of PPA in comparison with ADMM and MGL on a simulated network: nearest-neighbour network. The nearest-neighbour network is generated by modifying the data generating mechanism described in [52]. We set $n = 500$ and $L = 3$. For each $l = 1, 2, \dots, L$, we generate 10000 independently and identically distributed observations from a multivariate Gaussian distribution

$\mathcal{N}_n(0,(\Omega^{(l)})^{-1})$, where $\Omega^{(l)}$ is the precision matrix of the l-th class. The details of the generation of $\Omega^{(l)}$ are as follows. First of all, n points are randomly generated on a unit square, their pairwise distances are calculated, and m-nearest neighbours of each point in terms of distance are found. The nearest neighbour network is obtained by linking any two points that are m-nearest neighbours of each other. The integer m controls the degree of sparsity of the network, and we set $m = 5$ in our simulation. Subsequently, we add heterogeneity to the common structure by further creating individual links as follows: for each $\Omega^{(l)}$, a pair of symmetric zero elements is randomly selected and replaced with a value uniformly drawn from the interval $[-1,-0.5]\cup[0.5,1]$. This procedure is repeated $M/4$ times, where M is the number of edges in the nearest-neighbour graph. In our simulation, the true number of edges in the three networks is 3690.

There is a pair of tuning parameters λ_1 and λ_2 which must be specified. Following [17], we also reparameterize the GGL penalty parameters for the nearest neighbour networks in order to separate the regularization for “sparsity” and for “similarity”. In the FGL model, λ_1 drives sparsity and λ_2 drives similarity, and we say that λ_1 and λ_2 are the sparsity and similarity control parameters respectively. By contrast, in the GGL model, both parameters contribute to sparsity: λ_1 drives individual network edges to zero whereas λ_2 drives network edges to zero across all L network estimates at the same time. We reparameterize the tuning parameters for the GGL model in terms of

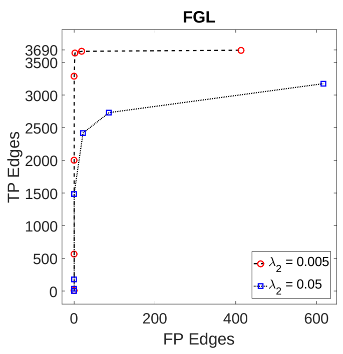
$$w_1 = \lambda_1 + \frac{1}{\sqrt{2}}\lambda_2,\ w_2 = \frac{1}{\sqrt{2}}\lambda_2/(\lambda_1 + \frac{1}{\sqrt{2}}\lambda_2),$$

which was found in [17] to reflect the levels of sparsity and similarity regularization. Likewise, we say that w_1 and w_2 are the sparsity and similarity control parameters of the GGL model, respectively. In order to show the diversity of sparsity in our experiments, we choose a series of λ_1 for the FGL model with λ_2 fixed. For the GGL model, we first fix w_2, then choose a series of w_1 for diversity. (We can obtain the corresponding (λ_1,λ_2) from (w_1,w_2).) Figure 4.1 characterizers the relative abilities of the FGL and GGL models to recover the network structures and to detect change-points.

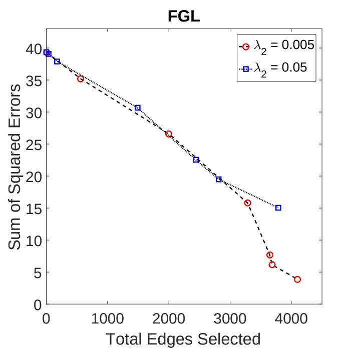
Figure 4.1a and Figure 4.1d display the number of true positive edges selected (i.e., TP edges) against the number of false edges selected (i.e., FP edges) for the FGL and GGL models respectively. We say that an edge (i, j) in the l-th network is selected in the estimate $\widehat{\Theta}^{(l)}$ if $\widehat{\Theta}^{(l)}_{ij} \neq 0$, and we say that the edge is true in the precision matrix $(\Sigma^{(l)})^{-1}$ if $((\Sigma^{(l)})^{-1})_{ij} \neq 0$ and false if $((\Sigma^{(l)})^{-1})_{ij} = 0$. We can see from the two figures that the FGL model with $\lambda_2 = 0.005$ and the GGL model with $w_2 = 0.2$ can recover almost all of the true positive edges without false positive edges. Figure 4.1a also indicates that for the FGL model the similarity control parameter $\lambda_2 = 0.005$ is much better than $\lambda_2 = 0.05$ in terms of the ability of true edges detection. When $\lambda_2 = 0.05$, the FGL model can merely detect about 3000 true positive edges while the the number of false positive edges is increased to over 600. One possible reason is that $\lambda_2 = 0.05$ is too large compared with the underlying optimal one in this case.

Figure 4.1b and Figure 4.1e illustrate the sum of squared errors between estimated edge values and true edge values, i.e., $\sum_{l=1}^{L} \sum_{i<j} \left(\widehat{\Theta}^{(l)}_{ij} - ((\Sigma^{(l)})^{-1})_{ij}\right)^2$, for the FGL and GGL models respectively. When the number of the total edges selected is increasing (i.e., the sparsity control parameter is decreasing), the error is decreasing and finally reaches a fairly low value.

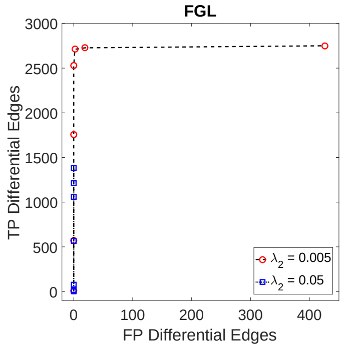
Figure 4.1c and Figure 4.1e plot the number of true positive differential edges against false positive differential edges for the FGL and GGL models respectively. A differential edge is an edge that differs between classes and thus corresponds to a change-point. We say that the (i, j) edge is estimated to be differential between the l-th and the $(l + 1)$-th networks if $|\widehat{\Theta}^{(l)}_{ij} - \widehat{\Theta}^{(l+1)}_{ij}| > 10^{-6}$, and we say that it is truly differential if $|((\Sigma^{(l)})^{-1})_{ij} - ((\Sigma^{(l+1)})^{-1})_{ij}| > 10^{-6}$. The number of differential edges is computed for all successive pairs of networks. The best point in Figure 4.1c is the red one which has approximately 2700 true positive differential edges and almost no false ones. While one satisfactory point in Figure 4.1f is the red one which has approximately 3000 true positive differential edges and almost no false ones. It is likely that the GGL model performances better in the change-point detection task of the nearest-neighbour networks in comparison with the FGL model. We can also see from Figure 4.1c that all the blue points have no false positive differential edge and small numbers of true positive differential edges. This might caused by the similarity control parameter $\lambda_2 = 0.05$ which compels excessively edges across L networks to be similar.



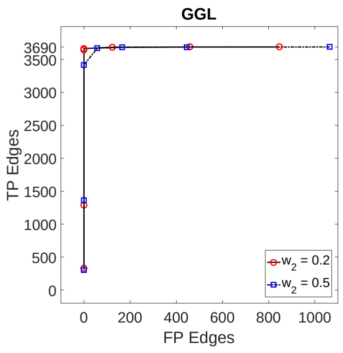
(a)



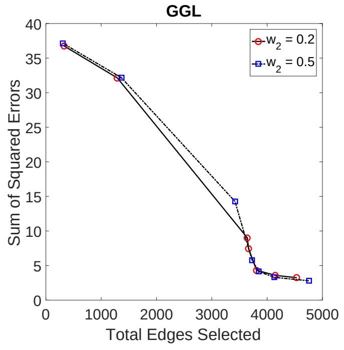
(b)



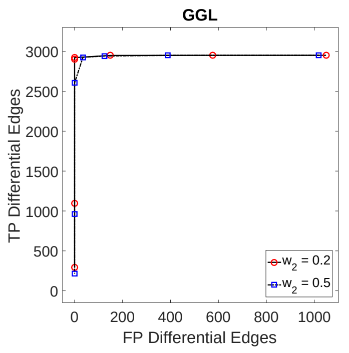
(c)



(d)



(e)



(f)

Figure 4.1: Performances of the GGL and FGL models on nearest-neighbour networks with $n = 500$ and $L = 3$. (a)(d) number of edges correctly identified to be nonzero (true positive edges) versus number of edges incorrectly identified to be nonzero (false positive edges); (b)(e) sum of squared errors in edge values versus the total number of edges estimated to be nonzero; (c)(f) number of edges correctly found to have values differing between successive classes (true positive differential edges) versus the number of edges incorrectly found to have values differing between successive classes (false positive differential edges).

4.4.3 Standard & Poor's 500 stock price

In this section, we compare PPA, ADMM, and MGL on the Standard & Poor's 500 stock price data sets. The stock price data sets contain daily returns of 500 stocks in a long period, and can be downloaded from the link `www.yahoo.com`. The dependency structures of different stocks vary along time. But it appears that the dependency networks change smoothly along time. Therefore, the GGL and FGL models might be able to find the interactions among these stocks and how they evolve over time.

The way of processing a data set will be described in the following paragraph. Once we have obtained a data set in the form of a matrix $H \in \mathbb{R}^{d\times n}$, where d is the number of days and n is the number of stocks, the data set is processed in the following way: (i) Split the matrix H into L small matrices by rows: $H = [H_1; H_2; \ldots; H_L]$. Each H_l records the prices of n stocks over a certain period. (ii) For $l = 1, 2, \ldots, L$, generate the sample covariance matrix $S^{(l)} = \mathrm{cov}(H_l)$.

We first consider a relatively short 3-year time period from January 2004 to December 2006. During this period, there are totally 755 days' returns of 370 stocks. We call this data set *SPX3a*. For each year, it contains approximately 250 days' returns of each stock. Considering the limited number of observations in each year and the interpretation of the results, we choose to analyse random smaller subsets of all involved stocks, whose sizes are chosen to be $n = 100$ and $n = 200$, over $L = 3$ periods.

In addition to the above data set over three years, a relatively long period from January 2004 to December 2014 is also considered in the numerical experiments, which is referred to as *SPX11b*. Since the time period is longer than the previous one, the number of stocks becomes smaller as some stocks might no longer exist. During the 11-year time period, there are 2769 daily closing prices of 272 stocks. We can set a relatively large parameter $L = 11$ according to years from January 2004 to December 2014. Again, we choose to analyse two random subsets of all existing stocks, of which the sizes are selected to be $n = 100$ and $n = 200$.

Table 4.1 shows the comparison of PPA, ADMM, and MGL on Standard & Poor's 500 stock price data sets *SPX3a* and *SPX11b* with 100 and 200 selected stocks. One outstanding observation from the table is that PPA is faster than ADMM and MGL for all instances in application of the GGL model and outperforms ADMM and MGL by an obvious margin for a large majority of all cases. In addition, we find that both PPA and ADMM succeeded in solving all instance; while MGL failed to solve two of them within 3 hours. This might imply that MGL is not robust for solving the models applied to stock price data sets. The numerical results show convincingly that our algorithm PPA can solve the GGL and FGL problems highly efficiently and robustly. The superior performance of our PPA algorithm can mainly be attributed to our ability to extract and exploit the sparsity structure (in the surrogate generalized Jacobian of $\mathrm{Prox}_{\mathcal{P}}$) within the SSN method to solve each PPA subproblem very efficiently.

Figure 4.2 displays the sparse patterns of 11 estimated precision matrices from year 2004 to year 2014 on the data set *SPX11b* with $(\lambda_1, \lambda_2) = $ (1e-4,1e-5). Note that this period covers the 2008 financial crisis. We manually split the time points into three stages (one stage corresponds to one row in Figure 4.2) to aid interpretation of the results. Each red pattern in the left panel presents the common structure across the estimated precision matrices in its stage. And each blue pattern visualizes the individual edges specific to its own precision matrix. Generally, one can hardly expect a meaningful common structure across all the 11 time points, and thus we provide here the common structure across parts of nearby precision matrices. The term "nnz" denotes the number of nonzero elements in the corresponding estimated precision matrix. One can clearly see that more edges are detected in the middle stage, and the number of the common edges across year 2007, 2008, 2009, 2010 is correspondingly larger than that in the earlier and later stages. The increased amount of interactions among the stocks over this period is likely due to the global financial crisis starting from 2007. Another observation is that the number of edges peaked in 2008, and then went down to a level still higher than that of the pre-crisis period (year 2004, 2005, 2006). The peak reflects the impact of the financial crisis, which is usually deemed fairly severe in 2008. The increased amount of interactions among stocks after the financial crisis compared to the pre-crisis period might indicate some essential changes of the financial landscape. To some degree, the observations coincide with those in a recent paper [86, Figure 1(a)].

Table 4.1: The performances of PPA, ADMM, and MGL on stock price data sets. Tolerance ε = 1e-6.

Model	Problem	(λ_1, λ_2)	Density	Iteration			Time			Error		
	(n, L)			P	A	M	P	A	M	P	A	M
		(1e-04,1e-05)	0.039	25	3701	6	06	33	31	6.0e-06	8.1e-06	1.1e-06
	SPX3a	(5e-05,5e-06)	0.138	24	3701	8	08	34	25	1.1e-05	8.2e-06	9.1e-06
	(100,3)	(2e-05,2e-06)	0.238	26	5216	20	12	49	15	1.2e-05	9.4e-06	1.1e-05
		(1e-04,1e-05)	0.025	24	3301	9	15	01:14	01:25	6.4e-06	5.1e-06	2.9e-06
	SPX3a	(5e-05,5e-06)	0.084	24	3301	16	22	01:20	02:31	5.1e-06	5.4e-06	4.8e-06
	(200,3)	(2e-05,2e-06)	0.150	26	6091	39	34	02:27	04:27	8.3e-06	1.1e-05	8.2e-06
GGL												
		(5e-04,5e-05)	0.030	24	3701	12	18	01:26	02:36	5.2e-06	4.5e-06	8.4e-07
	SPX11b	(1e-04,1e-05)	0.127	24	3701	110	26	01:59	03:12	5.1e-06	4.4e-06	5.0e-06
	(100,11)	(5e-05,5e-06)	0.206	24	3709	450	29	02:01	13:46	5.7e-06	4.6e-06	5.7e-06
		(5e-04,5e-05)	0.018	22	3501	31	57	04:23	43:14	2.0e-06	2.8e-06	1.5e-06
	SPX11b	(1e-04,1e-05)	0.082	24	3601	501	01:13	05:20	01:29:32	3.5e-06	2.3e-06	3.5e-06
	(200,11)	(5e-05,5e-06)	0.140	24	3568	1271	01:33	05:24	03:00:00	4.5e-06	2.9e-06	1.8e-05
		(1e-04,1e-05)	0.039	25	3701	6	06	33	33	6.0e-06	8.1e-06	1.1e-06
	SPX3a	(5e-05,5e-06)	0.144	24	3701	9	08	33	43	1.1e-05	8.3e-06	4.0e-06
	(100,3)	(2e-05,2e-06)	0.241	26	5359	19	11	49	09	1.3e-05	8.5e-06	1.2e-05
		(1e-04,1e-05)	0.025	24	3301	9	15	01:14	01:26	6.4e-06	5.1e-06	2.3e-06
	SPX3a	(5e-05,5e-06)	0.086	24	3301	17	22	01:15	03:22	5.1e-06	5.4e-06	4.7e-06
	(200,3)	(2e-05,2e-06)	0.150	26	5920	44	30	02:17	03:59	8.4e-06	1.1e-05	8.0e-06
FGL												
		(5e-04,5e-05)	0.028	24	3701	8	18	01:23	02:40	5.2e-06	4.5e-06	3.2e-06
	SPX11b	(1e-04,1e-05)	0.126	24	3701	111	27	01:55	04:59	5.1e-06	4.4e-06	4.9e-06
	(100,11)	(5e-05,5e-06)	0.206	24	3710	388	30	01:59	13:16	5.6e-06	4.5e-06	5.6e-06
		(5e-04,5e-05)	0.017	22	3501	28	54	03:58	46:47	2.0e-06	2.8e-06	1.7e-06
	SPX11b	(1e-04,1e-05)	0.081	24	3601	477	01:19	05:13	01:34:40	3.5e-06	2.3e-06	3.5e-06
	(200,11)	(5e-05,5e-06)	0.134	24	3573	1076	01:38	05:21	03:00:00	4.5e-06	2.8e-06	4.0e-05

4.4.4 University webpages

We evaluate in this section the numerical performances of PPA, ADMM, and MGL on a data set about university webpages, which is provided by a thesis [10] and can

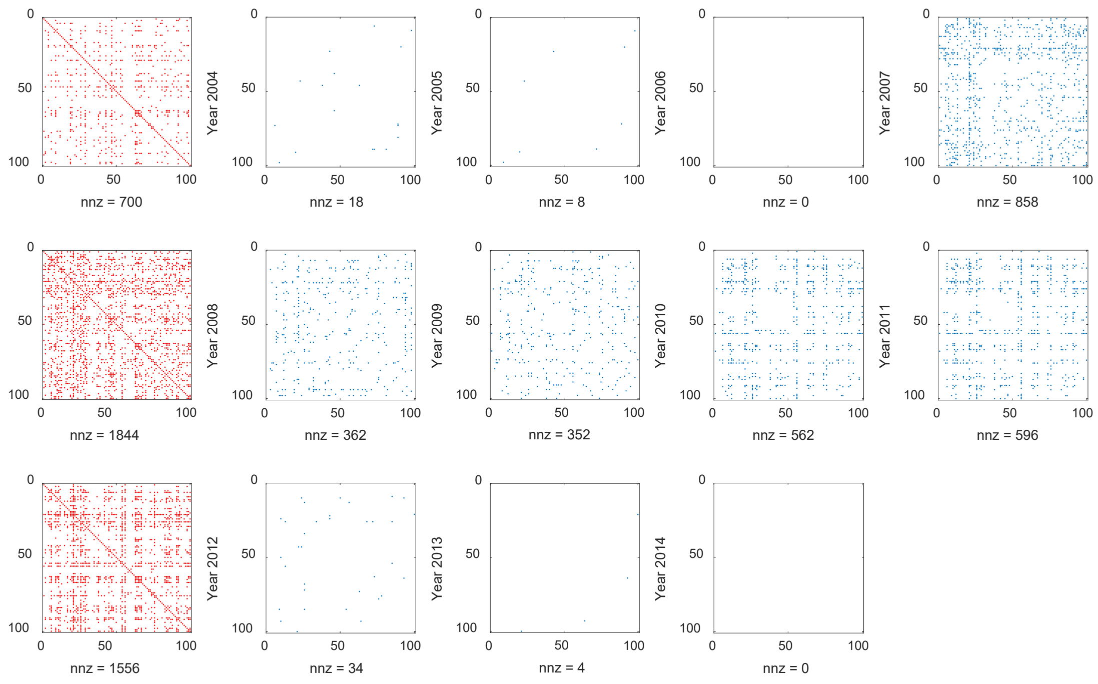


Figure 4.2: Patterns of estimated precision matrices over 11 years on the Standard & Poor's 500 stock price data sets. "nnz" denotes the number of nonzero elements, namely the number of dots in each pattern. The red pattern extracts the common structure of those in the same row. The blue pattern reflects individual edges specific to its own network.

be downloaded from the link

`http://ana.cachopo.org/datasets-for-single-label-text-categorization`.

These original pages were collected from computer science departments of various universities in 1997, manually classified into seven different classes: Student, Faculty, Staff, Department, Course, Project, and Other. For each class, the collection contains pages from four universities: Cornell, Texas, Washington, Wisconsin, and other miscellaneous pages collected from other universities. The classes Staff and Department were discarded because their pages were not enough for analysis. The class Other was also discarded because pages were very different among this class. That is, we selected four largest and meaningful classes in our numerical experiment. Furthermore, the original text data have been pre-processed by stemming techniques, that is, reducing words to their morphological root. The pre-processed data sets downloaded from the link above contain two files: two thirds of the pages were randomly chosen for training and the remaining third for testing. Table 4.2 presents the distribution of documents per class. In summary, we have a training data set, named by *Webtrain*, and a testing data set, named by *Webtest*.

Table 4.2: The distribution of documents of classes Student, Faculty, Course, and Project.

Class	#train docs	#test docs
Student	1097	544
Faculty	750	374
Course	620	310
Project	336	168
Total	2803	1396

The procedure of processing data and generating sample covariance matrices is described next, similar to the process used in [35]. Actually, previous work [35] on the data set *Webtest* applied a different penalty term to estimate multiple graphical models jointly. For given integer n, the sample covariance matrices $S^{(l)}$, $l = 1, 2, 3, 4$ were constructed from the data set *Webtest* in the following way: (i) Choose n words with highest frequency which appear in each class at least once. Namely, the words we analyse are a subset of all involved words. (ii) Obtain $X^{(1)} \in \mathbb{R}^{544\times n}$ from class Student, where the (i, j)-th element $X^{(1)}_{ij}$ denotes the number of times the j-th term appears in the i-th page of class Student. In the same way, $X^{(2)} \in \mathbb{R}^{374\times n}$, $X^{(3)} \in \mathbb{R}^{310\times n}$, and $X^{(4)} \in \mathbb{R}^{168\times n}$ can be obtained from class Faculty, Course, and Project, respectively. Denote their vertical concatenation by a new matrix $X = [X^{(1)}; X^{(2)}; X^{(3)}; X^{(4)}] \in \mathbb{R}^{1396\times n}$. (iii) The matrix P is obtained by normalizing X along each column: $P_{ij} = \frac{X_{ij}}{\sum_i X_{ij}}$. Then, the log-entropy weight of the j-th word is defined as $e_j = 1 + \frac{\sum_i P_{ij}(\ln P_{ij})}{\ln 1396}$. (iv) Compute $\overline{X}$ as follows: $\overline{X}_{ij} = e_j \ln(1+X_{ij})$, and split $\overline{X}$ by columns accordingly: $\overline{X} = [\overline{X}^{(1)}; \overline{X}^{(2)}; \overline{X}^{(3)}; \overline{X}^{(4)}]$. (v) Finally, generate sample covariance matrices $S^{(l)}$ from $\overline{X}^{(l)}$: $S^{(1)} = \mathrm{cov}(\overline{X}^{(1)})$, $l = 1, 2, 3, 4$. Following the procedure described above, we can also generate sample covariance matrices from the data set *Webtrain*. Due to the limited number of observations in each class, we conduct the numerical experiments on small subsets of all terms involved:

$n = 100$, $n = 200$, and $n = 300$.

First of all, we apply the FGL model to the *Webtest* data set for the purpose of the interpretation of the data. Due to the limited space, we merely analyse a subset of $n = 50$ words and do not show the results of the GGL model here. We choose tuning parameters that enforce high sparsity and similarity. In our experiment, we set $\lambda_1 = 0.005$ and $\lambda_2 = 0.003$.

The resulting common structure of four classes is displayed in Figure 4.3. The thickness of an edge is proportional to the magnitude of the average weight of that edge. Figure 4.3 shows that some standard phrases in computer science, like oper-system, distribut-system, softwar-engin, program-languag, possess high partial correlations among their constituent words in all four classes. It successfully demonstrates the effectiveness of the FGL model for exploring the similarity across related classes. On the other hand, we believe that the model can also detect the heterogeneity among different classes. As an example, Figure 4.4 illustrates the difference between the Course (Figure 4.4a) and Project (Figure 4.4b) classes. One can see that some course related terms, such as class and assign, are of high degree in Figure 4.4a; whereas they are not even connected in Figure 4.4b. Besides, some teaching related terms are linked only in the Course class, such as class-assign, assign-problem, class-project. Overall, it is likely that the FGL or GGL model is capable of identifying the common and individual structures of the webpages among related classes.

Table 4.3 shows the comparison of three methods PPA, ADMM, and MGL on the webpages data sets with data dimensionality $n = 100$, $n = 200$, and $n = 300$. Furthermore, Figures 4.5a and 4.5b present the performance profiles of PPA, ADMM, and MGL for all tested problems using GGL and FGL models respectively, which are presented in Table 4.3. 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, for both GGL and FGL problems, PPA outperforms ADMM and MGL by a large

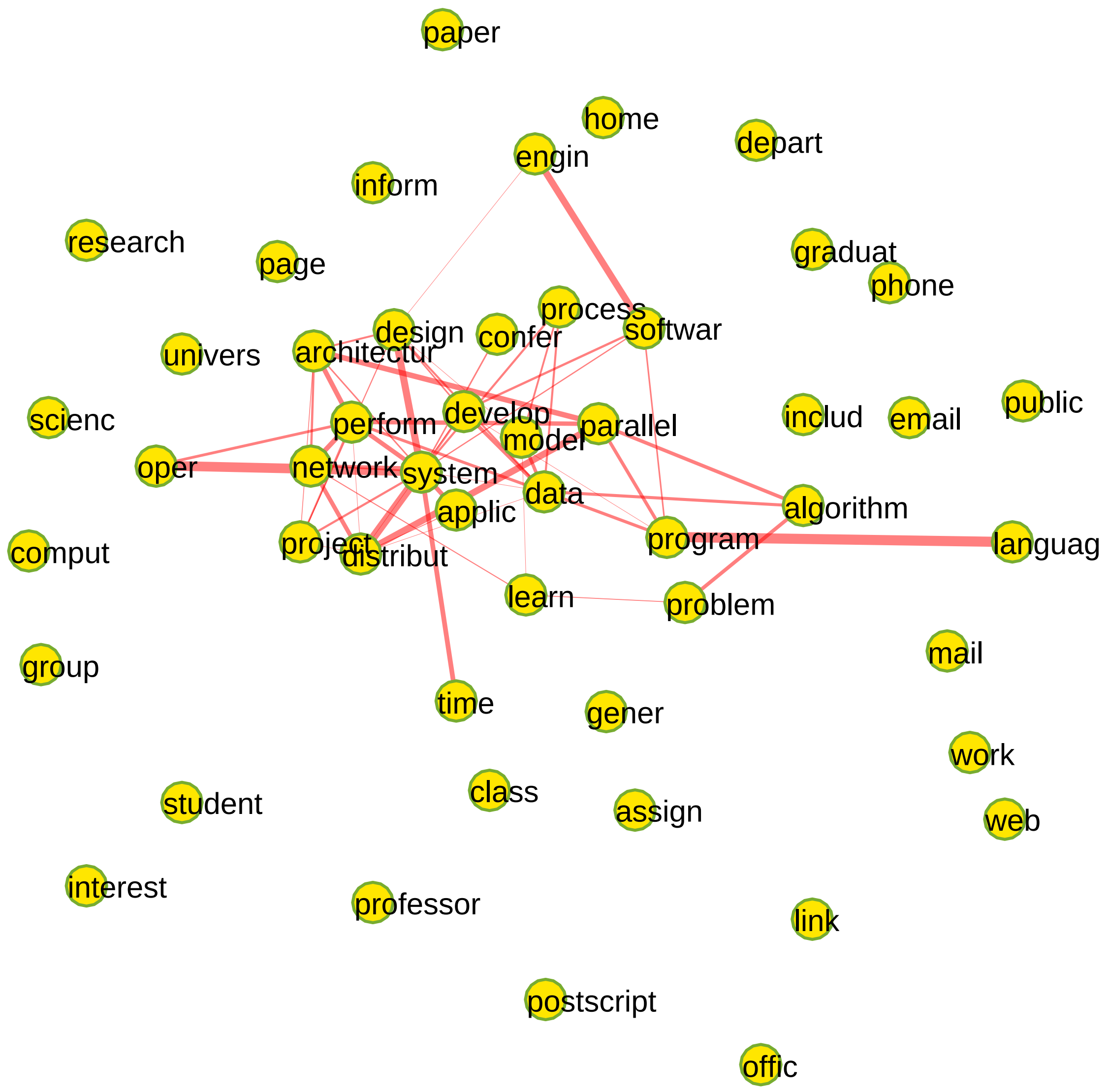
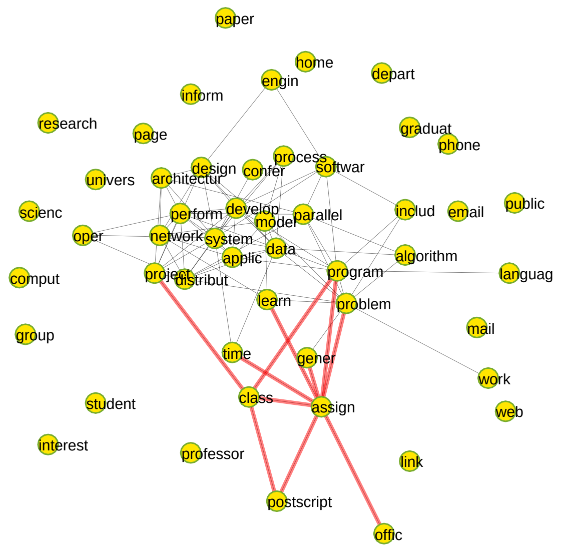
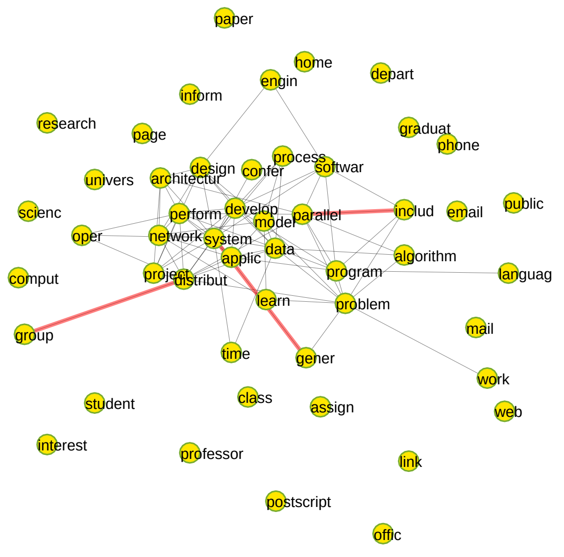


Figure 4.3: Common structure in the *Webtest* data. The nodes represent 50 words with highest frequencies. The width of an edge is proportional to the magnitude of the partial correlation.

margin for all tested webpages data sets. In particular, focusing on $y = 40\%$, we can see that PPA is around $3 \sim 5$ times faster in comparison with ADMM and MGL for over 60% of the tested instances. The results indicate again that our algorithm is efficient for solving the GGL and FGL problems.

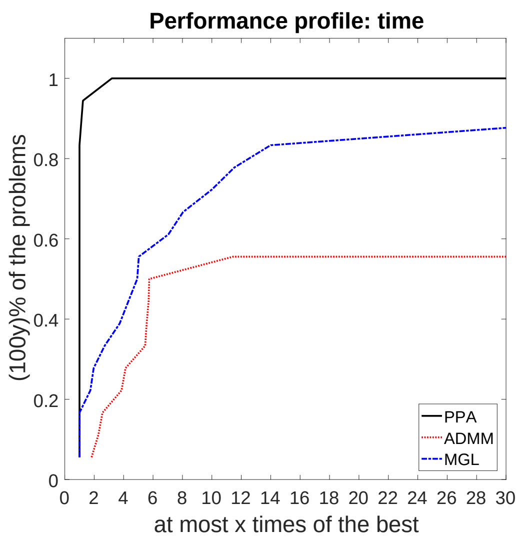


(a)

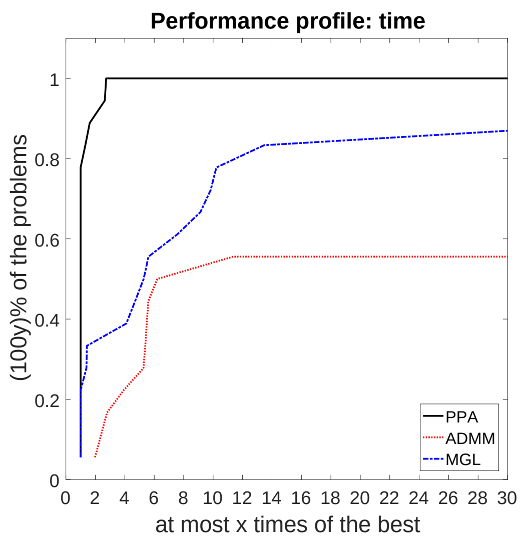


(b)

Figure 4.4: Dependency structures for class (a) Course (b) Project. The thin black lines are the edges appearing in both classes, and the thick red lines are the edges only appearing in one class.



(a)



(b)

Figure 4.5: Performance profiles of PPA, ADMM, and MGL on university webpages data sets for (a) the GGL problems and (b) the FGL problems.

Table 4.3: Performances of PPA, ADMM, and MGL on webpages data sets. Tolerance ε = 1e-6.

Model	Problem	(λ_1, λ_2)	Density	Iteration			Time			Error		
	(n, L)			P	A	M	P	A	M	P	A	M
		(1e-02,1e-03)	0.016	16	2401	4	07	25	11	5.8e-07	9.9e-07	2.9e-07
	Webtest	(5e-03,5e-04)	0.048	16	2401	6	08	31	15	5.6e-07	9.9e-07	1.1e-07
	(100,4)	(1e-03,1e-04)	0.225	15	701	33	05	09	01:10	5.7e-07	6.1e-07	9.6e-07
		(1e-02,1e-03)	0.008	18	2101	5	12	01:05	45	7.3e-07	9.4e-07	6.6e-08
	Webtest	(5e-03,5e-04)	0.026	18	2101	8	12	01:08	01:00	6.7e-07	9.4e-07	6.5e-07
	(200,4)	(1e-03,1e-04)	0.163	18	2101	89	13	01:13	07:25	5.5e-07	7.4e-07	9.5e-07
		(5e-03,5e-04)	0.016	18	2101	8	25	02:17	02:53	5.9e-07	9.4e-07	7.3e-08
	Webtest	(1e-03,1e-04)	0.125	17	2101	305	56	02:25	32:25	9.2e-07	1.0e-06	1.0e-06
	(300,4)	(5e-04,5e-05)	0.256	19	2901	1500	01:22	03:06	02:27:35	7.1e-07	1.1e-06	1.2e-06
GGL		(1e-02,1e-03)	0.012	24	20000	4	12	03:27	10	4.4e-06	4.1e-05	-7.9e-08
	Webtrain	(5e-03,5e-04)	0.033	24	20000	5	14	03:43	04	4.4e-06	4.1e-05	7.2e-07
	(100,4)	(1e-03,1e-04)	0.165	24	20000	27	14	04:02	39	4.2e-06	3.9e-05	3.6e-06
		(5e-03,5e-04)	0.016	24	20000	7	47	11:06	38	2.9e-06	3.0e-05	3.0e-08
	Webtrain	(1e-03,1e-04)	0.108	24	15226	39	44	08:26	03:38	2.9e-06	3.9e-06	2.7e-06
	(200,4)	(5e-04,5e-05)	0.219	24	20000	100	53	11:06	07:07	2.8e-06	2.9e-05	2.7e-06
		(5e-03,5e-04)	0.011	24	20000	9	01:32	21:56	06:39	1.9e-06	2.0e-05	1.1e-06
	Webtrain	(1e-03,1e-04)	0.080	24	20000	65	01:40	22:28	16:35	1.9e-06	2.0e-05	1.7e-06
	(300,4)	(5e-04,5e-05)	0.177	24	20000	194	01:59	22:35	22:46	1.8e-06	1.9e-05	1.8e-06
		(1e-02,1e-03)	0.015	16	2401	4	06	25	05	5.8e-07	9.9e-07	1.2e-07
	Webtest	(5e-03,5e-04)	0.047	16	2401	6	07	27	10	5.6e-07	9.9e-07	3.1e-07
	(100,4)	(1e-03,1e-04)	0.219	15	701	38	05	09	49	5.7e-07	6.1e-07	9.0e-07
		(1e-02,1e-03)	0.008	18	2101	7	11	01:03	59	7.3e-07	9.4e-07	8.6e-07
	Webtest	(5e-03,5e-04)	0.025	18	2101	8	12	01:04	01:05	6.8e-07	9.4e-07	4.7e-07
	(200,4)	(1e-03,1e-04)	0.156	18	2101	72	12	01:12	07:31	5.5e-07	7.4e-07	9.3e-07
		(5e-03,5e-04)	0.016	18	2101	9	25	02:12	03:44	5.9e-07	9.4e-07	3.7e-07
	Webtest	(1e-03,1e-04)	0.119	17	2101	258	49	02:16	39:58	8.7e-07	9.6e-07	1.0e-06
	(300,4)	(5e-04,5e-05)	0.244	19	2901	1393	01:18	03:07	02:22:23	6.8e-07	9.0e-07	1.0e-06
FGL		(1e-02,1e-03)	0.011	24	20000	3	12	03:11	04	4.4e-06	4.1e-05	3.4e-06
	Webtrain	(5e-03,5e-04)	0.030	24	20000	5	13	03:35	08	4.4e-06	4.1e-05	8.9e-07
	(100,4)	(1e-03,1e-04)	0.162	24	20000	22	14	03:52	01:06	4.2e-06	3.9e-05	3.5e-06
		(5e-03,5e-04)	0.015	24	20000	5	46	10:53	17	2.9e-06	3.0e-05	1.2e-06
	Webtrain	(1e-03,1e-04)	0.105	24	15227	33	44	08:23	03:02	2.9e-06	3.9e-06	2.6e-06
	(200,4)	(5e-04,5e-05)	0.210	24	20000	95	52	10:41	06:34	2.8e-06	2.9e-05	2.7e-06
		(5e-03,5e-04)	0.010	24	20000	7	01:31	21:31	02:07	1.9e-06	2.0e-05	-1.2e-08
	Webtrain	(1e-03,1e-04)	0.077	24	20000	52	01:33	22:06	20:47	1.9e-06	2.0e-05	1.8e-06
	(300,4)	(5e-04,5e-05)	0.168	24	20000	155	01:51	21:56	18:09	1.8e-06	1.9e-05	1.8e-06

4.4.5 20 Newsgroups

This section compares PPA, ADMM, and MGL on newsgroups data set, which is a popular text data set. The 20 newsgroups data set is a collection of newsgroup documents, partitioned nearly evenly across 20 different newsgroups. Different newsgroups correspond to different topics, and some of the newsgroups are closely related to each other (e.g., comp.sys.ibm.pc.hardware/comp.sys.mac.hardware), while others are highly unrelated (e.g., misc.forsale/soc.religion.christian). Table 4.4 lists the 20 newsgroups, partitioned according to subject matter [*]. According to the topics, our numerical experiments were conducted on the four sub-groups, which are likely to possess common semantic structures. The four sub-groups are highlighted in Table 4.4 and named as *NGcomp*, *NGrec*, *NGsci*, and *NGtalk* accordingly.

There are several classes in each sub-group, and we apply the GGL and FGL models to estimating jointly the precision matrices of different classes in each sub-group.

Table 4.4: Partition of 20 newsgroups by topics

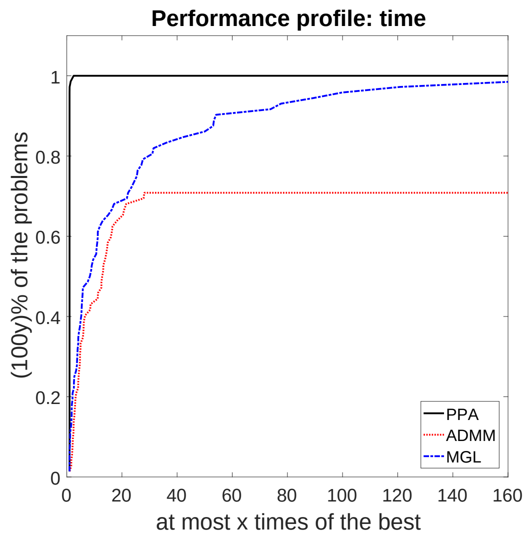
comp.graphics **comp.os.ms-windows.misc** **comp.sys.ibm.pc.hardware** **comp.sys.mac.hardware** **comp.windows.x**	**rec.autos** **rec.motorcycles** **rec.sport.baseball** **rec.sport.hockey**	**sci.crypt** **sci.electronics** **sci.med** **sci.space**
misc.forsale	**talk.politics.misc** **talk.politics.guns** **talk.politics.mideast**	talk.religion.misc alt.atheism soc.religion.christian

A processed version of the 20 newsgroups data set can be downloaded from Jason's page `http://qwone.com/~jason/20Newsgroups/`, and the downloaded data contains a training data set and a testing data set. We also adopted the procedure of generating sample covariance matrices described in the previous section 4.4.4 with a series of problem dimensionality $n = 100$, $n = 200$, and $n = 300$.

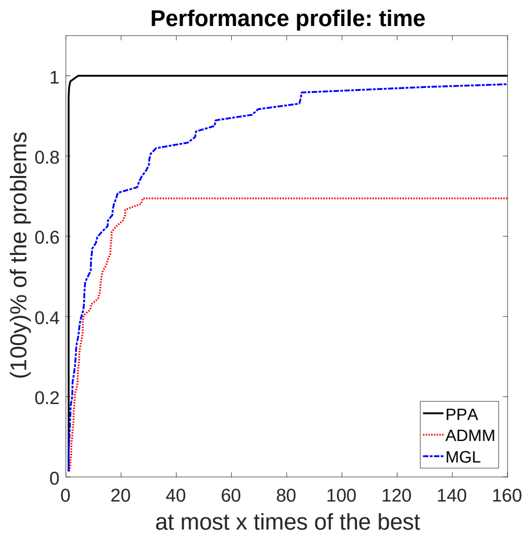
Table 4.5 shows the comparison of PPA, ADMM, and MGL on the testing and training data sets of four sub-groups with $n = 300$. The results for $n = 200$ and $n = 100$ are shown in Table 4.6 and Table 4.7, respectively. Furthermore, we summarize in Figure 4.6 all conducted instances in Table 4.5, Table 4.6, and Table

[*]The table is from the website: `http://qwone.com/~jason/20Newsgroups/`

4.7. One can clearly see from Figure 4.6 that for both GGL and FGL problems, PPA outperforms greatly ADMM and MGL. It truly suggests that our proposed algorithm is efficient for solving the GGL and FGL problems.



(a)



(b)

Figure 4.6: Performance profiles of PPA, ADMM, and MGL on newsgroups data sets with $n = 100$, $n = 200$, and $n = 300$ for (a) the GGL problems and (b) the FGL problems.

Table 4.5: The performances of PPA, ADMM, and MGL on newsgroups data sets. $n = 300$. Tolerance ε = 1e-6.

Model	Problem	(λ_1, λ_2)	Density	Iteration			Time			Error		
	(n, L)			P	A	M	P	A	M	P	A	M
	NGcomp	(5e-03,5e-04)	0.021	19	4201	41	51	05:17	42:31	5.9e-07	1.0e-06	7.6e-07
	test	(1e-03,1e-04)	0.099	15	1501	711	01:15	02:02	01:32:52	6.4e-07	5.3e-07	1.0e-06
	(300,5)	(5e-04,5e-05)	0.210	16	1342	1240	57	01:48	03:00:00	1.1e-06	1.0e-06	7.4e-06
	NGrec	(5e-03,5e-04)	0.004	25	20000	5	01:12	19:51	04:59	1.2e-06	1.8e-05	4.9e-07
	test	(1e-03,1e-04)	0.063	25	20000	14	01:19	21:43	04:49	1.2e-06	1.8e-05	7.6e-07
	(300,4)	(5e-04,5e-05)	0.143	24	20000	46	01:22	21:37	07:58	2.0e-06	1.8e-05	1.9e-06
	NGsci	(5e-03,5e-04)	0.006	22	16244	7	56	15:26	07:47	8.9e-07	2.1e-06	4.4e-07
	test	(1e-03,1e-04)	0.075	21	16230	23	01:06	17:30	11:51	1.5e-06	2.1e-06	1.4e-06
	(300,4)	(5e-04,5e-05)	0.167	21	16230	116	01:16	17:33	20:41	1.5e-06	2.0e-06	1.5e-06
	NGtalk	(5e-03,5e-04)	0.026	17	4179	16	51	04:19	23:10	7.3e-07	1.0e-06	6.9e-07
	test	(1e-03,1e-04)	0.115	17	1101	81	33	01:12	17:16	5.0e-07	6.1e-07	9.8e-07
	(300,3)	(5e-04,5e-05)	0.240	16	1101	482	35	01:12	58:44	9.7e-07	4.7e-07	1.0e-06
GGL	*NGcomp*	(5e-03,5e-04)	0.016	20	6023	13	36	07:32	18:53	8.6e-07	1.0e-06	6.4e-07
	train	(1e-03,1e-04)	0.080	20	6393	213	41	08:38	36:32	7.7e-07	1.3e-06	9.9e-07
	(300,5)	(5e-04,5e-05)	0.153	19	5856	743	56	07:52	01:52:17	1.1e-06	1.0e-06	1.1e-06
	NGrec	(5e-03,5e-04)	0.005	26	8842	5	01:29	07:08	02:30	1.5e-06	1.0e-06	4.0e-07
	train	(1e-03,1e-04)	0.068	24	8737	18	01:57	09:22	10:17	2.2e-06	1.0e-06	2.2e-06
	(300,4)	(5e-04,5e-05)	0.124	24	8623	68	02:06	09:20	11:47	2.2e-06	1.0e-06	2.1e-06
	NGsci	(5e-03,5e-04)	0.011	21	11166	10	41	11:12	11:44	1.0e-06	2.0e-06	1.6e-08
	train	(1e-03,1e-04)	0.086	20	11137	40	54	12:02	10:01	1.6e-06	1.9e-06	1.6e-06
	(300,4)	(5e-04,5e-05)	0.152	20	11505	273	01:07	12:31	27:14	1.7e-06	1.9e-06	1.7e-06
	NGtalk	(5e-03,5e-04)	0.026	22	20000	14	01:28	21:19	13:17	1.7e-06	5.9e-06	3.6e-07
	train	(1e-03,1e-04)	0.103	22	20000	59	43	21:40	10:38	1.7e-06	4.7e-06	1.6e-06
	(300,3)	(5e-04,5e-05)	0.204	22	20000	367	45	21:55	40:47	1.7e-06	4.4e-06	1.7e-06
	NGcomp	(5e-03,5e-04)	0.020	19	4201	35	53	05:27	41:26	5.9e-07	1.0e-06	6.2e-07
	test	(1e-03,1e-04)	0.094	16	1543	720	01:20	02:08	01:54:03	9.7e-07	1.0e-06	1.0e-06
	(300,5)	(5e-04,5e-05)	0.194	17	1391	1240	59	01:56	03:00:05	7.9e-07	1.0e-06	9.6e-06
	NGrec	(5e-03,5e-04)	0.004	25	20000	4	01:16	21:18	04:43	1.2e-06	1.8e-05	8.3e-07
	test	(1e-03,1e-04)	0.061	25	20000	13	01:20	22:10	04:38	1.2e-06	1.8e-05	4.6e-07
	(300,4)	(5e-04,5e-05)	0.134	24	20000	37	01:23	22:05	07:55	2.0e-06	1.8e-05	1.9e-06
	NGsci	(5e-03,5e-04)	0.006	22	16244	6	57	15:42	06:00	8.9e-07	2.1e-06	2.7e-07
	test	(1e-03,1e-04)	0.074	21	16230	25	01:06	17:54	10:22	1.5e-06	2.1e-06	1.3e-06
	(300,4)	(5e-04,5e-05)	0.156	21	16230	100	01:06	17:52	20:37	1.5e-06	2.0e-06	1.5e-06
	NGtalk	(5e-03,5e-04)	0.026	17	4179	14	52	04:23	14:33	7.4e-07	1.0e-06	2.7e-07
	test	(1e-03,1e-04)	0.111	17	1018	79	35	01:08	16:52	4.9e-07	1.0e-06	9.8e-07
	(300,3)	(5e-04,5e-05)	0.228	17	922	434	37	01:00	52:12	6.8e-07	1.0e-06	9.9e-07
FGL	*NGcomp*	(5e-03,5e-04)	0.016	20	6023	13	36	07:36	26:32	8.6e-07	1.0e-06	5.2e-07
	train	(1e-03,1e-04)	0.077	20	6393	153	40	08:59	36:18	7.7e-07	1.3e-06	9.7e-07
	(300,5)	(5e-04,5e-05)	0.142	19	5861	662	52	07:58	01:52:34	1.1e-06	1.0e-06	1.1e-06
	NGrec	(5e-03,5e-04)	0.004	26	8842	5	01:30	07:17	03:34	1.5e-06	1.0e-06	4.7e-07
	train	(1e-03,1e-04)	0.067	24	8737	17	01:58	09:31	06:32	2.2e-06	1.0e-06	1.7e-06
	(300,4)	(5e-04,5e-05)	0.119	24	8625	66	02:05	09:30	14:16	2.2e-06	1.0e-06	2.1e-06
	NGsci	(5e-03,5e-04)	0.011	21	11166	10	41	11:15	11:32	1.0e-06	2.0e-06	2.1e-08
	train	(1e-03,1e-04)	0.085	20	11137	41	01:03	12:17	13:45	1.6e-06	1.9e-06	1.5e-06
	(300,4)	(5e-04,5e-05)	0.146	20	11405	226	01:02	12:41	28:27	1.7e-06	1.9e-06	1.6e-06
	NGtalk	(5e-03,5e-04)	0.026	22	20000	12	01:29	21:04	13:18	1.7e-06	5.9e-06	1.4e-06
	train	(1e-03,1e-04)	0.101	22	20000	74	41	22:10	12:00	1.7e-06	4.8e-06	1.6e-06
	(300,3)	(5e-04,5e-05)	0.193	22	20000	402	40	23:15	45:12	1.7e-06	4.4e-06	1.6e-06

Table 4.6: The performances of PPA, ADMM, and MGL on newsgroups data sets. $n = 200$. Tolerance $\varepsilon = $ 1e-6.

Model	Problem	(λ_1, λ_2)	Density	Iteration			Time			Error		
	(n, L)			P	A	M	P	A	M	P	A	M
	NGcomp	(5e-03,5e-04)	0.034	17	1401	25	16	53	13:49	1.0e-06	9.5e-07	9.6e-07
	test	(1e-03,1e-04)	0.136	18	1580	369	26	01:05	39:09	6.5e-07	1.0e-06	1.0e-06
	(200,5)	(5e-04,5e-05)	0.258	15	2549	1492	33	01:44	01:30:52	9.6e-07	1.0e-06	1.0e-06
	NGrec	(5e-03,5e-04)	0.005	25	20000	5	35	09:09	36	1.9e-06	2.8e-05	1.4e-06
	test	(1e-03,1e-04)	0.097	25	20000	15	39	10:35	02:33	1.8e-06	2.7e-05	1.0e-06
	(200,4)	(5e-04,5e-05)	0.178	25	20000	37	39	10:41	02:53	1.8e-06	2.7e-05	1.7e-06
	NGsci	(5e-03,5e-04)	0.010	22	7720	5	23	03:22	01:27	1.2e-06	1.7e-06	3.3e-08
	test	(1e-03,1e-04)	0.112	21	16890	15	25	09:09	01:38	2.2e-06	2.8e-06	2.1e-06
	(200,4)	(5e-04,5e-05)	0.199	21	16890	62	26	08:58	04:07	2.2e-06	2.7e-06	2.0e-06
	NGtalk	(5e-03,5e-04)	0.047	19	4401	9	23	02:14	04:18	9.0e-07	8.6e-07	5.8e-07
	test	(1e-03,1e-04)	0.151	19	4301	72	28	02:18	03:30	6.2e-07	4.5e-07	9.5e-07
	(200,3)	(5e-04,5e-05)	0.277	17	1101	352	21	35	14:49	6.4e-07	9.5e-07	9.9e-07
GGL	*NGcomp*	(5e-03,5e-04)	0.026	20	7552	14	21	04:36	09:53	9.2e-07	1.8e-06	4.8e-07
	train	(1e-03,1e-04)	0.118	19	7236	115	20	04:50	08:17	1.5e-06	1.7e-06	1.5e-06
	(200,5)	(5e-04,5e-05)	0.197	19	7207	479	25	04:40	32:01	1.5e-06	1.6e-06	1.5e-06
	NGrec	(5e-03,5e-04)	0.007	26	20000	4	38	07:36	01:19	2.4e-06	6.5e-05	5.8e-07
	train	(1e-03,1e-04)	0.111	25	10021	13	53	05:17	01:20	4.0e-06	1.0e-06	3.4e-06
	(200,4)	(5e-04,5e-05)	0.163	25	9911	42	51	05:14	02:14	3.9e-06	1.0e-06	3.8e-06
	NGsci	(5e-03,5e-04)	0.018	21	9149	8	18	04:20	03:52	7.5e-07	1.8e-06	1.4e-08
	train	(1e-03,1e-04)	0.128	20	9005	36	16	04:44	02:46	1.3e-06	1.7e-06	1.1e-06
	(200,4)	(5e-04,5e-05)	0.191	20	8977	217	20	04:44	11:58	1.2e-06	1.7e-06	1.2e-06
	NGtalk	(5e-03,5e-04)	0.048	25	20000	11	48	10:14	04:18	2.4e-06	1.1e-05	6.3e-07
	train	(1e-03,1e-04)	0.144	22	20000	52	32	10:42	04:45	2.5e-06	7.8e-06	2.3e-06
	(200,3)	(5e-04,5e-05)	0.240	22	11808	241	28	06:16	12:10	2.4e-06	1.4e-06	2.4e-06
	NGcomp	(5e-03,5e-04)	0.033	17	1401	26	16	54	14:47	1.0e-06	9.5e-07	7.0e-07
	test	(1e-03,1e-04)	0.131	18	1551	377	23	01:04	32:28	8.4e-07	1.0e-06	1.0e-06
	(200,5)	(5e-04,5e-05)	0.242	18	2346	1288	29	01:33	01:33:40	5.4e-07	1.0e-06	1.0e-06
	NGrec	(5e-03,5e-04)	0.005	25	20000	4	35	09:25	58	1.9e-06	2.8e-05	1.5e-06
	test	(1e-03,1e-04)	0.096	25	20000	13	39	11:00	03:23	1.8e-06	2.7e-05	1.5e-06
	(200,4)	(5e-04,5e-05)	0.169	25	20000	37	39	10:49	03:13	1.8e-06	2.7e-05	1.6e-06
	NGsci	(5e-03,5e-04)	0.009	22	7720	6	24	03:30	01:46	1.2e-06	1.7e-06	1.0e-08
	test	(1e-03,1e-04)	0.110	21	16890	15	26	09:09	02:50	2.2e-06	2.8e-06	1.9e-06
	(200,4)	(5e-04,5e-05)	0.189	21	16890	64	26	09:04	04:59	2.2e-06	2.7e-06	2.1e-06
	NGtalk	(5e-03,5e-04)	0.046	19	4401	9	24	02:16	04:16	9.1e-07	8.6e-07	7.6e-07
	test	(1e-03,1e-04)	0.148	19	4314	59	26	02:22	03:30	6.2e-07	1.0e-06	9.9e-07
	(200,3)	(5e-04,5e-05)	0.265	17	1101	328	19	36	14:59	7.0e-07	7.1e-07	9.9e-07
FGL	*NGcomp*	(5e-03,5e-04)	0.025	20	7552	15	22	04:42	11:53	9.2e-07	1.8e-06	1.9e-08
	train	(1e-03,1e-04)	0.116	19	7236	101	20	04:50	10:01	1.5e-06	1.7e-06	1.5e-06
	(200,5)	(5e-04,5e-05)	0.186	19	7207	413	23	04:50	26:59	1.5e-06	1.6e-06	1.5e-06
	NGrec	(5e-03,5e-04)	0.007	26	20000	4	39	07:48	01:25	2.4e-06	6.5e-05	5.6e-07
	train	(1e-03,1e-04)	0.110	25	10021	15	53	05:22	01:17	4.0e-06	1.0e-06	3.0e-06
	(200,4)	(5e-04,5e-05)	0.160	25	9912	41	53	05:27	02:09	3.9e-06	1.0e-06	3.7e-06
	NGsci	(5e-03,5e-04)	0.017	21	9149	8	17	04:11	05:01	7.5e-07	1.8e-06	2.4e-08
	train	(1e-03,1e-04)	0.129	20	9005	42	16	04:51	04:02	1.3e-06	1.7e-06	1.2e-06
	(200,4)	(5e-04,5e-05)	0.185	20	8977	158	18	04:52	09:18	1.2e-06	1.7e-06	9.1e-07
	NGtalk	(5e-03,5e-04)	0.048	25	20000	13	48	10:39	07:41	2.4e-06	1.1e-05	2.3e-07
	train	(1e-03,1e-04)	0.143	22	20000	54	33	10:47	03:34	2.5e-06	7.8e-06	2.5e-06
	(200,3)	(5e-04,5e-05)	0.230	22	20000	265	29	10:45	12:57	2.4e-06	7.0e-06	2.4e-06

Table 4.7: The performances of PPA, ADMM, and MGL on newsgroups data sets. $n = 100$. Tolerance ε = 1e-6.

Model	Problem	(λ_1, λ_2)	Density	Iteration			Time			Error		
	(n, L)			P	A	M	P	A	M	P	A	M
	NGcomp	(5e-03,5e-04)	0.076	15	547	12	03	08	01:11	9.4e-07	1.0e-06	1.6e-07
	test	(2e-03,2e-04)	0.186	15	569	42	04	09	01:24	6.9e-07	9.9e-07	9.1e-07
	(200,5)	(1e-03,1e-04)	0.228	16	763	101	04	11	01:40	8.2e-07	1.0e-06	8.3e-07
	NGrec	(5e-03,5e-04)	0.011	25	20000	4	14	03:16	10	3.7e-06	5.6e-05	7.0e-07
	test	(2e-03,2e-04)	0.079	25	20000	6	15	03:42	19	3.6e-06	5.5e-05	-5.8e-08
	(200,4)	(1e-03,1e-04)	0.194	25	20000	8	17	03:55	20	3.5e-06	5.4e-05	2.2e-06
	NGsci	(5e-03,5e-04)	0.022	21	16832	4	08	02:50	15	2.3e-06	3.2e-06	1.9e-06
	test	(2e-03,2e-04)	0.156	21	16510	9	07	03:06	07	2.4e-06	3.1e-06	1.3e-07
	(200,4)	(1e-03,1e-04)	0.216	21	16366	11	07	03:07	14	2.3e-06	3.0e-06	2.0e-06
	NGtalk	(5e-03,5e-04)	0.119	18	2801	10	07	32	36	1.2e-06	1.2e-06	1.9e-09
	test	(2e-03,2e-04)	0.218	18	2771	15	08	32	32	1.1e-06	1.2e-06	4.9e-07
	(200,3)	(1e-03,1e-04)	0.237	15	2741	27	08	31	43	1.1e-06	1.3e-06	8.1e-07
GGL	*NGcomp*	(5e-03,5e-04)	0.050	20	5547	8	12	01:17	58	6.2e-07	1.3e-06	2.3e-07
	train	(2e-03,2e-04)	0.161	19	2701	12	12	38	32	1.0e-06	9.5e-07	3.5e-07
	(200,5)	(1e-03,1e-04)	0.216	17	1701	33	05	24	01:02	9.7e-07	1.2e-06	9.0e-07
	NGrec	(5e-03,5e-04)	0.016	26	20000	4	13	03:01	23	4.2e-06	8.6e-05	1.4e-07
	train	(2e-03,2e-04)	0.117	26	20000	9	15	03:43	42	4.2e-06	8.5e-05	1.2e-06
	(200,4)	(1e-03,1e-04)	0.226	26	20000	12	16	04:02	17	4.1e-06	8.3e-05	3.8e-06
	NGsci	(5e-03,5e-04)	0.045	15	1093	7	04	12	47	7.6e-07	1.0e-06	1.2e-07
	train	(2e-03,2e-04)	0.185	15	1093	10	05	13	16	7.2e-07	1.0e-06	8.2e-07
	(200,4)	(1e-03,1e-04)	0.237	15	322	19	04	04	20	6.3e-07	1.0e-06	6.1e-07
	NGtalk	(5e-03,5e-04)	0.121	23	20000	8	22	03:48	27	3.0e-06	2.3e-05	8.5e-07
	train	(2e-03,2e-04)	0.230	22	20000	14	21	03:49	08	5.2e-06	1.9e-05	2.8e-06
	(200,3)	(1e-03,1e-04)	0.239	22	20000	25	13	03:51	24	4.9e-06	1.6e-05	4.2e-06
	NGcomp	(5e-03,5e-04)	0.033	17	1401	26	16	54	14:47	1.0e-06	9.5e-07	7.0e-07
	test	(1e-03,1e-04)	0.131	18	1551	377	23	01:04	32:28	8.4e-07	1.0e-06	1.0e-06
	(200,5)	(5e-04,5e-05)	0.242	18	2346	1288	29	01:33	01:33:40	5.4e-07	1.0e-06	1.0e-06
	NGrec	(5e-03,5e-04)	0.005	25	20000	4	35	09:25	58	1.9e-06	2.8e-05	1.5e-06
	test	(1e-03,1e-04)	0.096	25	20000	13	39	11:00	03:23	1.8e-06	2.7e-05	1.5e-06
	(200,4)	(5e-04,5e-05)	0.169	25	20000	37	39	10:49	03:13	1.8e-06	2.7e-05	1.6e-06
	NGsci	(5e-03,5e-04)	0.009	22	7720	6	24	03:30	01:46	1.2e-06	1.7e-06	1.0e-08
	test	(1e-03,1e-04)	0.110	21	16890	15	26	09:09	02:50	2.2e-06	2.8e-06	1.9e-06
	(200,4)	(5e-04,5e-05)	0.189	21	16890	64	26	09:04	04:59	2.2e-06	2.7e-06	2.1e-06
	NGtalk	(5e-03,5e-04)	0.046	19	4401	9	24	02:16	04:16	9.1e-07	8.6e-07	7.6e-07
	test	(1e-03,1e-04)	0.148	19	4314	59	26	02:22	03:30	6.2e-07	1.0e-06	9.9e-07
	(200,3)	(5e-04,5e-05)	0.265	17	1101	328	19	36	14:59	7.0e-07	7.1e-07	9.9e-07
FGL	*NGcomp*	(5e-03,5e-04)	0.025	20	7552	15	22	04:42	11:53	9.2e-07	1.8e-06	1.9e-08
	train	(1e-03,1e-04)	0.116	19	7236	101	20	04:50	10:01	1.5e-06	1.7e-06	1.5e-06
	(200,5)	(5e-04,5e-05)	0.186	19	7207	413	23	04:50	26:59	1.5e-06	1.6e-06	1.5e-06
	NGrec	(5e-03,5e-04)	0.007	26	20000	4	39	07:48	01:25	2.4e-06	6.5e-05	5.6e-07
	train	(1e-03,1e-04)	0.110	25	10021	15	53	05:22	01:17	4.0e-06	1.0e-06	3.0e-06
	(200,4)	(5e-04,5e-05)	0.160	25	9912	41	53	05:27	02:09	3.9e-06	1.0e-06	3.7e-06
	NGsci	(5e-03,5e-04)	0.017	21	9149	8	17	04:11	05:01	7.5e-07	1.8e-06	2.4e-08
	train	(1e-03,1e-04)	0.129	20	9005	42	16	04:51	04:02	1.3e-06	1.7e-06	1.2e-06
	(200,4)	(5e-04,5e-05)	0.185	20	8977	158	18	04:52	09:18	1.2e-06	1.7e-06	9.1e-07
	NGtalk	(5e-03,5e-04)	0.048	25	20000	13	48	10:39	07:41	2.4e-06	1.1e-05	2.3e-07
	train	(1e-03,1e-04)	0.143	22	20000	54	33	10:47	03:34	2.5e-06	7.8e-06	2.5e-06
	(200,3)	(5e-04,5e-05)	0.230	22	20000	265	29	10:45	12:57	2.4e-06	7.0e-06	2.4e-06

Chapter 5

Conclusions and future works

In this thesis, 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.

One limitation is that the proposed algorithm does not take into account the possible overlaps among groups. Future research is therefore needed to develop an efficient second order information based algorithm for solving overlapping sparse group Lasso problems. The major difficulty is that we still failed to find the closed form expression of the proximal mapping associated with the ℓ_2 norm for groups (i.e., $\phi(x) := \sum_{l=1}^{g} \lambda_{2,l}\|x_{G_l}\|$ in Chapter 4). Because the groups are possibly overlapped, the optimization problem in the proximal mapping of ϕ cannot be decoupled into several solvable problems. Such a closed form expression is not only critical for the derivation of the generalized Jacobian in our algorithmic framework, but also can contribute to many first order methods. Moreover, second order information based algorithms for solving other large-scale convex composite problems may also deserve further explorations.

In the second part of the thesis, we have applied a proximal point algorithm to the primal formulations of the group graphical Lasso and the fused graphical Lasso problems. From a theoretical standpoint, we have showed that the proposed method is globally convergent, the sequence of the primal iterates is Q-linearly convergent, and the sequence of the dual objective values is R-linearly convergent. Numerically, we have demonstrated the superior efficiency and robust performance of the proposed method by comparing it with the extensively used alternating direction method of multipliers and the proximal Newton-type method on both synthetic and real data sets. In summary, the proposed semismooth Newton based proximal point algorithm is a highly efficient method for solving both the group graphical Lasso and fused graphical Lasso problems.

Base on the superior performance of the PPA for solving the group graphical Lasso and fused graphical Lasso problems, we can apply our algorithmic framework for solving multiple graphical Lasso problems with other penalties in future studies.

Bibliography

[1] A. Ahmed and E. P. Xing, *Recovering time-varying networks of dependencies in social and biological studies*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 11878–11883.

[2] G. Andrew and J. Gao, *Scalable training of L_1-regularized log-linear models*, in Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 33–40.

[3] 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).

[4] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont, *Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data*, Journal of Machine learning research, 9 (2008), pp. 485–516.

[5] O. Banerjee, L. E. Ghaoui, A. d'Aspremont, and G. Natsoulis, *Convex optimization techniques for fitting sparse gaussian graphical models*, in Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006, pp. 89–96.

[6] J. F. Bonnans and A. Shapiro, *Perturbation analysis of optimization problems*, Springer Science & Business Media, 2013.

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

[8] 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 (2011), pp. 1–122.

[9] 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 (2016), pp. 435–467.

[10] A. Cardoso-Cachopo, *Improving methods for single-label text categorization.* PhD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa, 2007.

[11] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, 2 (2011), pp. 27:1–27:27.

[12] 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 (2017), pp. 237–270.

[13] 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 (2003), pp. 39–56.

[14] F. H. Clarke, *Optimization and Nonsmooth Analysis*, Wiley, New York, 1983.

[15] 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).

[16] ——, *On the R-superlinear convergence of the KKT residues generated by the augmented Lagrangian method for convex composite conic programming*, Mathematical Programming, (2018), pp. 1–35.

[17] P. Danaher, P. Wang, and D. M. Witten, *The joint graphical lasso for inverse covariance estimation across multiple classes*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76 (2014), pp. 373–397.

[18] A. d'Aspremont, O. Banerjee, and L. El Ghaoui, *First-order methods for sparse covariance selection*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 56–66.

[19] 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 (2017), pp. 225–258.

[20] A. P. Dempster, *Covariance selection*, Biometrics, 28 (1972), pp. 157–175.

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

[22] J. Duchi, S. Gould, and D. Koller, *Projected subgradient methods for learning sparse gaussians*, in Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2008, pp. 153–160.

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

[24] F. Facchinei, *Minimization of SC^1 functions and the maratos effect*, Operations Research Letters, 17 (1995), pp. 131–138.

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

[26] J. Fan and J. Lv, *A selective overview of variable selection in high dimensional feature space*, Statistica Sinica, 20 (2010), pp. 101–148.

[27] Y. Fan and C. Y. Tang, *Tuning parameter selection in high dimensional penalized likelihood*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 75 (2013), pp. 531–552.

[28] 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 (2013), pp. 946–977.

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

[30] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical lasso*, Biostatistics, 9 (2008), pp. 432–441.

[31] ——, *A note on the group lasso and a sparse group lasso*, arXiv preprint arXiv:1001.0736, (2010).

[32] 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 (1976), pp. 17–40.

[33] A. J. Gibberd and J. D. Nelson, *Regularized estimation of piecewise constant gaussian graphical models: The group-fused graphical lasso*, Journal of Computational and Graphical Statistics, 26 (2017), pp. 623–634.

[34] 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 (1975), pp. 41–76.

[35] J. Guo, E. Levina, G. Michailidis, and J. Zhu, *Joint estimation of multiple graphical models*, Biometrika, 98 (2011), pp. 1–15.

[36] 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, ACM, 2015, pp. 387–396.

[37] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, *Network inference via the time-varying graphical lasso*, in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 205–213.

[38] S. Hara and T. Washio, *Learning a common substructure of multiple graphical gaussian models*, Neural Networks, 38 (2013), pp. 23–38.

[39] J.-B. Hiriart-Urrutty and C. Lemaréchal, *Convex analysis and minimization algorithms II: Advanced Theory and Bundle Methods*, vol. 306, Springer-Verlag, 1993.

[40] J. Honorio and D. Samaras, *Multi-task learning of gaussian graphical models*, in Proceedings of the 27th International Conference on Machine Learning, ACM, 2010, pp. 447–454.

[41] C.-J. Hsieh, I. S. Dhillon, P. K. Ravikumar, and M. A. Sustik, *Sparse inverse covariance matrix estimation using quadratic approximation*, in Advances in Neural Information Processing Systems, 2011, pp. 2330–2338.

[42] 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, 2010, pp. 883–891.

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

[44] 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, ACM, 2010, pp. 487–494.

[45] 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 (1996), pp. 437–472.

[46] J. Kim and H. Park, *Fast active-set-type algorithms for l_1-regularized linear regression*, in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, 2010, pp. 397–404.

[47] 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, 2013, pp. 379–388.

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

[49] ——, *Newton's method based on generalized derivatives for nonsmooth functions: convergence analysis*, in Advances in Optimization, vol. 382, Springer, 1992, pp. 171–194.

[50] X. Y. Lam, J. Marron, D. F. Sun, and K.-C. Toh, *Fast algorithms for large-scale generalized distance weighted discrimination*, Journal of Computational and Graphical Statistics, (2018), pp. 1–12.

[51] J. D. Lee, Y. Sun, and M. A. Saunders, *Proximal Newton-type methods for minimizing composite functions*, SIAM Journal on Optimization, 24 (2014), pp. 1420–1443.

[52] H. Li and J. Gui, *Gradient directed regularization for sparse gaussian concentration graphs, with applications to inference of genetic networks*, Biostatistics, 7 (2006), pp. 302–317.

[53] L. Li and K.-C. Toh, *An inexact interior point method for ℓ_1-regularized sparse covariance selection*, Mathematical Programming Computation, 2 (2010), pp. 291–315.

[54] 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 (2018), pp. 433–458.

[55] ——, *On efficiently solving the subproblems of a level-set method for fused lasso problems*, SIAM Journal on Optimization, 28 (2018), pp. 1842–1862.

[56] M. Lichman, *UCI machine learning repository*, 2013.

[57] J. Liu, S. Ji, and J. Ye, *SLEP: Sparse learning with efficient projections*, Arizona State University, (2009).

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

[59] Z. Lu, *Smooth optimization approach for sparse covariance selection*, SIAM Journal on Optimization, 19 (2009), pp. 1807–1827.

[60] ——, *Adaptive first-order methods for general sparse inverse covariance selection*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2000–2016.

[61] Z. Lu and Y. Zhang, *An augmented Lagrangian approach for sparse principal component analysis*, Mathematical Programming, 135 (2012), pp. 149–193.

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

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

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

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

[66] R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana, *Estimating time-varying brain connectivity networks from functional MRI time series*, NeuroImage, 103 (2014), pp. 427–443.

[67] J.-J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bulletin de la Société Mathématique de France, 93 (1965), pp. 273–299.

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

[69] J.-S. Pang and L. Qi, *A globally convergent Newton method for convex SC^1 minimization problems*, Journal of Optimization Theory and Applications, 85 (1995), pp. 633–648.

[70] 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 (2010), pp. 53–77.

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

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

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

[74] R. T. Rockafellar, *Convex analysis*, Princeton University Press, Princeton, NJ, 1970.

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

[76] ——, *Monotone operators and the proximal point algorithm*, SIAM Journal on Control and Optimization, 14 (1976), pp. 877–898.

[77] A. Shapiro, *On concepts of directional differentiability*, Journal of optimization theory and applications, 66 (1990), pp. 477–487.

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

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

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

[81] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society. Series B (Methodology), 58 (1996), pp. 267–288.

[82] 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 (2005), pp. 91–108.

[83] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Mathematical Programming, 117 (2009), pp. 387–423.

[84] G. Varoquaux, A. Gramfort, J.-B. Poline, and B. Thirion, *Brain covariance selection: better individual functional connectivity models using population prior*, in Advances in Neural Information Processing Systems, 2010, pp. 2334–2342.

[85] C. J. Wang, D. F. Sun, and K.-C. Toh, *Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm*, SIAM Journal on Optimization, 20 (2010), pp. 2994–3013.

[86] J. Yang and J. Peng, *Estimating time-varying graphical models*, arXiv preprint arXiv:1804.03811, (2018).

[87] J. F. Yang, D. F. Sun, and K.-C. Toh, *A proximal point algorithm for log-determinant optimization with group lasso regularization*, SIAM Journal on Optimization, 23 (2013), pp. 857–893.

[88] S. Yang, Z. Lu, X. Shen, P. Wonka, and J. Ye, *Fused multiple graphical lasso*, SIAM Journal on Optimization, 25 (2015), pp. 916–943.

[89] K. Yosida, *Functional analysis*, (1964).

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

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

[92] 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 (2006), pp. 49–67.

[93] ——, *Model selection and estimation in the Gaussian graphical model*, Biometrika, 94 (2007), pp. 19–35.

[94] X. Yuan, *Alternating direction method for covariance selection models*, Journal of Scientific Computing, 51 (2012), pp. 261–273.

[95] S. Yun, P. Tseng, and K.-C. Toh, *A block coordinate gradient descent method for regularized convex separable optimization and covariance selection*, Mathematical programming, 129 (2011), pp. 331–355.

[96] 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 (2013), pp. 163–186.

[97] Y. Zhang, N. Zhang, D. F. Sun, and K.-C. Toh, *An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems*, Mathematical Programming, (2019,(DOI:10.1007/s10107-018-1329-6)).

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

[99] Y. Zhou, J. Han, X. Yuan, Z. Wei, and R. Hong, *Inverse sparse group lasso model for robust object tracking*, IEEE Transactions on Multimedia, 19 (2017), pp. 1798–1810.

[100] Z. Zhou and A. M.-C. So, *A unified approach to error bounds for structured convex optimization problems*, Mathematical Programming, 165 (2017), pp. 689–728.

EFFICIENT HESSIAN BASED ALGORITHMS FOR SOLVING SPARSE GROUP LASSO AND MULTIPLE GRAPHICAL LASSO PROBLEMS

ZHANG YANGJING

NATIONAL UNIVERSITY OF SINGAPORE

2019

Efficient Hessian based algorithms for solving sparse group Lasso and multiple graphical Lasso problems   Zhang Yangjing   2019