SIAM J. MATH. DATA SCI.
Vol. 3, No. 2, pp. 524–543

An Efficient Linearly Convergent Regularized Proximal Point Algorithm for Fused Multiple Graphical Lasso Problems[*]

Ning Zhang[†], Yangjing Zhang[‡], Defeng Sun[§], and Kim-Chuan Toh[¶]

Abstract. Nowadays, analyzing data from different classes or over a temporal grid has attracted a great deal of interest. As a result, various multiple graphical models for learning a collection of graphical models simultaneously have been derived by introducing sparsity in graphs and similarity across multiple graphs. This paper focuses on the fused multiple graphical Lasso model, which encourages not only shared pattern of sparsity but also shared values of edges across different graphs. For solving this model, we develop an efficient regularized proximal point algorithm, 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 derive an explicit expression for the generalized Jacobian of the proximal mapping of the fused multiple graphical Lasso regularizer. Unlike those widely used first order methods, our approach has heavily exploited the underlying second order information through the semismooth Newton method. This not only can accelerate the convergence of the algorithm but also can improve its robustness. The efficiency and robustness of our proposed algorithm are demonstrated by comparing it with some state-of-the-art methods on both synthetic and real data sets.

Key words. fast linear convergence, network estimation, semismooth Newton method, sparse Jacobian

AMS subject classifications. 90C22, 90C25, 90C31, 62J10

DOI. 10.1137/20M1344160

1. Introduction.

Undirected graphical models have been especially popular for learning conditional independence structures among a large number of variables where the observations are drawn independently and identically from the same distribution. The Gaussian graphical model is one of the most widely used undirected graphical models. In the high-dimensional and low-sample-size settings, it is always assumed that the conditional independence structure or the precision matrix is sparse in a certain sense. In other words, its corresponding undirected

[*]Received by the editors June 9, 2020; accepted for publication (in revised form) February 10, 2021; published electronically April 21, 2021.
https://doi.org/10.1137/20M1344160
Funding: The work of the first author was funded by the National Natural Science Foundation of China (11901083). The work of the third author was funded by Hong Kong Research Grant Council grant PolyU 153014/18P. The work of the fourth author was funded by the Ministry of Education, Singapore, Academic Research Fund (grant R-146-000-257-112).

[†]School of Computer Science and Technology, Dongguan University of Technology, Dongguan 523808, China (zhangning@dgut.edu.cn).

[‡]Corresponding author. Department of Mathematics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 (zhangyangjing@u.nus.edu).

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

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

graph is expected to be sparse. To promote sparsity, there has been a great deal of interest in using the ℓ_1 norm penalty in statistical applications [2, 6]. In many conventional applications, a single Gaussian graphical model is typically enough to capture the conditional independence structure of the random variables. However, due to the heterogeneity or similarity of the data involved, it is increasingly appealing to fit a collection of such models jointly, such as inferring the time-varying networks and finding the change-points [1, 7, 9, 18, 25] and estimating multiple precision matrices simultaneously for variables from distinct but related classes [3, 8, 27].

Multiple graphical models refer to the models that can estimate a collection of precision matrices jointly. Specifically, let $\Delta^{(l)}$ be L random vectors (from different classes or over a temporal grid) drawn independently from different distributions $\mathcal{N}_p(\mu^{(l)},\Sigma^{(l)})$, $l=1,\dots,L$, $L\geq 2$. Assume that the multivariate random variable $\Delta^{(l)}$ has N_l observations $\delta_1^{(l)},\delta_2^{(l)},\dots,\delta_{N_l}^{(l)}$ for each $l\in\{1,\dots,L\}$. 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 $S^{(l)}=\frac{1}{N_l-1}\sum_{i=1}^{N_l}(\delta_i^{(l)}-\bar{\mu}^{(l)})(\delta_i^{(l)}-\bar{\mu}^{(l)})^T$, $l=1,\dots,L$. The multiple graphical model for estimating the precision matrices $(\Sigma^{(l)})^{-1}$, $l=1,\dots,L$, jointly is the model with the variable $\Theta=(\Theta^{(1)},\dots,\Theta^{(L)})\in\mathbb{S}^p\times\cdots\times\mathbb{S}^p$:

$$\min_{\Theta}\ \sum_{l=1}^{L}\left(-\log\det\Theta^{(l)}+\langle S^{(l)},\Theta^{(l)}\rangle\right)+\mathcal{P}(\Theta), \tag{1.1}$$

where $\mathcal{P}$ is a penalty function, which usually promotes sparsity in each $\Theta^{(l)}$ and similarities among different $\Theta^{(l)}$'s. Various penalties have been considered in the literature [1, 3, 7, 8, 18, 27].

In this paper, we focus on the following fused graphical Lasso (FGL) regularizer which was used by [1] and [27]:

$$\mathcal{P}(\Theta)=\lambda_1\sum_{l=1}^{L}\sum_{i\neq j}|\Theta_{ij}^{(l)}|+\lambda_2\sum_{l=2}^{L}\sum_{i\neq j}|\Theta_{ij}^{(l)}-\Theta_{ij}^{(l-1)}|. \tag{1.2}$$

We refer to problem (1.1) with the FGL regularizer $\mathcal{P}$ in (1.2) as the FGL problem. The FGL regularizer is in some sense a generalized fused Lasso regularizer [22]. It applies the ℓ_1 penalty to all the off-diagonal elements of the L precision matrices and the consecutive differences of the elements of successive precision matrices. Many elements with the same indices in the estimated matrices $\Theta^{(1)},\dots,\Theta^{(L)}$ will be similar or even identical when the parameter λ_2 is large enough. Therefore, the FGL regularizer encourages not only a shared pattern of sparsity but also shared values across different graphs. Throughout this paper, we assume that the optimal solution set of the FGL problem is nonempty. In fact, it was proved in [27] that the FGL problem has a unique optimal solution when the diagonal vector of each $S^{(l)}$ is positive, i.e., $\operatorname{diag}(S^{(l)})>0$, $l=1,\dots,L$.

Existing algorithms for solving the 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) [3, 7, 9]. Another algorithm for this class of problems is the forward-backward splitting (FBS) method [23]. Additionally, a proximal Newton-type method [11, 12] was implemented in [27] for solving the FGL problem. As we know, ADMM and FBS could be practical first order methods for finding approximate solutions of low or moderate accuracy. However, they hardly utilize any second order information, which generally must be used in order to obtain highly accurate solutions. Although the proximal Newton-type method does incorporate some forms of second order information, a complicated quadratic approximation problem has to be solved in each iteration, and this computation is usually time-consuming. Additionally, though the proximal Newton-type method is globally convergent, it is not known whether its local linear convergence can be guaranteed [27, p. 931]. It is worth mentioning that the regularizers are often introduced to promote certain structures in the estimated precision matrices, and the trade-off between biases and variances in the resulting estimators is controlled by the regularization parameters [4]. In practice, however, it is extremely hard to find the optimal regularization parameters. Therefore, a sequence of regularization parameters is applied in practice, and consequently, a sequence of corresponding optimization problems must be solved [5]. Under such a circumstance, a highly efficient and robust algorithm for solving the FGL model becomes particularly important.

In this paper, we will design a semismooth Newton (SSN) based regularized proximal point algorithm (rPPA) for solving the FGL problem, which is inspired by the work [16], where they have convincingly demonstrated the superior numerical performance of the SSN based augmented Lagrangian method (ALM), known as Ssnal, for solving the fused Lasso problem [22]. Thanks to the fact that the FGL problem has close connections to the fused Lasso problem, many of the virtues and theoretical insights of the Ssnal for solving the fused Lasso problem can be observed in our approach. However, we should emphasize that solving the FGL problem is much more challenging than solving the fused Lasso problem. Specifically, the difficulties are mainly due to the log-determinant function $\log\det(\cdot)$ and the matrix variables, as described below.

(a) Unlike the simple quadratic functions in the fused Lasso problem, the function $\log\det(\cdot)$ is defined on the space of positive definite matrices. Therefore, the FGL model requires the positive definiteness of their solutions. Besides, the gradient of the log-determinant function involves the inverse of a matrix. These greatly increase the difficulty and complexity of theoretical analysis and numerical implementation for rPPA as well as first order methods [3, 7, 9, 23] and proximal Newton-type method [27].

(b) An efficiently computable element in the generalized Jacobian of the proximal mapping of the fused Lasso regularizer is constructed in [16], which is an essential step for solving the fused Lasso problem. Based on the constructions, we could obtain an efficiently computable generalized Jacobian of the proximal mapping of the FGL regularizer. However, this process needs more complicated manipulations of coordinates for a collection of matrix variables, unlike the vector case of the fused Lasso problem.

The remaining parts of this paper are as follows. Section 2 presents preliminary results of proximal mappings. In section 3, we present a semismooth Newton based regularized proximal point algorithm for solving the FGL problem and its convergence properties. The numerical performance of our proposed algorithm on time-varying stock price data sets and categorical text data sets are evaluated in section 4. Section 5 gives the conclusion.

Notation. $\mathbb{S}^p_+$ ($\mathbb{S}^p_{++}$) denotes the cone of positive semidefinite (definite) matrices in the space of $p \times p$ real symmetric matrices $\mathbb{S}^p$. For any A, $B \in \mathbb{S}^p$, we denote $A \succeq B$ ($A \succ B$) if $A - B \in \mathbb{S}^p_+$ ($A - B \in \mathbb{S}^p_{++}$). In particular, $A \succeq 0$ ($A \succ 0$) indicates $A \in \mathbb{S}^p_+$ ($A \in \mathbb{S}^p_{++}$). We let $\mathcal{X} := \mathbb{S}^p_+ \times \cdots \times \mathbb{S}^p_+$ and $\mathcal{Y} := \mathbb{S}^p \times \cdots \times \mathbb{S}^p$ be the Cartesian product of L positive semidefinite cones $\mathbb{S}^p_+$ and that of L spaces of symmetric matrices $\mathbb{S}^p$, respectively. $\mathbb{R}^n$ denotes the n-dimensional Euclidean space, and $\mathbb{R}^{m\times n}$ denotes the set of all $m \times n$ real matrices. 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. For any $x \in \mathbb{R}^n$, $\|x\|_1 := \sum_{i=1}^n |x_i|$ and $\|x\| := \sqrt{\sum_{i=1}^n |x_i|^2}$. 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. For any matrix $A \in \mathbb{R}^{m\times n}$, A_{ij} denotes the (i, j)th element of A. For any $X := (X^{(1)}, \dots, X^{(L)}) \in \mathcal{Y}$, $X_{[ij]} := [X^{(1)}_{ij}; \dots; X^{(L)}_{ij}] \in \mathbb{R}^L$ denotes the column vector obtained by taking out the (i, j)th elements across all L matrices $X^{(l)}$, $l = 1, \dots, L$. $\mathrm{Diag}(D_1, \dots, D_n)$ denotes the block diagonal matrix whose ith diagonal block is the matrix D_i, $i = 1, \dots, n$. 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 two sequences of numbers $\{a_n\}$ and $\{b_n\}$, we say $a_n = \mathcal{O}(b_n)$ if there exists some positive constant c such that $|a_n| \le c|b_n|$ for sufficiently large n.

2. Preliminaries.

We will first present the properties related to the proximal mappings associated with the log-determinant function and the FGL regularizer since they are the building blocks for the algorithmic design in what follows. Let $\mathcal{E}$ be a finite-dimensional real Hilbert space, and let $\Xi : \mathcal{E} \to \mathbb{R} \cup \{+\infty\}$ be a proper and closed convex function. The Moreau–Yosida regularization [19, 28] of Ξ is defined by

$$\Psi_\Xi(u) := \min_{u'} \left\{\Xi(u') + \tfrac{1}{2}\|u' - u\|^2\right\} \ \forall\, u \in \mathcal{E}. \tag{2.1}$$

The proximal mapping associated with Ξ is the unique minimizer of (2.1) defined by

$$\mathrm{Prox}_\Xi(u) := \arg\min_{u'} \left\{\Xi(u') + \tfrac{1}{2}\|u' - u\|^2\right\} \ \forall\, u \in \mathcal{E}. \tag{2.2}$$

Moreover, $\Psi_\Xi(\cdot)$ is a continuously differentiable convex function [13, 21] with the gradient

$$\nabla\Psi_\Xi(u) = u - \mathrm{Prox}_\Xi(u) \ \forall\, u \in \mathcal{E}. \tag{2.3}$$

2.1. Properties of log-determinant function.

For notational convenience, define $\vartheta : \mathbb{S}^p \to \mathbb{R}\cup\{+\infty\}$ as follows: $\vartheta(A) = -\log\det\, A$ if $A \in \mathbb{S}^p_{++}$; $\vartheta(A) = +\infty$ otherwise. The properties of Prox_ϑ have been extensively studied [24, 26]; they need the following definitions of two scalar functions for given $\beta > 0$: $\phi^+_\beta(x) := (\sqrt{x^2 + 4\beta} + x)/2$, $\phi^-_\beta(x) := (\sqrt{x^2 + 4\beta} - x)/2$ $\forall\, x \in \mathbb{R}$. In addition, the matrix counterparts of these two scalar functions can be defined by

$$\phi^+_\beta(A) := Q\mathrm{Diag}(\phi^+_\beta(d_1), \dots, \phi^+_\beta(d_p))Q^T, \quad \phi^-_\beta(A) := Q\mathrm{Diag}(\phi^-_\beta(d_1), \dots, \phi^-_\beta(d_p))Q^T \tag{2.4}$$

for any $A \in \mathbb{S}^p$ with its eigenvalue decomposition $A = Q\mathrm{Diag}(d_1, d_2, \dots, d_p)Q^T$, where $d_1 \ge d_2 \ge \cdots \ge d_p$. 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 $A \in \mathbb{S}^p$. The next proposition gives the formulae of $\mathrm{Prox}_{\beta\vartheta}(\cdot)$, $\Psi_{\beta\vartheta}(\cdot)$, and the directional derivative of $\mathrm{Prox}_{\beta\vartheta}(\cdot)$ [24, Lemma 2.1(b)], [26, Proposition 2.3].

Proposition 2.1. *For any $A \in \mathbb{S}^p$, let $\phi_\beta^+(A)$ be defined by (2.4). Then the following hold:*

(a) $\mathrm{Prox}_{\beta\vartheta}(A) = \phi_\beta^+(A)$ *and* $\Psi_{\beta\vartheta}(A) = -\beta \log\det(\phi_\beta^+(A)) + \frac{1}{2}\|\phi_\beta^-(A)\|^2$.

(b) *The function* $\phi_\beta^+ : \mathbb{S}^p \to \mathbb{S}^p$ *is continuously differentiable, and its directional derivative* $(\phi_\beta^+)'(A)[B]$ *at A for any $B \in \mathbb{S}^p$ is given by* $(\phi_\beta^+)'(A)[B] = Q[\Gamma \odot (Q^T BQ)]Q^T$, *where* $A = Q\mathrm{Diag}(d_1, d_2, \ldots, d_p)Q^T$, $d_1 \geq d_2 \geq \cdots \geq d_p$, *and* $\Gamma \in \mathbb{S}^p$ *is defined by* $\Gamma_{ij} = (\phi_\beta^+(d_i) + \phi_\beta^+(d_j))/(\sqrt{d_i^2 + 4\beta} + \sqrt{d_j^2 + 4\beta})$, $i, j = 1, 2, \ldots, p$.

2.2. Properties of FGL regularizer.

In this section, we analyze the proximal mapping of the regularizer $\mathcal{P}$ defined by (1.2). For any $\Theta \in \mathcal{Y}$, one might observe that the penalty term $\mathcal{P}(\Theta)$ merely 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

$$\mathcal{P}(\Theta) = \textstyle\sum_{i\neq j} \varphi(\Theta_{[ij]}) \text{ with } \varphi(x) = \lambda_1\|x\|_1 + \lambda_2\|Bx\|_1 \ \forall\, x \in \mathbb{R}^L. \tag{2.5}$$

Here, the function φ is the fused Lasso regularizer [22], and the matrix $B \in \mathbb{R}^{(L-1)\times L}$ is defined by $Bx = [x_1 - x_2; \ldots; x_{L-1} - x_L]\ \forall\, x \in \mathbb{R}^L$. From (2.5), one can compute $\mathrm{Prox}_\mathcal{P}$ as follows: for any $X \in \mathcal{Y}$, $(\mathrm{Prox}_\mathcal{P}(X))_{[ij]} = \mathrm{Prox}_\varphi(X_{[ij]})$ if $i \neq j$; $(\mathrm{Prox}_\mathcal{P}(X))_{[ij]} = X_{[ij]}$ if $i = j$. The formulae for the proximal mapping Prox_φ and its generalized Jacobian $\widehat{\partial}\mathrm{Prox}_\varphi$ have been derived in [16] and will be summarized in section SM2.

Define a multifunction $\widehat{\partial}\mathrm{Prox}_\mathcal{P}(X) : \mathcal{Y} \rightrightarrows \mathcal{Y}$, the surrogate generalized Jacobian of $\mathrm{Prox}_\mathcal{P}$ at X, as follows:

$$\begin{cases} \mathcal{W} \in \widehat{\partial}\mathrm{Prox}_\mathcal{P}(X) \text{ if and only if there exist } M^{(ij)} \in \widehat{\partial}\mathrm{Prox}_\varphi(X_{[ij]}),\ i < j, \\ \text{such that } (\mathcal{W}[Y])_{[ij]} = \begin{cases} M^{(ij)}Y_{[ij]} & \text{if } i < j, \\ Y_{[ii]} & \text{if } i = j, \quad i, j = 1, \ldots, p, \quad \forall\, Y \in \mathcal{Y}. \\ M^{(ji)}Y_{[ij]} & \text{if } j < i. \end{cases} \end{cases} \tag{2.6}$$

From [16, Theorem 1], one can obtain the following theorem, which justifies why $\widehat{\partial}\mathrm{Prox}_\mathcal{P}(X)$ in (2.6) can be used as the surrogate generalized Jacobian of $\mathrm{Prox}_\mathcal{P}$ at X.

Theorem 2.2. *The surrogate generalized Jacobian* $\widehat{\partial}\mathrm{Prox}_\mathcal{P}(\cdot)$ *defined in (2.6) is a nonempty, compact valued, upper semicontinuous multifunction. Given any* $X \in \mathcal{Y}$, *any element in the set* $\widehat{\partial}\mathrm{Prox}_\mathcal{P}(X)$ *is self-adjoint and positive semidefinite. Moreover, there exists a neighborhood* $\mathcal{U}_X$ *of X such that for all* $Y \in \mathcal{U}_X$, $\mathrm{Prox}_\mathcal{P}(Y) - \mathrm{Prox}_\mathcal{P}(X) - \mathcal{W}[Y - X] = 0\ \forall\, \mathcal{W} \in \widehat{\partial}\mathrm{Prox}_\mathcal{P}(Y)$.

3. Regularized proximal point algorithm.

In this section, we present a regularized proximal point algorithm (rPPA) for solving the following problem, which is essentially an equivalent form of problem (1.1):

$$\min_{\Theta,\,\Omega} \left\{ f(\Theta, \Omega) := \sum_{l=1}^{L} \left(\vartheta(\Omega^{(l)}) + \langle S^{(l)}, \Theta^{(l)} \rangle \right) + \mathcal{P}(\Theta) \,\middle|\, \Theta - \Omega = 0 \right\}. \tag{3.1}$$

Given positive scalars $\sigma_k \uparrow \sigma_\infty \leq \infty$, the kth iteration of the proximal point algorithm (PPA) for solving (3.1) is given by

$$(\Theta^{k+1}, \Omega^{k+1}) \approx \arg\min_{\Theta,\,\Omega} \Big\{ f(\Theta, \Omega) + \tfrac{1}{2\sigma_k}(\|\Theta - \Theta^k\|^2 + \|\Omega - \Omega^k\|^2) \,\big|\, \Theta - \Omega = 0 \Big\}, \tag{3.2}$$

which only requires an inexact solution. There are many ways to solve (3.2). Inspired by recent progress in solving large scale convex optimization problems [15, 16, 26, 29], we shall adopt the approach of solving (3.2) via applying a sparse SSN method to its dual. The Lagrangian function of problem (3.1) is given by $\mathcal{L}(\Theta,\Omega,X) = f(\Theta,\Omega) - \langle \Theta-\Omega,\, X\rangle\ \forall\,(\Theta,\Omega,X)\in\mathcal{X}\times\mathcal{X}\times\mathcal{Y}$. Then, the Lagrangian dual of (3.2) is given by $\sup_X\{\Phi_k(X) := \inf_{\Theta,\Omega}\{\mathcal{L}(\Theta,\Omega,X) + \frac{1}{2\sigma_k}(\|\Theta-\Theta^k\|^2 + \|\Omega-\Omega^k\|^2)\}\}$. By the definition of the Moreau–Yosida regularization (2.1), we can write $\Phi_k(\cdot)$ explicitly as follows:

$$\begin{aligned}\Phi_k(X) &= \inf_{\Theta}\left\{\mathcal{P}(\Theta) + \langle\Theta, S-X\rangle + \tfrac{1}{2\sigma_k}\|\Theta-\Theta^k\|^2\right\}\\ &\qquad + \textstyle\sum_{l=1}^{L}\inf_{\Omega^{(l)}}\left\{\vartheta(\Omega^{(l)}) + \langle\Omega^{(l)}, X^{(l)}\rangle + \tfrac{1}{2\sigma_k}\|\Omega^{(l)} - (\Omega^{(l)})^k\|^2\right\}\\ &= \tfrac{1}{\sigma_k}\Psi_{\sigma_k\mathcal{P}}(\Theta^k + \sigma_k(X-S)) + \textstyle\sum_{l=1}^{L}\tfrac{1}{\sigma_k}\Psi_{\sigma_k\vartheta}((\Omega^k)^{(l)} - \sigma_k X^{(l)})\\ &\quad - \tfrac{1}{2\sigma_k}\|\Theta^k + \sigma_k(X-S)\|^2 + \tfrac{1}{2\sigma_k}\|\Theta^k\|^2 - \textstyle\sum_{l=1}^{L}\left(\tfrac{1}{2\sigma_k}\|(\Omega^k)^{(l)} - \sigma_k X^{(l)}\|^2 - \tfrac{1}{2\sigma_k}\|(\Omega^{(l)})^k\|^2\right),\end{aligned}$$

where $\Psi_{\sigma_k\mathcal{P}}$ and $\Psi_{\sigma_k\vartheta}$ are the Moreau–Yosida regularizations of $\sigma_k\mathcal{P}$ and $\sigma_k\vartheta$, respectively. Therefore, by Proposition 2.1 and the definition of the proximal mapping (2.2), the kth iteration of PPA (3.2) can be written as

$$\begin{cases}\Theta^{k+1} = \operatorname{Prox}_{\sigma_k\mathcal{P}}(\Theta^k + \sigma_k(X^{k+1} - S)),\\ (\Omega^{(l)})^{k+1} = \operatorname{Prox}_{\sigma_k\vartheta}\big((\Omega^{(l)})^k - \sigma_k(X^{(l)})^{k+1}\big),\ l = 1,2,\dots,L,\end{cases}$$

where X^{k+1} approximately solves the following problem: $X^{k+1}\approx \arg\max_X\ \Phi_k(X)$. Since $\Phi_k(\cdot)$ is not strongly concave in general, we consider the following rPPA.

Algorithm 3.1. A regularized proximal point algorithm (rPPA) for solving (3.1).

```
Choose $\Theta^0$, $\Omega^0$, $X^0 \in \mathcal{X}$. Iterate the following steps for $k = 0, 1, 2, \dots$:
  Step 1. Compute
     (3.3)   $X^{k+1} \approx \arg\max_X \left\{\widehat{\Phi}_k(X) := \Phi_k(X) - \frac{1}{2\sigma_k}\|X - X^k\|^2\right\}.$
  Step 2. Compute $\Theta^{k+1} = \operatorname{Prox}_{\sigma_k\mathcal{P}}(\Theta^k + \sigma_k(X^{k+1} - S))$ and for $l = 1, \dots, L$,
     $(\Omega^{(l)})^{k+1} = \operatorname{Prox}_{\sigma_k\vartheta}((\Omega^{(l)})^k - \sigma_k(X^{(l)})^{k+1}) = \phi^{+}_{\sigma_k}((\Omega^{(l)})^k - \sigma_k(X^{(l)})^{k+1}).$
  Step 3. Update $\sigma_{k+1} \uparrow \sigma_\infty \le \infty$.
```

We will use the following standard stopping criteria [20] for the inner subproblem (3.3):

(A) $\|\nabla\widehat{\Phi}_k(X^{k+1})\| \ \le\ \varepsilon_k/\sigma_k,\ \varepsilon_k \ge 0,\ \sum_{k=0}^{\infty}\varepsilon_k < \infty;$

(B) $\|\nabla\widehat{\Phi}_k(X^{k+1})\| \ \le\ (\delta_k/\sigma_k)\|(\Theta^{k+1},\Omega^{k+1}) - (\Theta^k,\Omega^k)\|,\ \delta_k \ge 0, \sum_{k=0}^{\infty}\delta_k < \infty.$

The reason for using the above stopping criteria is due to the fact that Algorithm 3.1 is equivalent to the primal-dual PPA in the sense of [20]. Note that the stopping criteria (A) and (B) are certainly satisfied (with $\varepsilon_k = \delta_k = 0$) if X^{k+1} is an exact solution of the subproblem

(3.3). However, in practice, problem (3.3) can only be solved inexactly. Therefore, we allow for a certain inexactness controlled by ε_k or δ_k. The choice of ε_k or δ_k should consider the tradeoff between the convergence speed of Algorithm 3.1 and the cost of solving the subproblem (3.3). A smaller ε_k will require a more accurate solution X^{k+1} to the subproblem (3.3), and a better solution sequence of the subproblems will generally converge faster. Similarly, as one will see in Theorem 3.3, a smaller δ_k will give rise to a better convergence rate μ_k. However, the price we have to pay for choosing smaller ε_k and δ_k is the increase in the cost of solving more difficult subproblems (3.3). To meet the summable condition, we usually take $\{\varepsilon_k\}$ and $\{\delta_k\}$ to be convergent geometric sequences.

3.1. Semismooth Newton method for solving subproblem (3.3).

Note that the key issue in the implementation of rPPA for solving the FGL model is the computation of the subproblem solution in each rPPA iteration. Therefore, in this section, we design a semismooth Newton method (SSN), which is specifically described by Algorithm 3.2, to solve subproblem (3.3). From (2.3) and Proposition 2.1, we know that $\widehat{\Phi}_k$ is a continuously differentiable, strongly concave function and $\nabla\Phi_k(X) = -\mathrm{Prox}_{\sigma_k\mathcal{P}}\big(U_k(X)\big) + \big(\phi^+_{\sigma_k}(W_k^{(1)}(X)),\ldots,\phi^+_{\sigma_k}(W_k^{(L)}(X))\big)$, where $U_k(X) := \Theta^k + \sigma_k(X - S)$ and $W_k^{(l)}(X) := (\Omega^k)^{(l)} - \sigma_k X^{(l)}$, $l = 1,\ldots,L$. Therefore, one can obtain the unique solution to problem (3.3) by solving the nonsmooth system

$$\nabla\widehat{\Phi}_k(X) = \nabla\Phi_k(X) - (X - X^k)/\sigma_k = 0. \tag{3.4}$$

Recall that $\phi^+_{\sigma_k}(\cdot)$ is differentiable, and its derivative is given by Proposition 2.1. Thus, the surrogate generalized Jacobian $\widehat{\partial}(\nabla\Phi_k)(X)$ of $\nabla\Phi_k$ at X is defined as follows:

$$\begin{cases} \mathcal{V} \in \widehat{\partial}(\nabla\Phi_k)(X) \text{ if and only if there exists } \mathcal{G} \in \widehat{\partial}\mathrm{Prox}_{\mathcal{P}}(U_k(X)/\sigma_k) \text{ such that} \\ \mathcal{V}[D] = -\sigma_k\mathcal{G}[D] - \sigma_k\big((\phi^+_{\sigma_k})'(W_k^{(1)}(X))[D^{(1)}],\ldots,(\phi^+_{\sigma_k})'(W_k^{(L)}(X))[D^{(L)}]\big)\ \forall\, D \in \mathcal{Y}. \end{cases}$$

With the generalized Jacobian of $\nabla\Phi_k$, we are ready to solve (3.4) by the SSN method, where the Newton systems are solved inexactly by the conjugate gradient (CG) method. In addition, we derive the convergence result of the SSN method (Algorithm 3.2).

Theorem 3.1. *Let $\{X^j\}$ be the infinite sequence generated by Algorithm 3.2. Then $\{X^j\}$ converges globally to the unique optimal solution $\widehat{X}$ of (3.4). Furthermore, the local rate of convergence is of order $1+\tau$, where the parameter $\tau \in (0,1]$ is from Algorithm 3.2; i.e., for all j sufficiently large,*

$$\|X^{j+1} - \widehat{X}\| = \mathcal{O}(\|X^j - \widehat{X}\|^{1+\tau}). \tag{3.5}$$

In fact, (3.5) for sufficiently large j means that there is a small neighborhood around the solution point $\widehat{X}$ that, once the iterates are trapped in it, will converge to the solution point at least superlinearly (at the rate of order $1+\tau$), and typically only a few additional iterations are enough to achieve high accuracy. On the other hand, the global convergence stated in Theorem 3.1 can guarantee that the iterates will always enter into this neighborhood and enjoy fast convergence.

Algorithm 3.2. A semismooth Newton (SSN) method for solving (3.4).

```
Given $\mu \in (0, 1/2)$, $\bar{\eta} \in (0, 1)$, $\tau \in (0, 1]$, and $\rho \in (0, 1)$. Choose $X^0 \in \mathbb{S}^p_{++} \times \cdots \times \mathbb{S}^p_{++}$. Iterate
the following steps for $j = 0, 1, \ldots$:
 Step 1. (Newton direction) Choose one specific map $\mathcal{V}_j \in \widehat{\partial}(\nabla\Phi_k)(X^j)$. Apply the CG
         method to find an approximate solution $D^j$ to
             $(\mathcal{V}_j - \sigma_k^{-1} I)[D] = -\nabla\widehat{\Phi}_k(X^j)$
         such that $\|(\mathcal{V}_j - \sigma_k^{-1} I)[D^j] + \nabla\widehat{\Phi}_k(X^j)\| \le \min(\bar{\eta}, \|\nabla\widehat{\Phi}_k(X^j)\|^{1+\tau})$.
 Step 2. (Line search) Set $\alpha_j = \rho^{m_j}$, where $m_j$ is the smallest nonnegative integer $m$ for
         which
             $\widehat{\Phi}_k(X^j + \rho^m D^j) \ge \widehat{\Phi}_k(X^j) + \mu\rho^m \langle \nabla\widehat{\Phi}_k(X^j), D^j \rangle$.
 Step 3. Set $X^{j+1} = X^j + \alpha_j D^j$.
```

3.2. Convergence analysis of rPPA.

The Karush–Kuhn–Tucker (KKT) optimality conditions [10] for (3.1) are given as follows:

$$\Theta - \mathrm{Prox}_{\mathcal{P}}(\Theta + X - S) = 0, \ \Omega^{(l)} - \mathrm{Prox}_{\vartheta}(\Omega^{(l)} - X^{(l)}) = 0, \ l = 1, \ldots, L, \ \Theta - \Omega = 0.$$

Define an operator $\mathcal{T}_{\mathcal{L}}$ by $\mathcal{T}_{\mathcal{L}}(\Theta, \Omega, X) := \{(\Theta', \Omega', X') \,|\, (\Theta', \Omega', -X') \in \partial\mathcal{L}(\Theta, \Omega, X)\}$. Since the function $\vartheta(\cdot)$ is strictly convex, we know that there exists a unique KKT point, denoted by $(\overline{\Theta}, \overline{\Omega}, \overline{X})$, and $\mathcal{T}_{\mathcal{L}}^{-1}(0) = \{(\overline{\Theta}, \overline{\Omega}, \overline{X})\}$. In order to ensure the local linear convergence rate of rPPA, we propose the following proposition.

Proposition 3.2. *There exists a nonnegative scalar κ such that for some $\varrho > 0$ it holds that*

$$\|(\Theta, \Omega, X) - (\overline{\Theta}, \overline{\Omega}, \overline{X})\| \le \kappa\|\Delta\| \ \ \forall\, \Delta \in \mathcal{T}_{\mathcal{L}}((\Theta, \Omega, X)) \text{ and } \|\Delta\| \le \varrho.$$

Based on Proposition 3.2, we can obtain the following convergence theorem.

Theorem 3.3. *Let $\{(\Theta^k, \Omega^k, X^k)\}$ be an infinite sequence generated by Algorithm 3.1 under stopping criterion (A). Then the sequence $\{(\Theta^k, \Omega^k)\}$ converges globally to the unique solution $(\overline{\Theta}, \overline{\Omega})$ of (3.1), and the sequence $\{X^k\}$ converges globally to the unique solution $\overline{X}$ of the dual problem of (3.1). Furthermore, if criterion (B) is also executed in Algorithm 3.1, there exists $\bar{k} \ge 0$ such that for all $k \ge \bar{k}$, $\|(\Theta^{k+1}, \Omega^{k+1}, X^{k+1}) - (\overline{\Theta}, \overline{\Omega}, \overline{X})\| \le \mu_k \|(\Theta^k, \Omega^k, X^k) - (\overline{\Theta}, \overline{\Omega}, \overline{X})\|$, where the convergence rate $1 > \mu_k := [\kappa(\kappa^2 + \sigma_k^2)^{-1/2} + \delta_k]/(1 - \delta_k) \to \mu_\infty = \kappa(\kappa^2 + \sigma_\infty^2)^{-1/2}$ ($\mu_\infty = 0$ if $\sigma_\infty = \infty$) and the parameter κ is from Proposition 3.2.*

3.3. Main ideas of rPPA and its extensions.

To summarize, it has been proven that our rPPA for solving the FGL problem not only is globally convergent but also has a linear convergent guarantee, and the convergence rate can be arbitrarily fast by choosing a sufficiently large proximal penalty parameter. Moreover, the SSN method for solving each of the rPPA subproblems has been shown to be superlinearly convergent. Furthermore, the generalized Jacobian of the proximal mapping of the FGL regularizer $\widehat{\partial}\mathrm{Prox}_{\mathcal{P}}(\cdot)$, a critical part of SSN, inherits the structured sparsity (referred to as second order sparsity) from that of the fused

Lasso regularizer $\widehat{\partial}\mathrm{Prox}_{\varphi}(\cdot)$. 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. Thus, based on these excellent convergent properties and the novel exploitation of the second order sparsity, we can expect the SSN based rPPA for solving the FGL problem to be highly efficient.

As the rPPA can be highly efficient for solving the FGL problems, a natural question is whether we can extend the algorithm to models with other forms of penalties. In fact, the key elements needed for the practical implementation of the rPPA are the efficient computation of $\mathrm{Prox}_{\mathcal{P}}(\cdot)$ and explicit characterization of $\widehat{\partial}\mathrm{Prox}_{\mathcal{P}}(\cdot)$. Therefore, if they can be efficiently computed and characterized, rPPA can be easily adapted to models with other penalties. For example, it can be extended to the case of group graphical Lasso and pairwise FGL penalties proposed in [3]; see, e.g., [17] and [30].

4. Numerical experiments.

In this section, we compare the performance of our algorithm rPPA with the alternating direction method of multipliers (ADMM) and the proximal Newton-type method [27] (referred to as MGL) for which the solver is available at http://senyang.info/. Our MATLAB code is available at https://blog.nus.edu.sg/mattohkc/softwares/graphical-lasso.

In all of the tables, "P," "A," and "M" stand for rPPA, ADMM, and MGL, respectively; "nnz" denotes the number of nonzero entries in the solution Θ obtained by rPPA using the estimation $\mathrm{nnz} := \min\{k \mid \sum_{i=1}^{k} |\hat{x}_i| \geq 0.999\|\hat{x}\|_1\}$, where $\hat{x} \in \mathbb{R}^{p^2L}$ is the vector obtained via sorting all elements in Θ by magnitude in a descending order; and "density" denotes the quantity $\mathrm{nnz}/(p^2L)$. The time is displayed in the format of "hours:minutes:seconds," and the fastest method in terms of running time is highlighted in red. The errors presented in the tables are the residuals η_P for rPPA and η_A for ADMM; the error for MGL is Δ_M; η_P, η_A, and Δ_M will be defined in the subsequent section.

4.1. Implementation details.

We first give some key implementation details of Algorithm 3.1 (rPPA) and Algorithm 3.2 (SSN).

4.1.1. Stopping conditions.

In this part, we describe the measurement of the accuracy of an approximate optimal solution and the stopping criteria of the three methods. Since both rPPA and ADMM can generate primal and dual approximate solutions, we can assess the accuracy of their solutions by the relative KKT residuals and duality gap. 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 rPPA. Based on the KKT optimality conditions for (3.1), the accuracy of an approximate optimal solution (Θ, Ω, X) generated by rPPA (Algorithm 3.1) is measured by the following relative residuals:

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

Here, pobj_P and dobj_P are the primal and dual objective values achieved by rPPA. Likewise, the accuracy of an approximate optimal solution generated by ADMM is measured by a similar residual η_A, defined in section SM3. In our numerical experiments, we terminate rPPA if it satisfies the condition $\eta_P < \varepsilon$ for a given accuracy tolerance ε; the same is true for ADMM with the stopping condition $\eta_A < \varepsilon$ or the maximum number of iterations 20000 is reached. Note that the terminating condition for MGL is different. Let "pobj_P" and "pobj_M" be the primal objective function values computed by rPPA and MGL, respectively. MGL will be terminated when the relative difference of its objective value with respect to that obtained by rPPA is smaller than the given tolerance ε, i.e., $\Delta_M := (\text{pobj}_M - \text{pobj}_P)/(1 + |\text{pobj}_M| + |\text{pobj}_P|) < \varepsilon$, or the maximum number of iterations 1500 is reached. All methods are terminated within three hours.

4.1.2. Warm-start strategy of rPPA.

We adopt a warm-start strategy to initialize rPPA. That is, we first run ADMM (with identity matrices as starting points) for a fixed number of iterations to generate a good initial point to warm-start rPPA. We also stop ADMM as soon as the residual η_A of the computed iterate is less than 100ε. Note that such a warm-start strategy is sound since in the initial phase of rPPA, where the iterates are not close to the optimal solution (as measured by η_P), it is computationally wasteful to use the more expensive rPPA 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 ADMM to generate the approximate solutions until the residual η_P has been sufficiently reduced.

4.1.3. Adjustment of parameters.

The appropriate adjustment of the parameters in Algorithm 3.1 and Algorithm 3.2 can accelerate the rPPA greatly in practice. We first describe the adjustment of σ_k, ε_k, δ_k in Algorithm 3.1. As revealed by Theorem 3.3, σ_k will affect the local convergence rate $\mu_k (< 1)$. If σ_k is increased, then μ_k will become smaller, and thus the local convergence rate is improved. On the other hand, σ_k will affect the condition number of the Newton system in Step 1 of Algorithm 3.2. When σ_k is very large such that the term $\sigma_k^{-1} I$ is negligible, the condition number of the map $\mathcal{V}_j - \sigma_k^{-1} I$ will be large. In this case, one needs extra CG iterations for finding the Newton direction. Therefore, the choice of σ_k should consider the trade-off between the rate of convergence of Algorithm 3.1 and the cost of solving the linear systems for the Newton directions in Algorithm 3.2. Based on our preliminary experiments, we set the initial parameter $\sigma_0 = \max\{0.01, \min\{1, \lambda_1, 1/\|S\|\}\}$. We adjust σ_k according to the experience that a larger σ_k usually leads to a better dual feasibility. An empirical characterization of dual feasibility can be given by the residuals $\frac{\|\Theta - \text{Prox}_{\mathcal{P}}(\Theta + X - S)\|}{1+\|\Theta\|}$, $\max_{1\le l\le L}\{\frac{\|\Omega^{(l)} X^{(l)} - I\|}{1+\sqrt{p}}\}$, or their maximum, which is denoted as χ_k for the kth iteration. When the improvement on the dual feasibility after one iteration is too small, we will increase σ_k. Specifically, we use the following strategy to update σ_k: if $\chi_k/\chi_{k-1} > 0.6$, set $\sigma_{k+1} = \zeta\sigma_k$; else, set $\sigma_{k+1} = \sigma_k$, where the factor ζ is set to $\zeta = 2$ if $\sigma_k < 10^7$ and $\zeta = 1.3$ otherwise. On the other hand, we set the initial parameter in stopping criterion (A) to be $\varepsilon_0 = 0.5$ and decrease it by a ratio $\varsigma \ge 1$, i.e., $\varepsilon_{k+1} = \varepsilon_k/\varsigma$. We empirically set ς to be 1.06 if the kth subproblem has been solved efficiently within a reasonable time (the normal situation); otherwise, we set ς to be 1. Finally, we update δ_k in the same fashion as that for ε_k.

Next we describe the adjustment of the parameters μ, $\bar{\eta}$, τ, ρ in Algorithm 3.2. For computing the Newton direction, we set $\bar{\eta} = 1$ if $k < 2$ (main rPPA iteration) and $j < 5$ (SSN iteration for solving the subproblem); otherwise, $\bar{\eta} = 0.1$. The choice of τ should consider the trade-off between the rate of convergence of Algorithm 3.2 and the cost of solving the linear systems for the Newton directions. As mentioned in [31], in order to achieve the quadratic convergence of SSN (i.e., $\tau = 1$), many more CG iterations will be needed to compute an accurate approximate Newton direction. Therefore, we usually take $\tau \in [0.1, 0.2]$ in practice. In the step on line search, we set $\mu = 10^{-4}$ and $\rho = 0.5$.

4.2. Numerical results. We now show the numerical performance on both synthetic and real data sets. Here, the real data sets include the stock price of `Standard & Poor's 500` and the `University Webpages` data set. Moreover, the experiments on a popular text data set named `20 Newsgroups` are given in section SM8.

4.2.1. Nearest-neighbor networks. In this section, we assess the effectiveness of the FGL model on a simulated network—the nearest-neighbor network. We set $p = 500$ and $L = 3$ and consider $N = 20p$, p, or $p/2$ observations. The nearest-neighbor network is generated by modifying the data generation mechanism described in [14]; see section SM5 for details.

There is a pair of tuning parameters λ_1 and λ_2 which must be specified. 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. In order to show the diversity of sparsity in our experiments, we choose a variety of λ_1 with λ_2 fixed. Figure 1 shows the relative ability of the FGL model to recover the network structures and to detect the change-points.

Figure 1(a) displays the number of true positive edges selected (i.e., TP edges) against the number of false edges selected (i.e., FP edges). We say that an edge (i, j) in the lth network is selected in the estimate $\widehat{\Theta}^{(l)}$ if $\widehat{\Theta}_{ij}^{(l)} \neq 0$, and we say that the edge is true in the precision matrix $\Omega^{(l)}$ if $\Omega_{ij}^{(l)} \neq 0$ and false if $\Omega_{ij}^{(l)} = 0$. The figure clearly shows that when more observations are sampled, the networks are better recovered by the FGL model. When the sample size $N = 10000$ is large enough, we can see that the FGL model with $\lambda_2 = 0.005$ can recover almost all of the true positive edges without false positive edges, and that the similarity control parameter $\lambda_2 = 0.005$ is much better than $\lambda_2 = 0.05$ in terms of the accuracy of true edge detection. When $\lambda_2 = 0.05$, $N = 10000$, 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 the case of enough samples.

Figure 1(b) illustrates the sum of squared errors between estimated edge values and true edge values, i.e., $\sum_{l=1}^{L} \sum_{i<j} \big(\widehat{\Theta}_{ij}^{(l)} - \Omega_{ij}^{(l)}\big)^2$. For sample size $N = 10000$, when the number of the total edges selected is increasing (i.e., the sparsity control parameter is decreasing), the errors are decreasing and finally reach fairly low values, which are much lower than those corresponding to sample sizes $N = 500$ and $N = 250$.

Figure 1(c) plots the number of true positive differential edges against false positive differential edges. 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 lth and the $(l + 1)$th networks if $|\widehat{\Theta}_{ij}^{(l)} - \widehat{\Theta}_{ij}^{(l+1)}| > 10^{-6}$, and we say that it is truly differential if $|\Omega_{ij}^{(l)} - \Omega_{ij}^{(l+1)}| > 10^{-6}$. The number of differential edges is computed for all successive pairs of networks. We find that the best result in Figure 1(c) is the one in the upper left corner which has approximately 2700 true positive differential edges and almost no false ones. We can also see that the red dashed curve has no false positive differential edge and small numbers of true

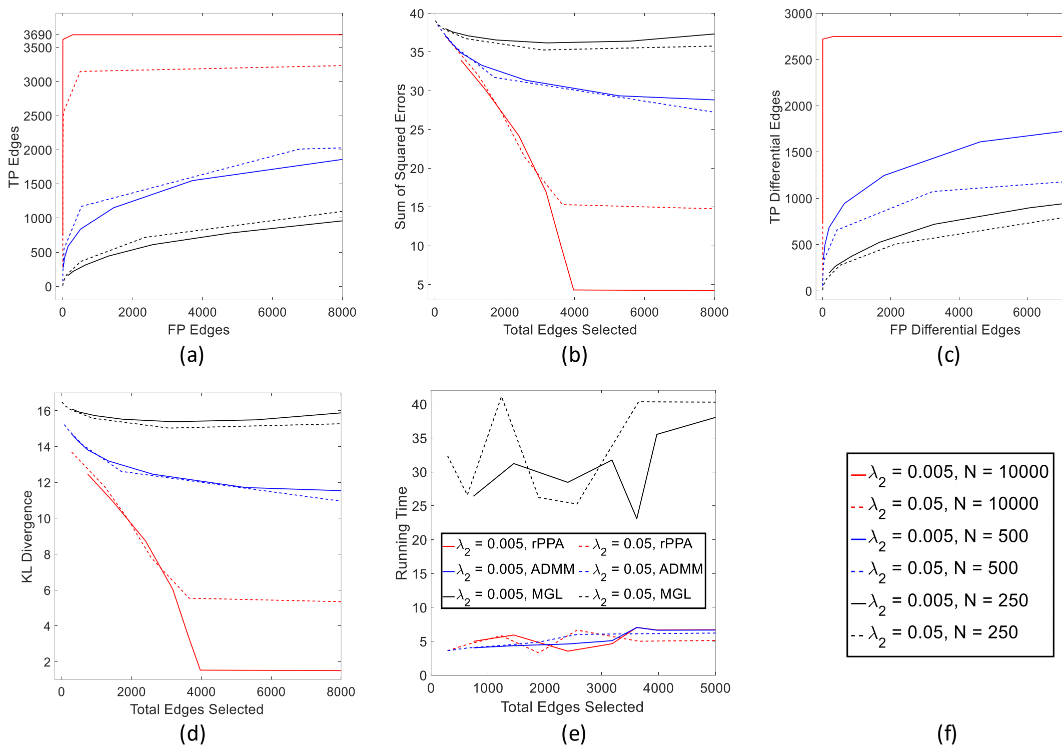

Figure 1. *Performances on nearest-neighbor networks with* $p = 500$*,* $L = 3$*, and various sample sizes* $N = 10000, 500, 250$*.* (a) *Number of edges correctly identified to be nonzero (true positive edges) versus number of edges incorrectly identified to be nonzero (false positive edges);* (b) *sum of squared errors in edge values versus the total number of edges estimated to be nonzero;* (c) *number of edges correctly found to have values differing between successive classes (true positive differential edges) versus number of edges incorrectly found to have values differing between successive classes (false positive differential edges);* (d) *KL divergence versus the total number of edges estimated to be nonzero;* (e) *running time of three methods when* $N = 10000$ *versus the total number of edges estimated to be nonzero;* (f) *legend for* (a)–(d)*.*

positive differential edges. This might be caused by the larger similarity control parameter $\lambda_2 = 0.05$, which forces an excessive number of edges across L networks to be similar. Again, with more samples, the recovery result would improve.

Figure 1(d) displays the Kullback–Leibler (KL) divergence between the true model and the estimated model: $\frac{1}{L}\sum_{l=1}^{L}(-\log\det(\widehat{\Theta}^{(l)}\Sigma^{(l)}) + \langle\widehat{\Theta}^{(l)}, \Sigma^{(l)}\rangle - p)$, where $\Sigma^{(l)}$, $l = 1,\dots,L$, are the true covariance matrices. It shows that the KL divergence deteriorates greatly with decreasing sample size. When the sample size is large enough such as $N = 10000$, the FGL model with appropriately chosen tuning parameters can attain a nearly zero KL divergence. When the sample size $N = 250$ is smaller than the number of features $p = 500$, we can see that the KL divergence decreases at first and then increases as the total number of edges selected increases, and we fail to attain a KL divergence value as low as those with the large sample size $N = 10000$.

Figure 1(e) plots the running times (in seconds) of the three methods for the case $N = 10000$. One can observe that the MGL is computationally more expensive than the ADMM and rPPA, while the latter two methods are comparably fast on these synthetic data instances.

In order to illustrate the performance on a relatively large L, we repeat the above experiments on simulations with $p = 500$, $L = 20$, and $N = 20p$, p, or $p/2$. The results are presented in Figure 2. Since the MGL is slow in solving nearest-neighbor network problems according to the numerical experiments conducted so far, we only report the running time of the rPPA and ADMM in Figure 2(e). The values in Figure 2 (TP Edges, FP Edges, Sum of Squared Errors, Total Edges Selected, TP Differential Edges, FP Differential Edges) are divided by $L = 20$ to improve readability. When L is increased from 3 to 20, the performances of the FGL model evaluated in Figure 2(a)–(d) remain similar, while we can see from Figure 2(e) that the running times of both rPPA and ADMM are about 10 times slower than those with $L = 3$. In addition to the results with $L = 3$, 20, we also repeat the above experiments on simulations with $L = 100$. The results are presented in section SM5.

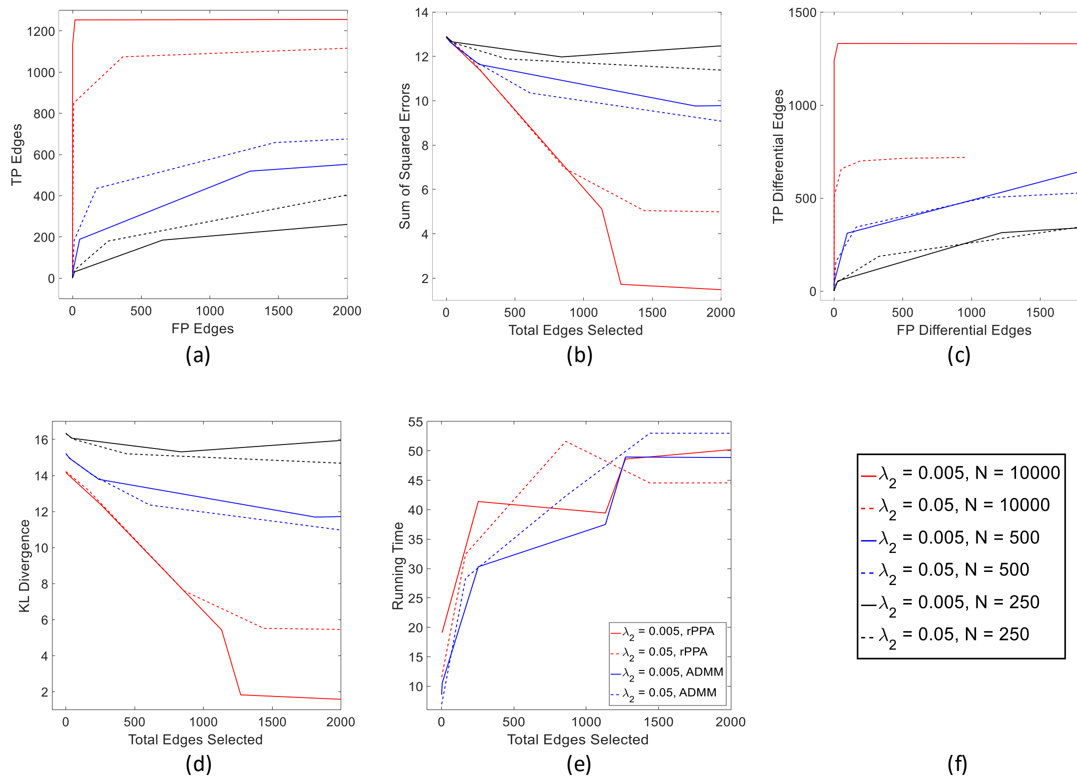

Figure 2. *Performances on nearest-neighbor networks with $p = 500$, $L = 20$, and various sample sizes $N = 10000, 500, 250$. (a)–(f) are as labeled in Figure 1. The values TP Edges, FP Edges, Sum of Squared Errors, Total Edges Selected, TP Differential Edges, and FP Differential Edges presented are divided by $L = 20$.*

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

We first consider a relatively short three-year time period from January 2004 to December 2006. During this period, there are in total 755 daily returns of 370 stocks. We call this data set *SPX3a*. For each year, it contains approximately 250 daily returns of each stock. Considering the limited number of observations in each year and the interpretation of the results, we choose to analyze random smaller subsets of all involved stocks, whose sizes are chosen to be $p = 100$ and $p = 200$, 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 experiments, which is referred to as *SPX11b*. Since this time period is longer than the previous one, the number of stocks becomes smaller as some stocks might disappear (the stocks that do not exist over the entire estimation period have been removed). During the 11-year time period, there are 2769 daily returns 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 analyze two random subsets of all existing stocks, of which the sizes are selected to be $p = 100$ and $p = 200$. To test on a large L, we can further split one year into two halves and obtain a data set with $L = 22$, $p = 100$. It is referred to as *SPX22c*.

For the tuning parameters (λ_1, λ_2), we first select three pairs manually, which results in various sparsity patterns ($1\% \sim 25\%$) so that we can see how sensitive the performance of a method is to the solution sparsity of the FGL problem. In addition, a 3-fold cross validation (CV) is performed with tuning grids

$$\lambda_{1,\text{grid}} = \lambda_{2,\text{grid}} = \{10^{-3}, 10^{-3.5}, 10^{-4}, 10^{-4.5}, 10^{-5}, 10^{-5.5}, 10^{-6}\}$$

to select the optimal pair $(\hat{\lambda}_1, \hat{\lambda}_2)$ (bold numbers in each table) that minimizes the CV score; see section SM4 for details.

Table 1

Performances of rPPA, ADMM, and MGL on stock price data. Tolerance ε = 1e-6.

Problem	(λ_1, λ_2)	Density	Iteration			Time			Error		
(p, L)			P	A	M	P	A	M	P	A	M
	$(10^{-4}, 10^{-5})$	0.024	24	5879	6	18	54	08	5.9e-07	1.0e-06	3.5e-07
SPX3a	$(5 \cdot 10^{-5}, 5 \cdot 10^{-6})$	0.127	24	5895	9	19	58	12	6.9e-07	1.0e-06	7.3e-08
(100,3)	$(2 \cdot 10^{-5}, 2 \cdot 10^{-6})$	0.236	24	10407	18	29	01:47	12	7.8e-07	1.0e-06	8.1e-07
	$\mathbf{(10^{-5}, 10^{-3})}$	0.305	28	13652	25	34	02:18	01:19	7.0e-07	1.0e-06	8.1e-07
	$(10^{-4}, 10^{-5})$	0.027	20	1508	6	25	33	25	6.6e-07	9.9e-07	5.9e-07
SPX3a	$(5 \cdot 10^{-5}, 5 \cdot 10^{-6})$	0.089	20	1599	23	29	36	03:12	6.5e-07	1.0e-06	8.3e-07
(200,3)	$(2 \cdot 10^{-5}, 2 \cdot 10^{-6})$	0.151	22	1958	55	29	46	02:32	9.4e-07	1.0e-06	8.5e-07
	$\mathbf{(10^{-4.5}, 10^{-4})}$	0.131	23	1704	24	32	39	01:51	9.1e-07	9.9e-07	5.2e-07
	$(5 \cdot 10^{-4}, 5 \cdot 10^{-5})$	0.028	23	4745	7	46	02:15	02:11	9.0e-07	1.0e-06	5.5e-07
SPX11b	$(10^{-4}, 10^{-5})$	0.130	23	4559	136	56	02:34	08:12	8.9e-07	1.0e-06	9.9e-07
(100,11)	$(5 \cdot 10^{-5}, 5 \cdot 10^{-6})$	0.214	23	4595	549	01:04	02:37	25:50	8.0e-07	1.0e-06	1.0e-06
	$\mathbf{(10^{-4.5}, 10^{-4})}$	0.340	26	4670	393	01:12	02:37	20:37	7.0e-07	1.0e-06	9.9e-07
	$(5 \cdot 10^{-4}, 5 \cdot 10^{-5})$	0.017	20	1201	26	48	01:20	45:40	8.9e-07	6.4e-07	8.4e-07
SPX11b	$(10^{-4}, 10^{-5})$	0.082	20	1287	552	01:05	01:46	01:46:52	8.8e-07	9.9e-07	9.9e-07
(200,11)	$(5 \cdot 10^{-5}, 5 \cdot 10^{-6})$	0.133	20	1289	840	01:11	01:49	03:00:00	7.9e-07	1.0e-06	4.8e-05
	$\mathbf{(10^{-4.5}, 10^{-4.5})}$	0.210	23	4662	992	03:38	06:48	03:00:00	7.6e-07	1.0e-06	5.2e-05
	$(5 \cdot 10^{-4}, 5 \cdot 10^{-5})$	0.021	33	14358	21	38	10:09	13:26	7.6e-07	1.0e-06	9.3e-07
SPX22c	$(10^{-4}, 10^{-5})$	0.097	36	14260	543	55	14:18	51:48	7.5e-07	1.0e-06	9.9e-07
(100,22)	$(5 \cdot 10^{-5}, 5 \cdot 10^{-6})$	0.176	33	14127	1500	51	14:09	02:22:31	7.3e-07	1.0e-06	4.6e-06
	$\mathbf{(10^{-4.5}, 10^{-4})}$	0.307	33	20000	568	01:44	18:54	01:51:03	7.3e-07	4.4e-06	9.8e-07

Table 1 shows the comparison of rPPA, ADMM, and MGL on the stock price data sets $SPX3a$, $SPX11b$, and $SPX22c$ with 100 and 200 selected stocks. For each instance, the first three pairs of (λ_1, λ_2) are selected manually for the diversity of sparsity, while the last bold pair is obtained from CV. One immediate observation from the table is that rPPA outperforms ADMM and MGL for a majority of instances. For the exceptional instances, rPPA is still faster than ADMM and comparable with MGL. In addition, we find that both rPPA and ADMM succeeded in solving all instances; while MGL failed to solve three of them within one hour. This might imply that MGL is not robust for solving the FGL model when applied to the stock price data sets. On the other hand, the running time of rPPA changes more smoothly when we vary the parameter values for λ_1 and λ_2 or the problem dimension. The numerical results show convincingly that our algorithm rPPA can solve the FGL problems efficiently and robustly. The superior performance of rPPA can mainly be attributed to our ability to extract and exploit the sparsity structure (in the surrogate generalized Jacobian of $\text{Prox}_{\mathcal{P}}$) within the SSN method to solve each rPPA subproblem very efficiently. Additional numerical results running over a variety of λ_2 (resp., λ_1) while holding $\lambda_1 = \hat{\lambda}_1$ (resp., $\lambda_2 = \hat{\lambda}_2$) at the CV selected value for the instance $SPX11b$, $p = 100$ are presented in Table 2.

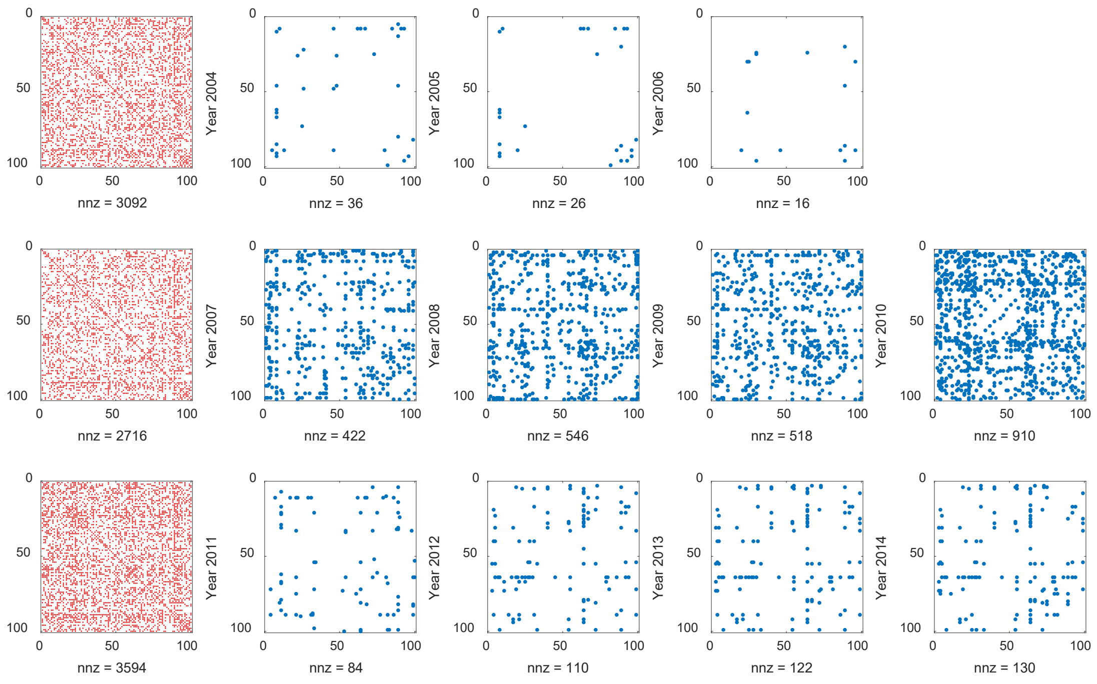

Figure 3. *Patterns of the estimated precision matrices over* 11 *years on the stock price data sets. The red pattern extracts the common structure of those in the same row. The blue pattern corresponds to individual edges specific to its own network.*

Figure 3 displays the sparse patterns of 11 estimated precision matrices from year 2004 to 2014 on the data set $SPX11b$ with CV selected tuning parameters $(\hat{\lambda}_1, \hat{\lambda}_2) = (10^{-4.5}, 10^{-4})$, $p = 100$. It should be noted 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 3) to

Table 2

Performances of rPPA, ADMM, and MGL on stock price data set SPX11b. $(p, L) = (100, 11)$. *Tolerance* $\varepsilon = 1e\text{-}6$. *The CV selected tuning parameter is* $(\hat{\lambda}_1, \hat{\lambda}_2) = (10^{-4.5}, 10^{-4.0})$.

λ_1	λ_2	Density	Iteration			Time			Error		
			P	A	M	P	A	M	P	A	M
	$10^{-3.0}$	0.286	26	4707	152	01:12	02:28	19:04	7.2e-07	1.0e-06	5.0e-07
	$10^{-3.2}$	0.289	23	5355	207	01:08	02:56	20:18	8.4e-07	1.0e-06	9.9e-07
	$10^{-3.4}$	0.297	23	4693	273	01:06	02:30	22:39	8.4e-07	1.0e-06	9.9e-07
	$10^{-3.6}$	0.308	23	5244	349	01:07	02:51	25:02	8.4e-07	1.0e-06	1.0e-06
$\mathbf{10^{-4.5}}$	$10^{-3.8}$	0.327	23	5245	345	01:07	02:51	23:18	8.3e-07	1.0e-06	1.0e-06
	$10^{-4.2}$	0.346	23	4703	420	01:11	02:32	16:27	8.1e-07	1.0e-06	1.0e-06
	$10^{-4.4}$	0.343	23	4781	579	01:12	02:34	19:37	8.7e-07	1.0e-06	9.9e-07
	$10^{-4.6}$	0.327	26	4898	769	01:17	02:39	24:47	8.4e-07	1.0e-06	1.0e-06
	$10^{-4.8}$	0.305	24	5039	971	01:21	02:45	30:13	6.3e-07	1.0e-06	1.0e-06
$\mathbf{10^{-4.5}}$	$\mathbf{10^{-4.0}}$	0.340	26	4670	393	01:12	02:37	20:37	7.0e-07	1.0e-06	9.9e-07
$10^{-3.5}$		0.046	23	4738	13	49	02:16	01:13	9.0e-07	1.0e-06	5.7e-07
$10^{-3.7}$		0.074	23	4726	19	49	02:23	02:23	8.9e-07	1.0e-06	8.7e-07
$10^{-3.9}$		0.124	23	4710	45	52	02:28	09:33	8.9e-07	1.0e-06	9.9e-07
$10^{-4.1}$		0.226	23	4572	84	55	02:27	04:10	8.8e-07	1.0e-06	9.6e-07
$10^{-4.3}$	$\mathbf{10^{-4.0}}$	0.305	26	4576	195	01:03	02:31	09:01	7.4e-07	1.0e-06	9.9e-07
$10^{-4.7}$		0.356	24	5177	589	01:20	02:47	21:37	7.5e-07	1.0e-06	1.0e-06
$10^{-4.9}$		0.378	22	2481	809	01:03	01:20	29:05	8.6e-07	1.0e-06	1.0e-06
$10^{-5.1}$		0.409	26	2485	1175	57	01:19	44:56	7.0e-07	9.9e-07	1.0e-06
$10^{-5.3}$		0.479	24	2644	1247	53	01:24	44:35	8.3e-07	1.0e-06	1.0e-06

aid the 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. One can clearly see that more individual edges are detected in the middle stage. The increased number of individual edges over this period is likely due to the 2008 global financial crisis and its sustained effects. Another observation is that the number of individual edges had a drastic increase in 2007, remained at a high level during the 2008 global financial crisis and a certain period after that, and then went down to a level still higher than that of the precrisis period (the years 2004, 2005, and 2006). The sudden increase in 2007 might be seen as a prediction of the oncoming financial crisis in 2008. The increased number of interactions among stocks after the financial crisis compared to that in the precrisis period may indicate some essential changes of the financial landscape. To a certain degree, the observations agree well with [25]. We further investigate the effect of financial crisis by analyzing the data in the middle stage (January 2007–December 2010) on a quarterly basis; see Figure SM2.

4.2.3. University webpages. In this section, we compare rPPA, ADMM, and MGL on the `university webpages` data.[1] The procedure of processing data and generating sample

[1]Available at http://ana.cachopo.org/datasets-for-single-label-text-categorization.

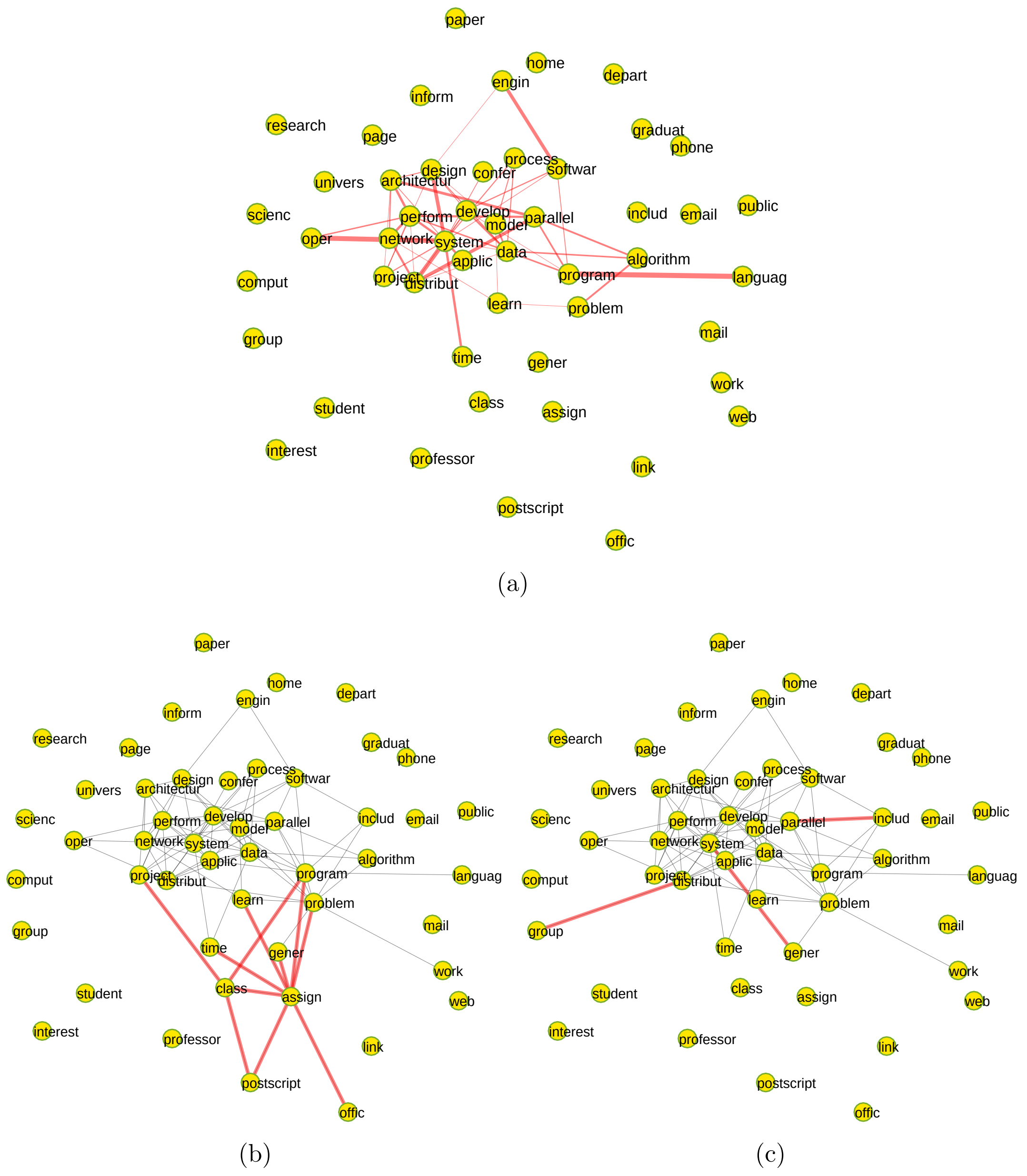

Figure 4. (a) *Common structure for four classes.* (b) *Dependency structure for class Course.* (c) *Dependency structure for class Project. The thin black lines are the edges appearing in both classes, and the thick red lines are the edges only appearing in one class.* $(\lambda_1, \lambda_2) = (0.005, 0.003)$.

covariance matrices is similar to the process in [8]. The original pages were collected from computer science departments of various universities in 1997. We selected the four largest and meaningful classes in our experiment: Student, Faculty, Staff, and Department. For each class, the collection contains pages from four universities, Cornell, Texas, Washington, Wisconsin, and other miscellaneous pages from other universities. Furthermore, the original text data have been preprocessed by stemming techniques, that is, reducing words to their morphological roots. The preprocessed data sets downloaded from the link above contain two files: two thirds of the pages were randomly chosen as a training set (*Webtrain*) and the remaining third as a testing set (*Webtest*). Table 3 presents the distribution of documents per class. See section SM7 for details on the procedure of data processing.

Table 3
Distribution of documents of classes Student, Faculty, Course, and Project.

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

We first apply the FGL model to the *Webtest* data set for the purpose of interpreting the data. We present the results with relatively large controlling parameters $\lambda_1 = 0.005$ and $\lambda_2 = 0.003$ to aid the interpretation of the relationships. Figure 4(a), (b), and (c) illustrate the common structure across all classes and the differences between the Course and Project classes. One can see that some course related terms, such as class and assign, are of high degree in Figure 4(b), whereas they are not even connected in Figure 4(c). Additionally, some teaching related terms are linked only in the Course category, such as class-assign, assign-problem, and class-project. Overall, it is likely that the FGL model is capable of identifying the common and individual structures of the webpages among related classes.

Table 4
Performances of rPPA, ADMM, and MGL on webpages data. Tolerance $\varepsilon = 1e$-6.

Problem	(λ_1, λ_2)	Density	Iteration			Time			Error		
(n, L)			P	A	M	P	A	M	P	A	M
	$(10^{-2}, 10^{-3})$	0.015	14	501	4	03	06	06	8.9e-07	9.9e-07	1.2e-07
Webtest	$(5\cdot10^{-3}, 5\cdot10^{-4})$	0.047	14	501	6	03	06	13	8.8e-07	9.9e-07	3.1e-07
(100,4)	$(10^{-3}, 10^{-4})$	0.219	14	549	38	03	07	01:00	8.5e-07	1.0e-06	9.0e-07
	$\mathbf{(10^{-3}, 10^{-2.5})}$	0.207	16	557	15	05	07	32	5.1e-07	1.0e-06	5.2e-07
	$(10^{-2}, 10^{-3})$	0.008	16	835	7	09	26	01:08	9.0e-07	9.9e-07	8.6e-07
Webtest	$(5\cdot10^{-3}, 5\cdot10^{-4})$	0.025	16	744	8	13	22	01:12	8.9e-07	9.8e-07	4.7e-07
(200,4)	$(10^{-3}, 10^{-4})$	0.156	16	562	72	08	17	08:24	8.9e-07	9.9e-07	9.4e-07
	$\mathbf{(10^{-3}, 10^{-5})}$	0.167	16	562	76	08	17	07:32	8.9e-07	9.9e-07	9.6e-07
	$(5\cdot10^{-3}, 5\cdot10^{-4})$	0.016	17	883	9	32	49	04:08	6.6e-07	1.0e-06	3.7e-07
Webtest	$(10^{-3}, 10^{-4})$	0.119	17	501	259	43	29	44:01	8.9e-07	7.9e-07	1.0e-06
(300,4)	$(5\cdot10^{-4}, 5\cdot10^{-5})$	0.244	18	701	1394	01:05	41	02:35:17	7.0e-07	7.8e-07	1.0e-06
	$\mathbf{(10^{-3}, 10^{-4.5})}$	0.127	17	501	291	47	29	48:45	9.7e-07	9.0e-07	1.0e-06
	$(10^{-2}, 10^{-3})$	0.011	20	2182	4	10	22	07	9.8e-07	9.9e-07	1.6e-08
Webtrain	$(5\cdot10^{-3}, 5\cdot10^{-4})$	0.030	20	2181	6	11	25	13	9.9e-07	1.0e-06	2.2e-09
(100,4)	$(10^{-3}, 10^{-4})$	0.162	20	2175	29	12	26	01:25	9.7e-07	1.0e-06	8.7e-07
	$\mathbf{(10^{-3}, 10^{-2})}$	0.181	21	1901	14	17	22	01:55	5.9e-07	7.1e-07	3.1e-07
	$(5\cdot10^{-3}, 5\cdot10^{-4})$	0.015	20	1682	6	24	48	25	5.8e-07	1.0e-06	1.1e-08
Webtrain	$(10^{-3}, 10^{-4})$	0.105	20	1799	42	26	56	03:47	5.7e-07	1.0e-06	8.8e-07
(200,4)	$(5\cdot10^{-4}, 5\cdot10^{-5})$	0.210	19	1796	121	27	55	08:33	1.0e-06	1.0e-06	9.9e-07
	$\mathbf{(10^{-3.5}, 10^{-2})}$	0.283	22	2318	28	01:04	01:08	05:06	9.6e-07	9.4e-07	9.0e-07
	$(5\cdot10^{-3}, 5\cdot10^{-4})$	0.010	19	1986	7	42	01:58	02:17	7.7e-07	9.9e-07	3.4e-08
Webtrain	$(10^{-3}, 10^{-4})$	0.077	19	1550	61	44	01:35	23:43	7.4e-07	1.0e-06	9.7e-07
(300,4)	$(5\cdot10^{-4}, 5\cdot10^{-5})$	0.168	19	1550	182	50	01:33	22:07	7.3e-07	1.0e-06	9.8e-07
	$\mathbf{(10^{-3.5}, 10^{-2})}$	0.230	22	3291	49	03:09	03:18	20:04	8.9e-07	1.0e-06	9.3e-07

Table 4 shows the comparison of the three methods rPPA, ADMM, and MGL on the webpages data sets with data dimension $p = 100$, $p = 200$, and $p = 300$. Again, we select three pairs of (λ_1, λ_2) which result in various sparsity $(1\% \sim 25\%)$ and one pair (bold numbers in the table) by a 3-fold CV with tuning grids

$$\lambda_{1,\text{grid}} = \lambda_{2,\text{grid}} = \{10^{-2}, 10^{-2.5}, 10^{-3}, 10^{-3.5}, 10^{-4}, 10^{-4.5}, 10^{-5}\}.$$

As can be seen, rPPA outperforms ADMM and MGL for most of the tested webpages data sets.

5. Conclusion.

We have designed an efficient and globally convergent regularized proximal point algorithm for solving the primal formulation of the fused graphical Lasso problem. From a theoretical perspective, we established the Lipschitz continuity of the solution mapping and consequently obtained that the primal and dual sequences are locally linearly convergent. This lays the foundation for the efficiency of the proposed algorithm. Moreover, the second order information was also fully exploited, which further leads to the high efficiency of the proposed algorithm. Numerically, we 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 [27] on both synthetic and real data sets. In summary, the proposed semismooth Newton based regularized proximal point algorithm is a highly efficient method for solving the fused graphical Lasso problem.

REFERENCES

[1] A. Ahmed and E. P. Xing, *Recovering time-varying networks of dependencies in social and biological studies*, Proc. Natl. Acad. Sci. USA, 106 (2009), pp. 11878–11883, https://doi.org/10.1073/pnas.0901910106.

[2] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont, *Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data*, J. Mach. Learn. Res., 9 (2008), pp. 485–516.

[3] P. Danaher, P. Wang, and D. M. Witten, *The joint graphical Lasso for inverse covariance estimation across multiple classes*, J. R. Stat. Soc. Ser. B Stat. Methodol., 76 (2014), pp. 373–397, https://doi.org/10.1111/rssb.12033.

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

[5] Y. Fan and C. Y. Tang, *Tuning parameter selection in high dimensional penalized likelihood*, J. R. Stat. Soc. Ser. B Stat. Methodol., 75 (2013), pp. 531–552, https://doi.org/10.1111/rssb.12001.

[6] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical Lasso*, Biostatistics, 9 (2008), pp. 432–441, https://doi.org/10.1093/biostatistics/kxm045.

[7] A. J. Gibberd and J. D. Nelson, *Regularized estimation of piecewise constant Gaussian graphical models: The group-fused graphical Lasso*, J. Comput. Graph. Statist., 26 (2017), pp. 623–634, https://doi.org/10.1080/10618600.2017.1302340.

[8] J. Guo, E. Levina, G. Michailidis, and J. Zhu, *Joint estimation of multiple graphical models*, Biometrika, 98 (2011), pp. 1–15, https://doi.org/10.1093/biomet/asq060.

[9] 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, New York, 2017, pp. 205–213, https://doi.org/10.1145/3097983.3098037.

[10] D. Han, D. F. Sun, and L. Zhang, *Linear rate convergence of the alternating direction method of multipliers for convex composite programming*, Math. Oper. Res., 43 (2018), pp. 622–637, https://doi.org/10.1287/moor.2017.0875.

[11] 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, Curran Associates, Inc., Red Hook, NY, 2011, pp. 2330–2338.

[12] J. D. Lee, Y. Sun, and M. A. Saunders, *Proximal Newton-type methods for minimizing composite functions*, SIAM J. Optim., 24 (2014), pp. 1420–1443, https://doi.org/10.1137/130921428.

[13] C. Lemaréchal and C. Sagastizábal, *Practical aspects of the Moreau–Yosida regularization: Theoretical preliminaries*, SIAM J. Optim., 7 (1997), pp. 367–385, https://doi.org/10.1137/S1052623494267127.

[14] 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, https://doi.org/10.1093/biostatistics/kxj008.

[15] X. Li, D. F. Sun, and K.-C. Toh, *A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems*, SIAM J. Optim., 28 (2018), pp. 433–458, https://doi.org/10.1137/16M1097572.

[16] X. Li, D. F. Sun, and K.-C. Toh, *On efficiently solving the subproblems of a level-set method for fused Lasso problems*, SIAM J. Optim., 28 (2018), pp. 1842–1866, https://doi.org/10.1137/17M1136390.

[17] M. Lin, Y.-J. Liu, D. F. Sun, and K.-C. Toh, *Efficient sparse semismooth Newton methods for the clustered Lasso problem*, SIAM J. Optim., 29 (2019), pp. 2026–2052, https://doi.org/10.1137/18M1207752.

[18] 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, https://doi.org/10.1016/j.neuroimage.2014.07.033.

[19] J.-J. Moreau, *Proximité et dualité dans un espace Hilbertien*, Bull. Soc. Math. France, 93 (1965), pp. 273–299.

[20] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optim., 14 (1976), pp. 877–898, https://doi.org/10.1137/0314056.

[21] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, Grundlehren Math. Wiss. 317, Springer, Berlin, 1998.

[22] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, *Sparsity and smoothness via the fused Lasso*, J. R. Stat. Soc. Ser. B Stat. Methodol., 67 (2005), pp. 91–108, https://doi.org/10.1111/j.1467-9868.2005.00490.x.

[23] F. Tomasi, V. Tozzo, A. Verri, and S. Salzo, *Forward-backward splitting for time-varying graphical models*, in International Conference on Probabilistic Graphical Models, Proc. Mach. Learn. Res., 72 (2018), pp. 475–486.

[24] C. J. Wang, D. F. Sun, and K.-C. Toh, *Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm*, SIAM J. Optim., 20 (2010), pp. 2994–3013, https://doi.org/10.1137/090772514.

[25] J. Yang and J. Peng, *Estimating time-varying graphical models*, J. Comput. Graph. Statist., 29 (2020), pp. 191–202, https://doi.org/10.1080/10618600.2019.1647848.

[26] J. Yang, D. F. Sun, and K.-C. Toh, *A proximal point algorithm for log-determinant optimization with group Lasso regularization*, SIAM J. Optim., 23 (2013), pp. 857–893, https://doi.org/10.1137/120864192.

[27] S. Yang, Z. Lu, X. Shen, P. Wonka, and J. Ye, *Fused multiple graphical Lasso*, SIAM J. Optim., 25 (2015), pp. 916–943, https://doi.org/10.1137/130936397.

[28] K. Yosida, *Functional Analysis*, Springer, Berlin, 1965.

[29] Y. Zhang, N. Zhang, D. F. Sun, and K.-C. Toh, *An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems*, Math. Program., 179 (2020), pp. 223–263, https://doi.org/10.1007/s10107-018-1329-6.

[30] Y. Zhang, N. Zhang, D. F. Sun, and K.-C. Toh, *A proximal point dual Newton algorithm for solving group graphical Lasso problems*, SIAM J. Optim., 30 (2020), pp. 2197–2220, https://doi.org/10.1137/19M1267830.

[31] X.-Y. Zhao, D. F. Sun, and K.-C. Toh, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM J. Optim., 20 (2010), pp. 1737–1765, https://doi.org/10.1137/080718206.