

An efficient GPU-Parallel Coordinate Descent Algorithm for Sparse Precision Matrix Estimation via Scaled Lasso

Seunghwan Lee · Sang Cheol Kim · Donghyeon Yu*

Received: date / Accepted: date

Abstract The sparse precision matrix plays an essential role in the Gaussian graphical model since a zero off-diagonal element indicates conditional independence of the corresponding two variables given others. In the Gaussian graphical model, many methods have been proposed and their theoretical properties are given as well. Among these, the sparse precision matrix estimation via scaled lasso (SPMESL) has an attractive feature to which the penalty level is automatically set to achieve the optimal convergence rate under the sparsity and invertibility conditions. Conversely, other methods need to be used in searching for the optimal tuning parameter. Despite such an advantage, the SPMESL has not been widely used due to its expensive computational cost. In this paper, we develop a GPU-parallel coordinate descent (CD) algorithm for the SPMESL and numerically show that the proposed algorithm is much faster than the least angle regression (LARS) tailored to the SPMESL. Several comprehensive numerical studies are conducted to investigate the scalability of the proposed algorithm and the estimation performance of the SPMESL. The results show that the SPMESL has the lowest false discovery rate for all cases and the best performance in the case where the level of the sparsity of the columns is high.

Keywords Gaussian graphical model · graphics processing unit · parallel computation · tuning-free

1 Introduction

The covariance matrix and its inverse are of main interest in multivariate analysis to model dependencies between variables. Traditionally, these two parameters have been estimated by the sample covariance matrix and its inverse based on the maximum likelihood (ML) estimation. Although these ML estimators are often asymptotically unbiased and simple to calculate, there are some weaknesses when the number of variables is greater than that of samples. This circumstance is also known as high-dimensional low-sample size (HDLSS) data. For HDLSS data, it is known that the sample covariance matrix becomes inefficient (Yao et al. 2015). In addition, the inverse of the sample covariance matrix is undefined since the sample covariance matrix is singular for HDLSS data. For the covariance matrix, many methods have been proposed under some structural conditions such as bandable structure and sparse structure to improve the estimation efficiency (Bickel and Levina 2008a,b; Rothman et al. 2009; Wu et al. 2009; Cai et al. 2010; Cai and Liu 2011a; Cai and Zhou 2012).

To obtain an estimator of the *precision matrix* (i.e., inverse covariance matrix) for HDLSS data, various approaches have been developed by adopting the sparsity assumption that there are many zero elements in the matrix. The existing methods can be categorized into four approaches: covariance estimation-induced approach, ML-based approach, regression approach, and constrained ℓ_1 -minimization approach.

Seunghwan Lee · Donghyeon Yu
Department of Statistics, Inha University, Incheon, Korea
E-mail: dyu@inha.ac.kr

Sang Cheol Kim
Division of Bio-Medical Informatics, Center for Genome Science, National Institute of Health, Korea Centers for Disease Control and Prevention, Cheongju, Korea

The covariance estimation-induced approach considers an indirect estimation using the inversion of the well-conditioned shrinkage covariance matrix estimators and applies multiple testing procedures to identify nonzero elements of the precision matrix (GeneNet, Schäfer and Strimmer (2005)). The ML-based approach directly estimates the precision matrix by maximizing the penalized likelihood function (Yuan and Lin 2007; Friedman et al. 2008; Witten et al. 2011; Mazumder et al. 2012). The regression approach considers the linear regression model and uses the fact that the nonzero regression coefficients correspond to the nonzero off-diagonal elements of the precision matrix (Meinshausen and Bühlmann 2006; Peng et al. 2009; Yuan 2010; Sun and Zhang 2013; Ren et al. 2015; Khare et al. 2015; Ali et al. 2017). The constrained ℓ_1 -minimization (CLIME) approach (Cai et al. 2011b) obtains the sparse precision matrix by solving the linear programming problem with the constraints on the proximity to the precision matrix where the objective function is the sum of absolute values of the design variables. The adaptive CLIME (Cai et al. 2016) improves the CLIME and attains the optimal rate of convergence.

Among the existing methods, sparse precision matrix estimation via the scaled Lasso (SPMESL) proposed by Sun and Zhang (2013) is a tuning-free procedure. Conversely, other existing methods require searching the optimal tuning parameter, GeneNet (Schäfer and Strimmer 2005) and the neighborhood selection (Meinshausen and Bühlmann 2006) require choosing the level of the false discovery rate; other penalized methods require choosing the optimal penalty level. In addition, SPMESL supports the consistency of the precision matrix estimation under the sparsity and invertibility conditions that are weaker than the irrerepresentable condition (van de Geer and Bühlmann 2009; Sun and Zhang 2013). However, it has not been widely used for the sparse precision matrix estimation due to the inefficiency of the implemented algorithm of the SPMESL using the least angle regression (LARS) algorithm (Efron et al. 2004). Even though the LARS algorithm efficiently provides a whole solution path of the Lasso problem (Tibshirani 1996), its computational cost is expensive and the SPMESL needs to solve p independent Lasso problems as the subproblems of the SPMESL. Thus, the scalability of the SPMESL is still challenging.

In this paper, we found the possibility to improve the computational efficiency of the SPMESL with the empirical observation that the SPMESL does not need a whole solution path of the Lasso problem based on the tuning-free characteristic of the SPMESL (see details in Section 3). Motivated by this empirical observation, we propose a more efficient algorithm based on the coordinate descent (CD) algorithm and the warm start strategy for the scaled Lasso and the SPMESL as applied to the standard lasso problem in Wu and Lange (2008). Moreover, we develop the *row-wise updating parallel CD algorithm using graphics processing units* (GPUs) adequate for the SPMESL. To efficiently implement the proposed parallel CD algorithm, we consider the *active response matrix* that consists of columns of active response variables that corresponds to the error variance estimate not converged at the current iteration. We will show the efficiency of the proposed algorithms using comprehensive numerical studies. In this paper, we also provide the numerical comparisons of the estimation performance of the SPMESL with three different penalty levels, which are theoretically suggested in previous literature (Sun and Zhang 2012, 2013) but there is no comparison result of them in terms of the estimation performance.

The remainder of the paper is organized as follows. Section 2 introduces the SPMESL and its original algorithm. We explain the proposed CD algorithm and GPU-parallel CD algorithm in Section 3. Section 4 provides a comprehensive numerical study, including the comparisons of computation times and estimation performances with other existing methods. We provide a summary of the paper and discussion in Section 5.

2 Sparse Precision Matrix Estimation via Scaled Lasso

In this section, we briefly introduce the scaled Lasso, the SPMESL, and their algorithms.

2.1 Scaled Lasso

The scaled Lasso proposed by Sun and Zhang (2012) is a variant of a penalized regression model with the Lasso penalty. To be specific, let $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\boldsymbol{\beta} \in \mathbb{R}^p$, and $\sigma > 0$. Consider a linear regression model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon$, where $\epsilon \sim N(0, \sigma^2 \mathbf{I}_n)$ and \mathbf{I}_n is the n -dimensional identity matrix. The scaled Lasso

considers the minimization of the following objective function $L_{\lambda_0}(\boldsymbol{\beta}, \sigma)$:

$$L_{\lambda_0}(\boldsymbol{\beta}, \sigma) = \frac{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2}{2n\sigma} + \frac{\sigma}{2} + \lambda_0\|\boldsymbol{\beta}\|_1, \quad (1)$$

where λ_0 is a given tuning parameter and $\|\mathbf{x}\|_1 = \sum_{j=1}^p |x_j|$ for $\mathbf{x} \in \mathbb{R}^p$. Thus, the scaled Lasso simultaneously obtains the estimate $\hat{\boldsymbol{\beta}}$ of the regression coefficient $\boldsymbol{\beta}$ and the estimate $\hat{\sigma}$ of the error standard deviation σ . It is proved that the objective function L_{λ_0} is jointly convex in $(\boldsymbol{\beta}, \sigma)$ and is strictly convex for σ in Sun and Zhang (2012). From the convexity of the objective function, the solution $(\hat{\boldsymbol{\beta}}, \hat{\sigma})$ of the scaled Lasso problem can be obtained by the block CD algorithm as follows:

(Step 1) With a fixed $\hat{\sigma}$, consider the minimization of $\hat{\sigma}L_{\lambda_0}(\boldsymbol{\beta}, \hat{\sigma})$:

$$\hat{\sigma}L_{\lambda_0}(\boldsymbol{\beta}, \hat{\sigma}) = \frac{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2}{2n} + \hat{\sigma}\lambda_0\|\boldsymbol{\beta}\|_1 + \frac{\hat{\sigma}^2}{2}.$$

The minimizer $\hat{\boldsymbol{\beta}}$ of $\hat{\sigma}L_{\lambda_0}(\boldsymbol{\beta}, \hat{\sigma})$ can be obtained by solving the following standard lasso problem with $\lambda = \hat{\sigma}\lambda_0$:

$$\min_{\boldsymbol{\beta}} \frac{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2}{2n} + \lambda\|\boldsymbol{\beta}\|_1. \quad (2)$$

(Step 2) With a fixed $\hat{\boldsymbol{\beta}}$, the minimizer $\hat{\sigma}$ of $L_{\lambda_0}(\hat{\boldsymbol{\beta}}, \sigma)$ is easily obtained by

$$\hat{\sigma} = \frac{\|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|_2}{\sqrt{n}}.$$

(Step 3) Repeat Steps 1 and 2 until convergence occurs.

In the original paper of the scaled Lasso, the standard lasso problem is solved by the LARS algorithm (Efron et al. 2004), which provides a whole solution path of the standard Lasso problem. During the block CD algorithm, the minimizer $\hat{\boldsymbol{\beta}}(\hat{\sigma}^{(r)})$ of $\hat{\sigma}^{(r)}L_{\lambda_0}(\boldsymbol{\beta}, \hat{\sigma}^{(r)})$ in Step 1 at the r -th iteration is obtained from the solution path of the standard Lasso problem with $\lambda = \hat{\sigma}^{(r)}\lambda_0$.

In addition to the joint estimation of $\boldsymbol{\beta}$ and σ , the scaled Lasso has attractive features as described in Sun and Zhang (2012). First, the scaled Lasso guarantees the consistency of $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}$ under two conditions: the penalty level condition $\lambda_0 > A\sqrt{2n^{-1}\log p}$ for $A > 1$ and the compatibility condition, which implies the oracle inequalities for the prediction and estimation (van de Geer and Bühlmann 2009). Second, the scaled Lasso estimates are scale-equivariant in \mathbf{y} in the sense that $\hat{\boldsymbol{\beta}}(\mathbf{X}, \alpha\mathbf{y}) = \alpha\hat{\boldsymbol{\beta}}(\mathbf{X}, \mathbf{y})$ and $\hat{\sigma}(\mathbf{X}, \alpha\mathbf{y}) = |\alpha|\hat{\sigma}(\mathbf{X}, \mathbf{y})$. Finally, the authors suggest using the universal penalty level $\lambda_0^U = \sqrt{2n^{-1}\log p}$ for λ_0 based on their numerical and real-data examples. We can consider the scaled Lasso with the universal penalty level as a tuning-free procedure. Note that the universal penalty level does not satisfy the theoretical requirement $\lambda_0 > A\sqrt{2n^{-1}\log p}$ for the consistency of $\hat{\sigma}$. The authors of the scaled Lasso provide several conditions that can weaken the required condition for λ_0 , in order to justify using a penalty level smaller than $A\sqrt{2n^{-1}\log p}$ for $A > 1$ (Sun and Zhang 2012).

2.2 Sparse precision matrix estimation via scaled Lasso

Let $\boldsymbol{\Sigma} = (\sigma_{jk})_{1 \leq j, k \leq p}$ and $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1} = (\omega_{jk})_{1 \leq j, k \leq p}$ be a covariance matrix and a corresponding precision matrix, respectively. Suppose that $\mathbf{x}^{(i)} = (X_{i1}, \dots, X_{ip})$ for $i = 1, 2, \dots, n$ are independently drawn from the multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$. Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a data matrix and $\mathbf{x}_k = (X_{1k}, \dots, X_{nk})^T$ be the k th column of \mathbf{X} . Consider following linear regression models for $k = 1, \dots, p$:

$$X_{ik} = \sum_{l \neq k} \beta_{lk} X_{il} + \epsilon_{ik}, \quad (3)$$

where ϵ_{ik} s are independent and identically distributed random variables drawn from the normal distribution with mean 0 and variance σ_k^2 . As applied in the regression approach for the sparse precision matrix

estimation, the elements of the precision matrix can be represented into the regression coefficient and the error variance by using the following relationships:

$$\omega_{jk} = -\frac{\beta_{jk}}{\sigma_k^2}, \quad \omega_{kk} = \frac{1}{\sigma_k^2} \quad \text{for } 1 \leq j \neq k \leq p. \quad (4)$$

As the scaled Lasso simultaneously estimates the regression coefficients (β_{jk}) and the error standard deviation σ_k as described in Section 2.1, we can use the scaled Lasso to estimate the precision matrix. Thus, the SPMESL considers solving the scaled Lasso problems column-wise and defines the SPMESL estimator that combines the estimates from the p scaled Lasso problems.

To be specific, let $\mathbf{B} = (b_{jk})_{1 \leq j, k \leq p}$ be a matrix of the regression coefficients such that $b_{jk} = \beta_{jk}$ for $j \neq k$ and $b_{jj} = -1$ for $j = 1, \dots, p$. Denote $\mathbf{b}^{(j)} = (b_{j1}, \dots, b_{jp})$ and $\mathbf{b}_k = (b_{1k}, \dots, b_{pk})^T$ as the j th row and the k th column of a matrix \mathbf{B} , respectively. We further let $\mathbf{S} = (s_{jk})$ be the sample covariance matrix. Subsequently, the precision matrix can be represented with \mathbf{B} and $\mathbf{D} = \text{diag}(\sigma_1^{-2}, \dots, \sigma_p^{-2})$ as follows:

$$\boldsymbol{\Omega} = -\mathbf{B}\mathbf{D} = (-\sigma_1^{-2}\mathbf{b}_1, \dots, -\sigma_p^{-2}\mathbf{b}_p).$$

To obtain the estimate of $\boldsymbol{\Omega}$, the SPMESL solves the following p independent scaled Lasso problems first: for $k = 1, \dots, p$,

$$(\hat{\mathbf{b}}_k, \hat{\sigma}_k) = \underset{\beta_k \in \mathbb{R}^p: \beta_{kk} = -1, \sigma_k > 0}{\text{argmin}} \quad \frac{\|\mathbf{X}_k - \sum_{j \neq k} \beta_{jk} \mathbf{X}_j\|_2^2}{2n\sigma_k} + \frac{\sigma_k}{2} + \lambda_0 \sum_{j \neq k} |\beta_{jk}|. \quad (5)$$

Note that the SPMESL in Sun and Zhang (2013) originally considers $\lambda_0 \sum_{j \neq k} \sqrt{s_{jj}} |\beta_{jk}|$ instead of $\lambda_0 \sum_{j \neq k} |\beta_{jk}|$ to penalize the coefficients on the same scale. In this paper, we assume that the columns of the data matrix \mathbf{X} are centered and scaled to $\mathbf{X}_k^T \mathbf{X}_k = n$ for $k = 1, \dots, p$. Thus, $s_{jj} = 1$ for $j = 1, \dots, p$. This assumption does not affect the estimation performance of the precision matrix as the scaled Lasso has the scale-equivariant property in the response variable as explained in the previous section. We can easily recover the estimate $\hat{\boldsymbol{\Omega}}^o = (\hat{\omega}_{jk}^o)_{1 \leq j, k \leq p}$ from the data in the original scale by the following Proposition 1:

Proposition 1 *Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{C}$ be a data matrix and the scaled data matrix with $\mathbf{C} = \text{diag}(s_{11}^{-1/2}, \dots, s_{pp}^{-1/2})$, where $s_{jj} > 0$ for $j = 1, \dots, p$. Denote $\hat{\boldsymbol{\Omega}}^o$ as the estimate of the precision matrix by applying the scaled Lasso in (5) with the penalty term $\lambda_0 \sum_{j \neq k} \sqrt{s_{jj}} |\beta_{jk}|$ column by column with \mathbf{X} . Similarly, denote $\hat{\boldsymbol{\Omega}}^C = (\hat{\omega}_{jk}^C)_{1 \leq j, k \leq p}$ as the estimate by the scaled Lasso in (5) with $\tilde{\mathbf{X}}$. Then, $\hat{\boldsymbol{\Omega}}^o = \mathbf{C} \hat{\boldsymbol{\Omega}}^C \mathbf{C}$.*

Proof By the definition of $\tilde{\mathbf{X}}$, the k th column of $\tilde{\mathbf{X}}$ is $\tilde{\mathbf{X}}_k = \mathbf{X}_k / \sqrt{s_{kk}}$. Let $\hat{\mathbf{B}}^C = (\hat{b}_{jk}^C)_{1 \leq j, k \leq p}$ and $(\hat{\sigma}_{k,C})_{1 \leq k \leq p}$ be the solutions of the p scaled Lasso problem in (5) with $\tilde{\mathbf{X}}$. We further let $\hat{\mathbf{B}}^o = (\hat{b}_{jk}^o)_{1 \leq j, k \leq p}$ and $(\hat{\sigma}_{k,o})_{1 \leq k \leq p}$ be the solutions of the following scaled Lasso problems: for $k = 1, \dots, p$,

$$(\hat{\mathbf{b}}_k^o, \hat{\sigma}_{k,o}) = \underset{\beta_k \in \mathbb{R}^p: \beta_{kk} = -1, \sigma_k > 0}{\text{argmin}} \quad \frac{\|\mathbf{X}_k - \sum_{j \neq k} \beta_{jk} \mathbf{X}_j\|_2^2}{2n\sigma_k} + \frac{\sigma_k}{2} + \lambda_0 \sum_{j \neq k} \sqrt{s_{jj}} |\beta_{jk}|. \quad (6)$$

By substituting $\tilde{\mathbf{X}}$ with $\mathbf{X}\mathbf{C}$ in (5) and the reparameterization with $\tilde{\beta}_{jk} = \beta_{jk} / \sqrt{s_{jj}}$ for $j \neq k$, the k th scaled Lasso problem becomes

$$(\hat{\mathbf{b}}_k^C, \hat{\sigma}_{k,C}) = \underset{\tilde{\beta}_k \in \mathbb{R}^p: \tilde{\beta}_{kk} = -1, \sigma_k > 0}{\text{argmin}} \quad \frac{\|\mathbf{X}_k / \sqrt{s_{kk}} - \sum_{j \neq k} \tilde{\beta}_{jk} \mathbf{X}_j\|_2^2}{2n\sigma_k} + \frac{\sigma_k}{2} + \lambda_0 \sum_{j \neq k} \sqrt{s_{jj}} |\tilde{\beta}_{jk}|. \quad (7)$$

From the forms of the problems (6) and (7), the relationship $\hat{\mathbf{b}}_k^o = \sqrt{s_{kk}} \hat{\mathbf{b}}_k^C$ and $\hat{\sigma}_{k,o} = \sqrt{s_{kk}} \hat{\sigma}_{k,C}$ hold by the scale equivariant property of the scaled Lasso. In addition, by the reparameterization, $\hat{b}_{jk}^o = \sqrt{s_{kk}/s_{jj}} \hat{b}_{jk}^C$ for $1 \leq j, k \leq p$. Combining the above relationships, the (j, k) th element $\hat{\omega}_{jk}^o$ of the precision matrix estimate $\hat{\boldsymbol{\Omega}}^o$ is represented as

$$\hat{\omega}_{jk}^o = -\hat{b}_{jk}^o \hat{\sigma}_{k,o}^{-2} = -(s_{jj}s_{kk})^{-1/2} \hat{b}_{jk}^C \hat{\sigma}_{k,C}^{-2} = (s_{jj}s_{kk})^{-1/2} \hat{\omega}_{jk}^C.$$

Hence, $\hat{\boldsymbol{\Omega}}^o = \mathbf{C} \hat{\boldsymbol{\Omega}}^C \mathbf{C}$.

Note that the result $\hat{\boldsymbol{\Omega}}^o = \mathbf{C}\hat{\boldsymbol{\Omega}}^C\mathbf{C}$ in Proposition 1 is consistent with the property of $(\text{Var}(\mathbf{A}\mathbf{z}))^{-1} = \mathbf{A}^{-T}\text{Var}(\mathbf{z})^{-1}\mathbf{A}^{-1}$ for a p -dimensional random vector \mathbf{z} and a nonsingular matrix $\mathbf{A} \in \mathbb{R}^{p \times p}$.

After solving p independent scaled Lasso problems, the estimate $\hat{\boldsymbol{\Omega}}_1 = -\hat{\mathbf{B}}\hat{\mathbf{D}} = (\hat{\omega}_{jk,1})_{1 \leq j,k \leq p}$ of the precision matrix is obtained. However, the estimate $\hat{\boldsymbol{\Omega}}_1$ is not symmetric in general. To find the symmetric estimate of the precision matrix using the current estimate $\hat{\boldsymbol{\Omega}}_1$, the SPMESL considers solving the following linear programming problem as in Yuan (2010):

$$\hat{\boldsymbol{\Omega}} = \underset{\mathbf{M}: \mathbf{M}^T = \mathbf{M}}{\text{argmin}} \|\mathbf{M} - \hat{\boldsymbol{\Omega}}_1\|_1. \quad (8)$$

Remark that the authors of the SPMESL consider the above linear programming problem for the symmetrization step in Sun and Zhang (2013), but they applied the following symmetrization step in the implemented R package `scalreg`:

$$\hat{\omega}_{jk} = \hat{\omega}_{kj} = \hat{\omega}_{jk,1}I(|\hat{\omega}_{jk,1}| \leq |\hat{\omega}_{kj,1}|) + \hat{\omega}_{kj,1}I(|\hat{\omega}_{jk,1}| > |\hat{\omega}_{kj,1}|), \quad (9)$$

which is applied in the CLIME and the theoretical properties are developed on this symmetrization (Cai et al. 2011b). In addition, for the high-dimensional data, the symmetrization applied in the CLIME is favorable as its computational cost is cheap and it is easily parallelizable. For these reasons, we apply the symmetrization step (9) in the proposed algorithm.

As stated in the previous section, the scaled Lasso guarantees the consistency of the regression coefficients and the error variance under the compatibility condition. Thus, the SPMESL also guarantees column-wise consistency of $\hat{\boldsymbol{\Omega}}_1$ under the compatibility conditions, which is independently defined in each column of $\hat{\boldsymbol{\Omega}}_1$, as the SPMESL applies the scaled Lasso column wise. To derive the overall consistency of $\hat{\boldsymbol{\Omega}}$, the authors of the SPMESL considers the capped ℓ_1 sparsity and the invertibility conditions as follows.

- (i) Capped ℓ_1 sparsity condition: For a certain ϵ_0 , λ_0^* not depending on j and an index set $T_j \subset \{1, 2, \dots, p\} \setminus \{j\}$, the capped ℓ_1 sparsity of the j th column with $t_j > 0$ is defined as

$$|T_j| + \sum_{k \neq j, k \notin S_j} \frac{|\omega_{kj}|\sqrt{\sigma_{kk}}}{(1 - \epsilon_0)\sqrt{\omega_{jj}\lambda_0^*}} \leq a_j.$$

- (ii) Invertibility condition: Let $\mathbf{W} = \text{diag}(\sigma_{11}, \dots, \sigma_{pp})$ and $\mathbf{R} = \mathbf{W}^{-1/2}\boldsymbol{\Sigma}\mathbf{W}^{-1/2}$. Further, let $T_j \subseteq Q_j \subseteq \{1, 2, \dots, p\} \setminus \{j\}$. The invertibility condition is defined as

$$\inf_j \left\{ \frac{\mathbf{u}^T \mathbf{R}_{-j, -j} \mathbf{u}}{\|\mathbf{u}_{Q_j}\|_2^2} : \mathbf{u} \in \mathbb{R}^p, \mathbf{u}_{Q_j} \neq 0, 1 \leq j \leq p \right\} \geq c_*$$

with a fixed constant $c_* > 0$. Note that the invertibility condition holds if the spectral norm of $\mathbf{R}^{-1} = \mathbf{D}^{1/2}\boldsymbol{\Omega}\mathbf{D}^{1/2}$ is bounded (i.e., $\|\mathbf{R}^{-1}\|_2 \leq c_*^{-1}$).

With some modifications on the capped ℓ_1 sparsity and the invertibility conditions, the authors of the SPMESL derive several conditions on λ_0^* that guarantees the estimation consistency of the precision matrix under the spectral norm. Among them, for practical usage, we consider two conditions on a penalty level $\lambda_0 \geq \lambda_0^*$ as follows:

- Union bound for p applications of the scaled Lasso (Theorem 2 in Sun and Zhang (2013)):

$$\lambda_0 = A\sqrt{4n^{-1}\log p} \text{ for } A > 1. \quad (10)$$

- Probabilistic error bound (Theorem 13 in Sun and Zhang (2013)):

$$\lambda_0 = AL_n(k/p) \text{ for } 1 < A \leq \sqrt{2}, \quad (11)$$

where k is a real solution of $k = L_1^4(k/p) + 2L_1^2(k/p)$, $L_n(t) = n^{-1/2}\Phi^{-1}(1 - t)$, and $\Phi^{-1}(t)$ is the standard normal quantile function.

Note that a real solution of the equation $k - L_1^4(k/p) + 2L_1^2(k/p) = 0$ can easily be found by applying the bisection method. For instance, we demonstrate two real solutions for $p = 100, 1000$ in Figure 1 with the values of $k - L_1^4(k/p) + 2L_1^2(k/p)$. For $p = 1000$ and $n = 100$, $\lambda_{ub} = \sqrt{4n^{-1} \log p} \approx 0.5257$ and $\lambda_{pb} = \sqrt{2L_n(k/p)} \approx 0.2810$ with $k = 23.4748$ while $\lambda_{univ} = \sqrt{2n^{-1} \log(p-1)} \approx 0.3717$, where λ_{ub} is the penalty level derived by the union bound, λ_{pb} is the penalty level derived by the probabilistic error bound, and λ_{univ} is the universal penalty level used in the scaled lasso. In the paper of Sun and Zhang (2013), the penalty level derived by the probabilistic error bound is suggested for the SPMESL. However, there are no comparison results for the three penalty levels λ_{univ} , λ_{ub} , and λ_{pb} . We conduct a comparison of performances for identifying the nonzero elements of Ω by the three penalty levels above in Section 4 to provide a guideline for the penalty level λ_0 .

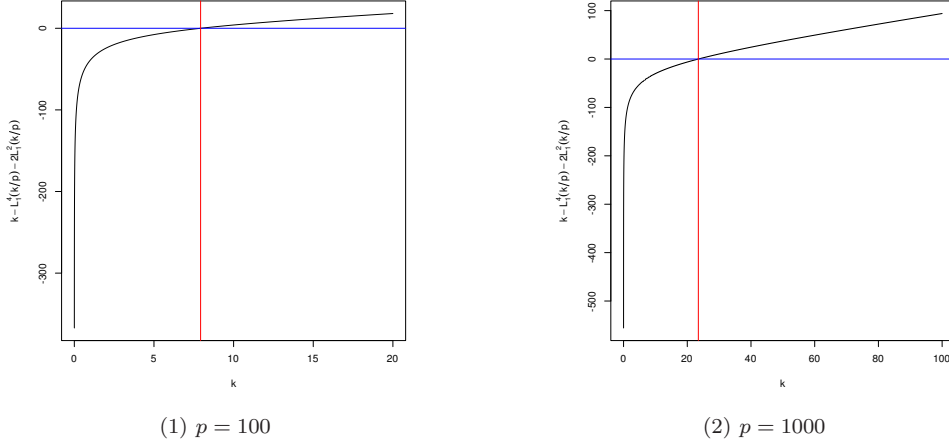


Fig. 1 Plots of $k - L_1^4(k/p) + 2L_1^2(k/p)$ for $p = 100, 1000$. Vertical red lines denote the solutions of $k - L_1^4(k/p) + 2L_1^2(k/p) = 0$ obtained by the bisection method.

3 Efficient Coordinate Descent Algorithm for SPMESL and its GPU-parallelization

The original algorithm for the scaled Lasso and the SPMESL adopt the LARS algorithm, which provides the whole solution path of the Lasso regression problem, and its implemented R package `scalreg` is available in the Comprehensive R Archive Network (CRAN) repository. As mentioned in the Introduction, we empirically observed that the block CD algorithms for the scaled Lasso and the SPMESL do not need a whole solution path of the standard Lasso problem in their sub-problems, where the sub-problem denotes the minimization problem in Step 1 of the scaled Lasso problem. To describe our empirical observation, we consider an example with a linear regression model $y_i = \mathbf{x}_i^T \beta + \epsilon_i$ and $\epsilon_i \sim N(0, \sigma^2)$ for $i = 1, 2, \dots, 250$, where the true parameter $\beta = (\beta_2^T, \beta_{-1}^T, \beta_0^T)^T \in \mathbb{R}^{500}$, $\beta_2 = (2, \dots, 2)^T \in \mathbb{R}^5$, $\beta_{-1} = (-1, \dots, -1)^T \in \mathbb{R}^5$, $\beta_0 = (0, \dots, 0)^T \in \mathbb{R}^{490}$, and $\sigma = 3, 5$. We set the initial value of (β, σ) as $(\mathbf{0}_{500 \times 1}, 1)$. As shown in Figure 2, the numbers of iterations for the convergence of $\hat{\sigma}$ are less than 10 when the true parameter $\sigma = 3, 5$ and $\lambda_0 = \sqrt{2n^{-1} \log p}$. This implies that we do not need to obtain the whole solution paths of p lasso problems with respect to all λ values.

Thus, the calculation of the whole solution path by the LARS algorithm is inefficient for the scaled Lasso and the SPMESL. In addition, the scaled Lasso and the SPMESL iteratively solve the lasso problem with the penalty $\lambda_r = \hat{\sigma}^{(r)} \lambda_0$ in their inner iteration, where λ_r denotes the penalty level at the r th iteration. As $\hat{\sigma}^{(r)}$ converges to the minimizer of (1), the difference between λ_r and λ_{r-1} decreases as the iteration proceeds. This denotes that the Lasso estimate in the current iteration is not far from that in the next iteration. From this observation, the warm start strategy, which denotes that the solution in the previous iteration is used as an initial value of the next iteration, is favorable and can efficiently accelerate the algorithm for the Lasso regression problem in the inner iteration.

To fully utilize the warm start strategy, we consider the coordinate descent (CD) algorithm for the Lasso regression problem in the inner iteration. This is because it is known that the CD algorithm with

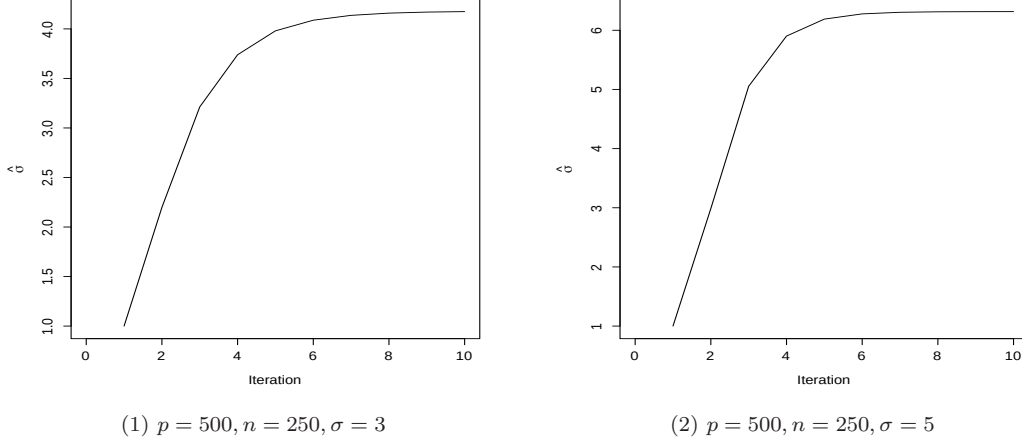


Fig. 2 Plots of the estimates of $\hat{\sigma}$ for $\sigma = 3, 5$ along with the number of iterations.

the warm start strategy is efficient for the Lasso regression problem and has an advantage for memory consumption (Wu and Lange 2008). In addition, the SPMESL needs solving p independent scaled Lasso problems to obtain the estimate of the precision matrix. We develop the GPU-parallel CD algorithm for the SPMESL, which updates p coordinates simultaneously with GPUs. In the following subsections, we introduce the CD algorithm for the subproblem of the scaled Lasso and GPU-parallel CD algorithm for the SPMESL in detail.

3.1 CD Algorithm for subproblem of the scaled Lasso

In this subsection, we focus on the following subproblem of the scaled Lasso with a given λ_0 :

$$\hat{\beta}^{(r)} = \underset{\beta}{\operatorname{argmin}} \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda^{(r-1)} \|\beta\|_1, \quad (12)$$

where $\mathbf{y} \in \mathbb{R}^n$ is a response vector, $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a design matrix, $\lambda^{(r-1)} = \hat{\sigma}^{(r-1)} \lambda_0$, and $\hat{\sigma}^{(r-1)} = \|\mathbf{y} - \mathbf{X}\hat{\beta}^{(r-1)}\|_2 / \sqrt{n}$ is the iterative solution for σ at the $(r-1)$ th iteration. For the notational simplicity, we use $\hat{\beta}$ to denote the r th iterative solution $\hat{\beta}^{(r)}$, and $\hat{\beta}^{[cur]}$ and $\hat{\beta}^{[next]}$ denote the current and the next iterative solution in the CD algorithm, respectively. To apply the warm start strategy, we set the initial value $\hat{\beta}^{[cur]}$ for $\hat{\beta}$ to $\hat{\beta}^{(r-1)}$. In this paper, we consider the cyclic CD algorithm with an ascending order (i.e., coordinate-wise update from the smallest index to the largest index). Subsequently, for $j = 1, \dots, p$, the CD algorithm updates $\hat{\beta}_j^{[next]}$ by following equations:

$$\mathbf{e}_j = \mathbf{y} - \sum_{l < j} \mathbf{X}_l \hat{\beta}_l^{[next]} - \sum_{l > j} \mathbf{X}_l \hat{\beta}_l^{[cur]}, \quad a_j = \mathbf{x}_j^T \mathbf{e}_j / n + \hat{\beta}_j^{[cur]}, \quad \hat{\beta}_j^{[next]} = \operatorname{Soft}_{\lambda}(a_j),$$

where $\operatorname{Soft}_{\lambda}(a_j) = \operatorname{sign}(a_j)(|a_j| - \lambda)_+$ is the soft-thresholding operator and $(x)_+ = \max(x, 0)$. The CD algorithm repeats the cyclic updates until the convergence occurs, where the common convergence criterion is the L_{∞} -norm of the difference between $\hat{\beta}^{[cur]}$ and $\hat{\beta}^{[next]}$ (i.e., $\|\hat{\beta}^{[next]} - \hat{\beta}^{[cur]}\|_{\infty}$). The whole CD algorithm with warm start strategy for the scaled Lasso is summarized in Algorithm 1.

3.2 CD Algorithm for SPMESL

As described in Section 2.2, the SPMESL estimates the precision matrix by solving the p scaled Lasso problems, where each column of the observed data matrix is considered as the response variable and the other columns are considered as the exploratory variables. To be specific, let $(\mathbf{x}^{(i)})^T = (X_{i1}, X_{i2}, \dots, X_{ip})^T \sim N(0, \mathbf{\Omega}^{-1})$ be the i th random sample and $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ be the precision matrix. Furthermore, let $\mathbf{x}_k =$

Algorithm 1 CD algorithm with warm start strategy for the scaled Lasso

Input: \mathbf{y} , \mathbf{X} , λ_0 , $\hat{\sigma}^{(0)} = 1$, $\hat{\beta}^{(0)} = \mathbf{0}$, convergence tolerance δ .

```

1: repeat  $r = 0, 1, 2, \dots$ 
2:    $\lambda \leftarrow \hat{\sigma}^{(r)} \lambda_0$  ▷ Initialization of lasso subproblem
3:    $\hat{\beta}^{[cur]} \leftarrow \hat{\beta}^{(r)}, \hat{\beta}^{[next]} \leftarrow \hat{\beta}^{[cur]}$  ▷ Warm start strategy
4:   repeat  $m = 0, 1, 2, \dots$ 
5:      $\hat{\beta}^{[cur]} \leftarrow \hat{\beta}^{[next]}$ 
6:     for  $j = 1, \dots, p$  do
7:        $\mathbf{e}_j = \mathbf{y} - \sum_{l < j} \mathbf{X}_l \hat{\beta}_l^{[next]} - \sum_{l > j} \mathbf{X}_l \hat{\beta}_l^{[cur]}$ 
8:        $a_j = \mathbf{X}_j^T \mathbf{e}_j / n + \hat{\beta}_j^{[cur]}$ 
9:        $\hat{\beta}_j^{[next]} = \text{Soft}_\lambda(a_j)$ 
10:    end for
11:    until  $\|\hat{\beta}^{[next]} - \hat{\beta}^{[cur]}\|_\infty < \delta$  ▷ End of lasso subproblem
12:     $\hat{\beta}^{(r+1)} \leftarrow \hat{\beta}^{[next]}$ 
13:     $\hat{\sigma}^{(r+1)} = \frac{\|\mathbf{y} - \mathbf{X} \hat{\beta}^{(r+1)}\|_2}{\sqrt{n}}$ 
14:  until  $|\hat{\sigma}^{(r+1)} - \hat{\sigma}^{(r)}| < \delta$ 
Output:  $\hat{\beta} \leftarrow \hat{\beta}^{(r+1)}, \hat{\sigma} \leftarrow \hat{\sigma}^{(r+1)}$ 

```

$(X_{1k}, \dots, X_{nk})^T$ be the k th column vector of the observed data matrix $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_p) \in \mathbb{R}^{n \times p}$. The CD algorithm with the warm start strategy for the SPMESL independently applies the CD algorithm in Section 3.1 to the subproblem (5) for $k = 1, 2, \dots, p$. The main procedures in the CD algorithm for the SPMESL are summarized in the following two steps:

- **Updating $\hat{\beta}_{-k}$ for $1 \leq k \leq p$:** Applying the CD algorithm with the warm-start strategy for the following lasso subproblem: for $k = 1, 2, \dots, p$,

$$\hat{\beta}_{-k} = \underset{\beta_{-k} : \beta_{kk}=0}{\text{argmin}} \frac{\|\mathbf{x}_k - \mathbf{X}_{-k} \beta_{-k}\|_2^2}{2n} + \hat{\sigma}_k \lambda_0 \|\beta_{-k}\|_1, \quad (13)$$

where $\beta_{-k} = (\beta_{1,k}, \dots, \beta_{k-1,k}, 0, \beta_{k+1,k}, \dots, \beta_{p,k})^T \in \mathbb{R}^p$.

- **Updating $\hat{\sigma}_k$ for $1 \leq k \leq p$:** For given λ_0 and $\hat{\beta}_{-k}$ s, $\hat{\sigma}_k$ s are obtained by the equation

$$\hat{\sigma}_k = \frac{\|\mathbf{x}_k - \mathbf{X}_{-k} \hat{\beta}_{-k}\|_2}{\sqrt{n}}, \quad (14)$$

where $\hat{\beta}_{-k} = (\hat{\beta}_{1,k}, \dots, \hat{\beta}_{k-1,k}, 0, \hat{\beta}_{k+1,k}, \dots, \hat{\beta}_{p,k})^T$ is the solution to the problem (13).

The CD algorithm for the SPMESL independently repeats the updating $\hat{\beta}_{-k}$ and $\hat{\sigma}_k$ until convergence occurs for $k = 1, 2, \dots, p$. After solving the p scaled Lasso problems, the final estimate $\hat{\boldsymbol{\Omega}}$ of the precision matrix by the SPMESL is obtained by the symmetrization (9). The whole CD algorithm with warm start strategy for the SPMESL is summarized in Algorithm 2.

3.3 Parallel CD algorithm for SPMESL using GPU

As we described in Section 3.2, the CD algorithm for the SPMESL solves p independent scaled Lasso problems. From this independence structure, p elements in $\mathbf{B} = (\beta_{jk})$ can be updated in parallel, where $(p-1)$ elements can simultaneously be updated in practice since $\beta_{kk} = 0$ is fixed. In addition, the computation of $\hat{\sigma}$ is independent as well in the sense that the update equation for $\hat{\sigma}_k$ only needs information

Algorithm 2 CD algorithm with warm start strategy for the SPMESL

Input: \mathbf{X} , λ_0 , $\hat{\sigma}^{(0)} = 1$, $\mathbf{B} = (\hat{\beta}_{ij}^{(0)}) = (\hat{\beta}_{-1}^{(0)}, \dots, \hat{\beta}_{-p}^{(0)}) = \mathbf{0}$, convergence tolerance δ .

```

1: for  $k = 1, \dots, p$  do
2:    $(\hat{\beta}_{-k}, \hat{\sigma}_k) \leftarrow$  Apply Algorithm 1 with  $(\mathbf{x}_k, \mathbf{X}_{-k} = (\mathbf{x}_1, \dots, \mathbf{x}_{k-1}, \mathbf{x}_{k+1}, \dots, \mathbf{x}_p), \lambda_0)$ 
3: end for
4: for  $j = 1, \dots, p$  do ▷ Calculation of initial estimate of  $\Omega$ 
5:    $\hat{\omega}_{jj} = \hat{\sigma}_j^{-2}$ 
6:   for  $k = 1, \dots, p$  do
7:     if  $k \neq j$  then
8:        $\hat{\omega}_{jk} = -\hat{\beta}_{jk} \hat{\omega}_{kk}$ 
9:     end if
10:  end for
11: end for
12: for  $j = 1, \dots, p-1$  do ▷ Symmetrization of  $\Omega$ 
13:   for  $k = j+1, \dots, p$  do
14:     if  $|\hat{\omega}_{jk}| > |\hat{\omega}_{kj}|$  then
15:        $\hat{\omega}_{jk} \leftarrow \hat{\omega}_{kj}$ 
16:     else
17:        $\hat{\omega}_{kj} \leftarrow \hat{\omega}_{jk}$ 
18:     end if
19:   end for
20: end for
Output:  $\hat{\Omega} = (\hat{\omega}_{jk})$ 

```

of $\hat{\beta}_{-k}$. To describe the proposed parallel CD algorithm, we consider the following joint minimization problem, which combines p scaled Lasso problems:

$$(\hat{\mathbf{B}}, \hat{\sigma}) = \underset{\{\beta_{-k}, \sigma_k\}_{k=1}^p}{\operatorname{argmin}} \sum_{k=1}^p \left\{ \frac{\|\mathbf{x}_k - \mathbf{X}\beta_{-k}\|_2^2}{2n\sigma_k} + \frac{\sigma_k}{2} + \lambda_0 \|\beta_{-k}\|_1 \right\}, \quad (15)$$

where $\mathbf{B} = (\beta_{-1}, \dots, \beta_{-p})$ and $\beta_{-k} = (\beta_{1,k}, \dots, \beta_{k-1,k}, 0, \beta_{k+1,k}, \dots, \beta_{p,k})^T$. As the updating equation (14) for $\hat{\sigma}_k$ is simple and easily parallelizable, we focus on the update for $\hat{\mathbf{B}}$ in this subsection. For the given iterative solution $\hat{\sigma}^{(r)} = (\hat{\sigma}_1^{(r)}, \dots, \hat{\sigma}_p^{(r)})$, the subproblem for updating $\hat{\mathbf{B}}^{(r+1)}$ can be represented as follows:

$$\begin{aligned} \hat{\mathbf{B}}^{(r+1)} &= \underset{\beta_{-1}, \dots, \beta_{-p}}{\operatorname{argmin}} \sum_{k=1}^p g(\beta_{-k}; \hat{\sigma}_k^{(r)}, \lambda_0) = \sum_{k=1}^p \left\{ \frac{\|\mathbf{x}_k - \mathbf{X}\beta_{-k}\|_2^2}{2n} + \lambda_k \|\beta_{-k}\|_1 \right\} \\ &= \underset{\mathbf{B}: b_{kk}=0, 1 \leq k \leq p}{\operatorname{argmin}} f(\mathbf{B}; \hat{\sigma}^{(r)}, \lambda_0) = \frac{1}{2n} \|\mathbf{X} - \mathbf{XB}\|_F^2 + \sum_{k=1}^p \lambda_k \|\beta_{-k}\|_1, \end{aligned} \quad (16)$$

where $\|\mathbf{A}\|_F = \operatorname{tr}(\mathbf{A}^T \mathbf{A}) = (\sum_{i,j} a_{ij}^2)^{1/2}$ is the Frobenius norm of a matrix \mathbf{A} and $\lambda_j = \hat{\sigma}_j^{(k)} \lambda_0$. For the notational simplicity, hereafter, we denote $f(\mathbf{B}; \hat{\sigma}^{(r)}, \lambda_0)$ and $\hat{\mathbf{B}}^{(r+1)}$ as $f(\mathbf{B})$ and $\hat{\mathbf{B}}$, respectively. As $f(\mathbf{B})$ is the sum of the smooth function of \mathbf{B} (the square of the Frobenius norm) and non-smooth convex functions, $f(\mathbf{B})$ satisfies the conditions of Theorem 4.1 in Tseng (2001). Thus, the iterative sequence $\{\hat{\mathbf{B}}^{(r)}\}$ in the CD algorithm with cyclic rule converges to the stationary point of $f(\mathbf{B})$, where $\hat{\mathbf{B}}^{(r)}$ denotes the iterative solution of the CD algorithm at the r th iteration, and each iteration is counted when one cycle is finished (i.e., $\beta_{12}, \dots, \beta_{p-1,p}$ have been updated.).

To develop the parallel CD (PCD) algorithm, we consider a row-wise update for $\hat{\mathbf{B}}$, which is one of the possible orderings in the cyclic rule. The main idea of the parallel CD algorithm is that the p Lasso subproblems are independent in the sense that $\beta_{j,k}$ does not need information $(\beta_{j,l})$ for $l \neq k$. To be specific, let $\hat{\beta}^{(j),[cur]} = (\hat{\beta}_{j,1}^{[cur]}, \dots, \hat{\beta}_{j,j-1}^{[cur]}, 0, \hat{\beta}_{j,j+1}^{[cur]}, \dots, \hat{\beta}_{j,p}^{[cur]})$ and $\hat{\beta}^{(j),[next]}$ be the j th row of the current

and next iterative solutions of $\hat{\mathbf{B}}$, respectively. The following Proposition 2 shows that the row of $\hat{\mathbf{B}}$ can be updated in parallel.

Proposition 2 Let $\mathbf{E} = (\mathbf{e}_1, \dots, \mathbf{e}_p)$ be a current residual matrix defined with $\mathbf{e}_k = \mathbf{x}_k - \mathbf{X}\hat{\boldsymbol{\beta}}_{-k}^{[cur]}$, and let $\hat{\mathbf{B}}^{[cur]}$ and $\hat{\mathbf{B}}^{[next]}$ be the current and next iterative solution for the coefficient of the joint Lasso subproblem, respectively. Suppose we update the rows of $\hat{\mathbf{B}}^{[next]}$ from the first row to the last row. Then, each row $\hat{\boldsymbol{\beta}}^{(j),[next]}$ of $\hat{\mathbf{B}}^{[next]}$ for $j = 1, \dots, p$ can simultaneously be updated by following updating equations:

$$\mathbf{a}_j = (a_{jk})_{k=1}^p = \mathbf{x}_j^T \mathbf{E} / n + \hat{\boldsymbol{\beta}}^{(j),[cur]}, \quad a_{jj} \leftarrow 0, \quad \hat{\boldsymbol{\beta}}^{(j),[next]} = \mathbf{S}_{\lambda_1, \dots, \lambda_p}(\mathbf{a}_j), \quad \mathbf{E} \leftarrow \mathbf{E} + \mathbf{x}_j(\hat{\boldsymbol{\beta}}^{(j),[cur]} - \hat{\boldsymbol{\beta}}^{(j),[next]}),$$

where $\mathbf{S}_{\lambda_1, \dots, \lambda_p}(\mathbf{x}) = (\text{Soft}_{\lambda_j}(x_j))_{1 \leq j \leq p}$, $\text{Soft}_{\lambda}(x) = \text{sign}(x)(|x| - \lambda)_+$, and $(x)_+ = \max(x, 0)$.

Proof As described in Section 3.1, the CD algorithm updates each element $\hat{\beta}_{j,k}^{[next]}$ in $\hat{\boldsymbol{\beta}}^{(j),[next]}$ by

$$\mathbf{e}_k = \mathbf{x}_k - \sum_{l < k} \mathbf{x}_l \hat{\beta}_{-l}^{[next]} - \sum_{l > k} \mathbf{x}_l \hat{\beta}_{-l}^{[cur]}, \quad a_{jk} = \mathbf{x}_j^T \mathbf{e}_k / n + \hat{\beta}_{j,k}^{[cur]}, \quad \hat{\beta}_{j,k}^{[next]} = \text{Soft}_{\lambda}(a_{jk}).$$

Consider updating the first row of $\hat{\mathbf{B}}^{[next]}$ by the CD algorithm. Then, for $k = 2, \dots, p$, the above updating equations becomes

$$\mathbf{e}_k = \mathbf{x}_k - \sum_{l=2}^p \mathbf{x}_l \hat{\beta}_{-l}^{[cur]}, \quad a_{1k} = \mathbf{x}_1^T \mathbf{e}_k / n + \hat{\beta}_{1,k}^{[cur]}, \quad \hat{\beta}_{1,k}^{[next]} = \text{Soft}_{\lambda}(a_{1k}).$$

The update for $\hat{\beta}_{1,k}^{[next]}$ only needs information on the current residual vector \mathbf{e}_k and $\hat{\beta}_{1,k}^{[cur]}$. Hence, the updates of $\hat{\beta}_{1,k}^{[next]}$ and $\hat{\beta}_{1,l}^{[next]}$ for $k \neq l$ are independent in the sense that $\hat{\beta}_{1,k}^{[next]}$ does not depend on $\hat{\beta}_{1,k}^{[next]}$, and vice versa. Combining these equations for $j = 2, \dots, p$, we can represent $(p-1)$ updating equations with the following vector form:

$$(0, \hat{\beta}_{12}^{[next]}, \dots, \hat{\beta}_{1p}^{[next]}) = (0, \text{Soft}_{\lambda_2}(a_{12}), \dots, \text{Soft}_{\lambda_p}(a_{1p})),$$

where $\lambda_j = \hat{\sigma}_j \lambda_0$ and $\mathbf{a}_1^T = (0, \mathbf{e}_2^T \mathbf{x}_1 / n, \dots, \mathbf{e}_p^T \mathbf{x}_1 / n) + (0, \hat{\beta}_{1,2}^{[cur]}, \dots, \hat{\beta}_{1,p}^{[cur]}) = \mathbf{x}_1^T \mathbf{E} / n + \hat{\boldsymbol{\beta}}^{(1),[cur]} - (\mathbf{x}_1^T \mathbf{e}_1 / n) \mathbf{i}_1$, where $\hat{\beta}_{1,1}^{[cur]} = 0$ and \mathbf{i}_j is the j th row of p -dimensional identity matrix. After updating $\hat{\boldsymbol{\beta}}^{(1),[next]}$, the current residual matrix is also updated by $\mathbf{E} \leftarrow \mathbf{E} + \mathbf{X}_1(\hat{\boldsymbol{\beta}}^{(1),[cur]} - \hat{\boldsymbol{\beta}}^{(1),[next]})$. Then, the updated \mathbf{e}_k becomes $\mathbf{x}_k - \mathbf{x}_1 \hat{\beta}_{1,k}^{[next]} - \sum_{l=2}^p \mathbf{x}_l \hat{\beta}_{l,k}^{[cur]}$. Using these equations, we can express a general form of updating equations as

$$\mathbf{a}_j = (a_{jk})_{k=1}^p = \mathbf{x}_j^T \mathbf{E} / n + \hat{\boldsymbol{\beta}}^{(j),[cur]}, \quad a_{jj} \leftarrow 0, \quad \hat{\boldsymbol{\beta}}^{(j),[next]} = \mathbf{S}_{\lambda_1, \dots, \lambda_p}(\mathbf{a}_j), \quad \mathbf{E} \leftarrow \mathbf{E} + \mathbf{x}_j(\hat{\boldsymbol{\beta}}^{(j),[cur]} - \hat{\boldsymbol{\beta}}^{(j),[next]}),$$

where, for computational simplicity, we calculate \mathbf{a}_j by $\mathbf{x}_j^T \mathbf{E} / n + \hat{\boldsymbol{\beta}}^{(j),[cur]}$ and then set $a_{jj} = 0$. Applying this sequence of updating equations from the first row ($j = 1$) to the last row ($j = p$) of $\hat{\mathbf{B}}^{[next]}$ is equivalent to the cyclic CD algorithm for the joint Lasso subproblem.

In Proposition 2, the row-wise updating equations consist of basic linear algebra operations such as matrix-matrix multiplication and element-wise soft-thresholding, which are adequate for parallel computation using GPUs. To fully utilize the GPUs, we use the cuBLAS library for linear algebra operations and develop CUDA kernel functions for the element-wise soft-thresholding, parallel update for $\hat{\boldsymbol{\sigma}}$, and symmetrization of $\hat{\boldsymbol{\Omega}}$. We refer this CD algorithm to the parallel CD (PCD) algorithm in the sense of that p elements in each row of $\hat{\mathbf{B}}$ are simultaneously updated if p GPUs (i.e., p CUDA cores) are available. Note that $\hat{\beta}_{jj}$ for $j = 1, \dots, p$ are fixed with 0 in the PCD algorithm, which is handled explicitly in the implementation. The whole PCD algorithm with warm start strategy for the SPMESL is summarized in Algorithm 3. In Algorithm 3, we use a convergence criterion $\|\mathbf{B}^{(r+1)} - \mathbf{B}^{(r)}\|_{\infty} < \delta$ to check the convergence of $\mathbf{B}^{(r)}$, which is different to the convergence criterion $\|\boldsymbol{\beta}_{-j}^{(r+1)} - \boldsymbol{\beta}_{-j}^{(r)}\|_{\infty} < \delta$ in the CD algorithm. To ensure that the CD and the PCD algorithm provide the same solution, we show that the PCD algorithm obtains a solution that is sufficiently close to the solution of the CD algorithm if two algorithms use the same initial values in the following Theorem 1:

Theorem 1 For a given vector $(\hat{\sigma}_1, \dots, \hat{\sigma}_p)$, a tuning parameter λ_0 , and a convergence tolerance $\delta > 0$, let $\beta_{-k}^{(r)}$ and $\mathbf{b}_k^{(r)}$ be iterative solutions at the r th iteration by the CD algorithm with a convergence criterion $\|\beta_{-k}^{(r+1)} - \beta_{-k}^{(r)}\|_\infty < \delta$ and the PCD algorithm with a convergence criterion $\|\mathbf{B}^{(r+1)} - \mathbf{B}^{(r)}\|_\infty < \delta$, respectively. Let $\hat{\beta}_{-j}$ and $\hat{\mathbf{b}}_j$ be the solutions of the CD and the PCD that satisfy the given convergence criteria. Suppose that the CD and the PCD algorithms use the same initial point ($\hat{\beta}_{-k}^{(0)} = \hat{\mathbf{b}}_k^{(0)}, 1 \leq j \leq p$). Then, $\|\hat{\mathbf{b}}_k - \hat{\beta}_{-k}\|_\infty$ is also bounded by δ .

Proof As the function $g(\beta_{-k}; \hat{\sigma}_k, \lambda_0)$ is convex with respect to β_{-k} , it is easy to show that the coordinate-wise minimization of $g(\beta_{-k}; \hat{\sigma}_k, \lambda_0)$ satisfies the conditions (B1)–(B3) and (C1) in Tseng (2001). To see this, let $\eta = (\eta_1, \dots, \eta_{p-1})^T = (\beta_{1k}, \dots, \beta_{k-1,k}, \beta_{k+1,k}, \dots, \beta_{pk})^T$, $\tau = (\tau_1, \dots, \tau_{p-1})^T$, and $\tau_m = \lambda_m$ for $1 \leq m \leq k-1$ and $\tau_m = \lambda_{m+1}$ for $k \leq m \leq p-1$. We further let $f_0(\eta) = \frac{1}{2n} \|\mathbf{x}_j - \mathbf{X}_{-j}\eta\|_2^2$ and $f_m(\eta_m) = \tau_m |\eta_m|$ for $1 \leq m \leq p-1$, where $\mathbf{X}_{-j} = (\mathbf{x}_1, \dots, \mathbf{x}_{j-1}, \mathbf{x}_{j+1}, \dots, \mathbf{x}_p)$. Then, we can represent $g(\beta_{-k}; \hat{\sigma}_k, \lambda_0)$ as $f(\eta) = f_0(\eta) + \sum_{m=1}^{p-1} f_m(\eta_m)$. With this representation, it is trivial that f_0 is continuous on $\text{dom} f_0$ (B1) and f_0, f_1, \dots, f_{p-1} are lower semi-continuous (B3). As f_0, f_1, \dots, f_{p-1} are convex functions, the function $\eta_m \mapsto f(\eta_1, \dots, \eta_{p-1})$ for each $m \in \{1, \dots, p-1\}$ and $(\eta_l)_{l \neq m}$ is also convex and hemivariate (B2). The function f_0 also satisfies that $\text{dom} f_0$ is open and f_0 tends to ∞ at every boundary point of $\text{dom} f_0$ because the domain of $f_0(\eta)$ is \mathbb{R}^{p-1} and $f_0(\eta)$ is the sum of squares of errors. Thus, by Theorem 5.1 in Tseng (2001), the cyclic CD algorithm guarantees that $\beta_{-k}^{(r)}$ converges to a stationary point of $g(\beta_{-k}; \hat{\sigma}_k, \lambda_0)$. As the same updating order ($1 \rightarrow 2 \rightarrow \dots \rightarrow p$) and equation are applied with the same initial values in the proposed CD and PCD algorithms, the sequence $\{\mathbf{b}_k^{(r)}\}$ by the PCD is equivalent to the sequence $\{\beta_{-k}^{(r)}\}$. That is, $\beta_{-k}^{(r)} = \mathbf{b}_k^{(r)}$ for $r \geq 0$. Let K_{CD} and K_{PCD} be the iteration numbers that satisfies the convergence criteria of the CD and the PCD, respectively. As $\|\beta_{-k}^{(r)} - \beta_{-k}^{(r-1)}\|_\infty = \|\mathbf{b}_k^{(r)} - \mathbf{b}_k^{(r-1)}\|_\infty \leq \|\mathbf{B}^{(r)} - \mathbf{B}^{(r-1)}\|_\infty$, it is satisfied that $K_{PCD} \geq K_{CD}$. As the p -dimensional Euclidean space with L_∞ -norm is Banach space, the convergent sequence $\{\beta_{-k}^{(r)}\}$ and $\{\mathbf{b}_k^{(r)}\}$ is a Cauchy sequence. Thus, from the definition of the Cauchy sequence, for a given $\delta > 0$, there exists K such that $\|\beta_{-k}^{(u)} - \beta_{-k}^{(v)}\|_\infty < \delta$ for $u, v \geq K$. Take $K = K_{CD}$, $u = K_{CD}$, and $v = K_{PCD}$. Then, $\beta_{-k}^{(K_{CD})} = \hat{\beta}_{-k}$ and $\beta_{-k}^{(K_{PCD})} = \hat{\mathbf{b}}_k$. Hence, the L_∞ -norm of the difference of $\hat{\beta}_{-k}$ and $\hat{\mathbf{b}}_k$ is bounded by δ .

For convergence of σ_j , we also use the same convergence criterion $|\sigma_j^{(r)} - \sigma_j^{(r-1)}| < \delta$ as in the CD algorithm. To reduce the computational costs in the PCD algorithm, at each iteration, we remove some columns of \mathbf{B} in the problem if the corresponding σ_j satisfies the convergence criterion $|\sigma_j^{(r)} - \sigma_j^{(r-1)}| < \delta$. This additional procedure needs the rearrangement of the coefficient matrix \mathbf{B} . In the implementation, we use an index vector and a convergence flag vector to implement the additional rearrangement procedure efficiently. Thus, the PCD algorithm requires more computational costs than the CD algorithm for the SPMESL as the CD algorithm runs consequently for $j = 1, \dots, p$ and does not need the rearrangement procedure. In the next section, however, we numerically show that the PCD algorithm becomes more efficient compared to the CD algorithm when either the number of variables or the sample size increases, although the PCD has more computational costs.

4 Numerical Study

4.1 Data construction and simulation settings

In this section, we numerically investigate the computational efficiency of the proposed CD and PCD algorithms and the estimation performance of the SPMESL with comparisons to other existing methods. To proceed the comparison on various circumstances, we first consider four network structures for the precision matrix defined as follows:

Algorithm 3 Parallel CD algorithm with warm start strategy for the SPMESL

Input: \mathbf{X} , λ_0 , $\hat{\sigma}^{(0)} = 1$, $\mathbf{B}^{(0)} = (\hat{\beta}_{ij}^{(0)}) = (\hat{\beta}_{-1}^{(0)}, \dots, \hat{\beta}_{-p}^{(0)}) = \mathbf{0}$, convergence tolerance δ .

1: Set $n_c = p$, $I = (1, \dots, p)$, and $F = (1, \dots, 1)$

2: **repeat** $r = 0, 1, 2, \dots$

3: $\lambda \leftarrow (\hat{\sigma}_{I_1}^{(r)} \lambda_0, \dots, \hat{\sigma}_{I_{n_c}}^{(r)} \lambda_0)$

▷ Initialization of joint lasso subproblem

4: $\hat{\mathbf{B}}^{[cur]} \leftarrow \hat{\mathbf{B}}^{(r)}$, $\hat{\mathbf{B}}^{[next]} \leftarrow \hat{\mathbf{B}}^{[cur]}$

▷ Warm start strategy

5: $\mathbf{E} = (\mathbf{x}_{I_1}, \dots, \mathbf{x}_{I_{n_c}}) - \mathbf{X} \hat{\mathbf{B}}^{[cur]}$

6: **repeat** $m = 0, 1, 2, \dots$

7: $\hat{\mathbf{B}}^{[cur]} \leftarrow \hat{\mathbf{B}}^{[next]}$

8: **for** $j = 1, \dots, p$ **do**

9: $\mathbf{a} = \mathbf{x}_j^T \mathbf{E} / n + \hat{\beta}^{(j), [cur]}$

10: $a_j \leftarrow 0$

11: $\hat{\beta}^{(j), [next]} = \mathbf{S}_{\lambda_1, \dots, \lambda_p}(\mathbf{a})$

12: $\mathbf{E} = \mathbf{E} + \mathbf{x}_i(\hat{\beta}^{(j), [cur]} - \hat{\beta}^{(j), [next]})$

13: **end for**

14: **until** $\|\hat{\mathbf{B}}^{[next]} - \hat{\mathbf{B}}^{[cur]}\|_\infty < \delta$

▷ End of joint lasso subproblem

15: $\hat{\mathbf{B}}^{(r+1)} \leftarrow \hat{\mathbf{B}}^{[next]}$

16: Update $\hat{\sigma}_{I_j}^{(r+1)} = \frac{\|\mathbf{x}_{I_j} - \mathbf{X} \hat{\beta}_{-I_j}^{(r+1)}\|_2}{\sqrt{n}}$ in parallel

17: Calculate $F_j = I(|\hat{\sigma}_j^{(r+1)} - \hat{\sigma}_j^{(r)}| \geq \delta)$ in parallel

18: Set $l = 0$

19: **for** $j = 1, \dots, n_c$ **do**

20: **if** $F_j = 1$ **then**

$l \leftarrow l + 1$, $I_l \leftarrow j$

end if

21: **end for**

22: Set $n_c = l$, $\hat{\mathbf{B}}^{[next]} \leftarrow (\hat{\beta}_{-I_1}^{[next]}, \dots, \hat{\beta}_{-I_{n_c}}^{[next]})$

23: **until** $n_c = 0$

24: Calculate $\hat{\omega}_{jj} = \hat{\sigma}_j^{-2}$ and $\hat{\omega}_{jk} = -\hat{\beta}_{jk} \hat{\sigma}_k^{-2}$ in parallel

▷ Initial estimate for Ω

25: Update $\hat{\omega}_{jk}$ in parallel

▷ Symmetrization

if $|\hat{\omega}_{jk}| > |\hat{\omega}_{kj}|$ **then**

$\hat{\omega}_{jk} \leftarrow \hat{\omega}_{kj}$

else

$\hat{\omega}_{kj} \leftarrow \hat{\omega}_{jk}$

end if

Output: $\hat{\Omega} = (\hat{\omega}_{jk})$

– AR(1): AR(1) network is also known as a chain graph. We define a precision matrix Ω for AR(1) as

$$\Omega = (\omega_{ij})_{1 \leq i, j \leq p} = \begin{cases} 1 & \text{if } i = j \\ 0.48 & \text{if } |i - j| = 1 \\ 0 & \text{otherwise} \end{cases}$$

– AR(4): In AR(4) network, each node is connected to neighborhood nodes whose distance is less than or equal to 4, where the distance of two nodes i and j is defined as $d(i, j) = |i - j|$. The precision matrix Ω corresponding to AR(4) network is defined as

$$\Omega = (\omega^{ij})_{1 \leq i, j \leq p} = \begin{cases} 0.6^{|i-j|} & \text{if } |i - j| \leq 4 \\ 0 & \text{otherwise} \end{cases}$$

- Scale-free: Degrees of nodes follow the power-law distribution having a form $P(k) \propto k^{-\alpha}$, where $P(k)$ is a fraction of nodes having k connections and α is a preferential attachment parameter. We set $\alpha = 2.3$, which is used in Peng et al. (2009), and we generate a scale-free network structure by using the Barabási and Albert (BA) model (Barabási and Albert 1999). With the generated network structure $G = (V, E)$, we define a precision matrix corresponding to $G = (V, E)$ by following the steps applied in (Peng et al. 2009):

$$(i) \tilde{\Omega} = (\tilde{\omega}_{ij})_{1 \leq i, j \leq p} = \begin{cases} 1 & \text{if } i = j \\ U & \text{if } (i, j) \in E, U \sim \mathcal{U}([-1, -0.5] \cup [0.5, 1]) \\ 0 & \text{otherwise} \end{cases}$$

$$(ii) \Omega = (\omega_{ij})_{1 \leq i, j \leq p} = \frac{\tilde{\omega}_{ij}}{1.5 \sum_{k \neq i} |\tilde{\omega}_{ik}|}$$

$$(iii) \Omega = (\Omega + \Omega^T)/2$$

$$(iv) \omega_{ii} = 1 \text{ for } i = 1, 2, \dots, p$$

- Hub: Following (Peng et al. 2009), for $p = 100$, a hub network consists of 10 hub nodes whose degrees are around 15 and 90 non-hub nodes whose degrees lie between 1 and 3. Edges in the hub network are randomly selected with the above conditions. With the given network structure $G = (V, E)$, we generate a precision matrix by the procedure described in the scale-free network generation.

To avoid nonzero elements having considerably small magnitudes (i.e., the absolute value of element), we generate $(p/100)$ subnetworks, each of which consists of 100 nodes, and set nonzero elements having magnitudes less than 0.1 to 0.1 (i.e., $\text{sign}(\omega_{ij})/10$) for the scale-free and hub networks. For example, we generate five subnetworks having 100 nodes when $p = 500$ for the scale-free and hub networks. We depict the generated four network structures in Figure 3 for $p = 500$. Among four networks, AR(1) and AR(4) networks correspond the circumstances that the variables are measured in a specific order, and the hub and scale-free networks are frequently observed in real-world problems such as the gene regulatory networks and functional brain networks.

For comparison of the computational efficiency, we consider the number of variables $p = 1000$, and sample size $n = 250,500$. To provide a benchmark of the computational efficiency, we also consider the well-known existing methods such as the CLIME (Cai et al. 2011b), graphical lasso (GLASSO) (Friedman et al. 2008), and convex partial correlation estimation method (CONCORD) (Khare et al. 2015). As the computation times of the algorithms are affected by the level of sparsity on the estimate of the precision matrix, we set $\lambda_0 = \sqrt{4n^{-1} \log p}$ for LARS-CPU (SPMESL-LARS) and CD-CPU (SPMESL-CD). Further, we search the tuning parameters for other methods to obtain the level of sparsity similar to the one obtained by the SPMESL. For example, for AR(1) and AR(4) networks with $p = 1000$, the averages of the estimated edges of all methods are around 1000, which is 0.2% of the total possible edges. With the chosen penalty levels, the computation time for each method is measured in CPU time (seconds) by using a workstation (Intel(R) Xeon(R) W-2175 CPU (base: 2.50 GHz, max-turbo: 4.30 GHz) and 128 GB RAM) with NVIDIA GeForce GTX 1080 Ti. We measure the average computation times and standard errors over 10 datasets.

For comparison of the estimation performance, we consider evaluating the estimation performance of the SPMESL with $\lambda_{pb} = \sqrt{2L_n(k/p)}$, $\lambda_{univ} = \sqrt{2n^{-1} \log(p-1)}$, and $\lambda_{ub} = \sqrt{4n^{-1} \log p}$ as there are three different suggestions for λ_0 without a comparison of the estimation performance, where k is a real solution of $k = L_1^4(k/p) + 2L_1^2(k/p)$, $L_n(t) = n^{-1/2} \Phi^{-1}(1-t)$, and $\Phi^{-1}(t)$ is the standard normal quantile function. Here, we denote the SPMESL with λ_{pb} , λ_{univ} , λ_{ub} as SPMESL-P, SPMESL-2, SPMESL-4, respectively. As described in the computational efficiency comparison, we apply the three existing methods (CLIME, GLASSO, CONCORD) to provide a benchmark of the estimation performance. Hereafter, we referred these three methods to the *tuning-search* methods. To measure the estimation performance, we consider five performance measures—sensitivity (SEN), specificity (SPE), false discovery rate (FDR), miss-classification error rate (MISR), and Matthew’s correlation coefficients (MCC)—for identifying the true edges and the Frobenius norm of $\Omega^0 - \hat{\Omega}$ for estimation error, where Ω^0 and $\hat{\Omega}$ denote the true precision matrix and the estimate of the precision matrix, respectively. The five performance measures

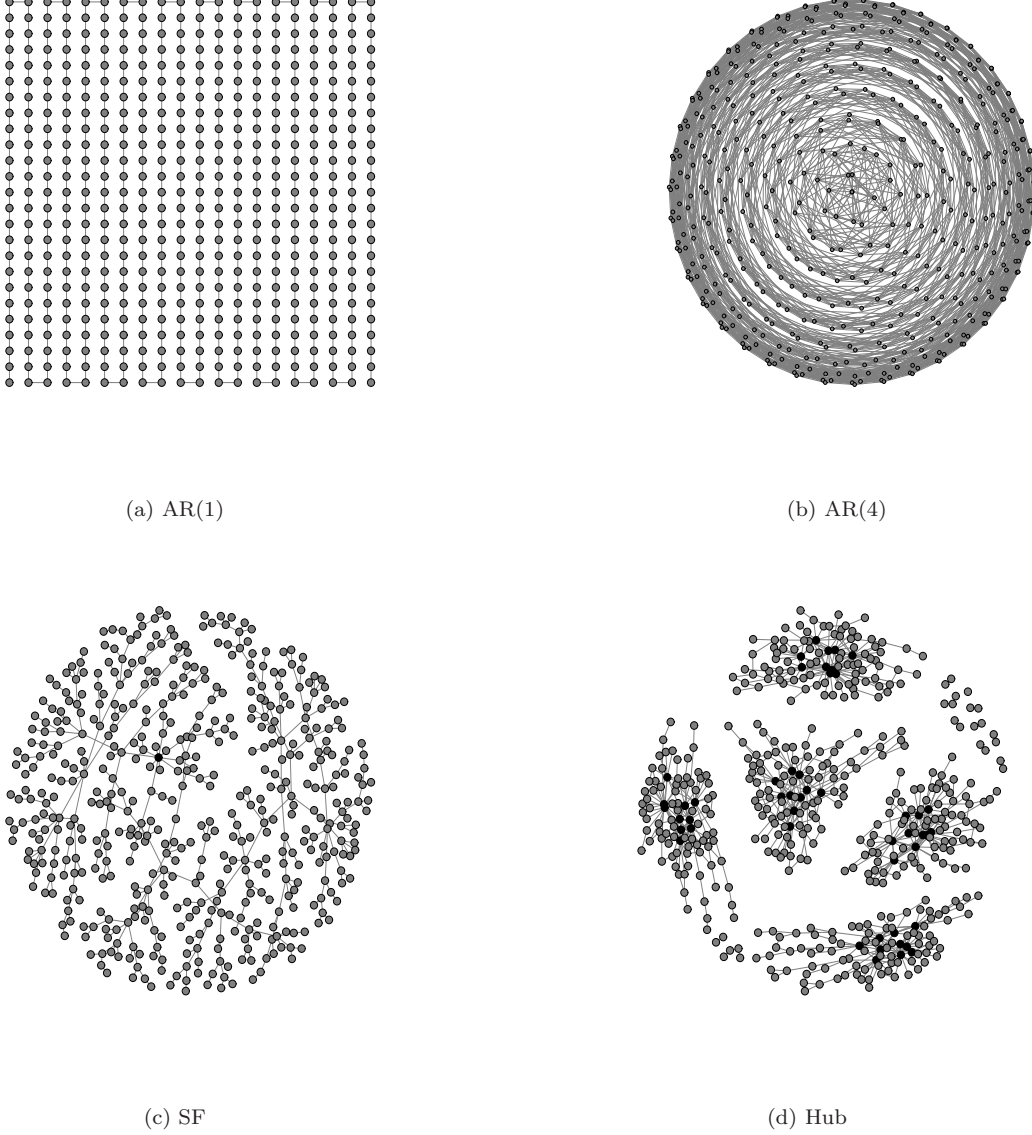


Fig. 3 Four network structures of precision matrix: (a) AR(1), (b) AR(4), (c) Scale-free (SF), and (d) Hub-network (Hub). Nodes in black denote nodes whose degrees are more than 9.

for identification of the true edges are defined as follows:

$$\begin{aligned} \text{SEN} &\equiv \frac{\text{TP}}{(\text{TP} + \text{FN})}, \text{SPE} \equiv \frac{\text{TN}}{(\text{TN} + \text{FP})}, \text{FDR} \equiv \frac{\text{FP}}{(\text{TP} + \text{FP})}, \\ \text{MISR} &\equiv \frac{(\text{FP} + \text{FN})}{p(p-1)/2}, \text{MCC}(\hat{\Omega}, \Omega^0) \equiv \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}, \end{aligned}$$

where $\text{TP} = \sum_{j < k} I(\hat{\omega}_{jk} \neq 0, \omega_{jk}^0 \neq 0)$, $\text{FP} = \sum_{j < k} I(\hat{\omega}_{jk} \neq 0, \omega_{jk}^0 = 0)$, $\text{TN} = \sum_{j < k} I(\hat{\omega}_{jk} = 0, \omega_{jk}^0 = 0)$, and $\text{FN} = \sum_{j < k} I(\hat{\omega}_{jk} = 0, \omega_{jk}^0 \neq 0)$,

In the comparison of the estimation performance, we set the number of variables and sample size as 500 and 250, respectively. We generate samples $\mathbf{X}^1, \mathbf{X}^2, \dots, \mathbf{X}^n \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1}$ and $\boldsymbol{\Omega}$ is from a given network structure. Unlike the comparison of computational efficiency, we need to choose criteria for the selection of the optimal tuning parameter of the CLIME, GLASSO, and CONCORD. For a fair

comparison, we adopt the Bayesian information criterion (BIC), which is widely used in model selection, and search the optimal tuning parameter over an equally spaced grid $(0.10, 0.12, \dots, 0.88, 0.90)$. We generate 50 datasets, each of which we search the optimal tuning parameters for the CLIME, GLASSO, and CONCORD and apply λ_{pb} , λ_{univ} , λ_{ub} for the SPMESL.

In this numerical study, we implement R package `cdscalreg` for the CD algorithm with a warm start strategy for the scaled Lasso and SPMESL, which is available at <https://sites.google.com/view/seunghwan-lee/soft>. For the other methods in this study, we use R packages `fastclime` for FASTCLIME, `glasso` for GLASSO, `gconcord` for CONCORD, `scalreg` for the original algorithm for SPMESL. Note that consider FASTCLIME (Pang et al. 2014) for the CLIME, which obtains the CLIME estimator efficiently and provides a solution path of the CLIME applying the parametric simplex method. It is also worth noting that the GPUs we used are more efficient for conducting operations with single-precision values than double-precision values, but the R programming language only supports the double-precision that makes the efficiency of the GPU-parallel computation decrease. Although this PCD implementation could decrease its computational efficiency, we develop an R function for the PCD algorithm to provide an efficient and convenient tool for R users. If readers want to utilize GPU-parallel computation maximally, PyCUDA (Klöckner et al. 2012) is one of the convenient and favorable ways to implement CUDA GPU-parallel computation.

4.2 Comparison results for computational efficiency

Table 1 reports the average computation times and standard errors over 10 datasets. From Table 1, we numerically verify that the proposed algorithm based on the CD with warm-start strategy is more efficient than the original algorithm based on the LARS, where the proposed algorithm (CD-CPU) is 110.2 and 466.3 times faster than LARS-CPU for the worst case (AR(1), $n = 250$) and the best case (AR(4), $n = 500$), respectively. For comparison of efficiency with other methods, overall, the CD-CPU is faster than FASTCLIME and slower than GLASSO and CONCORD. GLASSO is the most efficient algorithm in our numerical study. Its efficiency is from the sub-procedure that reduces the computational cost by the pre-identification procedure for nonzero block diagonals of the estimate that rearranges the order of variables for a given tuning parameter described in Witten et al. (2011). The CONCORD is faster than the CD-CPU in general because the CD algorithm for the CONCORD is applied to minimize its objective function directly. However, the CD algorithm in the CD-CPU is repeatedly applied to solve the lasso subproblems. Even though the GLASSO and CONCORD are faster than the CD-CPU, the GLASSO and CONCORD are tuning-search methods while the CD-CPU is not. Thus, the CD-CPU becomes the most efficient when the GLASSO and CONCORD need to evaluate more than five tuning parameters. For the efficiency of the PCD-GPU, we can see that the PCD-GPU is faster than CD-CPU for all cases except for the case of (Hub, $n = 250$). In addition, Table 1 also shows that the efficiency of the PCD-GPU increases as the sample size increases. For example, all computation times of CD-CPU significantly increase when the sample size increases from 250 to 500. However, there is no significant difference on the computation times of PCD-GPU between the sample sizes 250 and 500. This might show that the GPU device has idle processing units when $n = 250$. Note that the estimator of the SPMESL can be obtained by solving p scaled Lasso problems independently on multi CPU cores instead of on GPUs. However, the cost per core of CPU is more expensive than that of GPU. Moreover, the average computation times of the parallel computation of the CD-CPU with 16 CPU cores with the R package `doParallel` (PCD-MPI in Table 1) are around 20 seconds, which are worse than CD-CPU. This inefficiency might be from the communication cost and the number of CPU cores not enough for large p .

To verify the efficiency of PCD-GPU compared to CD-CPU, we conduct additional numerical studies for CD-CPU and PCD-GPU with $p = 500, 1000, 2000, 5000$ and $n = 250, 500, 1000$. Table 2 reports the average computation times and standard errors measured in CPU time (seconds) for CD-CPU and PCD-GPU. As shown in Table 2, PCD-GPU becomes more efficient than CD-CPU when either the number of variables or the sample size increases. For example, CD-CPU is 1.05~1.94 times faster than PCD-GPU only for the Hub-network cases of $(p, n) = (500, 250), (500, 500), (1000, 250)$; however, PCD-GPU outperforms CD-CPU for all the other cases and 4.71~11.65 times faster than CD-CPU when $p = 5000$. In Table 2, we also find an advantage of the GPU-parallel computation. Originally, the parallel computation in PCD-GPU applied to reduce the computational cost depends on the number of variables. However, the additional numerical studies support that the parallel computation in PCD-GPU also reduces the

computation times when the number of samples increases for a fixed p . This advantage is from the efficiency of the GPU-parallel computation for the matrix-matrix and matrix-vector multiplications. Thus, the additional numerical studies show that PCD-GPU is favorable for the cases where either p or n is sufficiently large.

Table 1 The averages of the computation times (sec.) over 10 datasets. Numbers in the parentheses denote the standard errors.

Network	p	n	scalreg	CD-CPU	PCD-GPU	PCD-MPI	FASTCLIME	GLASSO	CONCORD
AR(1)	1000	250	937.7358 (4.4662)	8.5027 (0.0519)	4.4356 (0.0749)	18.8599 (0.7913)	241.0991 (4.0303)	0.7091 (0.0081)	6.4348 (0.1541)
		500	4345.7155 (3.9313)	13.9459 (0.0335)	4.2743 (0.0397)	21.4523 (0.8523)	227.3807 (2.2561)	1.2521 (0.1210)	12.4249 (0.0830)
AR(4)	1000	250	803.3684 (0.9841)	5.9992 (0.0158)	1.7896 (0.0274)	19.2027 (1.0094)	233.5897 (1.9512)	0.8030 (0.0460)	3.4643 (0.0410)
		500	4043.0187 (4.4534)	8.6698 (0.0220)	2.2892 (0.0317)	20.0631 (0.8708)	259.5524 (2.0104)	8.9721 (0.2280)	7.5435 (0.2210)
Scale-Free	1000	250	790.8713 (0.6122)	5.1367 (0.0376)	3.0949 (0.0545)	18.8765 (1.3815)	238.7474 (1.7989)	0.6852 (0.0010)	3.5295 (0.0481)
		500	3819.9712 (8.2088)	9.1574 (0.0190)	3.0781 (0.0255)	19.2749 (0.7189)	246.0107 (1.2268)	0.7945 (0.0014)	7.0473 (0.0775)
Hub	1000	250	787.6731 (0.8286)	4.7785 (0.0247)	7.3326 (0.1355)	18.8566 (1.3197)	245.2853 (1.6676)	0.7512 (0.0099)	6.1875 (0.1024)
		500	3769.7531 (23.6487)	9.2215 (0.0458)	7.7584 (0.1942)	18.7167 (0.5727)	250.2303 (2.0064)	2.2454 (0.1917)	12.7924 (0.2707)

Table 2 The averages of the computation times (sec.) over 10 datasets. Numbers in the parentheses denote the standard errors.

Network	n	$p = 500$		$p = 1000$		$p = 2000$		$p = 5000$	
		CD	PCD	CD	PCD	CD	PCD	CD	PCD
AR(1)	250	2.2800 (0.0224)	1.9528 (0.0414)	8.7067 (0.0268)	4.4149 (0.0765)	38.1056 (0.0841)	12.0651 (0.1121)	297.7769 (0.2836)	60.1581 (0.9851)
	500	3.4519 (0.0134)	1.7117 (0.0330)	13.8964 (0.0419)	4.2744 (0.0445)	59.2341 (0.0736)	12.8309 (0.1152)	436.7230 (2.7395)	64.4670 (0.7545)
	1000	5.8935 (0.0265)	1.7364 (0.0217)	23.5556 (0.0636)	4.9784 (0.0561)	99.7887 (0.1460)	15.8946 (0.1447)	700.8538 (1.0246)	85.5756 (0.6645)
AR(4)	250	1.4538 (0.0077)	0.8794 (0.0258)	6.1486 (0.0265)	1.7679 (0.0271)	27.9208 (0.0666)	4.9945 (0.2905)	232.7940 (0.2484)	20.9527 (0.3062)
	500	2.1363 (0.0116)	0.9950 (0.0177)	8.5920 (0.0254)	2.2517 (0.0372)	37.8047 (0.0777)	6.2053 (0.2523)	305.4613 (0.3319)	26.2313 (0.6105)
	1000	6.1868 (0.0312)	1.5552 (0.0155)	21.5172 (0.1308)	4.1366 (0.0284)	80.3323 (0.3212)	12.1282 (0.0558)	499.1462 (2.8728)	57.1606 (0.7015)
Scale-Free	250	1.2493 (0.0067)	1.1620 (0.0328)	4.9605 (0.0227)	2.9021 (0.0426)	22.1757 (0.0362)	6.8921 (0.0762)	186.5103 (0.2680)	35.0760 (0.6203)
	500	2.1820 (0.0094)	1.1572 (0.0109)	8.7524 (0.0196)	2.9931 (0.0276)	38.4489 (0.0646)	8.3851 (0.1162)	300.9288 (0.6319)	44.9643 (0.5134)
	1000	3.7781 (0.0074)	1.2210 (0.0122)	15.0969 (0.0141)	3.6376 (0.0298)	65.1003 (0.1194)	11.1231 (0.0762)	479.8427 (1.4722)	63.4638 (0.9511)
Hub	250	1.2021 (0.0075)	2.3266 (0.0434)	4.5724 (0.0207)	6.9508 (0.1257)	19.9179 (0.0456)	11.6372 (0.2135)	168.9348 (0.2214)	35.8963 (0.6634)
	500	2.2632 (0.0078)	2.3915 (0.0335)	8.9738 (0.0220)	7.6092 (0.1866)	38.4337 (0.0630)	15.7290 (0.3070)	300.2697 (0.4684)	53.2606 (0.4222)
	1000	3.9263 (0.0108)	2.6366 (0.0342)	15.6040 (0.0279)	9.6065 (0.1127)	66.8115 (0.1204)	21.4761 (0.1523)	492.7247 (1.1698)	77.6366 (0.4552)

4.3 Comparison results for estimation performance

Tables 3 and 4 report the averages of the number of estimated edges ($|\hat{E}|$) and the six performance measures over 50 data sets for AR(1), AR(4), Scale-free, and Hub networks. From the results in Tables 3 and 4, we can find several interesting features. First, focusing on comparing SPMESL-P, SPMESL-2, and SPMESL-4, the SPMESL-P obtains the smallest estimation error in Frobenius norm, and the SPMESL-4 has the largest estimation error for all network structures. The estimation error of SPMESL-2 is located near the middle of an interval defined by the estimation errors of the SPMESL-P and SPMESL-4. However, for the performance in the identification of the true edges, the SPMESL-4 has the smallest SEN and FDR for all network structures, and the SPMESL-2 obtains the lowest MISR and the highest MCC for Scale-free and Hub networks while the SPMESL-P has the highest MISR and the lowest MCC for all network structures. For AR(1) and AR(4) network structures, the MISR and the MCC of the SPMESL-2 are worse than those of the SPMESL-4, but the differences of the SPMESL-2 and the SPMESL-4 are small. For example, the difference in the MCC of the SPMESL-2 and the SPMESL-4 are 0.0195 and 0.0095 in an original scale for AR(1) and AR(4), respectively.

Second, by comparing the SPMESL and the tuning-search methods (CLIME, GLASSO, CONCORD), the SPMESL-2 and SPMESL-4 obtain considerably small FDRs compared to the tuning-search methods. For example, the FDRs by the tuning-search methods are over 27% for all cases, but the FDRs by the SPMESL-2 and the SPMESL-4 are less than 6.2%. The SPMESL-P has the lowest MCC and the highest FDR for Scale-free and Hub networks and obtains the second-lowest MCC and the second-highest FDR for AR(4) networks, where only the GLASSO is worse than the SPMESL-P in terms of the MCC and FDR. For AR(1) network, the SPMESL with all penalty levels are better than the tuning-search methods for identifying the true edges, and the estimation errors of the SPMESL-P and SPMESL-2 are less than those of the tuning-search methods.

Finally, we compare the estimation performance of the tuning-search methods. For the estimation error in the Frobenius norm, the CONCORD outperforms CLIME and GLASSO for all cases, where the CONCORD obtains similar estimation errors to the SPMESL-P for AR(4), Scale-free, and Hub networks. However, for the identification of the true edges, the CLIME obtains slightly better performance than the CONCORD for all networks except the AR(1) network. For the AR(1) network, the CONCORD outperforms CLIME and GLASSO for estimation error and identification of the true edges. In our numerical study, the CONCORD is favorable among the three tuning-search methods we consider. Note that we adopt the BIC for three tuning-search methods for a fair comparison, but the comparison results might be changed if we apply other model selection criteria.

From the results of the estimation performance comparison, we recommend the SPMESL with the universal penalty level λ_{univ} if the target problem can accept the FDR level around 5%; the uniform-bound penalty level is only preferred when the problem only accepts the small FDR less than 1%. Note that we do not recommend using the probabilistic bound λ_{pb} as the SPMESL-P has the highest FDR among three penalty levels for the SPMESL, which are over 50% for Scale-free and Hub networks, although the SPMESL-P obtains the lowest estimation error in Frobenius norm.

5 Conclusion

In this paper, we proposed an efficient coordinate descent algorithm with the warm start strategy for sparse precision matrix estimation using the scaled lasso motivated by the empirical observation that the iterative solution for the diagonal elements of the precision matrix needs only a few iterations. In addition, we also develop the parallel coordinate descent algorithm (PCD) for the SPMESL by representing p Lasso subproblems as the unified minimization problem. In the PCD algorithm, we use a different convergence criterion $\|\mathbf{B}^{(k+1)} - \mathbf{B}^{(k)}\|_\infty < \delta$ to check the convergence of the PCD algorithm for the unified minimization problem. We show that the difference in the iterative solutions of the CD and PCD caused by the difference of the convergence criteria is also bounded by the convergence tolerance δ .

Our numerical study shows that the proposed CD algorithm is much faster than the original algorithm of the SPMESL, which adopts the LARS algorithm to solve the Lasso subproblems. Moreover, the PCD algorithm with GPU-parallel computation becomes more efficient than the CD algorithm when either the number of variables or the sample size increases. For the optimal tuning parameter for the SPMESL, there are three suggestions without the comparison of the estimation performance. In the additional simulation,

Table 3 For AR(1) and AR(4) networks with $p = 500$ and $n = 250$, the averages of the number of estimated edges, the five performance measures and the Frobenius norms of difference of the estimate and true precision matrix over 50 datasets. Numbers in the parentheses denote the standard errors.

Network	Method	$ \hat{E} $	SEN	SPE	FDR	MISR	MCC	$\ \hat{\Omega} - \Omega\ _F$
AR(1) ($ E = 499$)	CLIME	897.80 (5.20)	100.00 (0.00)	99.68 (0.00)	44.33 (0.31)	0.32 (0.00)	74.48 (0.21)	9.17 (0.02)
	GLASSO	2134.48 (14.52)	100.00 (0.00)	98.68 (0.01)	76.57 (0.16)	1.31 (0.01)	48.07 (0.17)	12.74 (0.02)
	CONCORD	722.16 (2.93)	100.00 (0.00)	99.82 (0.00)	30.84 (0.29)	0.18 (0.00)	83.08 (0.17)	4.96 (0.01)
	SPMESL-P	649.28 (1.68)	100.00 (0.00)	99.88 (0.00)	23.12 (0.20)	0.12 (0.00)	87.62 (0.11)	3.65 (0.01)
	SPMESL-2	524.74 (0.77)	100.00 (0.00)	99.98 (0.00)	4.90 (0.14)	0.02 (0.00)	97.51 (0.07)	4.55 (0.01)
	SPMESL-4	504.40 (0.36)	100.00 (0.00)	100.00 (0.00)	1.07 (0.07)	0.00 (0.00)	99.46 (0.04)	6.39 (0.01)
AR(4) ($ E = 1990$)	CLIME	908.58 (2.66)	32.55 (0.08)	99.79 (0.00)	28.70 (0.15)	1.29 (0.00)	47.65 (0.08)	22.35 (0.01)
	GLASSO	988.96 (10.79)	28.86 (0.07)	99.66 (0.01)	41.66 (0.52)	1.47 (0.01)	40.36 (0.17)	23.17 (0.01)
	CONCORD	979.40 (3.33)	33.03 (0.08)	99.74 (0.00)	32.86 (0.19)	1.33 (0.00)	46.52 (0.10)	18.46 (0.01)
	SPMESL-P	1206.32 (2.97)	36.31 (0.07)	99.61 (0.00)	40.09 (0.16)	1.40 (0.00)	45.99 (0.09)	18.40 (0.01)
	SPMESL-2	545.30 (0.94)	25.71 (0.02)	99.97 (0.00)	6.18 (0.15)	1.21 (0.00)	48.77 (0.05)	20.57 (0.01)
	SPMESL-4	499.00 (0.17)	25.05 (0.01)	100.00 (0.00)	0.11 (0.02)	1.20 (0.00)	49.72 (0.01)	22.72 (0.01)

Table 4 For Scale-Free and Hub networks with $p = 500$ and $n = 250$, the averages of the number of estimated edges, the five performance measures and the Frobenius norms of differences of the estimate and true precision matrix over 50 datasets. Numbers in the parentheses denote the standard errors.

Network	Method	$ \hat{E} $	SEN	SPE	FDR	MISR	MCC	$\ \hat{\Omega} - \Omega\ _F$
Scale-Free ($ E = 495$)	CLIME	643.08 (2.19)	91.38 (0.16)	99.85 (0.00)	29.63 (0.24)	0.19 (0.00)	80.10 (0.17)	8.33 (0.01)
	GLASSO	810.52 (7.21)	92.84 (0.19)	99.72 (0.01)	43.08 (0.52)	0.31 (0.01)	72.53 (0.31)	7.79 (0.02)
	CONCORD	682.36 (4.00)	93.04 (0.17)	99.82 (0.00)	32.39 (0.40)	0.21 (0.00)	79.20 (0.23)	5.33 (0.01)
	SPMESL-P	1022.56 (3.66)	94.60 (0.14)	99.55 (0.00)	54.18 (0.17)	0.47 (0.00)	65.66 (0.14)	5.15 (0.01)
	SPMESL-2	457.88 (1.06)	87.94 (0.15)	99.98 (0.00)	4.92 (0.16)	0.07 (0.00)	91.40 (0.11)	6.57 (0.01)
	SPMESL-4	348.34 (0.85)	70.33 (0.17)	100.00 (0.00)	0.06 (0.02)	0.12 (0.00)	83.79 (0.10)	8.83 (0.01)
Hub ($ E = 551$)	CLIME	644.02 (2.31)	84.36 (0.23)	99.86 (0.00)	27.80 (0.21)	0.21 (0.00)	77.93 (0.17)	8.43 (0.01)
	GLASSO	881.76 (10.37)	87.44 (0.27)	99.68 (0.01)	45.03 (0.59)	0.38 (0.01)	69.10 (0.31)	7.89 (0.02)
	CONCORD	731.22 (5.43)	89.03 (0.19)	99.81 (0.00)	32.73 (0.51)	0.24 (0.00)	77.24 (0.27)	5.55 (0.01)
	SPMESL-P	1094.14 (3.33)	91.32 (0.14)	99.52 (0.00)	53.99 (0.14)	0.51 (0.00)	64.62 (0.12)	5.37 (0.01)
	SPMESL-2	474.36 (1.55)	81.22 (0.22)	99.98 (0.00)	5.64 (0.15)	0.10 (0.00)	87.49 (0.13)	6.78 (0.01)
	SPMESL-4	319.72 (1.20)	57.96 (0.21)	100.00 (0.00)	0.11 (0.02)	0.19 (0.00)	76.02 (0.14)	8.90 (0.01)

we numerically investigate the estimation performance of the SPMESL with the three penalty levels and the other three tuning-search methods. From the comparison results, the SPMESL with the uniform bound level and the universal penalty level outperform the three tuning-search methods. Specifically, the probabilistic bound level $\lambda_{pb} = \sqrt{2}L_n(k/p)$ provides the estimate that has the smallest estimation

error in terms of the Frobenius norm; the uniform bound level $\lambda_{ub} = \sqrt{4n^{-1}\log(p)}$ provides the estimate that has the smallest FDR less than 1%; and the universal penalty level $\lambda_{univ} = \sqrt{2n^{-1}\log(p-1)}$ obtains either the best or second-best in terms of MCC and FDR. Overall, we recommend the SPMESL with the universal penalty level and apply the CD algorithm if p is less than or equal to 1000 and the PCD algorithm when the target problem has more than 1000 variables and GPU-parallel computation is available.

Acknowledgements

Sang Cheol Kim’s research was supported by funding (2019-NI-092-00) from the Research of Korea Centers for Disease Control and Prevention. Donghyeon Yu’s research was supported by the INHA UNIVERSITY Research Grant and the National Research Foundation of Korea (NRF-2020R1F1A1A01048127).

References

- Ali, A., Khare K., Oh S.-Y., and Rajaratnam, B. (2017). Generalized pseudolikelihood methods for inverse covariance estimation. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, **54**, 280–288.
- Barabási, A. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, **286**, 509–512. doi:10.1126/science.286.5439.509
- Bickel, P. J. and Levina, E. (2008a). Regularized estimation of large covariance matrices. *The Annals of Statistics*, **36**, 199–227. doi:10.1214/009053607000000758
- Bickel, P. J. and Levina, E. (2008b). Covariance regularization by thresholding. *The Annals of Statistics*, **36**, 2577–2604. doi:10.1214/08-AOS600
- Cai, T. T., Zhang, C. H., and Zhou H. H. (2010). Optimal rates of convergence for covariance matrix estimation. *The Annals of Statistics*, **38**, 2118–2144. doi:10.1214/09-AOS752
- Cai, T. and Liu, W. (2011a). Adaptive thresholding for sparse covariance matrix estimation. *Journal of the American Statistical Association*, **106**, 672–684. <https://doi.org/10.1198/jasa.2011.tm10560>
- Cai, T. T., Liu, W. and Luo, X. (2011b). A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, **106**, 594–607. <https://doi.org/10.1198/jasa.2011.tm10155>
- Cai, T. T. and Zhou, H. H. (2012). Minimax estimation of large covariance matrices under ℓ_1 norm. *Statistica Sinica*, **22**, 1319–1349. <http://dx.doi.org/10.5705/ss.2010.253>
- Cai, T. T., Liu, W., and Zhou, H. H. (2016). Estimating sparse precision matrix: optimal rates of convergence and adaptive estimation. *The Annals of Statistics*, **44**, 455–488. doi:10.1214/13-AOS1171
- Bradely Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, **32**, 407–499. doi:10.1214/009053604000000067
- Friedman, J., Hastie, T. and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical Lasso. *Biostatistics*, **9**, 432–441. <https://doi.org/10.1093/biostatistics/kxm045>
- Khare, K., Oh, S.-Y., and Rajaratnam, B. (2015). A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees. *Journal of the Royal Statistical Society: Series B*, **77**, 803–825. <http://www.jstor.org/stable/24775310>
- Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., and Fasih, A. (2012). PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. *Parallel Computing*, **38**, 157–174. <https://doi.org/10.1016/j.parco.2011.09.001>
- Mazumder, R. and Hastie, T. (2012). The graphical lasso: New insights and alternatives. *Electronic Journal of Statistics*, **6**, 2125–2149. doi:10.1214/12-EJS740
- Meinshausen, N. and Bühlmann, P. (2006). High dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, **34**, 1436–1462. doi:10.1214/009053606000000281
- Pang, H., Liu, H., and Vanderbei, R. (2014). The FASTCLIME package for linear programming and large-scale precision matrix estimation in R. *Journal of Machine Learning Research*, **15**, 489–493.
- Peng, J., Wang, P., Zhou, N., and Zhu, J. (2009). Partial correlation estimation by joint sparse regression models. *Journal of the American Statistical Association*, **104**, 735–746. doi:10.1198/jasa.2009.0126
- Ren, Z., Sun, T., Zhang, C.-H., and Zhou H. H. (2015). Asymptotic normality and optimalities in estimation of large Gaussian graphical model. *The Annals of Statistics*, **43**, 991–1026. doi:10.1214/14-AOS1286
- Rothman, A. J., Levina, E. and Zhu, J. (2009). Generalized thresholding of large covariance matrices. *Journal of the American Statistical Association*, **104**, 177–186. <https://doi.org/10.1198/jasa.2009.0101>

- Schäfer, J., and Strimmer, K. (2005). A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, *4*, 32. <https://doi.org/10.2202/1544-6115.1175>
- Sun, T. and Zhang, C. H. (2012). Scaled sparse linear regression. *Biometrika*, *99*, 879–898. doi:10.1093/biomet/ass043
- Sun, T. and Zhang, C. H. (2013). Sparse matrix inversion with scaled lasso *Journal of Machine Learning Research*, *14*, 3385–3418.
- Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society Series B*, *58*, 267–288. <https://doi.org/10.1111/j.2517-6161.1996.tb02080.x>
- Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, *109*, 475–494. <https://doi.org/10.1023/A:1017501703105>
- van de Geer, S. and Bühlmann, P. (2009). On the conditions used to prove oracle results for the Lasso. *Electronic Journal of Statistics*, *3*, 1360–1392. doi:10.1214/09-EJS506
- Witten, D., Friedman, J., and Simon, N. (2011). New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, *20*, 892–900. <https://doi.org/10.1198/jcgs.2011.11051a>
- Wu, T. T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics*, *2*, 224–244. doi:10.1214/07-AOAS147
- Wu, W. B. and M. Pourahmadi (2009). Banding sample autocovariance matrices of stationary processes. *Statistica Sinica*, *19*, 1755–1768.
- Yao, J., Zheng, S., and Bai, Z. (2015). *Large sample covariance matrices and high-dimensional data analysis*, New York, NY: Cambridge University Press. <https://doi.org/10.1017/CB09781107588080>
- Yuan, M. and Y. Lin (2007). Model selection and estimation in the Gaussian graphical model. *Biometrika*, *94*, 19–35. <https://doi.org/10.1093/biomet/asm018>
- Yuan, M. (2010). Sparse inverse covariance matrix estimation via linear programming. *The Journal of Machine Learning Research*, *11*, 2261–2286.