A Fast and Scalable Joint Estimator for Learning Multiple Related Sparse Gaussian Graphical Models

Beilun Wang
University of Virginia

Ji Gao
University of Virginia

Yanjun Qi
University of Virginia

Abstract

Estimating multiple sparse Gaussian Graphical Models (sGGMs) jointly for many related tasks (large K) under a high-dimensional (large p) situation is an important task. Most previous studies for the joint estimation of multiple sGGMs rely on penalized log-likelihood estimators that involve expensive and difficult non-smooth optimizations. We propose a novel approach, FASJEM for fast and scalable joint structure-estimation of multiple sGGMs at a large scale. As the first study of joint sGGM using the M-estimator framework, our work has three major contributions: (1) We solve FASJEM through an entry-wise manner which is parallelizable. (2) We choose a proximal algorithm to optimize FASJEM. This improves the computational efficiency from $O(Kp^3)$ to $O(Kp^2)$ and reduces the memory requirement from $O(Kp^2)$ to $O(K)$. (3) We theoretically prove that FASJEM achieves a consistent estimation with a convergence rate of $O(\log(Kp)/n_{tot})$. On several synthetic and four real-world datasets, FASJEM shows significant improvements over baselines on accuracy, computational complexity and memory costs.

1 Introduction

The past decade has seen a revolution in collecting large-scale heterogeneous data from many scientific fields. For instance, genomic technologies have delivered fast and accurate molecular profiling data across many cellular contexts (e.g., cell lines or stages) from national projects like ENCODE[1]. Given such data, understanding and quantifying variable graphs across multiple contexts is a fundamental analysis task. Such variable graphs can significantly simplify network-driven studies about diseases or drugs[2]. The number of contexts those applications need to consider grows extremely fast. For example, the ENCODE [1] project, being generated over ten years with contributions from bio-labs across the world, contains expression data from 147 different human cell types (i.e., the number of tasks $K = 147$) in 2016. Besides, the number of variables (denoted as p) is also quite large, ranging from thousands (e.g., gene) to hundreds of thousands (e.g., SNP[3]).

We formulate this data analysis problem as jointly estimating K conditional dependency graphs $G^{(1)}, G^{(2)}, \ldots, G^{(K)}$ from data samples accumulated from K distinct conditions. For homogeneous data samples from a given i-th condition, one typical approach in the literature is the sparse Gaussian Graphical Model(sGGM)[4, 5, 6]. sGGM assumes data samples are independently and identically drawn from $N_p(\mu^{(i)}, \Sigma^{(i)})$, a multivariate normal distribution with mean $\mu^{(i)}$ and covariance matrix $\Sigma^{(i)}$. The graph structure $G^{(i)}$ is encoded by the sparsity pattern of the inverse covariance matrix, also named precision matrix, $\Omega^{(i)}$. $\Omega^{(i)} := (\Sigma^{(i)})^{-1}$. In $G^{(i)}$ an edge does not connect j-th node and k-th node (i.e., conditional independent) if and only if $\Omega^{(i)}_{jk} = 0$. sGGM imposes an ℓ_1 penalty on the parameter $\Omega^{(i)}$. For heterogeneous data samples, rather than estimating sGGM of each condition separately, a multi-task formulation that jointly estimates K different but related sGGMs can lead to a better generalization[7].

Most previous studies[8, 9, 10, 11, 12, 13, 14, 15] for joint estimation of multiple sGGMs relied on optimizing ℓ_1 regularized likelihood function plus an extra penalty function $\mathcal{R}'$. This extra regularizer $\mathcal{R}'$, which varies in different estimators, enforces similarity among multiple estimated networks. Since the penalized likelihood framework includes two regularization functions ($\ell_1 + \mathcal{R}'$), these approaches cannot avoid the steps like SVD [8] and matrix multiplication [8, 9]. Both steps need $O(Kp^3)$ time complexity for computation. Besides, most studies in this category require all tasks' co-

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

variance matrices to locate in the main memory[8, 9, 10] (for their optimization). Storing all elements needs $O(Kp^2)$ memory space. As a result, this category of models are difficult to scale up when the dimension p or the number of tasks K are large.

In this paper, we propose a novel model, namely fast and scalable joint estimator for multiple sGGM (FASJEM), for estimating multiple sGGMs jointly. Briefly speaking, this paper makes the following contributions:

- **Novel approach:** FASJEM presents a new way of learning multi-task sGGMs by extending the elementary estimator [16]. (Section 3)
- **Fast optimization:** We optimize FASJEM through an entry-wise and group-entry-wise manner that can dramatically improve the time complexity to $O(Kp^2)$. (Section 3 and Section 3.3)
- **Scalable optimization:** The optimization of our estimators is scalable. We reduce the memory cost to $O(K)$ (i.e., requiring to store at most K entries in the main memory). (Section 5)
- **Method variations:** We propose two variations of FASJEM: (1) FASJEM-G uses a group-2 norm to connect multiple sGGMs. (2) FASJEM-I uses a group-infinite norm to connect multiple related sGGMs. Both methods show better performance over their corresponded "Joint graphical lasso" (JGL) baselines. (Section 3 and Section 6)
- **Convergence rate:** We theoretically prove the convergence rate of FASJEM as $O(\log(Kp)/n_{tot})$. This rate shows the benefit of joint estimation, which significantly improves the convergence rate $O(\frac{\log p}{n})$ of single task sGGM (with n samples). (Section 5)
- **Evaluation:** FASJEM is evaluated using several synthetic datasets and four real-world biomedical datasets. It performs better than the baselines not only on accuracy but also with respect to the time and storage requirements. (Section 6)

Att: Due to space limit, we have put details of certain contents (e.g., proofs) in the appendix. Notations with "S:" as prefix in the numbering mean the corresponding contents are in the appendix. For example, full proofs are in Section S:8.

Notations: We focus on the problem of estimating K sGGMs from a p-dimensional aggregated dataset in the form of K different data matrices. $X^{(i)}_{n_i \times p}$ describes the data matrix for the i-th task, which includes n_i data samples described by p different feature variables. The total number of data samples is $n_{tot} = \sum_{i=1}^{K} n_i$. We use notation Ω for the precision matrices and $\widehat{\Sigma}$ for the estimated covariance matrices. Given a p-dimensional vector $x = (x_1, x_2, \ldots, x_p)^T \in \mathbb{R}^p$, $||x||_1 = \sum_i |x_i|$ represent the l_1-norm of x. $||x||_\infty = \max_i |x_i|$ is the l_∞-norm of x. $||x||_2 = \sqrt{\sum_i x_i^2}$, ℓ_2-norm of x.

2 Background

Single-task sGGM: The classic formulation of sparse Gaussian Graphical model [6] for a single given task (or context) (single sGGM) is the "graphical lasso" estimator (GLasso) [6, 17] that solves the following penalized maximum likelihood estimation (MLE) problem:

$$\underset{\Omega>0}{\operatorname{argmin}} -\log\det(\Omega) + <\Omega, \Sigma> + \lambda_n ||\Omega||_1 \qquad (2.1)$$

Elementary estimator for single sGGM (EE-sGGM): Recently the seminal study[18] generalized this formulation into a so-called M-estimator framework:

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta) + \lambda_n \mathcal{R}(\theta) \qquad (2.2)$$

where $\mathcal{R}(\cdot)$ represents a decomposable regularization function in [18] and $\mathcal{L}(\cdot)$ represents a loss function (e.g., the negative log-likelihood function $-L(\cdot)$ in sGGM). Using this framework, recent studies[19, 16] propose a new category of estimators named "Elementary estimator" [1], whose solution achieves the same optimal convergence rate as Eq. (2.2) when satisfying certain conditions. These estimators have the following general formulation:

$$\begin{aligned} &\underset{\theta}{\operatorname{argmin}}\, \mathcal{R}(\theta) \\ &\text{subject to:} \mathcal{R}^*(\theta - \widehat{\theta}_n) \le \lambda_n \end{aligned} \qquad (2.3)$$

Where $\mathcal{R}^*(\cdot)$ is the dual norm of $\mathcal{R}(\cdot)$,

$$\mathcal{R}^*(v) := \sup_{u \neq 0} \frac{<u, v>}{\mathcal{R}(u)} = \sup_{\mathcal{R}(u) \le 1} <u, v>. \qquad (2.4)$$

$\widehat{\theta}_n$ represents the backward mapping of θ. We provide detailed explanations of backward mapping and backward mapping for Gaussian case in the Appendix Section S:1. For sGGM, it is easy to derive Ω through the backward mapping on its covariance matrix Σ, which is Σ^{-1}. However, under the high-dimensional setting, when $p > n$, the sample covariance matrix $\widehat{\Sigma}$ is not full rank, therefore is not invertible. Thus the authors of [16] proposed a proxy backward mapping on the covariance matrix $\widehat{\Sigma}$ under high-dimensional settings as $(T_v(\widehat{\Sigma}))^{-1}$. Here $[T_v(M)]_{ij} := \rho_v(M_{ij})$ where $\rho_v(\cdot)$ is chosen to be a soft-thresholding function. [16] proves that this approximation will not change the convergence rate of sGGM estimation. Using Eq. (2.3), [16] proposed a new estimator for sGGM (the so-called elementary estimator for sGGM):

$$\begin{aligned} &\underset{\Omega}{\operatorname{argmin}}\, ||\Omega||_1 \\ &\text{subject to:} ||\Omega - [T_v(\widehat{\Sigma})]^{-1}||_\infty \le \lambda_n \end{aligned} \qquad (2.5)$$

This estimator has a closed-form solution[16] and has been shown to be more practical and scalable than GLasso in large-scale settings. v is a hyperparameter

[1] We denote this category of estimators as "elementary estimator" or "EE" in the rest of paper.

which ensures that $T_v(\widehat{\Sigma})$ is invertible.

Multi-task sGGM (Multi-sGGM): For joint estimation of multiple sGGMs, most studies focus on adding a second norm which enforces the group property among multiple tasks. Previous studies on the joint estimation of multiple sGGMs can be summarized using Eq. (2.6),

$$\begin{aligned}\underset{\Omega^{(i)}\succ 0}{\operatorname{argmin}} \sum_{i=1}^{K}(-L(\Omega^{(i)}) + \lambda_n \sum_{i=1}^{K} ||\Omega^{(i)}||_1 \\ + \lambda_n' \mathcal{R}'(\Omega^{(1)}, \Omega^{(2)}, \ldots, \Omega^{(K)})\end{aligned} \tag{2.6}$$

where $\Omega^{(i)}$ denotes the precision matrix for i-th task. $\Omega^{(i)} \succ 0$ means that $\Omega^{(i)}$ needs to be a positive definite matrix. $\mathcal{R}'(\cdot)$ represents the second penalty function for multi-tasking.

Superposition structured estimator (SS estimator) : The above Eq. (2.6) is a special case (explained in Section 3) of the following superposition structured estimators [18]:

$$\underset{(\theta_\alpha)_{\alpha\in I}}{\operatorname{argmin}} \mathcal{L}(\sum_{\alpha\in I}\theta_\alpha) + \sum_{\alpha\in I}\lambda_\alpha \mathcal{R}_\alpha(\theta_\alpha). \tag{2.7}$$

$\{\mathcal{R}_\alpha(\cdot)|\alpha \in I\}$ are a set of regularization functions and $(\lambda_\alpha)_{\alpha\in I}$ are the regularization penalties. The target parameter is $\theta = \sum_{\alpha\in I}\theta_\alpha$, a superposition of θ_α.

Elementary superposition-structured moment estimator (ESS moment estimator): Similar to Eq. (2.3), a recent study[20] extends the elementary estimator for sparse covariance matrices to the case of superposition-structured moments and named this extension as "Elem-Super-Moment" (ESM) estimator. [2]

$$\begin{aligned}&\underset{\theta_1,\theta_2,\ldots,\theta_{|I|}}{\operatorname{argmin}} \sum_{\alpha\in I}\lambda_\alpha \mathcal{R}_\alpha(\theta_\alpha) \\ &\text{Subject to: } \mathcal{R}_\alpha^*(\widehat{\theta} - \sum_{\alpha\in I}\theta_\alpha) \le \lambda_\alpha \qquad \forall \alpha \in I.\end{aligned} \tag{2.8}$$

3 Method: A fast and scalable joint estimator for multi-sGGM

The penalized likelihood framework for multi-task sGGMs in Eq. (2.6) involves a hybrid of two regularization functions ($\ell_1 + \mathcal{R}'$). Studies in this direction cannot avoid the expensive steps like SVD and matrix multiplication and also require to store K covariance matrices in the main memory. Since this paper aims to design a scalable joint estimator for multi-sGGM under large-scale settings, extending the elementary estimator of single-task sGGM [16] to multi-task formulation becomes a natural choice.

For multi-task sGGMs, we can denote that $\Omega_{tot} = (\Omega^{(1)}, \Omega^{(2)}, \ldots, \Omega^{(K)})$ and $\Sigma_{tot} = (\Sigma^{(1)}, \Sigma^{(2)}, \ldots, \Sigma^{(K)})$. Ω_{tot} and Σ_{tot} are both $p \times Kp$ matrices (i.e., Kp^2 parameters to estimate). Now define an inverse function as $\text{inv}(A_{tot}) := (A^{(1)^{-1}}, A^{(2)^{-1}}, \ldots, A^{(K)^{-1}})$, where A_{tot} is a given $p \times Kp$ matrix with the same structure as Σ_{tot}. Furthermore, we add a new hyperparameter variable $\epsilon = \frac{\lambda_n'}{\lambda_n}$.

Let $I = \{1, 2\}$ and $\theta_1 = \theta_2 = \frac{1}{2}\Omega_{tot}$. We can clearly tell that Eq. (2.6) is a special case of the superposition structured estimation in Eq. (2.7). The ESS (elementary superposition structured) moment estimator (Eq. (2.8)) extends the elementary estimator of structured covariance matrix to elementary superposition-structured estimator for estimating covariance matrices with a hybrid structure (e.g., sparse + low rank). This motivates us to propose the following elementary superposition estimator for learning multi-task sGGM:

$$\begin{aligned}&\underset{\Omega_{tot}}{\operatorname{argmin}} ||\Omega_{tot}||_1 + \epsilon\mathcal{R}'(\Omega_{tot}) \\ &s.t. ||\Omega_{tot} - \text{inv}(T_v(\widehat{\Sigma}_{tot}))||_\infty \le \lambda_n \\ &\mathcal{R}'^*(\Omega_{tot} - \text{inv}(T_v(\widehat{\Sigma}_{tot}))) \le \epsilon\lambda_n\end{aligned} \tag{3.1}$$

Here $||\cdot||_1^* = ||\cdot||_\infty$ (the dual norm of l_1-norm is l_∞-norm). $\mathcal{R}'(\cdot)$ represents a regularizer on Ω_{tot} to enforce that $\{\Omega^{(i)}\}$ share certain similarity. $\mathcal{R}'^*(\cdot)$ is the dual norm of $\mathcal{R}'(\cdot)$. We name this novel formulation as FASJEM. By varying $\mathcal{R}'(\cdot)$, we can get a variety of FASJEM estimators.

Section 5 theoretically proves the convergence rate of FASJEM as $O(\log(Kp)/n_{tot})$. Our theory proof is inspired by the ESS moment estimator [20], the SS estimator [21] and the EE-sGGM [16].

3.1 Method I: FASJEM-G

For multi-task regularization, the first $\mathcal{R}'(\cdot)$ we try is the $\mathcal{G}, 2$-norm (i.e., $\mathcal{R}'(\cdot) = ||\cdot||_{\mathcal{G},2}$). This norm is inspired by JGL-group lasso[8]. $\mathcal{G}, 2$-norm constrains the parameters in the same group to have the same level of sparsity. In multi-task sGGMs, group set $\mathcal{G} := \{g_{j,k}\}$, where $g_{j,k} = \{\Omega_{j,k}^{(i)} | i = 1, \ldots, K\}$. Suppose g is an arbitrary group in group set $\mathcal{G}$ and totally we have p^2 groups. $||\Omega_{tot}||_{\mathcal{G},2} = \sum_{j=1}^{p}\sum_{k=1}^{p} ||(\Omega_{j,k}^{(1)}, \Omega_{j,k}^{(2)}, \ldots, \Omega_{j,k}^{(i)}, \ldots, \Omega_{j,k}^{(K)})||_2$. When $\mathcal{R}'(\cdot) = ||\cdot||_{\mathcal{G},2}$, we name Eq. (3.1) as FASJEM-G (short form of FASJEM-Group2). We solve FASJEM-G using a parallel proximal based optimization formulation from[22]. Algorithm 1 summarizes the detailed optimization steps and the four proximity operators implemented on GPU are listed in Table 1[3]. The optimization sequence of Algorithm 1 converges Q-linearly (See Eq. (S:2–10)).

[2] [20] has proved that this class of ESM estimators achieves the same convergence rate as the corresponding estimators (with the same superposition of structures) using the penalized MLE formulation under certain conditions.

[3] The non-GPU version of the four proximity operators are in Eq. (S:2–2) to Eq. (S:2–5).

3.2 Method II: FASJEM-I

As shown in Section 4, most previous models for multi-task sGGMs varied the second norm R' to obtain different models. Similarly we can easily change $R'(\cdot)$ in Eq. 3.1 into any other desired norm to extend our FASJEM. For instance, we can change $R'(\cdot)$ to group-infinity norm $||\cdot||_{\mathcal{G},\infty}$.

$$||\Omega_{tot}||_{\mathcal{G},\infty} = \sum_{j=1}^{p}\sum_{k=1}^{p} ||(\Omega_{j,k}^{(1)}, \Omega_{j,k}^{(2)}, \ldots, \Omega_{j,k}^{(i)}, \ldots, \Omega_{j,k}^{(K)})||_{\infty}.$$

This norm is inspired by a multi-task sGGM proposed by [11]. When using group-infinity norm, we get FASJEM-I(short for FASJEM-Groupinf). We can derive the optimization for FASJAM-I by changing two proximities in Algorithm 1. Considering that the original formulation in [11] is similar with JGL[8], in the rest of this paper, we call the model from [11] as JGL-groupinf or JGL-I (the corresponded baseline for FASJEM-I).

3.3 Proximal Algorithm for Optimization

Eq. (3.1) includes a convex programming task since the norms we choose are convex. By simplifying notations and adding another parameter, we reformulate it to:

$$\begin{aligned} &\underset{\theta_1,\theta_2}{\operatorname{argmin}}\ f_1(\theta_1) + f_2(\theta_2) \\ \text{subject to :}&||\theta_1 - \operatorname{inv}(T_v(\widehat{\Sigma}_{tot}))||_{\infty} \le \lambda_n \\ &\mathcal{R}'^{*}(\theta_2 - \operatorname{inv}(T_v(\widehat{\Sigma}_{tot}))) \le \epsilon\lambda_n \\ &\theta_1 = \theta_2 \end{aligned} \tag{3.2}$$

Where $f_1(\cdot) = ||\cdot||_1$ and $f_2(\cdot) = \epsilon||\cdot||_{\mathcal{G},2}$. Then we convert Eq. (3.2) to the following equivalent and distributed formulation:

$$\begin{aligned} &\underset{\theta_1,\theta_2,\theta_3,\theta_4}{\operatorname{argmin}}\ f_1(\theta_1) + f_2(\theta_2) + f_3(\theta_3) + f_4(\theta_4) \\ &\text{subject to: } \theta_1 = \theta_2 = \theta_3 = \theta_4 \end{aligned} \tag{3.3}$$

Here $f_3(\theta) = \mathcal{I}_{\{||\theta - inv(T_v(\Sigma_{tot}))||_{\infty} \le \lambda_n\}}(\theta)$ and $f_4(\theta) = \mathcal{I}_{\{||\theta - inv(T_v(\Sigma_{tot}))||^{*}_{\mathcal{G},2} \le \epsilon\lambda_n\}}(\theta)$. $\mathcal{I}_C(\cdot)$ represents the indicator function of a convex set C as $\mathcal{I}_C(x) = 0$ when $x \in C$. Otherwise $\mathcal{I}_C(x) = \infty$. To solve Eq. (3.3), we choose a parallel proximal based algorithm[22] summarized in Algorithm 1. Besides the distributed nature, the proximal algorithm also bring in the benefit that many proximity operators are entry-wise operators for the targeted parameters. The four proximal operators for four functions $\{f_1, f_2, f_3, f_4\}$ (for CPU platform implementation) are included in the Equations Eq. (S:2–2) to Eq. (S:2–5) in Section S:2. With the benefits as proximal operators, Eq. (S:2–2) and Eq. (S:2–4) are entry-wise and Eq. (S:2–3) and Eq. (S:2–5) are group entry-wise.

3.4 GPU Implementation of FASJEM-G

We further revise Algorithm 1 to take advantage of the advanced computational architecture–GPU. This algorithm cannot be directly parallelized on GPU, because GPU is slow for multiple branches based predictions.

Table 1: Four proximity operators implemented on GPU platform.

$[\text{prox}_{\gamma f_1}(x)]_{j,k}^{(i)}$	$\max((x_{j,k}^{(i)} - \gamma), 0) + \min(0, (x_{j,k}^{(i)} + \gamma))$
$\text{prox}_{\gamma f_2}(x_g)$	$x_g \max((1 - \frac{\gamma}{||x_g||_2}), 0)$
$[\text{prox}_{\gamma f_3}(x)]_{j,k}^{(i)}$	$\min(\max(x_{j,k}^{(i)} - a_{j,k}^{(i)}, -\lambda_n), \lambda_n) + a_{j,k}^{(i)}$
$\text{prox}_{\gamma f_4}(x_g)$	$\max(\frac{\lambda_n}{||x_g - a_g||_2}, 1)(x_g - a_g) + a_g$

Therefore, we convert those four operators prox$(\cdot)$ in Algorithm 1 into single soft-threshold based operators which only include simple algorithmic operations like + or max. These operators can be easily parallelized on GPU[23]. The four proximity operators we use to implement FASJEM-G on GPU are summarized in Table 1. More details are included in Section S:2.

Algorithm 1 Parallel proximal algorithm[4]

input K given data blocks $X^{(1)}, X^{(2)}, \ldots, X^{(K)}$. Hyper-parameter:$\alpha$, ϵ, v, λ_n and γ. Learning rate: $0 < \rho < 2$. Max iteration number $iter$.

output Ω_{tot}

Compute Σ_{tot} from $X^{(1)}, X^{(2)}, \ldots, X^{(K)}$
Initialize $\theta^0 = \operatorname{inv}(T_v(\Sigma_{tot})), \theta_j^0 = \operatorname{inv}(T_v(\Sigma_{tot}))$ for $j \in \{1,2,3,4\}$ and $a = \operatorname{inv}(T_v(\Sigma_{tot}))$.
for $i = 0$ **to** $iter$ **do**
  $p_1^i = \text{prox}_{4\gamma f_1}\theta_1^i$
  $p_2^i = \text{prox}_{4\gamma f_2}\theta_2^i$
  $p_3^i = \text{prox}_{4\gamma f_3}\theta_3^i$
  $p_4^i = \text{prox}_{4\gamma f_4}\theta_4^i$
  $p^i = \frac{1}{4}(\sum_{j=1}^{4}\theta_j^i)$
  for $j = 1, 2, 3, 4$ **do**
    $\theta_j^{i+1} = \theta_j^i + \rho(2p^i - \theta^i - p_j^i)$
  end for
  $\theta^{i+1} = \theta^i + \rho(p^i - \theta^i)$
end for
$\Omega_{tot} = \theta^{iter}$

output Ω_{tot}

4 Connecting to Relevant Studies

Estimators for Single task sGGM: Sparse GGM is a highly active topic in the recent literature. Roughly speaking, there exist three major categories of estimators for sGGM. A key class of estimators is based on the regularized maximum likelihood optimization. The popular estimator "graphical lasso"(GLasso) considers maximizing a ℓ_1 penalized normal likelihood [6, 17, 24, 25]. As the second type, CLIME estimator[26] learns sGGM by solving a constrained ℓ_1 optimization. The CLIME formulation can be converted into multiple subproblems of linear programming and has shown more favorable theoretical properties than GLasso. The linear programming while convex, is computationally expensive for large-scale tasks. Recently as a third group of stud-

[4]Four proximity operators used on GPU are defined in Table 1. Hyperparameters are explained in Section 6. Here $j, k = 1, \ldots, p$, $i = 1, \ldots, K$ and $g \in \mathcal{G}$.

ies, soft-thresholding based elementary estimators[16] have been introduced for inferring undirected sparse Graphical models. This paper mostly follows the third category. Besides, there exists quite a number of recent studies trying to scale up single sGGM to a large scale. For example, the BigQUIC algorithm [27] proposes an asymptoticly quadratic optimization to estimate sGGM. The elementary estimator [16] of sGGM is an advanced version of BigQUIC.

Multi-sGGM: Previous Likelihood based Estimators. Most previous methods to jointly estimate multiple sGGMs can be formulated using the penalized MLE Eq. (2.6), including for instance, (1) Joint graphical lasso (JGL-group uses $\mathcal{G}, 2$-norm in Section 3.1 [8]), (2) Node-perturbed JGL [9], (3) Simone [10], and (4) multi-task sGGM proposed by [11]. These methods differ with respect to the second regularization function $\mathcal{R}'(\cdot)$they used to enforce their assumption of similarity among tasks.

Optimization and Computational Comparison: We use JGL-group and the model proposed by [11] (we name it as JGL-groupInf) as baselines in our experiments. As we mentioned in Section 1, the bottleneck of optimizing multi-sGGM in JGL-group is the step of SVD that needs $O(Kp^3)$ time complexity and requires storing K covariance matrix ($O(Kp^2)$ memory cost). Differently, JGL-Groupinf chose a coordinate descent method and proved that their optimization is equivalent to p sequences of quadratic subproblems, each of which costs $O(K^3p^3)$ computation. Therefore the total computational complexity of JGL-Groupinf is $O(K^3p^4)$. Besides, this coordinate descent method needs to store all K covariance matrices in the main memory($O(Kp^2)$ memory cost). Table 2 compares our model with two baselines in terms of time and space cost. Solving our model relies totally on entry-wise and group-entry-wise procedures. Its time complexity is $O(Kp^2)$. This is much faster than the baselines, especially in high-dimensional settings (Table 2). [5] Moreover, in our optimization, learning the parameters for each group $\{\Omega^{(i)}_{j,k}|i = 1, \ldots, K\}$ does not rely on other groups. This means we only need to store K entries of the same group in the memory for computing Eq. (S:2–4) and Eq. (S:2–5). The space complexity $O(K)$ is much smaller than previous methods' $O(Kp^2)$ requirement. [6] The comparisons are in Table 2.

[5]Note that the discussion of time complexity is for each iteration in optimization. We show the Q-linear convergence for all first-order multi-task sGGM estimators in Eq. (S:2–10). Since the baselines and our methods all use first-order optimization, we assume the number of iterations is the same among all methods.

[6]We have provided a GPU implementation of FASJEM in Section 3.3. Although SVD or matrix inversion can also be speed up by GPU parallelization, these method cannot avoid the $O(Kp^2)$ memory cost, which is a huge bottleneck for large-scale problems. In Section 5 we prove that our estimator is completely group entry-wise and asynchronously optimizable, this makes FASJEM only require $O(K)$ memory storage.

Table 2: Comparison to Previous multi-sGGM methods

References	Computational Complexity	Memory Cost
JGL-Group [8]	$O(Kp^3)$	$O(Kp^2)$
JGL-GroupInf [11]	$O(K^3p^4)$	$O(Kp^2)$
FASJEM Models	$O(Kp^2)$ (if paralleling completely, $O(K)$)	$O(K)$

Previous Studies using Elementary based Estimators: Most previous studies of multi-sGGMs follow the penalized MLE framework. Few works of Multi-task sGGM follow the CLIME formulation, since it is not easy to transfer two regularizers into the CLIME formulation (summarized in Table S:1). Based on the authors' knowledge, no previous multi-sGGM studies have followed the elementary estimators(EE) formulation. As a simple soft-thresholding based estimator, elementary estimators (EE) have been used for other tasks as well. Table S:1 summarizes three different types of previous tasks for which EE can be applied: high-dimensional regression, single sGGM and multi-sGGM. For comparison, we show how these tasks have been solved through the penalized likelihood framework in the second column and use the the third column to show studies following the CLIME formulation.

Convergence Rate Analysis: Although previous joint sGGMs work well on datasets whose K and p are relatively small, two important questions remain unanswered: (1) what's the statistical convergence rate of these joint estimators? and (2) what's the benefits of joint learning? The convergence rate of estimating single-task sGGM has been well investigated[6, 17, 24, 25]. These studies proved that the estimator of single-task sGGM holds a consistent convergence rate $O(\sqrt{\frac{\log p}{n}})$ if given n data samples. However, none of the previous joint-sGGM studies provide such theoretical analysis. Experimental evaluations in previous joint-sGGM papers have shown better performance of running joint estimators over running single-task sGGM estimators on each dataset separately. However, it hasn't been proven that theoretically this joint estimation is better. We successfully answer these two remaining questions in Section 5.

5 Theoretical Analysis

In this section, we prove that our estimator can be optimized asynchronously in a group entry-wise manner. We also provide the proof of the theoretical error bounds of FASJEM.

5.1 Group entry-wise and parallelizing optimizable

Theorem 5.1. ***(FASJEM is Group entry-wise optimizable)*** *Suppose we use FASJEM to infer multiple inverse of covariance matrices summarized as* $\widehat{\Omega}_{tot}$. $\{\widehat{\Omega}^{(i)}_{j,k}|i=1,\ldots,K\}$ *describes a group of* K *entries at* (j,k) *position. Varying* $j\in\{1,2,\ldots,p\}$ *and* $k\in\{1,2,\ldots,p\}$*, we have totally* $p\times p$ *groups. If these groups are independently estimated by FASJEM, then we have,*

$$\bigcup_{j=1}^{p}\bigcup_{k=1}^{p}\{\widehat{\Omega}^{(i)}_{j,k}|i=1,\ldots,K\}=\widehat{\Omega}_{tot}. \tag{5.1}$$

Proof. Eq. (S:2–6) and Eq. (S:2–8) are soft-thresholding based operators on each entry. Eq. (S:2–7) and Eq. (S:2–9) are soft-thresholding operators on each group of entries. □

Corollary 5.2. *We can decompose FASJEM into* $p\times p$ *subproblems that are independent from each other, and solve each subproblem at a time. Therefore our estimator only requires* $O(K)$ *memory storage for computation.*

This corollary proves the claims we showed in section 4. Through Theorem (5.1), it is important to notice that the optimization on multiple groups of entries can be totally **parallelized**.

5.2 Theoretical error bounds

In this subsection, we first provide the error bounds for elementary super-position estimator (ESS estimator) under $I=\{1,2\}$. We then use this general bound to prove the error bound for FASJEM-G. All the proofs are included in Section S:8. We also include the error bounds for elementary estimator (EE) in Section S:7.

Extension to ESS: For the multiple-task case, we need to consider two or more regularization functions. For instance, in FASJEM-G we assume the sparsity of parameter and the group sparsity among tasks. Since we only consider the models with two regularization function, we consider the error bounds of the following elementary super-position estimator formulation in the rest of the section.

$$\begin{aligned}&\underset{\theta_1,\theta_2}{\operatorname{argmin}}\lambda_1\mathcal{R}_1(\theta_1)+\lambda_2\mathcal{R}_2(\theta_2)\\ &\text{subject to:}\mathcal{R}^*_i(\widehat{\theta}_n-(\theta_1+\theta_2))\le\lambda_i\ ,i=1,2\end{aligned} \tag{5.2}$$

This equation restricts the number of penalty functions to 2. Similar to the single-task error bounds (in Section S:7) , we naturally extend condition **(C2)** to the following condition:

(C3) $\text{proj}_{\mathcal{M}_i^{\perp}}(\theta_i^*)=0,\ i=1,2.$

We borrow the following condition from [21], which is a structural incoherence condition ensuring that the non-interference of different structures.

(C4) Let $\Phi:=\max\{2+\frac{3\lambda_1\Psi_1(\bar{\mathcal{M}}_1)}{\lambda_2\Psi_2(\bar{\mathcal{M}}_2)},2+\frac{3\lambda_2\Psi_2(\bar{\mathcal{M}}_2)}{\lambda_1\Psi_1(\bar{\mathcal{M}}_1)}\}$.
$\max\{\sigma_{\max}(\mathcal{P}_{\bar{\mathcal{M}}_1}\mathcal{P}_{\bar{\mathcal{M}}_2}),$
$\sigma_{\max}(\mathcal{P}_{\bar{\mathcal{M}}_1}\mathcal{P}_{\bar{\mathcal{M}}_2^{\perp}})\sigma_{\max}(\mathcal{P}_{\bar{\mathcal{M}}_1^{\perp}}\mathcal{P}_{\bar{\mathcal{M}}_2^{\perp}})\}\le\frac{1}{16\Phi^2}$

where $\mathcal{P}_{\bar{\mathcal{M}}}$ is the matrix corresponding to the projection operator for the subspace $\bar{\mathcal{M}}$. The definition of $\Psi(\cdot)$ are included in Definition (S:7.1).

With these two conditions, we have the following theorem:

Theorem 5.3. *Suppose that the true parameter* θ^* *satisfies the conditions* ***(C3)(C4)*** *and* $\lambda_i\ge\mathcal{R}^*_i(\widehat{\theta}-\theta^*)$*, then the optimal point* $\widehat{\theta}$ *of Eq. (5.2) has the following error bounds:*

$$\mathcal{R}^*_i(\widehat{\theta}-\theta^*)\le 2\lambda_i,\ i=1,2 \tag{5.3}$$

$$\mathcal{R}_i(\widehat{\theta}-\theta^*)\le\frac{32}{\lambda_i}(\max_i\lambda_i\Psi(\bar{\mathcal{M}}_i))^2,\ i=1,2 \tag{5.4}$$

$$||\widehat{\theta}-\theta^*||_F\le 8\max_i\lambda_i\Psi(\bar{\mathcal{M}}_i) \tag{5.5}$$

Notice that for FASJEM-G model, $\mathcal{R}_1=||\cdot||_1$ and $\mathcal{R}_2=||\cdot||_{\mathcal{G},2}$. Based on the results in[18], $\Psi(\bar{\mathcal{M}}_1)=\sqrt{s}$ and $\Psi(\bar{\mathcal{M}}_2)=\sqrt{s_{\mathcal{G}}}$, where s is the number of nonzero entries in Ω_{tot} and s_g is the number of groups in which there exists at least one nonzero entry. Clearly $s>s_g$. Also in practice, to utilize group information, we have to choose hyperparameter $\lambda_n>\lambda'_n$ ($\lambda_1>\lambda_2$ in Eq. (5.5)). Therefore by Theorem (5.3), we have the following theorem,

Theorem 5.4. *Suppose that* $\mathcal{R}_1=||\cdot||_1$ *and* $\mathcal{R}_2=||\cdot||_{\mathcal{G},2}$ *and the true parameter* Ω^*_{tot} *satisfies the conditions* ***(C3)(C4)*** *and* $\lambda_i\ge\mathcal{R}^*_i(\widehat{\Omega}_{tot}-\Omega^*_{tot})$*, then the optimal point* $\widehat{\Omega}_{tot}$ *of Eq. (3.1) has the following error bounds:* $||\widehat{\Omega}_{tot}-\Omega^*_{tot}||_F\le 8\sqrt{s}\lambda_n$.

We then derive a corollary of Theorem (5.4) for FASJEM-G. A prerequisite is to show that $inv(T_v(\widehat{\Sigma}_{tot}))$ is well-defined. The following conditions define a broad class of sGGM that satisfy the requirement. Similar results are also introduced by[16].

Conditions for elementary estimator of sGGM:
C-MinInfΣ The true parameter Ω^*_{tot} of Eq. (5.2) has bounded induced operator norm, i.e., $|||{\Omega^{(i)}}^*|||_\infty:=\sup_{w\neq 0\in\mathbb{R}^p}\frac{||{\Sigma^{(i)}}^*w||_\infty}{|w|_\infty}\le\kappa_1\forall i.$

C-SparseΣ The true multiple covariance matrices $\Sigma^*_{tot}:=inv(\Omega^*_{tot})$ are "approximately sparse" along the lines [28] : for some positive constant D, ${\Sigma^{(i)}_{j,j}}^*\le D$ for all diagonal entries. Moreover, for some $0\le q<1$ and $c_0(p)$, $\max_i\sum_{j=1}^{p}|{\Sigma^{(i)}_{j,k}}^*|^q\le c_0(p)\forall i$. If $q=0$, then this condition reduce to Σ^* being sparse. We additionally require $\inf_{w\neq 0\in\mathbb{R}^p}\frac{|{\Omega^{(i)}}^*w|_\infty}{|w|_\infty}\ge\kappa_2$.

Error bounds of FASJEM-group: In FASJEM, $\theta_1^* = \theta_2^* = \frac{1}{2}\theta^*$. θ_i is the parameter w.r.t a subspace pair$(\mathcal{M}_i, \bar{\mathcal{M}}_i^{\perp})$, where $i = 1, 2$.

Here $\mathcal{R}_1 = ||\cdot||_1$ and $\mathcal{R}_2 = ||\cdot||_{\mathcal{G},2}$. We assume the true parameter θ^* satisfies **C-MinInf**Σ and **C-Sparse**Σ conditions. Using the above theorems, we have the following corollary:

Corollary 5.5. *If we choose hyperparameters $\lambda'_n < \lambda_n$. Let $v := a\sqrt{\frac{\log p'}{n_{tot}}}$ for $p' = \max(Kp, n_{tot})$. Then for $\lambda_n := \frac{4\kappa_1 a}{\kappa_2}\sqrt{\frac{\log p'}{n_{tot}}}$ and $n_{tot} > c \log p'$, with a probability of at least $1 - 2C_1\exp(-C_2 Kp\log(Kp))$, the estimated optimal solution $\widehat{\Omega}_{tot}$ has the following error bound:* $||\widehat{\Omega}_{tot} - \Omega^*_{tot}||_F \leq 32\frac{4\kappa_1 a}{\kappa_2}\sqrt{\frac{s\log p'}{n_{tot}}}\}$

where a, c, κ_1 and κ_2 are constants.

The convergence rate of single-task sGGM is $O(\log p/n_i)$. In high-dimensional setting, $p' = Kp$ since $Kp > n_{tot}$. Assuming $n_i = \frac{n_{tot}}{K}$, the convergence rate of single sGGM is $O(K\log p/n_{tot})$. Clearly, since $K\log p > \log(Kp)$, the convergence rate of FASJEM is better than single-task sGGM.

6 Experiment

Multiple simulated datasets and four real-world biomedical datasets are used to evaluate FASJEM.

6.1 Experimental Settings

Baseline: We compare (1)FASJEM-G versus JGL-group [8]; (2)FASJEM-I versus JGL-groupinf [11]. This is because the specific FASJEM estimator and its baseline share the same second-penalty function. [7] Three evaluation metrics are used for such comparisons.

- **Precision:** We use the edge-level false positive rate (FPR) and true positive rate (TPR) to measure the predicted graphs versus true graph. Repeating the process 10 times, we obtain average metrics for each method we tests. Here, FPR $= \frac{\text{FP}}{\text{FP + TN}}$ and TPR $= \frac{\text{TP}}{\text{TP + FN}}$. TP (true positive) and TN (true negative) mean the number of true nonzero entries and the number of true zero entries estimated by the predicted precision matrices. The FPR vs. TPR curve shows multi-point performance of a method over a range of the tuning parameter. The bigger the area under a FPR-TPR curve, the better a method has achieved overall.
- **Speed:** The time (log(second)) between the whole program's start and end indicates the speed of a certain method under a specific configuration of hyper-parameters. To be fair, we set up two types of comparisons. The first one fixes the number of tasks (K) but varies the dimension (p). This shows the performance of each method under a high-dimensional setting. The other type fixes the dimension (p) but varies the number of tasks (K). This measures the performance of each method when having a large number of tasks.
- **Memory:** For each method, we vary the number of tasks (K) and the dimension (p) until a specific method terminates due to the "out of memory" error. This measures the memory capacity of the corresponding method.

Our implementation: We implement FASJEM on two different architectures: (1)CPU only and (2)GPU [8]. Similar to the JGL-group from [8], we implement the CPU version FASJEM-G and FASJEM-I with R. We choose torch7 [29] (LUA based) to program FASJEM on GPU machine. [9]

Selection of hyper-parameters: In this experiment, we need to choose the value of three hyper-parameters. The first one v is unique for elementary-estimator based sGGM models. The second λ_n(in some models also noted as λ_1) is the main hyper-parameter we need to tune. The third ϵ equals to $\frac{\lambda'_n}{\lambda_n}$ (The notation λ_2 is normally used in related works instead of λ'_n).

- v: We pre-choose v in the set $\{0.001i | i = 1, 2, \ldots, 1000\}$ to guarantee $T_v(\Sigma_{tot})$ is invertible.
- λ_n [10]: Recent research studies from [18] and [16] conclude that the regularization parameter λ_{n_i} of a single task with n_i samples should be chosen with $\lambda_{n_i} \propto \sqrt{\frac{\log p}{n_i}}$. Combining this result and our convergence rate analysis in Section 5, we choose $\lambda_n = \alpha\sqrt{\frac{\log Kp}{n_{tot}}}$ where α is a hyper-parameter. The hyperparameter γ in Algorithm 1 equals to λ_n.
- ϵ: We select the best ϵ from the set $\{0.1i | i = 1, 2, \ldots, 10\}$ using cross-validation.

6.2 Experiments on simulated datasets

Using the following "Random Graph Model"(RGM), we first generate a set of synthetic multivariate Gaussian datasets, each of which includes samples of K tasks described by p variables. From [25], this "Random Graph Model" assumes $\Omega^{(i)} = B^{(i)} + \delta^{(i)}I$, where each off-diagonal entry in $B^{(i)}$ is generated independently, equals to 0.5 with probability $0.05i$ and, equals to 0 with probability $1 - 0.05i$. $\delta^{(i)}$ is selected large enough to guarantee the positive definiteness of precision matrix.

For each case of p, we use this model to generate K random sparse graphs. For each graph (task), $n = p/2$

[7] Since single-sGGM EE has a closed-form solution (i.e., no iterative steps are needed in optimization), we do not include it as baseline.

[8] Information of Experiment Machines: The machine that we use for experiments includes Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz with a 8GB memory. The GPU that we use for experiments is Nvidia Tesla K40c with 2880 cores and 12GB memory.

[9] Though the ideal memory requirement of FASJEM is only $O(K)$, IO costs should also be taken into account in real implementations. As being proved, Ω_{tot} is group-entry-wise optimizable. The parameter groups are independently estimated in the parallelized style. When implementing FASJEM in a single machine (our experimental setting), we prefer to choose smaller m to make full use of the main memory, where m is the number of parameter groups which are estimated at the same time.

[10] $\lambda_n = 0.1$ used for time and memory experiments

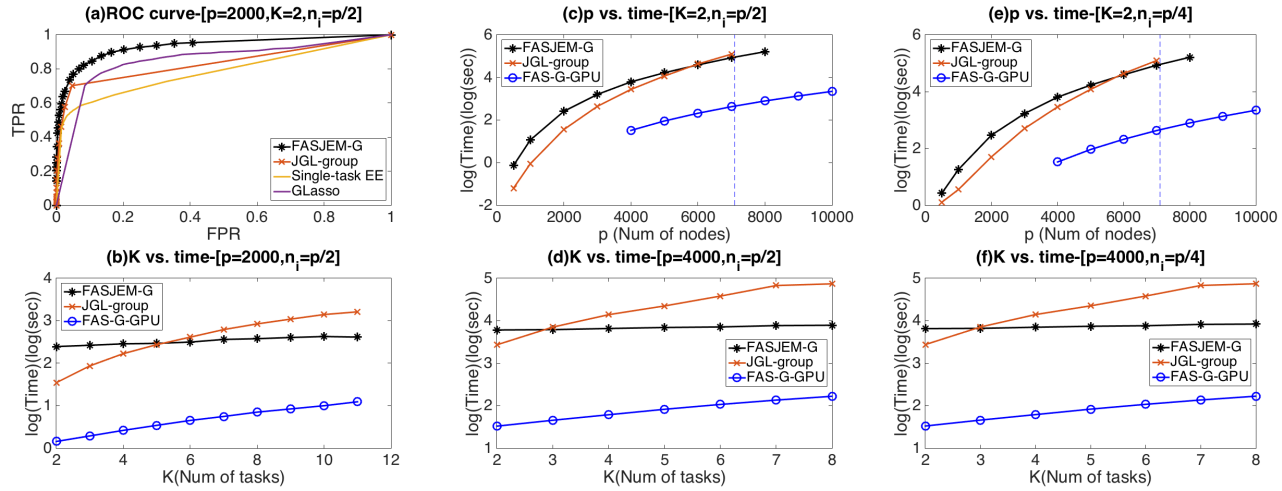

Figure 1: FASJEM-G versus JGL-group with respect to accuracy, speed and memory capacity. **(a):** FPR-TPR curves of two methods and two single-sGGM baselines on the simulated dataset using Random Graph Model when $p = 2000$ and $K = 2$. (AUC number–FASJEM-G:0.9332, JGL-group:0.5803, EE for sGGM:0.7852, GLasso:0.8504) **(c) and (e):** Time versus p(the number of variables) curves from FASJEM-G, JGL-group and FASJEM-G's GPU implementation. (c) uses $n_i = p/2$ and (e)$n_i = p/4$. **(b), (d) and (f):** the time versus K(the number of tasks) curves for two methods plus FASJEM-G-GPU. (b) uses $p = 2000$ and $n_i = p/2$, (d) uses $p = 4000$ and $n_i = p/2$ and (f) uses $p = 4000$ and $n_i = p/4$.

data samples are generated randomly by following $N(0, (\Omega^{(i)})^{-1})$. For each (K, p) parameter setting we test in the experiment, we use this RGM process to generate 10 different datasets (with different random seeds). Then we apply our methods and baseline methods on these datasets to obtain estimated sGGM networks. All results or curves we show in the rest of this section are average scores/curves over 10 trials for each case of parameter configuration.

Figure 1(a) and S:2(a) present FPR vs. TPR curves of two proposed methods: FASJEM-G and FASJEM-I versus their corresponding baselines: JGL-group and JGL-groupinf, on the simulated datasets. We choose $p = 2000$ and $K = 2$. FPR-TPR curve plots are obtained by varying its tuning parameter λ_n over a range of $\{0.05 \times \sqrt{\frac{\log Kp}{n_{tot}}} \times i | i \in \{1, 2, 3, \ldots, 30\}\}$ and interpolating the obtained performance points (We pre-choose v and ϵ and the methods are introduced in Section 6.1). The two subfigures of "ROC curve" clearly show that FASJEM-G and FASJEM-I obtain better under-plot areas than corresponding JGL-group and JGL-groupinf.

Then in Figure 1(c)(e) and S:2(c)(e) we show the curves of Time vs. Dimension p comparing FASJEM-G and FASJEM-I versus their baselines. Sub-figure 1(c) and S:2(c) choose $n_i = p/2$. Sub-Figures 1(e) and S:2(e) use $n_i = p/4$. The CPU curves are obtained by varying p in the set of $\{1000i | i = 0.5, 1, 2, 3, \ldots, 8\}$. GPU curves are obtained by varying p in the set of $\{1000i | i = 4, 5, 6, \ldots, 10\}$. The subfigure (c) "$p$ versus time-$[K = 2, n_i = p/2]$" and subfigure (e) "p versus time-$[K = 2, n_i = p/4]$" in Figure 1 show that though JGL-group obtains a slightly better performance than our method under lower-dimension cases, when reaching high dimensional stages, FASJEM-G performs similarly and trains much faster than the baseline method. Figure S:2(c) and Figure S:2(e) provide similar conclusions for FASJEM-I vs JGL-groupinf. In addition, the baselines cannot handle $p \geq 8000$ because these approaches require too much memory. Clearly our proposed FASJEM methods can still perform reasonable well for the large-scale cases. This shows that our methods makes better usage of memory. Moreover, both FASJEM-G-GPU and FASJEM-I-GPU implementations spend only $\frac{1}{10}$ of train time against its CPU implementations. This proves that GPU parallelization can speed up FASJEM significantly.

Figure 1(b)(d)(f) and S:2(b)(d)(f) show the curves about "Time vs. Number of tasks-K" comparing our methods FASJEM-G and FASJEM-I versus two baseline methods JGL-group and JGL-groupinf respectively. These sub-figures use the varying K as the x-axis over a range of $\{2, 3, \ldots, 8\}$. Sub-figures (b) use $p = 2000$, $n_i = p/2$, sub-figures (d) use $p = 4000$, $n_i = p/2$ and sub-figures (f) choose $p = 4000$, $n_i = p/4$. These figures show that the JGL-group and JGL-groupinf obtain a slightly better speed than two FASJEM, under small K cases. For larger K, our methods perform faster than the baseline methods. The conclusion hold across three cases with different pairs of (p, n_i), indicating that the advantage of our methods do not change by working on graphs and datasets of different sizes. In addition, when $p = 4000$, JGL-group and JGL-groupinf cannot handle $K \geq 5$ (i.e., the R program died) due to the memory issue on our experiment machine, while both FASJEM-G and FASJEM-I can. This proves that FASJEM requires a lower memory cost than the base-

lines. In all subfigures (b), (d) and (f), FASJEM curves are roughly linear. The experimental results match with the computational complexity analysis we have performed in Table 2 (the computation cost of FASJEM is linear to K). Moreover, subfigures (b), (d) and (f) show that both FASJEM-G-GPU and FASJEM-I-GPU implementations spend only $\frac{1}{10}$ time of their CPU implementations respectively. This confirms that GPU-parallelization can speed up FASJEM significantly.

Furthermore in Section S:6, we compare FASJEM-G and JGL-group on four different real-world datasets. FASJEM-G consistently outperforms JGL-group on all four datasets in recovering more known edges.

References

[1] ENCODE Project Consortium et al. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57–74, 2012.

[2] Trey Ideker and Nevan J Krogan. Differential network biology. *Molecular systems biology*, 8(1):565, 2012.

[3] Tim Beck, Robert K Hastings, Sirisha Gollapudi, Robert C Free, and Anthony J Brookes. Gwas central: a comprehensive resource for the comparison and interrogation of genome-wide association studies. *European Journal of Human Genetics*, 22(7):949–952, 2014.

[4] Steffen L Lauritzen. *Graphical models*. Oxford University Press, 1996.

[5] Kantilal Varichand Mardia, John T Kent, and John M Bibby. Multivariate analysis. 1980.

[6] Ming Yuan and Yi Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[7] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[8] Patrick Danaher, Pei Wang, and Daniela M Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2013.

[9] Karthik Mohan, Palma London, Maryam Fazel, Su-In Lee, and Daniela Witten. Node-based learning of multiple gaussian graphical models. *arXiv preprint arXiv:1303.5145*, 2013.

[10] Julien Chiquet, Yves Grandvalet, and Christophe Ambroise. Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553, 2011.

[11] Jean Honorio and Dimitris Samaras. Multi-task learning of gaussian graphical models. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 447–454, 2010.

[12] Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu. Joint estimation of multiple graphical models. *Biometrika*, page asq060, 2011.

[13] Bai Zhang and Yue Wang. Learning structural changes of gaussian graphical models in controlled experiments. *arXiv preprint arXiv:1203.3532*, 2012.

[14] Yi Zhang and Jeff G Schneider. Learning multiple tasks with a sparse matrix-normal penalty. In *Advances in Neural Information Processing Systems*, pages 2550–2558, 2010.

[15] Yunzhang Zhu, Xiaotong Shen, and Wei Pan. Structural pursuit over multiple undirected graphs. *Journal of the American Statistical Association*, 109(508):1683–1696, 2014.

[16] Eunho Yang, Aurélie C Lozano, and Pradeep K Ravikumar. Elementary estimators for graphical models. In *Advances in Neural Information Processing Systems*, pages 2159–2167, 2014.

[17] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

[18] Sahand Negahban, Bin Yu, Martin J Wainwright, and Pradeep K Ravikumar. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems*, pages 1348–1356, 2009.

[19] Eunho Yang, Aurelie Lozano, and Pradeep Ravikumar. Elementary estimators for high-dimensional linear regression. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 388–396, 2014.

[20] Eunho Yang, Aurélie C Lozano, and Pradeep D Ravikumar. Elementary estimators for sparse covariance matrices and other structured moments. In *ICML*, pages 397–405, 2014.

[21] Eunho Yang and Pradeep K Ravikumar. Dirty statistical models. In *Advances in Neural Information Processing Systems*, pages 611–619, 2013.

[22] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing.

In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[23] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.

[24] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[25] Adam J Rothman, Peter J Bickel, Elizaveta Levina, Ji Zhu, et al. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

[26] Tony Cai, Weidong Liu, and Xi Luo. A constrained l1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

[27] Cho-Jui Hsieh, Matyas A Sustik, Inderjit S Dhillon, and Pradeep D Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. In *NIPS*, pages 2330–2338, 2011.

[28] Peter J Bickel and Elizaveta Levina. Covariance regularization by thresholding. *The Annals of Statistics*, pages 2577–2604, 2008.

[29] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.