# Efficient Computational Methods for Advanced Sparse Estimation

Jie Chen

MARCH 2022

# Efficient Computational Methods for Advanced Sparse Estimation

A dissertation submitted to

THE GRADUATE SCHOOL OF ENGINEERING SCIENCE

OSAKA UNIVERSITY

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN SCIENCE

BY

Jie Chen

MARCH 2022

**Abstract**

It is difficult to infer statistical objects in regression and classification when the number of samples is small and the number of features is large. Sparse estimation addresses the problem by regularizing the objective function, plus the constraint multiplied by a constant $\lambda$, and selecting the appropriate features without overfitting. Least absolute shrinkage and selection operator (Lasso) is a classic sparse estimation; its objective function to be minimized is convex so that we can find the solution efficiently. This thesis considers convex optimization problems for sparse estimation, particularly joint graphical Lasso (JGL) and convex biclustering (CB).

To solve JGL, thus far, the alternating direction method of multipliers (ADMM) has been the main approach. However, converging to obtain a high-accuracy solution for ADMM often takes time, which is not feasible in large data sets. In the first part (Chapter 3) of this thesis, we propose proximal gradient algorithms with and without a backtracking option for JGL and further show the boundedness for the solution of the JGL problem and the iterates in the algorithms, which guarantee the linear convergence of the proximal gradient method. For the procedure with backtracking, we reduce the updated iterative steps to subproblems that can be solved efficiently and accurately by Lasso-type problems. We modified the step-size selection by extending the strategy in [86] to one without backtracking, which significantly reduces the computational time needed to evaluate objective functions.

The existing techniques to solve CB were unsatisfactory. The convex biclustering algorithm (COBRA) solves twice the standard convex clustering problem with a nondifferentiable function optimization. In the second part (Chapter 4) of this thesis, we instead convert the original optimization problem to a differentiable one. Then, we combine the basic procedures in the augmented Lagrangian method (ALM) with the accelerated gradient descent method (Nesterov's accelerated gradient method), which can attain a $O(1/k^2)$ convergence rate. The conventional algorithms are sensitive to the tuning parameter $\lambda$, which is not feasible in some biclustering applications that are needed to solve a wide range of $\lambda$. However, our proposed method is not greatly influenced by the tuning parameter $\lambda$.

The experimental results indicate that the proposed algorithms can achieve high accuracy, and their efficiency is competitive with state-of-the-art algorithms, even for large-scale problems.

# Contents

# List of Figures



# List of Tables

# 1 Introduction

## 1.1 Introduction

We often require the analysis of high-dimensional (the number of features $p$ is much larger than the sample size $n$) and large-scale data sets for mining, such as gene regulatory networks, recommendation systems, text mining, and social networks. With the volume and complexity of data growth, people often tend to be distracted by irrelevant information. Thus, we need to extract useful information representing an accurate model from massive data sets. Specifically, in statistical learning, estimation leads to solving the following minimization problem,

$$\min_{\theta} l(\theta) \tag{1}$$

For example, the loss function $l(\theta)$ can be the least square in linear regression. However, minimizing the loss function $l(\theta)$ in a naive manner may lead to overfitting, especially in a high-dimensional setting.

To extract useful information and avoid overfitting from high-dimensional data, variable selection will be unavoidable [71]. We choose a subset of relevant features from the origin, enhancing the models' readability and learning performance and decreasing computational complexity. Regularization is an essential principle in variable selection by adding a penalty term $P(\theta)$ to (1). While the model (1) is combined with different penalty terms (such as $\ell_1$ norm and $\ell_2$ norm), we can obtain various sparse models for different demands and prevent overfitting. We describe the minimization problem as follows:

$$\min_{\theta} l(\theta) + \lambda P(\theta), \tag{2}$$

where $\lambda \geq 0$ is the tuning parameter, determining the degree to penalize the model.

Accurate fitting models and efficient computations to solve the models are of great importance when faced with enormous data sets. The least absolute shrinkage and selection operator (Lasso) [82] is an attractive regularization model with efficient solvers [82, 5, 85]. Moreover, the properties with Lasso have recently been well studied in [26, 100, 10, 99, 53, 84]. There are many variants of Lasso-type models. Table 1 summarizes some sparse estimation methods extended by Lasso. We will describe them in detail in the following chapters.

Table 1: Sparse estimation models

| Model | Author | Optimization problem |
|---|---|---|
| Group Lasso | Friedman et al. [25] | $\frac{1}{2}\|\|y - \sum_{g=1}^{G} X\theta_g\|\|_2^2 + \lambda \sum_{g=1}^{G} \|\|\theta_g\|\|_1$ |
| Fused Lasso | Tibshirani et al. [83] | $\frac{1}{2}\|\|y - X\theta\|\|_2^2 + \lambda \sum_{i<j} \|\theta_i - \theta_j\|$ |
| Convex Clustering | Hocking et al. [39] | $\frac{1}{2}\|X - U\|_F^2 + \lambda \sum_{i<j} \omega_{ij} \|U_{i\cdot} - U_{j\cdot}\|_2$ |
| Sparse Convex Clustering | Wang et al. [87] | $\frac{1}{2}\|X - U\|_F^2 + \lambda_1 \sum_{i<j} \omega_{ij} \|U_{i\cdot} - U_{j\cdot}\|_2 + \lambda_2 \sum_i \|\|U_{i\cdot}\|\|_1$ |
| Graphical Lasso | Friedman et al. [23] | $\log\det\Theta - \mathrm{trace}(S\Theta) - \lambda\|\|\Theta\|\|_1$ |

We often use the notions of the Lasso penalty for sparse purposes. For instance, the graphical Lasso [53, 52, 23] is a sparse estimation for the Gaussian graphical model, using the Lasso penalty to promote zero values in the precision matrix (the inverse of the covariance matrix). Wang et al. [87] proposed sparse convex clustering, which uses the Lasso penalty to detect uninformative features in convex clustering.

Meanwhile, the increasing models of sparse estimation evolve with the demands of computational methods. Specifically, in the optimization problem (2), because penalty terms such as $\ell_1$ and $\ell_2$ norms are convex. Moreover, if the loss function $l(\theta)$ is convex, we can find the solution efficiently, and the result is often sparse. The convex formulation has substantial advantages in not only theoretical guarantees but also practical benefits. The most appealing one is that the convex formulation ensures that the result will reflect globally optimal solutions regardless of the initialization settings [7], leading to affordable conveniences. In addition, we can apply various scalable and powerful algorithms for convex optimization problems. With the development of convex optimization, several researchers reformulate some estimation problems into convex formulations to utilize the numerous advantages of convexity, such as [11, 39, 15, 79]. Despite these advantages, the optimization problems of the variants and extensions of the Lasso are more challenging than Lasso due to the complexity of the penalty and the multiple terms in the models.

To find the solution of the estimators $\theta$, we need to solve the optimization problem (2). It is sometimes possible to directly obtain closed-form solutions from the problem (2), such as least-squares regression. However, it is often difficult to obtain closed-form solutions. Hence, we need iterative algorithms to attain the optimality conditions by letting the gradient or subgradient of (2) be or tend to be zero [7].

The penalty term $P(\theta)$ may be difficult to solve in optimization procedures, especially when nondifferentiable or indecomposable. The alternating direction method of multipliers (ADMM), which was introduced in [29, 27, 7], is a procedure for solving convex optimization problems for general purposes. Many studies use ADMM to solve sparse estimation models that contain nondifferentiable or indecomposable terms; these studies include [17, 13, 91, 92].

However, ADMM takes time before converging to a high-accuracy solution [8]. Hence, it is challenging for ADMM to solve large-scale problems. In addition, we need to solve the problem for several $\lambda$ values and select a single solution by criteria such as AIC, BIC, and cross-validation. Consequently, we hope to solve each single-value problem in the procedures efficiently. The gradient descent method is one of the most popular first-order techniques to solve convex optimization in machine learning and deep learning. In the first-order method, we mean to use the first-order gradient information, avoiding the calculations of the time-consuming Hessian matrices. Meanwhile, first-order methods can attain moderate accuracy, lending themselves to many applications. The accelerated gradient descent method, such as Nesterov's accelerated gradient method [60, 58], has been proposed and applied to several machine learning problems due to the competitive convergence rate being improved from the original $O(1/k)$ to $O(1/k^2)$. Nevertheless, we cannot apply the standard gradient descent method in some nondifferentiable cases due to the lack of an efficient way to obtain a gradient.

Some modifications to the gradient descent method can efficiently solve nondifferentiable problems. The proximal gradient method is an appealing technique for solving nondifferentiable terms in large-scale problems. We can regard the essential operation in the proximal gradient method as a convex optimization problem, which can be solved quickly using traditional methods [61]. This approach has recently acquired popularity for solving Lasso-type models. For example, Yuan et al. [95] proposed using the proximal gradient method to solve the group Lasso problem. Guillot et al. [31] used the proximal gradient method to solve the graphical Lasso problem. Regarding improvement, Beck and Teboulle [6] provided the accelerated version of the proximal gradient method with a $O(1/k^2)$ convergence rate by extending Nesterov's accelerated gradient method.

With the development of sparse estimation, the models have become more complicated in recent decades, and the related optimization problems have become quite challenging. This thesis aims to propose efficient computational methods to solve the corresponding optimization problems of two advanced sparse estimation models.

Our two main objective models are joint graphical Lasso (JGL) and convex biclustering (CB), which have drawn much recent attention and attracted many users. Nevertheless, the existing algorithms for these models are time-consuming, especially for large data sets, which impedes the development of the models.

First, we propose two efficient proximal gradient methods for JGL, one with backtracking line search and another without the backtracking line search option. To obtain the step size in the algorithm, we need the Lipschitz parameter, but it is difficult to obtain the parameter of the joint graphical Lasso, which is why the backtracking line search is required. We offered two methods because the evaluations in the backtracking line search are sometimes expensive. Furthermore, we provided the boundness of the solution of the joint graphical Lasso and the iterates in our algorithms, guaranteeing the linear convergence of the proximal gradient method.

Then, we propose an efficient and stable algorithm for convex biclustering. The existing algorithms, such as ADMM and COBRA, are sensitive to the model's tuning parameter $\lambda$. When $\lambda$ grows, the computation becomes enormous. Many applications must solve a wide range of $\lambda$ to explore the resulting changes in the biclustering solutions, and the conventional methods are not feasible. Moreover, the ADMM-based algorithms show a lower convergence rate than COBRA. We combined the basic procedures of the augmented Lagrangian method with the accelerated gradient descent method, which is not greatly influenced by the parameter $\lambda$, and the convergence rate was competitive with the conventional methods.

## 1.2 Overview of the thesis

The remainder of this thesis is organized as follows. In Chapter 2, we first briefly introduce some concepts in convex optimization and sparse estimation. Then, we review several important algorithms, namely, the gradient descent method, the proximal gradient descent method, their extensions, and ADMM, which serve as basic knowledge throughout this thesis. In Chapter 3, we propose two efficient algorithms for joint graphical Lasso with and without backtracking options. Moreover, we provide the boundness of the solution of the joint graphical Lasso and the iterates in the proximal gradient algorithm, which can guarantee the linear convergence rate of the algorithm. In Chapter 4, we introduce clustering, convex clustering, and convex biclustering. We propose an efficient algorithm for convex biclustering and provide numerical experiments to show its efficiency and accuracy. In Chapter 5, we present the proof of the propositions of Chapters 3 and 4. In

Chapter 6, we draw some conclusions from the main results and point out the further improvement of the research contents.

Notation: We use $||x||_p$ to denote the $\ell_p$ norm of a vector $x \in \mathbb{R}^d$, $||x||_p := (\sum_{i=1}^{d} |x_i|^p)^{\frac{1}{p}}$ for $p \in [1, \infty)$, and $||x||_\infty := \max_i |x_i|$. For a matrix $X \in \mathbb{R}^{p \times q}$, we use $||X||_F$ to denote the Frobenius norm, $||X||_2$ to denote the spectral norm, $||X||_\infty := \max_{i,j} |x_{i,j}|$, and $||X||_1 := \sum_{i=1}^{p} \sum_{j=1}^{q} |x_{i,j}|$ if not specified. The inner product is defined by $\langle X, X \rangle := \text{trace}(X^T X)$.

# 2 Background

This chapter reviews the basic knowledge for the convex optimization problem and sparse estimation. For the functions $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^p \to \mathbb{R}$, we consider the corresponding optimization method in the following form:

$$\min_x F(x) := f(x) + g(Ax) \tag{3}$$

with variables $x \in \mathbb{R}^n$, and matrix $A \in \mathbb{R}^{p\times n}$. We are mainly focused on (3) since many problems or models in machine learning and statistics fields can be reformulated in the form (3). Moreover, all of the problems we consider in this thesis can be expressed in the form (3).

## 2.1 Basic definition

This subsection introduces some basic definitions and terms in convex optimization that will be used in the sequel.

**Convex set:** We say a set $C$ is *convex set*, if for any $x, y \in C$, and $\alpha \in [0, 1]$, it satisfies

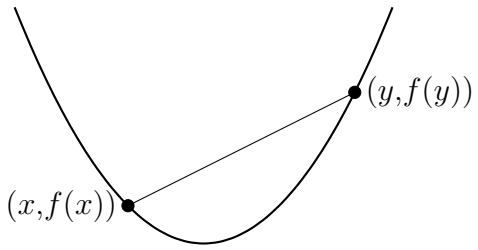
$$\alpha x + (1 - \alpha)y \in C.$$

**Convex function:** We say a function $f$ is *convex*, if $domf$ (the domain of $f$) is a convex set, and $\forall x, y \in domf$ and $0 \le \theta \le 1$,

$$f(\theta x + (1 - \theta)y) \le \theta f(x) + (1 - \theta)f(y). \tag{4}$$

The following is a figure that illustrates the convexity (4) of the convex function.

Figure 1: An example of convex function




We say a function $f$ is *proper* if it does not attain the value $-\infty$ and there exists $x$ such that $f(x) < \infty$, meaning that $domf$ is nonempty. Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a *closed proper convex* function, which means that its *epigraph*,

$$epif = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} | f(x) \le t\},$$

is a nonempty closed convex set.

The *effective domain* of $f$ is defined as follows:

$$\mathrm{dom} f = \{x \in \mathbb{R}^n | f(x) < +\infty\},$$

which means $f$ takes on finite values for the set.

**Proximal operator:** Given a function $h : \mathbb{R}^n \to \mathbb{R}$, denote the proximal operator $\mathrm{prox}_h : \mathbb{R}^n \to \mathbb{R}^n$ as follows:

$$\mathrm{prox}_h(z) := \arg\min_{\theta \in \mathbb{R}^n} \{\frac{1}{2}\|z - \theta\|_2^2 + h(\theta)\}. \tag{5}$$

Note: The proximal operator of function $h = \lambda||\theta||_1$ is the soft-thresholding operator: the absolute value $|\theta_i|$ being either $\theta_i - \mathrm{sgn}(\theta_i)\lambda$ or zero (if $|\theta_i| < \lambda$). We use the following function for this operator in the following chapters:

$$[\mathcal{S}_\lambda(\theta)]_i = \mathrm{sgn}(\theta_i)(|\theta_i| - \lambda)_+ \tag{6}$$

where $(x)_+ := \max(x, 0)$.

**Conjugate function:** Define the *conjugate* of a function $f : \mathbb{R}^n \to \mathbb{R}$ by

$$f^*(y) = \sup_{x \in \mathrm{dom} f} (y^T x - f(x)),$$

It is known that $f^*$ is closed proper convex function and $(f^*)^* = f$ when $f$ is closed and convex, and that Moreau's decomposition [56] is available: Let $f : \mathbb{R}^n \to \mathbb{R}$ be closed and convex. For any $x \in \mathbb{R}^n$ and $\gamma > 0$, we have

$$prox_{\gamma f}(x) + \gamma prox_{\gamma^{-1} f^*}(\gamma^{-1} x) = x. \tag{7}$$

**Differentiable:** We say a function $f$ is *differentiable* if $\exists\ g$, such that

$$\lim_{d \to 0} \frac{f(x + d) - f(x) - \langle g, d \rangle}{||d||_2} = 0$$

**Lipschitz-continuous:** A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ has *Lipschitz-continuous gradient* if $\exists\ L > 0$ (Lipschitz constant), such that

$$||\nabla f(x) - \nabla f(y)||_2 \leq L||x - y||_2, \forall x, y \in \mathbb{R}^n. \tag{8}$$

**Firm nonexpansivity [16, Lemma 2.4]:** For a proper closed convex function $f$, $\forall x, y \in dom f$, the following inquality holds:

$$||prox_f(x) - prox_f(y)|| \leq ||x - y||.$$

## 2.2 Sparse estimation

We are interested in the sparse estimation problems, which assume that the estimators/features are sparse. To be more specific, sparse means that many parameters are zero in the model. A sparsity model with few features is easier to explain and cheaper to implement. It has attracted tremendous attention in data mining, statistics, and machine learning areas over the past decades.

One of the most straightforward ways to enforce sparsity is to add the penalty term to the loss function to promote zeros in the solution. Given a data matrix $X \in \mathbb{R}^{n\times p}$, sparse estimation models can be formulated as the following minimization problem,

$$\beta = \arg\min_{\beta} Loss(X, \beta) + \lambda P(\beta). \tag{9}$$

The first term is the loss function to fit the model to the data, and the second term is the penalty term that promotes the sparsity of the model. $\lambda \geq 0$ is the regularization parameter, and the value of $\lambda$ works as a trade-off between fitness and sparsity in the two terms.

### 2.2.1 Lasso-type models

Starting from the linear regression setting, given $n$ samples, $\{(x_i, y_i)\}_{i=1}^n$, with $p-$dimensional feature vector $x_i = (x_{i1}, \ldots, x_{ip})^t$ and associated response variable $y_i \in \mathbb{R}$, the least-squares loss function is formulated as follows:

$$Loss(X, y, \beta) := \min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \right\}. \tag{10}$$

With different types of penalty terms $P(\beta)$ in (9) to constrain (10), we can achieve the various sparsity targets. Least absolute shrinkage and selection operator (Lasso) [82] is a very popular method to select and estimate features simultaneously. It solves the following constrained optimization problem:

$$\begin{aligned} \min_{\beta} \quad & \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \right\} \\ s.t. \quad & \sum_{j=1}^{p} |\beta_j| \leq t. \end{aligned}$$

where $t \geq 0$ is a given value to constrain the absolute value of the estimator $\beta$, and it determines the value $\lambda$ in (9). After centering, standardizing ($\frac{1}{n}\sum_{i=1}^n x_{ij} = 0$, $\frac{1}{n}\sum_{i=1}^n x_{ij}^2 = 1$), and eliminating

the intercept $\beta_0$, we can obtain the following Lasso problem in Lagrangian form:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2n} ||y - X\beta||_2^2 + \lambda ||\beta||_1 \right\}. \tag{11}$$

There are many efficient techniques to solve the above Lasso problem [21, 19, 62, 88, 24]. What is more, Lasso has a variety of applications such as [12, 9, 93]. Although Lasso is a common way of getting sparse solutions, it still has some limitations. Lasso cannot process the highly correlated variables, and it may lead to excessive shrinkage. However, to identify homogeneous subgroups of variables and select correlated variables, fused Lasso [82] was proposed with pairwise differences of features in penalty term. The fused Lasso can be expressed as the following optimization problem:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} ||y - X\beta||_2^2 + \lambda \sum_{i<j} |\beta_i - \beta_j|. \tag{12}$$

In some cases, such as gene expression analyses, it is preferable to have the features within the same group reach the same value. The group Lasso [25] was designed for this purpose. The optimization of group Lasso can be formulated as follows:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} ||y - \sum_{g=1}^{G} X\beta_g||_2^2 + \lambda \sum_{g=1}^{G} ||\beta_g||_1, \tag{13}$$

the above can select the variables $\beta$ in the same groups $g$.

**Solution path:** When $\lambda = 0$, there is no penalty constraint, and Lasso becomes the ordinary least square problem. As the value of $\lambda$ increases, the term works as penalization. The solution path means the figure of solution variation under various $\lambda$ values. Figure 2 exhibits an example of a Lasso solution path.

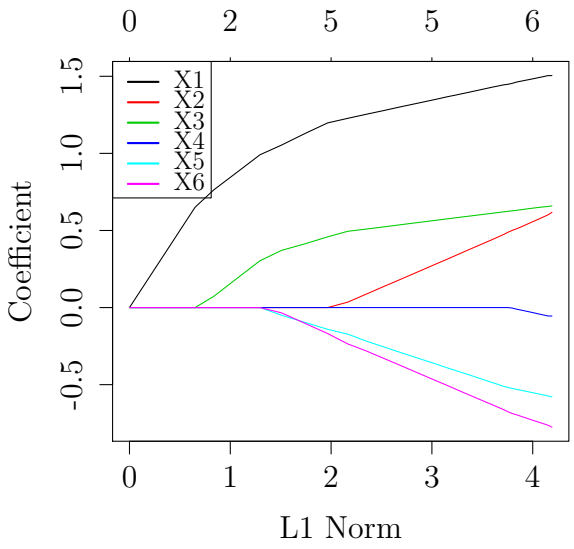



Figure 2: An example of the Lasso solution path [76]. Different colors represent the different features, and the x-axis for the different value of $\lambda$.

Therefore, efficient algorithms are needed to compute the whole solution path or a sequence of $\lambda$ values. There are wide applications of the Lasso penalty $||\beta||_1$ and fused Lasso penalty $\sum_{i<j} |\beta_i - \beta_j|$ for the purpose of sparsity and detecting group variables, respectively. The following are two examples corresponding to the Lasso penalty in graphical Lasso and the fused Lasso penalty in convex clustering.

**Graphical Lasso**. Banerjee et al. [2] proposed a way to estimate the inverse covariance matrix $\Theta$ by solving the following optimization problem:

$$\underset{\Theta}{\text{maximize}} \left\{ log \text{ likelihood} - \lambda ||\Theta||_1 \right\}, \tag{14}$$

**Convex clustering** [39]. Given a data matrix $X \in \mathbb{R}^{N \times p}$, non-negative weight $\omega_{ij} \geq 0$ and $\lambda > 0$, the convex clustering calculate a clustering matrix $U$ as follows:

$$\frac{1}{2}\|X - U\|_F^2 + \lambda \sum_{i<j} \omega_{ij} \|U_{i\cdot} - U_{j\cdot}\|_2, \tag{15}$$

where $X_{i\cdot}$ and $U_{i\cdot}$ denote the $i$-th rows of $X$ and $U$.

Both formulations are convex and have many advantages in optimization. We will illustrate the details of the above two models later, which are two main research contents in this thesis.

## 2.3 Algorithms

The problem (3) can be solved directly if the closed-form expression can be found. However, in many cases, it is hard to obtain the closed-form solution of the problem (3). Hence, we need to iterate the update steps in algorithms from an initial value, generate a sequence of iterates and find a convergent solution. We mainly consider the first-order method in the thesis, which is a method that contains first-order gradient information. The gradient calculation will not cost much for large data sets problems. Moreover, the accuracy of the first-order method is enough for most problems.

### 2.3.1 Gradient descent method

For problem (3), if the objective function $F(x)$ is convex and differentiable, then one popular method to solve the optimization problem is the gradient descent method. The main update step in gradient descent method is $x_{t+1} = x_t - \eta_t \nabla F(x_t)$, where $\eta_t > 0$ is the step size and $-\nabla F(x_t)$ is the descent direction. The search direction and step size are computed at the new point, and the process is repeated until convergence. The whole procedure is summarized in Algorithm 1.

**Algorithm 1** Gradient Method

**Input**: initial point $x_0$.

**While** $t < t_{\max}$ (until convergence) **do**

1: Choose a stepsize $\eta_t > 0$

2: Compute $x_{t+1} = x_t - \eta_t \nabla F(x_t)$

3: $t \leftarrow t + 1$

**end**

**Output**: $x^*$.

There are some strategies to choose the step size $\eta_t$. For example:

- Armijo rule: $\eta_t$ satisfies $F(x_t - \eta_t \nabla F(x_t)) \leq F(x_t) + c\eta_t \nabla F(x_t)$, for $c \in (0, 1)$.

- Constant step size: $\eta_t := \eta > 0$ (specified constant).
- Exact optimal step size: $\eta_t := \arg\min_{\eta_t} F(x_t - \eta_t \nabla F(x_t))$.

When computing the step size $\eta_t$, we need to trade off between the reduction of the objective function $F$ and the time of choosing the step size $\eta_t$. Some strategies may require the evaluation condition to select $\eta_t$, such as the Armijo rule, which is a procedure that requires evaluating the objective function many times. Those evaluations may add considerable computational costs.

### 2.3.2 Nesterov's accelerated gradient method

The following is an accelerated version of gradient descent method, called Nesterov's Accelerated Gradient Method (NAGM) [60]. It has $O(\frac{1}{k^2})$ convergence rate while the (conventional) gradient descent method has $O(\frac{1}{k})$ [60].

**Algorithm 2** NAGM

**Input**: Lipschitz constant $L$, initial value $x^0 = y^0$, $t^1 = 1$.

**While** $k < k_{\max}$ (until convergence) **do**

1: $x^{k+1} = y^k - \frac{1}{L}\nabla F(y^k)$

2: $t^{k+1} = \frac{1+\sqrt{4t^{k^2}+1}}{2}$

3: $y^{k+1} = x^{k+1} + \frac{t^k - 1}{t^{k+1}}(x^{k+1} - x^k)$

4: $k = k + 1$

**End while**

Algorithm 2 replaces the gradient descent $y^{k+1} = y^k - \frac{1}{L}\nabla F(y^k)$ by Steps 1 to 3: Step 1 executes the gradient descent to obtain $x^{k+1}$ from $y^k$, Steps 2 and 3 calculate new $y^{k+1}$ based on the previous $x^k, x^{k+1}$, and then return to the gradient descent in Step 1.

### 2.3.3 Proximal gradient method

The gradient method is not feasible when the function $F$ contains a nondifferentiable term because we cannot calculate $\nabla F$. The proximal gradient method has proved to be a better choice with an efficient convergence rate ($O(1/k)$ convergence rate) than the subgradient method ($O(1/\sqrt{k})$ convergence rate) for the nondifferentiable problem [5]. It has broad applications in handling problems in various domains such as image and signal processing [6].

Here, for illustration, we consider the case when the function $g(Ax)$ is nondifferentiable and the function $f(x)$ is differentiable. The proximal gradient method is a method that "proximal" means the proximal operator of the nondifferentiable term $g(Ax)$, and "gradient" means the gradient of the differentiable term $f(x)$. We describe the detailed procedure as follows: define the quadratic approximation $Q_\eta : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ w.r.t. $f(x)$ and $\eta > 0$ by

$$Q_\eta(x', x) := f(x) + \langle x' - x, \nabla f(x) \rangle + \frac{1}{2\eta} ||x' - x||_F^2, \tag{16}$$

then we can describe the proximal gradient procedures by following,

$$x_{t+1} = \arg\min_x \{Q_{\eta_t}(x, x_t) + g(x)\} \tag{17}$$

$$= \text{prox}_{\eta_t g}(x_t - \eta_t \nabla f(x_t)), \tag{18}$$

given initial value $x_0$, where the value of step size $\eta_t > 0$ may change at each iteration $t = 1, 2, \ldots$, for efficient convergence purpose. The whole procedure is summarized in Algorithm 3.

**Algorithm 3** Proximal Gradient Method

**Input**: initial point $x_0$.

**While** $t < t_{\max}$ (until convergence) **do**

1: Choose a stepsize $\eta_t > 0$

2: Compute $x_{t+1} = \text{prox}_{\eta_t g}(x_t - \eta_t \nabla f(x_t))$

3: $t \leftarrow t + 1$

**end**

**Output**: $x^*$.

We summarized some problems and corresponding solvers solved by gradient descent-based method introduced above in Table 2.

### 2.3.4 Accelerated version of ISTA (FISTA)

Fast iterative shrinkage-thresholding algorithm (FISTA) is an extension of NAGM to ISTA, proposed by Beck and Teboulle [6], which enhances the convergence rate from $O(1/k)$ to $O(1/k^2)$. Algorithm 4 shows the detailed procedure of FISTA.

Table 2: Several problems solved by gradient-based methods, where $\sigma$ denotes the indicator function.

| Problem | Update equation | Name of solver |
|---|---|---|
| $\min_x f(x)$ | $x_{k+1} = x_k - \eta_k \nabla f(x_k)$ | Gradient descent method |
| $\min_{x\in\sigma} f(x)$ | $P_C(x)$ | Projection problem |
| $\min_x \{f(x) + \lambda\|\|x\|\|_1\}$ | $sgn(x)[\|x\| - \lambda]_+$ | Proximal gradient method (ISTA) [5] |
| $\min_x \{f(x) + \lambda\|\|x\|\|_2\}$ | $(1 - \frac{\lambda}{\max\{\|\|x\|\|,\lambda\}})x$ | Proximal gradient of Euclidean norm |

**Algorithm 4** FISTA

**Input**: initial value $y^1 = x^0$, $h^1 = 1$.

**While** $t < t_{\max}$ (until convergence) **do**

1: Choose a stepsize $\eta_t > 0$

2: Compute $x_{t+1} = \text{prox}_{\eta_t g}(y_t - \eta_t \nabla f(y_t))$

3: $h^{t+1} = \frac{1+\sqrt{4h^{t2}+1}}{2}$

4: $y^{t+1} = x^{t+1} + \frac{h^t - 1}{h^{t+1}}(x^{t+1} - x^t)$

5: $t \leftarrow t + 1$

**end**

**Output**: optimal solution $x^*$.

### 2.3.5 ADMM

This subsection shows a more general algorithm than the gradient method, which does not require the differentiability of the objective function to solve convex optimization problems, called alternating direction method of multipliers (ADMM) [7].

Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^p \to \mathbb{R}$ be convex. We consider the following constrained optimization problem of (3):

$$\begin{aligned} &\min_{x,y} f(x) + g(Ax) \\ &\text{subject to } Ax = y, \end{aligned} \tag{19}$$

with variables $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^p$, and matrix $A \in \mathbb{R}^{p\times n}$. Define the augmented Lagrangian function as the following,

$$L_\nu(x, y, u) := f(x) + g(y) + \langle u, Ax - y\rangle + \frac{\nu}{2}||Ax - y||_2^2 \ , \tag{20}$$

where $\nu > 0$ is an augmented Lagrangian parameter, and $u \in \mathbb{R}^p$ is the Lagrangian multipliers.

ADMM is a procedure to find the solution to the problem (19) by iterating

$$\begin{aligned} x^{k+1} &:= \arg\min_x L_\nu(x, y^k, u^k), \\ y^{k+1} &:= \arg\min_y L_\nu(x^{k+1}, y, u^k), \\ u^{k+1} &:= \arg\min_u L_\nu(x^{k+1}, y^{k+1}, u), \end{aligned}$$

given the initial values $y^1$ and $u^1$.

From the procedure, ADMM split the target composite problem (19), and handle the functions $f$ and $g$ w.r.t variables $x$ and $y$ separately.

## 2.4 Convergence rate

This subsection introduces the notion of the convergence rate of the algorithm. If $x^*$ satisfies $\nabla F(x^*) = 0$, then we say $x^*$ is an *optimal* point. The convergence rate provides an upper bound of the complexity of an algorithm to obtain the optimal solution $x^*$ or $x$ close to $x^*$ with some tolerance $\epsilon$. There are several ways to illustrate the convergence rate of an algorithm, here we introduce the two most used criteria.

- $\lim_{k\to\infty} \frac{||x^{k+1}-x^*||}{x^k-x^*} = C$,
- $F(x^k) - F(x^*) \leq \epsilon$,

where $k$ represents the $k-$th iteration in the algorithm. For the first one: If $C = 1$, then the sequence $x^k$ is sublinear convergence to $x^*$; If $0 < C < 1$, then the sequence $x^k$ is linear convergence to $x^*$; If $C = 0$, then the sequence $x^k$ is superlinear convergence to $x^*$. However, many algorithms cannot attain the linear convergence rate. Therefore, we use the second criterion for finer expression to compare the sublinear convergence rate level. For example, if the following inequality holds:

$$F(x^k) - F(x^*) \leq \frac{x^0 - x^*}{Ck},$$

where $C$ is constant, and $x^0$ is an initial value, then we say the rate that $F(x^k)$ converges to $F(x^*)$ is $O(1/k)$, and $O(1/k)$ represents the upper bound of the tolerance $F(x^k) - F(x^*)$, when $k$ goes large, then the tolerance goes small. Both belong to the sublinear convergence rate, while it is evident that the rate $O(1/k^2)$ can be significantly faster than $O(1/k)$.

The convergence rate is quite related to the choice of step size in the above algorithms. We below introduce a lemma, which provides theoretical help for the choice of step size.

**Lemma 1.** *[Descent lemma] Let $f$ be a Lipschitz continuous gradient function, and $L$ is corresponding Lipschitz constant, then $\forall x, y \in \mathrm{dom} f$, the following inequality holds:*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}||x - y||^2.$$

Based on Lemma 1, the following Lemma can be proved for the convergence rate of the proximal gradient method.

**Lemma 2.** *[5] Let $\{x^k\}_{k\geq 0}$ be the sequence generated by Algorithm 3 (proximal gradient descent algorithm) to solve problem (3) with Lipschitz continuous gradient assumption of $f$ (Lipschitz constant: $L$), then*

$$F(x^k) - F(x_*) \leq \frac{L||x_0 - x_*||^2}{2k}.$$

The above lemma exhibits that the $k$-th iteration can bring us a certain tolerance of $F(x^k) - F(x^*)$ that related to the number of $k$, i.e. $O(1/k)$ rate of convergence [7, 20, 4].

The following lemma illustrates the convergence rate of the accelerated version of ISTA, which attains $O(1/k^2)$ rate of convergence.

**Lemma 3.** *[5]Let $\{x^k\}_{k\geq 0}$ be the sequence generated by Algorithm 4 (FISTA) to solve problem (3) with Lipschitz continuous gradient assumption of $f$ (Lipschitz constant: $L$), then*

$$F(x^k) - F(x^*) \leq \frac{2L||x^0 - x^*||^2}{(k+1)^2}.$$

# 3 Efficient proximal gradient algorithms for joint graphical Lasso

## 3.1 Introduction

Graphical models are widely used to describe the relationships among interacting objects [47]. Such models have been extensively used in various domains, such as bioinformatics, text mining, and social networks.

In this thesis, we only consider learning Gaussian graphical models that are expressed by undirected graphs. It represents the relationship among continuous variables that follow a joint Gaussian distribution. In a joint Gaussian distribution with random variables $X$, $Y$ and $Z$, we say $X$ is independent of $Y$ conditioned on $Z$ if $P(X, Y|Z) = P(X|Z)P(Y|Z)$ and denote as $X \perp\!\!\!\perp Y|Z$.

In an undirected graph, $\mathcal{G} = (V, E)$, edge set $E$ represents the conditional dependencies among the variables in vertex set $V$. Let $X_1, \ldots, X_p$ $(p \geq 1)$ be Gaussian variables with covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$, and $\boldsymbol{\Theta} := \boldsymbol{\Sigma}^{-1}$ be, if it exists, the precision matrix. We remove the edges so that the variables $X_i$, $X_j$ are conditionally independent given the other variables if and only if the $(i, j)$-th element $\theta_{i,j}$ in $\boldsymbol{\Theta}$ is 0:

$$\{i, j\} \notin E \iff \theta_{i,j} = 0 \iff X_i \perp\!\!\!\perp X_j | X_{V\setminus\{i,j\}},$$

where each edge is expressed as a set of two elements in $\{1, \ldots, p\}$. In this sense, constructing a Gaussian graphical model is equivalent to estimating a precision matrix.

Suppose that we estimate the undirected graph from data consisting of $n$ samples of $p$ variables and that dimension $p$ is much higher than sample size $n$. However, it is almost impossible to estimate the locations of the nonzero elements in $\boldsymbol{\Theta}$ by obtaining the inverse of sample covariance matrix $\boldsymbol{S} \in \mathbb{R}^{p\times p}$, which is the estimator of $\boldsymbol{\Sigma}$. In fact, if $p > n$, then no inverse $\boldsymbol{S}^{-1}$ exists because the rank of $\boldsymbol{S} \in \mathbb{R}^{p\times p}$ is, at most, $n$.

In order to address this situation, two directions are suggested:

1. Sequentially find the variables on which each variable depends via regression so that the quasi-likelihood is maximized [54].

2. Find the locations in $\boldsymbol{\Theta}$, the values of which are zeros, so that the $\ell_1$ regularized log-likelihood is maximized [97, 23, 2, 70].

We follow the second approach because we assume Gaussian variables, also known as graphical Lasso (GL). The $\ell_1$ regularized log-likelihood is defined by

$$\underset{\boldsymbol{\Theta}}{\text{maximize}} \left\{ \log\det \boldsymbol{\Theta} - \text{trace}(\boldsymbol{S}\boldsymbol{\Theta}) - \lambda ||\boldsymbol{\Theta}||_1 \right\}, \tag{21}$$

where tuning parameter $\lambda$ controls the amount of sparsity, and $||\boldsymbol{\Theta}||_1$ denotes the sum of the absolute value of the off-diagonal elements in $\boldsymbol{\Theta}$. Several optimization techniques [1, 52, 31, 18, 23, 42] have been studied for the optimization problem of (21).

In particular, we consider a generalized version of the abovementioned GL. For example, suppose that the gene regulatory networks of thirty case and seventy control patients are different. One might construct a gene regulatory network separately for each of the two categories. However, estimating each on its own does not provide an advantage if a common structure is shared. Instead, we use one hundred samples to construct two networks simultaneously. Intuitively speaking, using both types of data improves the reliability of the estimation by increasing the sample size for the genes that show similar values between case and control patients, while using only one type of data leads to a more accurate estimate for genes that show significantly different values.

Danaher et al. [17] proposed a joint graphical Lasso (JGL) model by including an additional convex penalty (fused or group Lasso penalty) to the graphical Lasso objective function for $K$ classes. For example, $K$ is equal to two for the case/control patients in the example. JGL includes fused graphical Lasso with fused Lasso penalty and group graphical Lasso with group Lasso penalty. Although there are several approaches to handle the multiple graphical models, such as those of [41], [32], [98], and [35], the JGL is considered the most promising.

Our main goal is to improve efficiency in terms of solving the JGL problem. For the GL, relatively efficient solving procedures exist. If we differentiate the $\ell_1$ regularized log-likelihood (21) by $\boldsymbol{\Theta}$, then we have an equation to solve [23]. Moreover, several improvements have been considered for the GL, such as proximal Newton [42] and proximal gradient [31] procedures. However, for the JGL, even if we derive such an equation, we have no efficient way to handle it.

Instead, the alternating direction method of multipliers (ADMM) [29], which is a procedure for solving convex optimization problems for general purposes, has been the main approach taken [17, 81, 33, 28]. However, ADMM does not scale well concerning feature dimension $p$ and number of classes $K$. It usually takes time to converge to a high-accuracy solution [8].

## 3.2 Preliminaries

### 3.2.1 Graphical Lasso

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$ be $n \geq 1$ observations of dimension $p \geq 1$ that follow the Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^p$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$, where without loss of generality, we assume $\boldsymbol{\mu} = \mathbf{0}$. Let $\boldsymbol{\Theta} := \boldsymbol{\Sigma}^{-1}$, and the empirical covariance matrix $\boldsymbol{S} := \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T$. Given penalty parameter $\lambda > 0$, the graphical Lasso (GL) is the procedure to find the positive definite $\boldsymbol{\Theta} \in \mathbb{R}^{p \times p}$ such that:

$$\underset{\boldsymbol{\Theta}}{\text{minimize}} \left\{ -\log\det \boldsymbol{\Theta} + \text{trace}(\boldsymbol{S}\boldsymbol{\Theta}) + \lambda \|\boldsymbol{\Theta}\|_1 \right\}, \tag{22}$$

where $||\boldsymbol{\Theta}||_1 = \sum_{j \neq k} |\theta_{j,k}|$. If we regard $V := \{1, \ldots, p\}$ as a vertex set, then we can construct an undirected graph with edge set $\{\{j,k\} | \theta_{j,k} \neq 0\}$, where set $\{j,k\}$ denotes an undirected edge that connects the nodes $j, k \in V$.

If we take the subgradient of (22), then we find that the optimal solution $\boldsymbol{\Theta}_*$ satisfies the condition:

$$-\boldsymbol{\Theta}_*^{-1} + \boldsymbol{S} + \lambda \boldsymbol{\Phi} \ni 0, \tag{23}$$

where $\boldsymbol{\Phi} = (\Phi_{j,k})$ is

$$\Phi_{j,k} = \begin{cases} 1, & \theta^*_{j,k} > 0 \\ [-1, 1], & \theta^*_{j,k} = 0 \\ -1, & \theta^*_{j,k} < 0 \end{cases}.$$

### 3.2.2 ISTA for graphical Lasso

Then, we introduce the method for solving the GL problem (22) by the iterative shrinkage-thresholding algorithm (ISTA) proposed by Guillot et al. [31], which is a proximal gradient method usually employed in dealing with nondifferentiable composite optimization problems.

Specifically, the general ISTA solves the following composite optimization problem:

$$\underset{\boldsymbol{x}}{\text{minimize}}\ F(\boldsymbol{x}) := f(\boldsymbol{x}) + g(\boldsymbol{x}), \tag{24}$$

where $f$ and $g$ are convex, with $f$ differentiable and $g$ possibly being nondifferentiable.

For the GL problem (22), we denote $f, g : \mathbb{R}^{p \times p} \to \mathbb{R}$ as

$$f(\boldsymbol{\Theta}) := -\log\det\boldsymbol{\Theta} + \operatorname{trace}(\boldsymbol{S}\boldsymbol{\Theta}),$$

and

$$g(\boldsymbol{\Theta}) := \lambda\|\boldsymbol{\Theta}\|_1.$$

If we define the quadratic approximation $Q_\eta : \mathbb{R}^{p\times p} \times \mathbb{R}^{p\times p} \to \mathbb{R}$ w.r.t. $f(\boldsymbol{\Theta})$ and $\eta > 0$ by

$$Q_\eta(\boldsymbol{\Theta}', \boldsymbol{\Theta}) := f(\boldsymbol{\Theta}) + \langle \boldsymbol{\Theta}' - \boldsymbol{\Theta}, \nabla f(\boldsymbol{\Theta})\rangle + \frac{1}{2\eta}||\boldsymbol{\Theta}' - \boldsymbol{\Theta}||_F^2, \tag{25}$$

then we can describe the ISTA as a procedure that iterates

$$\boldsymbol{\Theta}_{t+1} = \arg\min_{\boldsymbol{\Theta}}\{Q_{\eta_t}(\boldsymbol{\Theta}, \boldsymbol{\Theta}_t) + g(\boldsymbol{\Theta})\} \tag{26}$$

$$= \operatorname{prox}_{\eta_t g}(\boldsymbol{\Theta}_t - \eta_t \nabla f(\boldsymbol{\Theta}_t)), \tag{27}$$

given initial value $\boldsymbol{\Theta}_0$, where the value of step size $\eta_t > 0$ may change at each iteration $t = 1, 2, \ldots,$ for efficient convergence purpose.

It is known that if we choose $\eta_t = \frac{1}{L}$ ($L$ is Lipschitz constant) for each step in the ISTA that minimizes $F(\cdot)$ , then the convergence rate is, at most, as follows by Lemma 2:

$$F(\boldsymbol{\Theta}_t) - F(\boldsymbol{\Theta}_*) = O(\frac{1}{t}). \tag{28}$$

However, for the GL problem (22), we know neither the exact value of the Lipschitz constant $L$ nor any nontrivial upper bound. Guillot et al. [31] implement a backtracking line search option in Step 1 of Algorithm 5 below to handle this issue.

The backtracking line search enables us to compute the $\eta_t$ value for each time $t = 1, 2, \ldots$ by repeatedly multiplying $\eta_t$ by a constant $c \in (0, 1)$ until $\boldsymbol{\Theta}_{t+1} \succ 0$ ($\boldsymbol{\Theta}$ is positive definite) and

$$f(\boldsymbol{\Theta}_{t+1}) \le Q_{\eta_t}(\boldsymbol{\Theta}_{t+1}, \boldsymbol{\Theta}_t), \tag{29}$$

for the $\boldsymbol{\Theta}_{t+1}$ in (27). Additionally, (29) is a sufficient condition for (28), which was derived in [5] (see the relationship between Lemma 2.3 and Theorem 3.1 in [5]).

The whole procedure is given in Algorithm 5.

**Algorithm 5** G-ISTA for problem (22)

**Input**: $\boldsymbol{S}$, tolerance $\epsilon > 0$, backtracking constant $0 < c < 1$, initial value $\eta_0$, $\boldsymbol{\Theta}_0$, $t = 0$.

**While** $t < t_{\max}$ (until convergence) **do**

1: Backtracking line search: Continue to multiply $\eta_t$ by c until

$$\boldsymbol{\Theta}_{t+1} \succ 0 \quad \text{and} \quad f(\boldsymbol{\Theta}_{t+1}) \leq Q_{\eta_t}(\boldsymbol{\Theta}_{t+1}, \boldsymbol{\Theta}_t)$$

for $\boldsymbol{\Theta}_{t+1} := prox_{\eta_t g}(\boldsymbol{\Theta}_t - \eta_t \nabla f(\boldsymbol{\Theta}_t))$.

2: Update iterate: $\boldsymbol{\Theta}_{t+1} \leftarrow prox_{\eta_t g}(\boldsymbol{\Theta}_t - \eta_t \nabla f(\boldsymbol{\Theta}_t))$.

3: Set next initial step size $\eta_{t+1}$ by the Barzilai—Borwein method.

4: $t \leftarrow t + 1$

**end**

**Output**: $\epsilon$-optimal solution to problem (22), $\boldsymbol{\Theta}_* = \boldsymbol{\Theta}_{t+1}$.

### 3.2.3 Composite self-concordant minimization

The notion of the self-concordant function was proposed in [59, 67, 58]. In the following, we say a convex function $f$ is self-concordant with parameter $M \geq 0$ if

$$|f'''(\boldsymbol{x})| \leq M f''(\boldsymbol{x})^{3/2}, \text{ for all } \boldsymbol{x} \in \text{dom } f.$$

where dom $f$ is the domain of $f$.

Tran-Dinh et al. [86] considered a composite version of self-concordant function minimization and provided a way to efficiently calculate the step size for the proximal gradient method for the GL problem without relying on the Lipschitz gradient assumption in (8). They proved that

$$f(\boldsymbol{\Theta}) := -\log\det\boldsymbol{\Theta} + \text{trace}(S\boldsymbol{\Theta})$$

in (22) is self-concordant and considers the following minimization:

$$F^* := \underset{\boldsymbol{x}}{\text{minimize}} \ \{F(\boldsymbol{x}) := f(\boldsymbol{x}) + g(\boldsymbol{x})\},$$

where $f$ is convex, differentiable, and self-concordant, and $g$ is convex and possibly nondifferentiable. As for Algorithm 5, without using the backtracking line search, we can compute direction $\boldsymbol{d}_t$ with initial step size $\eta_t$ as follows:

$$\boldsymbol{d}_t := prox_{\eta_t g}(\boldsymbol{\Theta}_t - \eta_t \nabla f(\boldsymbol{\Theta}_t)) - \boldsymbol{\Theta}_t, \tag{30}$$

where the operator *prox* is defined by (5). Then, we use the modified step size $\alpha_t$ to update $\boldsymbol{\Theta}_{t+1} := \boldsymbol{\Theta}_t + \alpha_t \boldsymbol{d}_t$, which can be determined by the direction $\boldsymbol{d}_t$. After defining two parameters related to the direction: $\beta_t := \eta_t^{-1}||\boldsymbol{d}_t||_F^2$ and $\lambda_t := (\langle \nabla^2 f(\boldsymbol{\Theta}_t)\boldsymbol{d}_t, \boldsymbol{d}_t\rangle)^{1/2}$, the modified step size can be obtained by

$$\alpha_t := \frac{\beta_t}{\lambda_t(\lambda_t + \beta_t)} . \tag{31}$$

By Lemma 12 in [86], if the modified step size $\alpha_t \in (0, 1]$, then it can ensure a decrease in the objective function and guarantee convergence in the proximal gradient scheme. From (31), if $\lambda_t \geq 1$, then the condition $\alpha_t \in (0, 1]$ is satisfied. Therefore, we only need to check the case when $\lambda_t < 1$. If the condition $\alpha_t \in (0, 1]$ does not hold, we can change the value of the initial $\eta_t$ (such as the bisection method) to influence the value of $\boldsymbol{d}_t$ in (30) until the condition is satisfied.

### 3.2.4 Joint graphical Lasso

Let $N \geq 1, p \geq 1, K \geq 2$, and $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^p \times \{1, \ldots, K\}$, where each $\boldsymbol{x}_i$ is a row vector. Let $n_k$ be the number of occurrences in $y_1, \ldots, y_N$ such that $y_i = k$, so that $\sum_{k=1}^K n_k = N$.

For each $k = 1, \ldots, K$, we define the empirical covariance matrix $\boldsymbol{S}^{(k)} \in \mathbb{R}^{p\times p}$ of the data $\boldsymbol{x}_i$ as follows:

$$\boldsymbol{S}^{(k)} := \frac{1}{n_k} \sum_{i:y_i=k} \boldsymbol{x}_i^T \boldsymbol{x}_i.$$

Given the penalty parameters $\lambda_1 > 0$ and $\lambda_2 > 0$, the joint graphical Lasso (JGL) is the procedure to find the positive definite matrix $\boldsymbol{\Theta}^{(k)} \in \mathbb{R}^{p\times p}$ for $k = 1, \ldots, K$, such that

$$\underset{\boldsymbol{\Theta}}{\text{minimize}} \left\{ -\sum_{k=1}^K n_k \{\log\det \boldsymbol{\Theta}^{(k)} - \text{trace}(\boldsymbol{S}^{(k)}\boldsymbol{\Theta}^{(k)})\} + \lambda_1 \sum_{k=1}^K \sum_{i\neq j} |\theta_{k,i,j}| + P(\boldsymbol{\Theta}) \right\}, \tag{32}$$

where $P(\boldsymbol{\Theta})$ penalizes $\boldsymbol{\Theta} := [\boldsymbol{\Theta}^{(1)}, \ldots, \boldsymbol{\Theta}^{(K)}]^T$. For example, Danaher et al. [17] suggested the following fused and group Lasso penalties:

$$P_F(\boldsymbol{\Theta}) := \lambda_2 \sum_{k<l} \sum_{i,j} |\theta_{k,i,j} - \theta_{l,i,j}|$$

and

$$P_G(\boldsymbol{\Theta}) := \lambda_2 \sum_{i\neq j} \left\{ \sum_{k=1}^K \theta_{k,i,j}^2 \right\}^{1/2},$$

where $\theta_{k,i,j}$ is the $(i,j)$-th element of $\mathbf{\Theta}^{(k)} \in \mathbb{R}^{p\times p}$ for $k = 1, \ldots, K$.

## 3.3 Optimization problem and algorithms

We propose two efficient algorithms to solve the JGL problem. One is an extended ISTA based on the G-ISTA, and the other utilized the self-concordant properties.

### 3.3.1 ISTA for JGL

To describe the JGL problem, we define $f, g : \mathbb{R}^{K\times p\times p} \to \mathbb{R}$ by

$$f(\mathbf{\Theta}) := -\sum_{k=1}^{K} n_k \left\{\log\det \mathbf{\Theta}^{(k)} - \mathrm{trace}(\boldsymbol{S}^{(k)}\mathbf{\Theta}^{(k)})\right\}, \tag{33}$$

$$g(\mathbf{\Theta}) := \lambda_1 \sum_{k=1}^{K}\sum_{i\neq j} |\theta_{k,i,j}| + P(\mathbf{\Theta}). \tag{34}$$

Then, the optimization problem (32) reduces to:

$$\underset{\mathbf{\Theta}}{\text{minimize}}\ F(\mathbf{\Theta}) := f(\mathbf{\Theta}) + g(\mathbf{\Theta})\ ,$$

where the function $f$ is convex and differentiable, and $g$ is convex and nondifferentiable. Therefore, ISTA is available for solving the JGL problem (32).

If we define the quadratic approximation $Q_{\eta_t} : \mathbb{R}^{K\times p\times p} \to \mathbb{R}$ of $f(\mathbf{\Theta}_t)$ by

$$Q_{\eta_t}(\mathbf{\Theta}, \mathbf{\Theta}_t) := f(\mathbf{\Theta}_t) + \sum_{k=1}^{K}\left\langle \mathbf{\Theta}^{(k)} - \mathbf{\Theta}_t^{(k)}, \nabla f(\mathbf{\Theta}_t^{(k)})\right\rangle + \frac{1}{2\eta_t}\sum_{k=1}^{K} ||\mathbf{\Theta}^{(k)} - \mathbf{\Theta}_t^{(k)}||_F^2,$$

then the update iteration is simplified as:

$$\begin{aligned}\mathbf{\Theta}_{t+1} &= \underset{\mathbf{\Theta}}{\mathrm{argmin}}\ \{Q_{\eta_t}(\mathbf{\Theta}, \mathbf{\Theta}_t) + g(\mathbf{\Theta})\} \\ &= \mathrm{prox}_{\eta_t g}(\mathbf{\Theta}_t - \eta_t \nabla f(\mathbf{\Theta}_t)).\end{aligned}$$

Nevertheless, the Lipschitz gradient constant of $f(\mathbf{\Theta})$ is unknown over the whole domain in the JGL problem. Therefore, our approach needs a backtracking line search to calculate step size $\eta_t$. We show the details in Algorithm 6.

**Algorithm 6** ISTA for problem (32)

**Input**: $\boldsymbol{S}$, tolerance $\epsilon > 0$, backtracking constant $0 < c < 1$, initial step size $\eta_0$, initial iterate $\boldsymbol{\Theta}_0$.

**For** $t = 0, 1, \cdots,$ (until convergence) **do**

1: Backtracking line search: Continue to multiply $\eta_t$ by c until

$$f(\boldsymbol{\Theta}_{t+1}) \leq Q_{\eta_t}(\boldsymbol{\Theta}_{t+1}, \boldsymbol{\Theta}_t) \text{ and } \boldsymbol{\Theta}_{t+1}^{(k)} \succ 0 \text{ for } k = 1, \cdots, K. \tag{35}$$

for $\boldsymbol{\Theta}_{t+1} := \text{prox}_{\eta_t g}(\boldsymbol{\Theta}_t - \eta_t \nabla f(\boldsymbol{\Theta}_t))$.

2: Update iterate: $\boldsymbol{\Theta}_{t+1} \leftarrow \text{prox}_{\eta_t g}(\boldsymbol{\Theta}_t - \eta_t \nabla f(\boldsymbol{\Theta}_t))$.

3: Set next initial step size $\eta_{t+1}$. See details in Section 3.4.

**end**

**Output**: optimal solution to problem (32), $\boldsymbol{\Theta}_* = \boldsymbol{\Theta}_{t+1}$.

In the update of $\boldsymbol{\Theta}_{t+1}$, we need to compute the proximal operators for fused and group Lasso penalties. In the following, for each of them, the problem can be divided into the fused Lasso problems [83] and group Lasso problems [74, 25] for $\boldsymbol{\theta}_{i,j} \in \mathbb{R}^K$, $i, j = 1, \ldots, p$. We apply the solutions given by (38) and (39) below.

### A. Fused Lasso penalty $P_F$

By the definition of the proximal operator in the update step, we have

$$\begin{aligned}\boldsymbol{\Theta}_{t+1} = \arg\min_{\boldsymbol{\Theta}} \Bigg\{ \frac{1}{2} \sum_{k=1}^{K} ||\boldsymbol{\Theta}^{(k)} - \boldsymbol{\Theta}_t^{(k)} + \eta_t \nabla f(\boldsymbol{\Theta}_t^{(k)})||_F^2 + \eta_t \lambda_1 \sum_{k=1}^{K} \sum_{i \neq j} |\theta_{k,i,j}| \\ + \eta_t \lambda_2 \sum_{k<l} \sum_{i,j} |\theta_{k,i,j} - \theta_{l,i,j}| \Bigg\}.\end{aligned} \tag{36}$$

Problem (36) is separable with respect to the elements $\theta_{k,i,j}$ in $\boldsymbol{\Theta}^{(k)} \in \mathbb{R}^{p \times p}$; hence, the proximal operator can be computed in componentwise manner: Let $\boldsymbol{A} = \boldsymbol{\Theta}_t - \eta_t \nabla f(\boldsymbol{\Theta}_t)$; then, problem (36) reduces to the following for $i = 1, \cdots, p$, $j = 1, \cdots, p$ :

$$\underset{\theta_{1,i,j}, \cdots, \theta_{K,i,j}}{\text{argmin}} \left\{ \frac{1}{2} \sum_{k=1}^{K} (\theta_{k,i,j} - a_{k,i,j})^2 + \eta_t \lambda_1 1_{i \neq j} \sum_{k=1}^{K} |\theta_{k,i,j}| + \eta_t \lambda_2 \sum_{k<l} |\theta_{k,i,j} - \theta_{l,i,j}| \right\}, \tag{37}$$

where $1_{i \neq j}$ is an indicator function, the value of which is 1 only when $i \neq j$.

The problem (37) is known as the fused Lasso problem [83, 40] given $a_{k,i,j}$ for $k = 1, \ldots, K$. In particular, let $\alpha := \eta_t \lambda_1 1_{i \neq j}$ and $\beta := \eta_t \lambda_2$. When $i \neq j$, $\alpha \neq 0$ and $\beta > 0$, the solution to (37) can

be obtained through soft thresholding operator based on the solution when $\alpha = 0$ by the following lemma.

**Lemma 4.** *([22]) Denote the solution to parameters $\alpha$ and $\beta$ as $\theta_i(\alpha, \beta)$, then the solution $\theta_i(\alpha, \beta)$ of the fused Lasso problem*

$$\frac{1}{2}\sum_{i=1}^{n}(y_i - \theta_i)^2 + \alpha\sum_{i=1}^{n}|\theta_i| + \beta\sum_{i=1}^{n-1}|\theta_i - \theta_{i+1}| \tag{38}$$

*is given by $[S_\alpha(\theta(0, \beta))]_i$ when $y_1, \cdots, y_n \in \mathbb{R}$ are given for $n \geq 1$.*

Additionally, rather efficient algorithms are available for solving the fused Lasso problem (38) when $\alpha = 0$ (i.e., $\theta(0, \beta)$ ) such as [40, 85, 43].

### B. Group Lasso penalty $P_G$

By definition, the update of $\mathbf{\Theta}_{t+1}$ for the group Lasso penalty $P_G(\mathbf{\Theta})$ is as follows:

$$\mathbf{\Theta}_{t+1} = \arg\min_{\mathbf{\Theta}} \left\{ \frac{1}{2}\sum_{k=1}^{K} ||\mathbf{\Theta}^{(k)} - \mathbf{\Theta}_t^{(k)} + \eta_t \nabla f(\mathbf{\Theta}_t^{(k)})||_F^2 + \eta_t\lambda_1 \sum_{k=1}^{K}\sum_{i\neq j}|\theta_{k,i,j}| + \eta_t\lambda_2 \sum_{i\neq j}(\sum_{k=1}^{K}\theta_{k,i,j}^2)^{1/2} \right\}$$

Similarly, let $\boldsymbol{A} = \mathbf{\Theta}_t - \eta_t \nabla f(\mathbf{\Theta}_t)$; then, the problem becomes the following for $i = 1, \cdots, p$, $j = 1, \cdots, p$:

$$\underset{\theta_{1,i,j},\cdots,\theta_{K,i,j}}{\operatorname{argmin}} \left\{ \frac{1}{2}\sum_{k=1}^{K}(\theta_{k,i,j} - a_{k,i,j})^2 + \eta_t\lambda_1 1_{i\neq j}\sum_{k=1}^{K}|\theta_{k,i,j}| + \eta_t\lambda_2 1_{i\neq j}(\sum_{k=1}^{K}\theta_{k,i,j}^2)^{1/2} \right\}.$$

We have $\theta_{k,i,j} = a_{k,i,j}$ for $i = j$. And for $i \neq j$, the solution [96, 25, 76] is given by

$$\theta_{k,i,j} = \mathcal{S}_{\eta_t\lambda_1}(a_{k,i,j}) \left( 1 - \frac{\eta_t\lambda_2}{\sqrt{\sum_{k=1}^{K}\mathcal{S}_{\eta_t\lambda_1}(a_{k,i,j})^2}} \right)_+. \tag{39}$$

#### 3.3.2 Modified ISTA for JGL

Thus far, we have seen that $f(\mathbf{\Theta})$ in the JGL problem (32) is not globally Lipschitz gradient continuous. The ISTA may not be efficient enough for the JGL case because it includes the backtracking line search procedure for this case, which needs to evaluate the objective function and the positive definiteness of $\mathbf{\Theta}_{t+1}$ in (35) and is inefficient when the evaluation is expensive.

We modify Algorithm 6 to Algorithm 7 based on the step-size selection strategy, which takes advantage of the properties of the self-concordant function. The self-concordant function does not

**Algorithm 7** Modified ISTA (M-ISTA)

**Input**: $\boldsymbol{S}$, tolerance $\epsilon > 0$, initial step size $\eta_0$, initial iterate $\boldsymbol{\Theta}_0$.

**For** $t = 0, 1, \cdots,$ (until convergence) **do**

1: Initialize $\eta_t$.

2: Compute

$$\boldsymbol{d}_t := \text{prox}_{\eta_t g}(\boldsymbol{\Theta}_t - \eta_t \nabla f(\boldsymbol{\Theta}_t)) - \boldsymbol{\Theta}_t. \tag{40}$$

3: Compute

$$\beta_t := \eta_t^{-1} ||\boldsymbol{d}_t||_F^2$$

and

$$\lambda_t := \sum_{k=1}^{K} \sqrt{n_k} ||(\boldsymbol{\Theta}_t^{(k)})^{-1} \boldsymbol{d}_t^{(k)}||_F.$$

4: Determine the step size $\alpha_t := \frac{\beta_t}{\lambda_t(\lambda_t + \beta_t)}$.

5: If $\alpha_t > 1$, then set $\eta_t := \eta_t / 2$ and go back to Step 2.

6: Update $\boldsymbol{\Theta}_{t+1} := \boldsymbol{\Theta}_t + \alpha_t \boldsymbol{d}_t$.

**end**

**Output**: optimal solution to problem (32), $\boldsymbol{\Theta}_* = \boldsymbol{\Theta}_{t+1}$.

rely on the Lipschitz gradient assumption on the function $f(\boldsymbol{\Theta})$ [86], and we can eliminate the need for the backtracking line search.

**Lemma 5.** *([7]) Self-concordance is preserved by scaling and addition: if $f$ is a self-concordant function and a constant $a \leq 1$, then $af$ is self-concordant. If $f_1, f_2$ are self-concordant, then $f_1 + f_2$ is self-concordant.*

The function $f(\boldsymbol{\Theta})$ is self-concordant by Lemma 5. In Algorithm 7, for the initial step size of $\eta_t$ in each iteration, we use the Barzilai-Borwein method [3]. And we apply the self-concordant step size mechanism, which is employed in Steps 3-5 of Algorithm 7.

There is no backtracking procedure in this algorithm that guarantees the positive definiteness of $\boldsymbol{\Theta}_{t+1}$, as in (35) of Algorithm 6. We next show how to ensure the positive definiteness of $\boldsymbol{\Theta}_{t+1}$

in the iterations of Algorithm 7.

**Lemma 6.** *([57], Theorem 2.1.1) Let $f$ be a self-concordant function, and let $\boldsymbol{x} \in \mathrm{dom}\ f$. Additionally, if*

$$W(\boldsymbol{x}) = \{\boldsymbol{y} | \left(\langle \nabla^2 f(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x}), \boldsymbol{y}-\boldsymbol{x}\rangle\right)^{1/2} \leq 1\},$$

*then $W(\boldsymbol{x}) \subset dom f$.*

In Algorithm 7, because we know $\alpha_t := \frac{\beta_t}{\lambda_t(\lambda_t+\beta_t)} < 1$ with $\beta_t > 0$ and $\lambda_t > 0$ by Steps 3-5. Thus, we have $\alpha_t\lambda_t < 1$:

$$\alpha_t\lambda_t := \alpha_t \left(\langle \nabla^2 f(\boldsymbol{\Theta}_t)\boldsymbol{d}_t, \boldsymbol{d}_t\rangle\right)^{1/2} < 1,$$

which implies,

$$\left(\langle \nabla^2 f(\boldsymbol{\Theta}_t)(\boldsymbol{\Theta}_{t+1}-\boldsymbol{\Theta}_t), \boldsymbol{\Theta}_{t+1}-\boldsymbol{\Theta}_t\rangle\right)^{1/2} < 1.$$

Hence, from Lemma 6, we see that $\boldsymbol{\Theta}_{t+1}$ stays in the domain and maintains positive definiteness.

## 3.4 Theoretical analysis

For multiple Gaussian graphical models, Honorio and Samaras [41] and Hara and Washio [35] provided lower and upper bounds for the optimal solution $\boldsymbol{\Theta}_*$. However, the models they considered are different from the JGL. To the best of our knowledge, no related research has provided the bounds of the optimal solution $\boldsymbol{\Theta}_*$ for the JGL problem (32).

In the following, we show the bounds of the optimal solution $\boldsymbol{\Theta}_*$ for the JGL and the iterates $\boldsymbol{\Theta}_t$ generated by Algorithms 6 and 7, which are applied to both fused and group Lasso-type penalties.

**Proposition 1.** *The optimal solution $\boldsymbol{\Theta}_*$ of the problem (32) satisfies*

$$\max_{1\leq k\leq K} \frac{n_k}{p\lambda_c + n_k||\boldsymbol{S}^{(k)}||_2} \leq ||\boldsymbol{\Theta}_*^{(k)}||_2 \leq \frac{Np}{\lambda_1} + \sum_{k=1}^{K}\sum_{i=1}^{p}(s_{k,i,i})^{-1},$$

*where $\lambda_c := \sqrt{K\lambda_1^2 + 2K\lambda_1\lambda_2 + \lambda_2^2}$, and $s_{k,i,i}$ is the $i$-th diagonal element of $\boldsymbol{S}^{(k)}$.*

For the proof, see Section 5.1. Note that the objective function value $F(\boldsymbol{\Theta})$ is always decreasing with the increase of iteration in both algorithms due to [5, Remark 3.1] and Lemma 12 in [86].

Therefore, the following inequality holds for Algorithm 6 and Algorithm 7:

$$F(\boldsymbol{\Theta}_{t+1}) \leq F(\boldsymbol{\Theta}_t) \qquad \text{for } t = 0, 1, \ldots. \tag{41}$$

Then, based on the condition (41), we provide the explicit bounds of iterates $\{\boldsymbol{\Theta}_t\}_{t=0,1\ldots}$ in Algorithms 6 and 7 for the JGL problem (32).

**Proposition 2.** *Sequence $\{\boldsymbol{\Theta}_t\}_{t=0,1,\cdots}$, generated by Algorithms 6 and 7 can be bounded:*

$$m \leq ||\boldsymbol{\Theta}_t||_2 \leq M,$$

*where* $M := ||\boldsymbol{\Theta}_0||_F + \frac{2Np}{\lambda_1} + 2\sum_{k=1}^{K}\sum_{i=1}^{p} {s_{k,i,i}}^{-1}$, $m := e^{-\frac{C_1}{n_m}} M^{(1-Kp)}$, $n_m = \max_k n_k$, *and constant* $C_1 := F(\boldsymbol{\Theta}_0)$.

For the proof, see Section 5.2.

With the help of Proposition 1, Proposition 2, and the following lemma, we can obtain the range of the step size that ensures the linear convergence rate of Algorithm 6.

**Lemma 7.** *Let $\boldsymbol{\Theta}_t$ be t-th iterate in Algorithm 6. And denote $\lambda_{min}$ and $\lambda_{max}$ as the minimum and maximum eigenvalues of the corresponding matrix, respectively. Define*

$$a_k := \min\{\lambda_{\min}(\boldsymbol{\Theta}_t^{(k)}), \lambda_{\min}(\boldsymbol{\Theta}_*^{(k)})\}, \; b_k := \max\{\lambda_{\max}(\boldsymbol{\Theta}_t^{(k)}), \lambda_{\max}(\boldsymbol{\Theta}_*^{(k)})\}$$

*and* $n_l = \min_{k=1\ldots,K} n_k$, $n_m = \max_{k=1\ldots,K} n_k$, $a_l = \min_{k=1\ldots,K} a^{(k)}$ *and* $b_m = \max_{k=1\ldots,K} b^{(k)}$. *The sequence $\{\boldsymbol{\Theta}_t\}_{t=0,1,\ldots}$ generated by Algorithm 6 satisfy*

$$||\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}_*||_F \leq \gamma_t ||\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_*||_F$$

*with the convergence rate* $\gamma_t := \max\{\frac{\eta_t n_m}{a_l^2} - 1, 1 - \frac{\eta_t n_l}{b_m^2}\}$.

*Proof.* It can be easily extended by Lemma 3 in [31]. □

Lemma 7 implies that to obtain the convergence rate $\gamma_t < 1$, we require

$$0 < \eta_t < \frac{2a_l^2}{n_m}. \tag{42}$$

After using Proposition 1 and Proposition 2, we can obtain the bounds of $a_l$. Further, we can obtain the step size $\eta_t$ that satisfies (42) and guarantee the linear convergence rate ($\gamma_t < 1$). However, the step size is quite conservative in practice. Hence, we consider the Barzilai-Borwein method for implementation and regard the step size $\eta_t$ that satisfies (42) as a safe choice. When the number of backtracking iterations in Step 1 of Algorithm 6 exceeds the given maximum number to fulfill the back tracking condition, we can use the safe step size $\eta_t$ for the subsequent calculations. In Section 3.5.5, we confirm the linear convergence rate of the proposed ISTA by experiment.

## 3.5 Simulation

In this section, we evaluate the performance of the proposed methods on both synthetic and real datasets, and we compare the following algorithms:

- ADMM: the general ADMM method proposed by Danaher et al. [17].
- FMGL: the proximal Newton-type method proposed by Yang et al. [94].
- ISTA: the proposed method in Algorithm 6.
- M-ISTA: the proposed method in Algorithm 7.

We perform all the tests in R Studio on a Macbook Air with 1.6 GHz Intel Core i5 and 8 GB memory. The wall times are recorded as the run times for the four algorithms.

### 3.5.1 Stopping criteria and model selection

In the experiments, we consider two stopping criteria for the algorithms.

1. Relative error stopping criterion:
$$\frac{\sum_{k=1}^{K} ||\mathbf{\Theta}_{t+1}^{(k)} - \mathbf{\Theta}_{t}^{(k)}||_F}{\max\{\sum_{k=1}^{K} ||\mathbf{\Theta}_{t}^{(k)}||_F, 1\}} \leq \epsilon.$$
2. Objective error stopping criterion:
$$F(\mathbf{\Theta}_t) - F(\mathbf{\Theta}_*) \leq \epsilon.$$

$\epsilon$ is a given accuracy tolerance; we terminate the algorithm if the above error is smaller than $\epsilon$ or the maximum number of iterations exceeds $1,000$. We use the objective error for convergence rate analysis and the relative error for the time comparison.

The JGL model is affected by regularized parameters $\lambda_1$ and $\lambda_2$. For selecting the parameters, we use the $D$-fold cross-validation method. First, the dataset is randomly split into $D$ segments of equal size, a single subset (test data), estimated by the other $D-1$ subsets (training data), is evaluated, and the subset is changed for the test to repeat $D$ times so that each subset is used.

Let $\boldsymbol{S}_d^{(k)}$ be the sample covariance matrix of the $d$-th ($d = 1, \ldots, D$) segment for class $k = 1, \ldots, K$. We estimate the inverse covariance matrix by the remaining $D-1$ subsets $\hat{\mathbf{\Theta}}_{\lambda,-d}^{(k)}$ and choose $\lambda_1$ and $\lambda_2$, which minimize the average predictive negative log-likelihood as follows:

$$CV(\lambda_1, \lambda_2) = \sum_{d=1}^{D} \sum_{k=1}^{K} \left\{ n_k \text{trace}(\boldsymbol{S}_d^{(k)} \hat{\mathbf{\Theta}}_{\lambda,-d}^{(k)}) - \text{logdet} \hat{\mathbf{\Theta}}_{\lambda,-d}^{(k)} \right\}$$

### 3.5.2 Synthetic data

The performance of the proposed methods was assessed on synthetic data in terms of the number of iterations, the execution time, the squared error, and the receiver operating characteristic (ROC) curve. We follow the data generation mechanism described in [49] with some modifications for the JGL model. We put the details in Appendix.

### 3.5.3 Time comparison experiments

We vary $p, N, K$ and $\lambda_1$ to compare the execution time of our proposed methods with that of the existing methods. We consider only the fused penalty in our proposed method for a fair comparison in the experiments because the FMGL algorithm applies only to the fused penalty. First, we compare the performance among different algorithms under various dimensions $p$, which are shown in Figure 3.

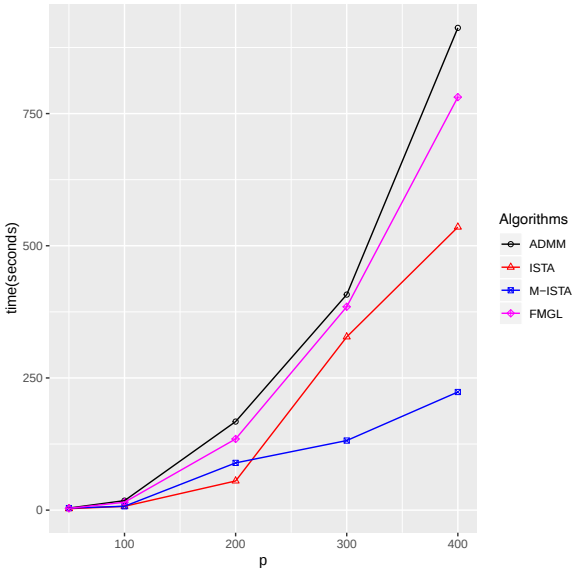



Figure 3: Plot of time comparison under different $p$. Setting $\lambda_1 = 0.1$, $\lambda_2 = 0.05$, $K = 2$ and $N = 200$.

Figure 3 shows that the execution time of the FMGL and ADMM increases rapidly as $p$ increases. In particular, we observe that the M-ISTA significantly outperforms when $p$ exceeds 200. The ISTA shows better performance than the three methods when $p$ is less than 200, but it requires more time as $p$ grows, compared to the M-ISTA. It is reasonable to consider that evaluating

the objective function in the backtracking line search at every iteration increases the computational burden, especially when $p$ increases, which means that the M-ISTA is a good choice for these cases. Furthermore, the ISTA can be a good candidate when the evaluation is inexpensive.

Table 3: Computational time under different settings.

| Parameters setting | | | | | | Computational time | | | |
|---|---|---|---|---|---|---|---|---|---|
| $p$ | $K$ | $N$ | $\lambda_1$ | $\lambda_2$ | precision $\epsilon$ | ADMM | FMGL | ISTA | M-ISTA |
| 20 | 2 | 60 | 0.1 | 0.05 | 0.00001 | 10.506 secs | **1.158 secs** | 2.174 secs | 1.742 secs |
| | 3 | | | | | 1.879 mins | 4.267 secs | **3.357 secs** | 3.668 secs |
| | 5 | | 1 | 0.5 | | 1.123mins | 10.556 secs | 4.216 secs | **2.874 secs** |
| 30 | 2 | 120 | 0.1 | 0.05 | 0.0001 | 10.095 secs | 5.259 secs | **2.690 secs** | 4.857 secs |
| | 3 | | | | | 2.014 mins | 38.562 secs | **14.722 secs** | 31.870 secs |
| | 5 | | 1 | 0.5 | | 2.447 mins | 15.819 secs | 22.431 secs | **12.113 secs** |
| 50 | 2 | 600 | 0.02 | 0.005 | 0.0001 | 6.427 secs | 10.228 secs | 7.213 secs | **4.625 secs** |
| | | | 0.03 | | | 6.240 secs | 8.925 secs | 6.645 secs | **4.023 secs** |
| | | | 0.04 | | | 7.025 secs | 9.381 secs | 6.144 secs | **3.993 secs** |
| 200 | 2 | 400 | 0.09 | 0.05 | 0.0001 | 4.050 mins | 1.874 mins | 2.289 mins | **35.038 secs** |
| | | | 0.1 | | | 4.569 mins | 1.137 mins | 1.340 mins | **24.852 secs** |
| | | | 0.12 | | | 3.848 mins | 1.881 mins | 1.443 mins | **18.367 secs** |

Table 3 summarizes the performance of the four algorithms under different parameter settings to achieve a given precision, $\epsilon$, of the relative error. The results presented in Table 3 reveal that when we increase the number of classes $K$, all the algorithms require more time than usual. Moreover, the execution time of ADMM becomes huge among them. When we vary $\lambda_1$, the algorithms become more efficient as the value increases. For most instances, the M-ISTA and ISTA outperform the existing methods, such as ADMM and FMGL. For the exceptional cases ($p = 20, k = 2, N = 60, \lambda_1 = 0.1$ and $\lambda_2 = 0.05$), the M-ISTA and ISTA are still comparable with the FMGL and faster than ADMM.

### 3.5.4 Algorithm assessment

We generate the simulation data as described in Appendix and regard the synthetic inverse covariance matrices $\mathbf{\Theta}^{(k)}$ as the true values for our assessment experiments.

First, we assessed our proposed method by drawing a ROC curve, which displays the number of true positive edges (i.e., TP edges) selected compared to the number of false positive edges (i.e., FP edges) selected. We say that an edge, $(i, j)$, in the $k$-th class is selected in estimate $\hat{\mathbf{\Theta}}^{(k)}$ if element $\hat{\theta}_{k,i,j} \neq 0$ and the edges are true positive edges selected if the precision matrix element $\theta_{k,i,j} \neq 0$ and false positive edges selected if the precision matrix element $\theta_{k,i,j} = 0$, where the two quantities are defined by

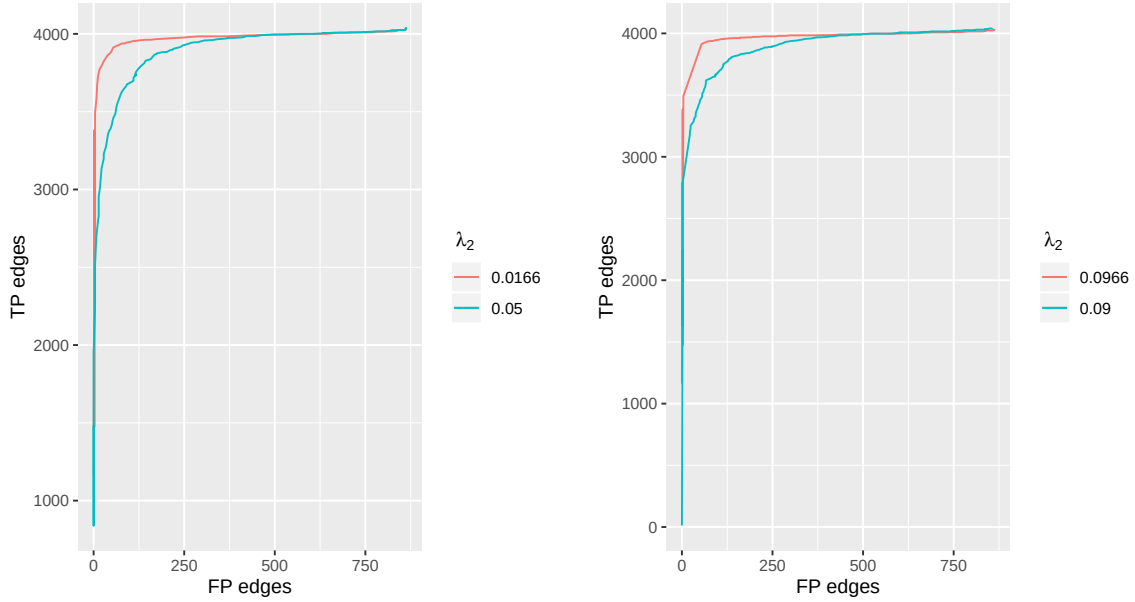
$$TP = \sum_{k=1}^{K} \sum_{i,j} 1(\theta_{k,i,j} \neq 0) \cdot 1(\hat{\theta}_{k,i,j} \neq 0)$$

and

$$FP = \sum_{k=1}^{K} \sum_{i,j} 1(\theta_{k,i,j} = 0) \cdot 1(\hat{\theta}_{k,i,j} \neq 0) \ ,$$

where $1(\cdot)$ is the indicator function.

To confirm the validity of the proposed methods, we compare the ROC figures of the fused penalty and group penalty. We fix the parameters $\lambda_2$ for each curve and change the $\lambda_1$ value to obtain various numbers of selected edges because the sparsity penalty parameter $\lambda_1$ can control the number of selected total edges.




(a) The fused penalty.

(b) The group penalty.

Figure 4: Plot of true positive edges vs. false positive edges selected. Setting $p = 50$, $K = 2$.

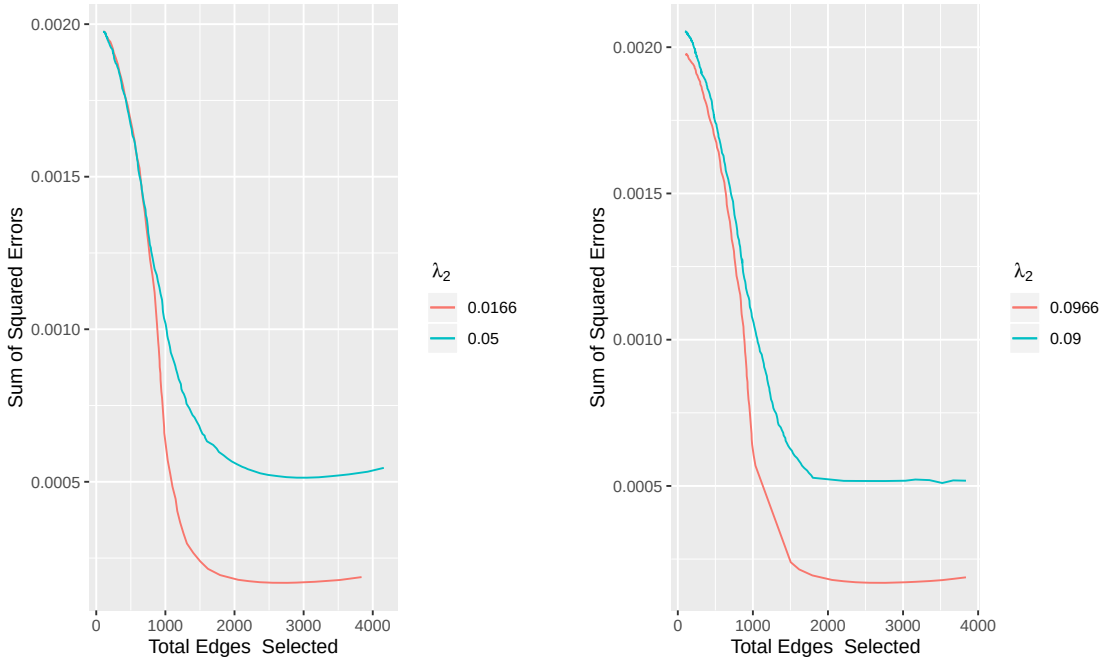
We show the ROC curves for fused and group Lasso penalties in Figures 4a and 4b, respectively.

From the figures, we observe that both penalties show highly accurate predictions for the edge selections. The result of $\lambda_2 = 0.0166$ in the fused penalty case is better than that in $\lambda_2 = 0.05$. Additionally, the result of $\lambda_2 = 0.0966$ in the group penalty case is better than that in $\lambda_2 = 0.09$, which means that if we select the tuning parameters properly, then we can obtain precise results while simultaneously meeting our different model demands.

Then, Figures 5a and 5b display the mean squared error (MSE) between the estimated values and true values.

$$MSE = \frac{2}{Kp(p-1)} \sum_{k=1}^{K} \sum_{i<j} (\hat{\theta}_{k,i,j} - \theta_{k,i,j})^2,$$

where $\hat{\theta}_{k,i,j}$ is the value estimated by the proposed method, and $\theta_{k,i,j}$ is the true precision matrix value we used in the data generation.




(a) The fused penalty.

(b) The group penalty.

Figure 5: Plot of the mean squared errors vs. total edges selected. Setting $p = 50$, $K = 2$.

The figures illustrate that when the total number of edges selected increases, the errors decrease and finally achieve relatively low values.

Overall, the proposed method shows competitive efficiency not only in computational time but also in accuracy.

### 3.5.5 Convergence rate

This section shows the convergence rate of the ISTA for solving the JGL problem (32) in practice, with $\lambda_1 = 0.1$, 0.09 and 0.08. We recorded the number of iterations to achieve the different tolerance of $F(\boldsymbol{\Theta}_t) - F(\boldsymbol{\Theta}_*)$ and ran it on a synthetic dataset, with $p = 200$, $K = 2$, $\lambda_2 = 0.05$ and $N = 400$. The figure reveals that as $\lambda_1$ decreases, more iterations are needed to converge to the specified tolerance. Moreover, the figure shows the linear convergence rate of the proposed ISTA method, which corroborate the theoretical analysis in Section 3.4.

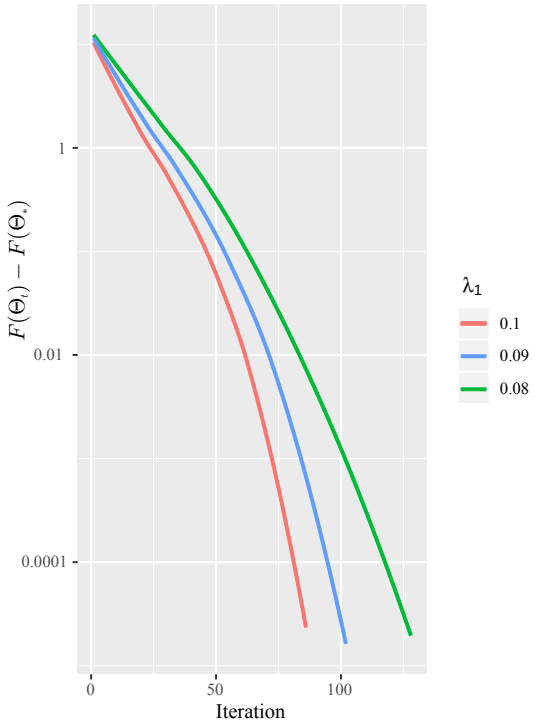



Figure 6: Plot of $\log(F(\boldsymbol{\Theta}_t) - F(\boldsymbol{\Theta}_*))$ vs. the number of iterations with different $\lambda_1$ values. Setting $p = 200, N = 400, K = 2$ and $\lambda_2 = 0.05$.

### 3.5.6 Real data

In this section, we use two different real datasets to demonstrate the performance of our proposed method and visualize the result.

Firstly, we use the presidential speeches dataset in [92] for the experiment to jointly estimate common links across graphs and show the common structure. The dataset contains 75 most used words (features) from several big speeches of the 44 US presidents (samples). And we use the

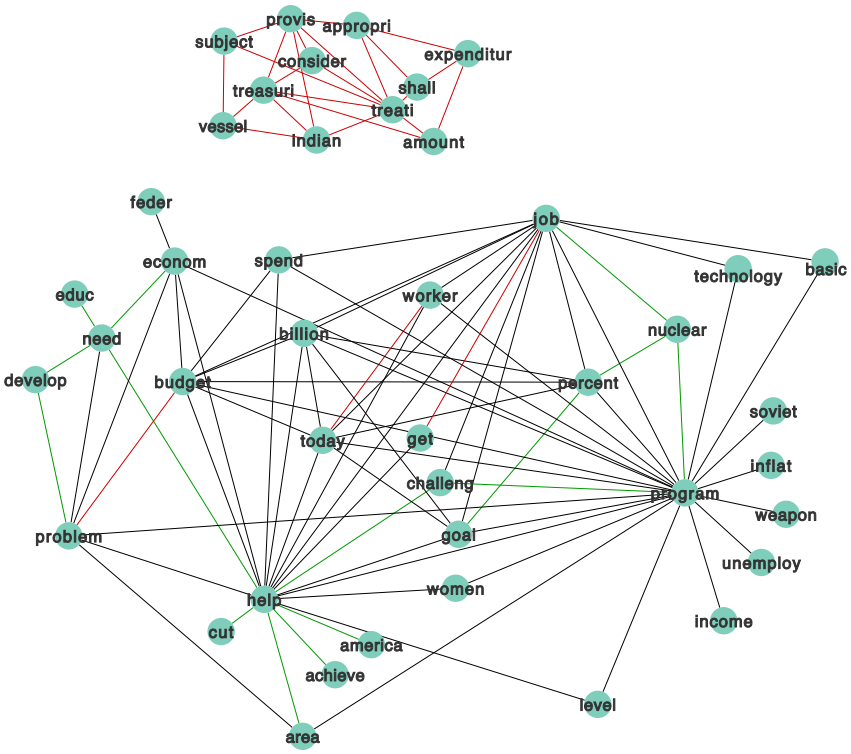



Figure 7: Network figure of the words in president speeches dataset.

clustering result in [92], where the authors split the 44 samples into two groups with similar features, then we obtain two classes of samples ($K = 2$).

We use Cytoscape [72] to visualize the results when $\lambda_1 = 1.9, \lambda_2 = 0.16$. We choose these relatively large tuning parameters for better interpretation of the network figure. Figure 7 shows the relationship network graph of the high-frequency words identified by the JGL model with the proposed method. As shown in the figure, each node represents a word, and the edges demonstrate the relationships between words. We use different colors to show various structures. The black edges are a common structure between the two classes, the red edges are the specific structures for the first class ($k = 1$), and the green edges are for the second class ($k = 2$). Figure 7 shows a sub-network on the top with red edges, meaning there are relationships among those words, and the connections only exist in the first group.

We compare the time cost among four algorithms and show the results in Table 4. We used the cross-validation method ($D = 6$) described in Section 3.5.1 to select the optimal tuning parameters ($\lambda_1 = 0.1$, $\lambda_2 = 0.05$). And we manually chosen the other two pairs of parameters for more

comparisons.

Table 4: Time comparison result of two real datasets.

| Dataset | Parameters setting | | | Computational time | | | |
|---|---|---|---|---|---|---|---|
| | $\lambda_1$ | $\lambda_2$ | precision $\epsilon$ | ADMM | FMGL | ISTA | M-ISTA |
| Speeches | 0.1 | 0.05 | 0.0001 | 19.969 secs | 4.977 mins | **11.829 secs** | 12.867 secs |
| | 0.2 | 0.1 | | 4.661 mins | 3.209 mins | **11.560 secs** | 12.682 secs |
| | 0.5 | 0.25 | | 5.669 mins | 1.490 mins | **11.043 secs** | 12.788 secs |
| Breast cancer | 0.1 | 0.0166 | 0.0001 | 3.809 mins | 7.937 mins | 1.305 mins | **1.158 mins** |
| | 0.2 | 0.02 | | 6.031 mins | 5.198 mins | 1.503 mins | **1.230 mins** |
| | 0.3 | 0.03 | | 5.499 mins | 2.265 mins | 1.188 mins | **1.061 mins** |

The table 4 shows that ISTA outperforms the other three algorithms, and our proposed methods offer stable performance when varying the parameters, while ADMM is the slowest in most cases.

Secondly, the other one is the breast cancer dataset [55] for the time comparison. There are 250 samples and 1,000 genes in the dataset, with 192 control samples and 58 case samples ($K = 2$). Furthermore, we extract 200 genes with the highest variances among the original genes. The tuning parameter pair ($\lambda_1 = 0.01$, $\lambda_2 = 0.0166$) was chosen by the cross-validation method. Table 4 exhibits that our proposed methods (ISTA and M-ISTA) outperform ADMM and FMGL, and M-ISTA shows the best performance in the breast cancer dataset.

# 4 Efficient algorithms for convex biclustering

## 4.1 Introduction

Clustering is a popular unsupervised learning method with applications in a wide variety of fields such as statistics, computer science, and machine learning. By clustering, we usually mean dividing $N$ samples, each consisting of $p$ covariate values, into several categories (called clusters), where $N, p \geq 1$.

The traditional clustering likes hierarchical clustering [44, 38] and k-means clustering [37] may suffer from poor performance due to the non-convexity of the models, which causes difficulty to find the global optimal solutions. To handle this issue, convex clustering was proposed.

### 4.1.1 Convex clustering

The convex clustering [39, 50, 63, 15] was originally developed as a convex relaxation of the hierarchal clustering [39]. Denote the $N$ data observations by $x_1, \cdots, x_N$ in a data matrix $X \in \mathbb{R}^{N \times p}$, where each row denotes a data point, and $p$ is the number of covariates. The convex clustering optimization problem is formulated through clustering the rows as follows. Given a data matrix $X \in \mathbb{R}^{N \times p}$ and $\lambda > 0$, we compute a matrix $U$ of the same size as $X$.

$$\frac{1}{2}\|X - U\|_F^2 + \lambda \sum_{i<j} \omega_{ij} \|U_{i\cdot} - U_{j\cdot}\|_q \tag{43}$$

where $X_{i\cdot}$ and $U_{i\cdot}$ denote the the $i$-th rows of $X$ and $U$, and $q$ can be $1, 2$ or $\infty$.

When $\lambda = 0$, each point belongs to one cluster. As the value of $\lambda$ increases, the cluster begin to fuse, and we assign $X_i$ and $X_j$ to be in the same cluster if $U_i = U_j$. When $\lambda$ is extremely large, every point belongs to one cluster.

The formulation guarantees global optimal solution regardless of initial initializations because of convexity. There are many researches regarding the computational and statistical properties of the convex clustering. Mainly, from the theoretical perspective, Zhu et al. [101] provided the recovery guarantees for the convex clustering model, and Tan and Witten [77] studied several statistical properties of convex clustering. Radchenko and Mukherjee [65] studied the sample behavior of convex clustering with theoretical support. Chi and Steinerberger [14] provided the conditions such that the solution path of convex clustering can recover a tree.

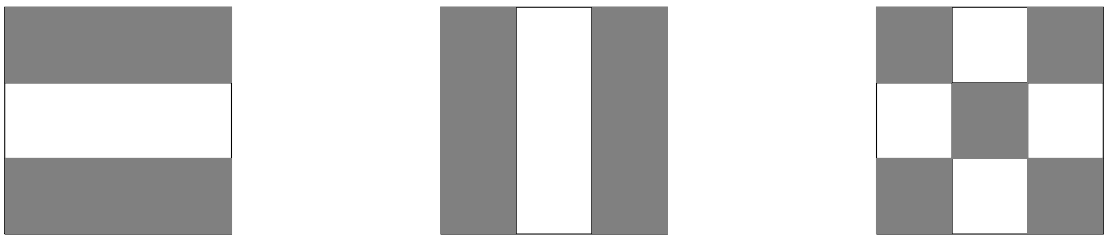
From the computational perspective, Chi and Lange [13] proposed ADMM and AMA to solve the convex clustering problem. Weylandt et al. [92] proposed a fast way to approximate the dendrogram of convex clustering. Sun et al. [75] used an efficient semi-smooth Newton-based augmented Lagrangian method to solve the convex clustering problem. Han and Zhang [34] proposed screening rules for convex clustering in preprocessing to reduce the dimension of problem.

Furthermore, some variants of convex clustering are proposed. Wang et al. [87] proposed sparse convex clustering, which can perform clustering and feature selection simultaneously. The robust convex clustering [90] was proposed to detect the uninformative features. Chi et al. [15] proposed convex biclustering as a biclustering technique with convex formulation.

### 4.1.2 Convex biclustering

In recent years, biclustering [36] has become a ubiquitous data mining technique with varied applications, such as text mining, recommendation system, and bioinformatics. It is an extended notion of clustering, where divide both $\{1, \ldots, N\}$ and $\{1, \ldots, p\}$ based on the data simultaneously. If we are given a data matrix in $\mathbb{R}^{N \times p}$, then the rows and columns within the shared group exhibit similar characteristics. Figure 8 illustrates an intuitive difference between standard clustering and biclustering. A comprehensive survey of biclustering has been given by [80, 64, 51].

However, as noted in [80], biclustering is an NP-hard problem. To avoid the same non-convexity problems in traditional clustering, Chi et al. [15] extended convex clustering to convex bi-clustering.



(a) Row clustering (b) Column clustering (c) Biclustering

Figure 8: While standard clustering divides either rows or columns, the biclustering divides the both.

Given a data matrix $X = (x_{ij})$ consisting of $N$ observations, $i = 1, \ldots, N$, w.r.t. $p$ features $j = 1, \ldots, p$. Our task is to assign each observation to one of the non-overlapped row clusters $C_1, \cdots, C_R \subseteq \{1, \ldots, N\}$ and assign each feature to one of the non-overlapped column clusters

$D_1, \cdots, D_K \subseteq \{1, \ldots, p\}$. We assume that the clusters $C_1, \cdots, C_R$ and $D_1, \cdots, D_K$ and the values of $R$ and $K$ are not known a priori.

More precisely, the convex biclustering in this paper is formulated as follows:

$$\min_{U \in \mathbb{R}^{N \times p}} \frac{1}{2}||X - U||_F^2 + \lambda \left( \sum_{\substack{i,j=1 \\ i<j}}^{N} \omega_{ij} ||U_{i\cdot} - U_{j\cdot}||_2 + \sum_{\substack{m,n=1 \\ m<n}}^{p} \tilde{\omega}_{mn} ||U_{\cdot m} - U_{\cdot n}||_2 \right), \tag{44}$$

where $U_{i\cdot}$ and $U_{\cdot j}$ are the $i$-th row and $j$-th column of $U \in \mathbb{R}^{N \times p}$. Chi et al. [15] suggested a requirement on the weights selection:

$$\omega_{ij} := 1_{i,j}^k \exp\left(-\phi \|x_{i\cdot} - x_{j\cdot}\|_2^2\right)$$

and

$$\tilde{\omega}_{mn} := 1_{m,n}^k \exp\left(-\tilde{\phi} \|x_{\cdot m} - x_{\cdot b}\|_2^2\right),$$

where $1_{i,j}^k$ is 1 if $j$ belongs to the $i$'s $k$-nearest-neighbors and 0 otherwise and $1_{m,n}^k$ is defined similarly (the parameter $k$ should be specified beforehand), and $x_{i\cdot}$ and $x_{\cdot j}$ are the $i$-th row and $j$-th column of the matrix $X$. They suggested that the constants $\phi$ and $\tilde{\phi}$ are determined so that the sums $\sum_{i<j} \omega_{ij}$ and $\sum_{m<n} \tilde{\omega}_{mn}$ are $N^{-1/2}$ and $p^{-1/2}$, respectively.

The convex biclustering achieves checker-board-like biclusters by penalizing both the rows and columns of $U$. When $\lambda$ (the tuning parameter for $\{\|U_{i\cdot} - U_{j\cdot}\|\}_{i \neq j}$ and $\{\|U_{\cdot m} - U_{\cdot n}\|\}_{m \neq n}$) is zero, each $(i, j)$ occupies a unique bicluster $\{(i, j)\}$ and $x_{ij} = u_{ij}$ for $i = 1, \ldots, N$ and $j = 1, \ldots, p$ when $X = (x_{ij})$ and $U = (u_{ij})$. As $\lambda$ increases, the bicluster begins to fuse. For sufficiently large $\lambda$, all $(i, j)$ merge into one single bicluster $\{(i, j) | i = 1, \ldots, N, j = 1, \ldots, p\}$. The convex formulation guarantees a globally optimal solution and demonstrates superior performance to competing approaches. Chi et al. [15] claims that the convex biclustering performs better than the dynamic tree-cutting algorithm [46] and sparse biclustering algorithm [78] in their experiments.

## 4.2 Related work

Chi et al. [15] proposed Convex Bi-ClusteRing Algorithm (COBRA) using a Dykstra-like proximal algorithm to solve the convex biclustering problem. However, it yields subproblems, including the convex clustering problem, which requires expensive computations for large-scale problems due to the high per iteration cost [39, 13]. Essentially, COBRA is a splitting method that separately solves a composite optimization problem containing three terms. Additionally, it is sensitive to tuning

parameter $\lambda$. Therefore, getting the solutions under a wide range of parameters $\lambda$ takes time, which is not feasible for broad applications and different demands for users. Weylandt [91] proposed using ADMM and its variant, Generalized ADMM, to solve the problem. ADMM generally solves the problem by breaking it into smaller pieces and updating the variables alternately. Still, at the same time, it also introduces several subproblems which may cost much time. To be more specific, the ADMM requires solving the Sylvester equation in the step of updating the variable $U$. Hence, the Schur decomposition requires solving the Sylvester equation based on the numerical method, which is complicated and time-consuming. Additionally, it is known that ADMM exhibits $O(1/k)$, where $k$ is the number of iterations, convergence in general [30]. It often takes time to achieve relatively high precision [7], which is not feasible in some highly accurate applications.

## 4.3 Optimization problem and algorithm

We propose NAGM to solve the optimization problem of convex biclustering after converting to differentiable terms. At first, we show that the whole terms in the augmented Lagrangian function of (44) can be differentiable w.r.t. $U$ after introducing two dual variables. Therefore, we use NAGM rather than FISTA in [73], where assumes the objective function contains a nondifferentiable term.

In order to make the notation clear, we formulate the problem (44) in another way. Let $\epsilon_1$ and $\epsilon_2$ be the sets $\{(i,j)|\omega_{ij} > 0, i < j\}$ and $\{(m,n)|\tilde{\omega}_{mn} > 0, m < n\}$, respectively, and denote the cardinality of a set $S$ by $|S|$. We define the matrices $C \in \mathbb{R}^{|\epsilon_1|\times n}$ and $D \in \mathbb{R}^{p\times|\epsilon_2|}$ by

$$C_{l,i} = 1,\ C_{l,j} = -1,\ C_{l,k} = 0,\ k \neq i,j \Longleftrightarrow l = (i,j) \in \epsilon_1,$$

and

$$D_{m,l} = 1,\ D_{n,l} = -1,\ D_{k,l} = 0,\ k \neq m,n \Longleftrightarrow l = (m,n) \in \epsilon_2\ ,$$

respectively.

Then, the optimization problem (44) can be reformulated as follows,

$$\min_{U\in\mathbb{R}^{N\times p}} \frac{1}{2}||X-U||_F^2 + \lambda\left(\sum_{l\in\epsilon_1}\omega_l||C_{l,\cdot}U||_2 + \sum_{l\in\epsilon_2}\tilde{\omega}_l||UD_{\cdot,l}||_2\right). \tag{45}$$

### 4.3.1 The ALM formulation

To implement ALM, we further construct the problem (45) into the following constrained optimization problem by introducing the dual variables $V \in \mathbb{R}^{|\epsilon_1|\times p}$ and $Z \in \mathbb{R}^{N\times|\epsilon_2|}$,

$$\begin{aligned} \min_{U,V,Z} \quad & \frac{1}{2}||X-U||_F^2 + \lambda\left(\sum_{l\in\epsilon_1}\omega_l||V_l||_2 + \sum_{l\in\epsilon_2}\tilde{\omega}_l||Z_l||_2\right) \\ \text{subject to} \quad & C_{l,\cdot}U - V_l = 0, \quad \forall l \in \epsilon_1, \\ & UD_{\cdot,l} - Z_l = 0, \quad \forall l \in \epsilon_2 \ , \end{aligned} \tag{46}$$

where $V_l$ and $Z_l$ are the $l$-th row and $l$-th column of $V$ and $Z$, respectively. If we introduce the following functions,

$$\begin{aligned} f(U) &:= \frac{1}{2}||X-U||_F^2 \\ h(V) &:= \lambda\sum_{l\in\epsilon_1}\omega_l||V_l||_2 \\ g(Z) &:= \lambda\sum_{l\in\epsilon_2}\tilde{\omega}_l||Z_l||_2 \ , \end{aligned}$$

then the problem in (46) becomes

$$\min_{U,V,Z}\left\{f(U)+h(V)+g(Z)\right\}. \tag{47}$$

The augmented Lagrangian function of the problem (46) is given by

$$\begin{aligned} L_\nu(U,V,Z,\Lambda_1,\Lambda_2) :=& f(U)+h(V)+\sum_{l\in\epsilon_1}\langle\Lambda_{1l}, C_{l,\cdot}U-V_l\rangle + \frac{\nu}{2}\sum_{l\in\epsilon_1}||C_{l,\cdot}U-V_l||_2^2 \\ &+ g(Z)+\sum_{l\in\epsilon_2}\langle\Lambda_{2l}, UD_{\cdot,l}-Z_l\rangle + \frac{\nu}{2}\sum_{l\in\epsilon_2}||UD_{\cdot,l}-Z_l||_2^2, \end{aligned} \tag{48}$$

where $\nu > 0$ is an augmented Lagrangian penalty, $\Lambda_1 \in \mathbb{R}^{|\epsilon_1|\times p}$ and $\Lambda_2 \in \mathbb{R}^{N\times|\epsilon_2|}$ are Lagrangian multipliers, and $\Lambda_{1l}$ and $\Lambda_{2l}$ are the $l$-th row and $l$-th column of $\Lambda_1$ and $\Lambda_2$, respectively.

Hence, the ALM procedure of the problem (46) consists of the following three steps:

$$(U^k, V^k, Z^k) = \arg\min_{U,V,Z} L_\nu(U,V,Z,\Lambda_1^{k-1},\Lambda_2^{k-1}), \tag{49}$$

$$\Lambda_{1l}^k = \Lambda_{1l}^{k-1} + \nu(C_{l,\cdot}U^k - V_l^k), \quad \forall l \in \epsilon_1, \tag{50}$$

$$\Lambda_{2l}^k = \Lambda_{2l}^{k-1} + \nu(C_{l,\cdot}U^k - V_l^k), \quad \forall l \in \epsilon_2. \tag{51}$$

### 4.3.2 The proposed method

We construct our proposed method: repeatedly minimizing the augmented Lagrangian function in Equation (49) w.r.t $U, V_l, Z_l$ and updating the Lagrange multipliers in Equations (50) and (51). The whole procedure is summarized in Algorithm 8.

**Step 1: update $U$.** In the $U$-update, if we define the following function in Equation (48),

$$\begin{aligned} F(U) :&= f(U) + \min_{V,Z} \{L_\nu(U, V, Z, \Lambda_1, \Lambda_2)\} \\ &= f(U) + \min_{V,Z} \Bigg\{ h(V) + \sum_{l\in\epsilon_1} \langle \Lambda_{1l}, C_{l,\cdot}U - V_l \rangle + \frac{\nu}{2} \sum_{l\in\epsilon_1} ||C_{l,\cdot}U - V_l||_2^2 \\ &\quad + g(Z) + \sum_{l\in\epsilon_2} \langle \Lambda_{2l}, UD_{\cdot,l} - Z_l \rangle + \frac{\nu}{2} \sum_{l\in\epsilon_2} ||UD_{\cdot,l} - Z_l||_2^2 \Bigg\}, \end{aligned} \tag{52}$$

then the update of $U$ in (49) can be written as

$$U^{k+1} := \arg\min_U F(U^k). \tag{53}$$

We find that (52) is differentiable, and obtain the following proposition.

**Proposition 3.** *The function $F(U)$ is differentiable with respect to $U$, and*

$$\nabla_U F(U) = -X + U + C^T \left(prox_{\nu h^*}(\nu CU + \Lambda_1)\right) + \left(prox_{\nu g^*}(\nu UD + \Lambda_2)\right) D^T. \tag{54}$$

For the proof, please see Section 5.3.

With Proposition 3, we can use NAGM to update $U$ by solving the differentiable optimization problem (53).

**Step 2: update $V_l$ and $\Lambda_1$.** By step (49) in the ALM procedure, we must minimize the functions in Equation (52) corresponding to the vector $V_l$ by updating the following,

$$\begin{aligned} V_l^k &= \arg\min_{V_l} \left\{ \lambda\omega_l ||V_l||_2 + \frac{\nu}{2}||V_l||_2^2 - \sum_{l\in\epsilon_1} \langle \Lambda_{1l}^{k-1} + \nu C_{l,\cdot}U^k, V_l \rangle \right\} \\ &= \arg\min_{V_l} \left\{ \frac{\nu}{2}||V_l - (C_{l,\cdot}U^k + \nu^{-1}\Lambda_{1l}^k)||_2^2 + h_l(V_l) \right\} \\ &= \text{prox}_{h_l/\nu}(C_{l,\cdot}U^k + \nu^{-1}\Lambda_{1l}^{k-1}) \ , \end{aligned}$$

where $h_l(V_l) := \lambda\omega_l||V_l||_2$ denotes the $l$-th term in $h(V)$.

We substitute the optimal $V_l^k$ in the $k$-th iteration into step (50)

$$\Lambda_{1l}^k \leftarrow \Lambda_{1l}^{k-1} + \nu(C_{l,\cdot}U^k - V_l^k), \quad \forall l \in \epsilon_1,$$

to obtain

$$\Lambda_{1l}^k \leftarrow \Lambda_{1l}^{k-1} + \nu C_{l,\cdot} U^k - \nu \text{prox}_{h_l/\nu}(C_{l,\cdot} U^k + \nu^{-1} \Lambda_{1l}^{k-1}). \tag{55}$$

By Moreau's decomposition (7), we further simplify the update (55) as follows,

$$\Lambda_{1l}^k \leftarrow \text{prox}_{\nu h_l^*}(\Lambda_{1l}^{k-1} + \nu C_{l,\cdot} U^k), \tag{56}$$

which means the updates of $V_l$ and $\Lambda_{1l}$ become one update (56). Hence, there is no longer a need to store and compute the variable $V_l$ in the ALM updates which reduces computational costs.

In the update (56), the conjugate function $h_l^*(y)$ of the $\ell_2$ norm is an indicator function [7, Example 3.26]:

$$h_l^*(y) = \begin{cases} 0, & \text{if } ||y||_2 \leq \lambda \omega_l, \\ \infty, & \text{otherwise.} \end{cases} \tag{57}$$

Moreover, the proximal operator of the indicator function (56) is the projection problem [4, Theorem 6.24]:

$$\text{prox}_{\nu h_l^*}(\nu C_{l,\cdot} U^k + \Lambda_{1l}^{k-1}) = P_{\mathcal{B}_l}(\nu C_{l,\cdot} U^k + \Lambda_{1l}^{k-1}) \ , \tag{58}$$

where $\mathcal{B}_l := \{y : ||y||_2 \leq \lambda \omega_l\}$, and the operator $P_{\mathcal{B}_l}$ denotes the projection onto the ball $\mathcal{B}_l$. It solves the problem $P_{\mathcal{B}_l}(x) := \arg\min_{u \in \mathcal{B}_l} ||u - x||_2^2$, i.e.,

$$P_{\mathcal{B}_l}(x) = \begin{cases} x, & \text{if } ||x||_2 \leq \lambda \omega_l, \\ \lambda \omega_l, & \text{otherwise.} \end{cases} \tag{59}$$

This projection problem completes in $O(p)$ operations for a $p$-dimensional vector $x \in \mathbb{R}^p$.

**Step 3: update $Z$ and $\Lambda_2$.** Similarly, we can derive the following equations:

$$\begin{aligned} Z_l^{k+1} &= \arg\min_{Z_l} \left\{ \lambda \tilde{\omega}_l ||Z_l||_2 + \frac{\nu}{2} ||Z_l||_2^2 - \sum_{l \in \epsilon_2} \langle \Lambda_{2l}^{k-1} + \nu U^k D_{\cdot,l}, Z_l \rangle \right\} \\ &= \arg\min_{Z_l} \left\{ \frac{\nu}{2} ||Z_l - (U^k D_{\cdot,l} + \nu^{-1} \Lambda_{2l}^{k-1})||_2^2 + g_l(Z_l) \right\} \\ &= \text{prox}_{g_l/\nu}(U^k D_{\cdot,l} + \nu^{-1} \Lambda_{2l}^{k-1}) \end{aligned}$$

where $g_l(Z_l) := \lambda \tilde{\omega}_l ||Z_l||_2$. Then, the dual variable $\Lambda_2$ update becomes

$$\Lambda_{2l}^k \leftarrow \text{prox}_{\nu g_l^*}(\Lambda_{2l}^{k-1} + \nu U^k D_{\cdot,l}), \quad \text{for } l \in \epsilon_2.$$

If we write in projection operator, then it becomes:

$$\Lambda_{2l}^k \leftarrow P_{\tilde{\mathcal{B}}_l}(\Lambda_{2l}^{k-1} + \nu U^k D_{\cdot,l}) \tag{60}$$

where $\tilde{\mathcal{B}}_l := \{\tilde{y} : ||\tilde{y}||_2 \leq \lambda\tilde{\omega}_l\}$.

**Algorithm 8** Proposed method for convex biclustering.

**Input**: Data $X$, matrices $C$ and $D$, Lipschitz constant $L$ calculated by (61), penalties $\lambda$ and $\nu$, initial value $\Lambda_1^0, \Lambda_2^0$, $Y^1$, $t^1 = 1$.

**While** $k < k_{\max}$ (until convergence) **do**

1: Calculate the gradient $\nabla F(Y^k)$ .

2: Update iterate: $U^k \leftarrow Y^k - \frac{1}{L}\nabla F(Y^k)$.

3: Update iterate: $\Lambda_{1l}^k \leftarrow P_{\mathcal{B}_l}(\Lambda_{1l}^{k-1} + \nu C_{l,\cdot} U^k)$, for $l \in \epsilon_1$, where $\mathcal{B}_l := \{y : ||y||_2 \leq \lambda\omega_l\}$.

4: Update iterate: $\Lambda_{2l}^k \leftarrow P_{\tilde{\mathcal{B}}_l}(\Lambda_{2l}^{k-1} + \nu U^k D_{\cdot,l})$, for $l \in \epsilon_2$, where $\tilde{\mathcal{B}}_l := \{\tilde{y} : ||\tilde{y}||_2 \leq \lambda\tilde{\omega}_l\}$.

5: $t^{k+1} = \frac{1+\sqrt{1+4t^{k\,2}}}{2}$

6: $Y^{k+1} = U^k + \frac{t^k - 1}{t^{k+1}}(U^k - U^{k-1})$

7: $k = k + 1$

**Output**: optimal solution to problem (44), $U^* = U^k$.

Our proposed method only uses first-order information. Furthermore, we just need to calculate the gradient of the function $F$ and proximal operators in each iteration, where the proximal operators are easy to obtain by solving the projection problem.

### 4.3.3 Lipschitz constant and convergence rate

By deriving the Lipschitz constant $L$ of $\nabla_U F(U)$ as in the following proposition, we can obtain the step size in Algorithm 8 for efficient descent.

**Proposition 4.** *The Lipschitz constant of $\nabla_U F(U)$ is upperbounded by*

$$1 + \nu\lambda_{\max}(C^T C) + \nu\lambda_{\max}(D^T D) \ , \tag{61}$$

*where $\lambda_{\max}$ denotes the maximum eigenvalue of the corresponding matrix.*

*Proof.* By definition in (8) and Proposition 3, we derive the Lipschitz constant as follows,

$$\begin{aligned}
||\nabla_U F(U_1) - \nabla_U F(U_2)||_2 = ||U_1 - U_2 + C^T \text{prox}_{\nu h^*}(\nu C U_1 + \Lambda_1) - C^T \text{prox}_{\nu h^*}(\nu C U_2 + \Lambda_1) \\
+ \left(\text{prox}_{\nu g^*}(\nu U_1 D + \Lambda_2)\right) D^T - \left(\text{prox}_{\nu g^*}(\nu U_2 D + \Lambda_2)\right) D^T ||_2 \\
\leq ||U_1 - U_2||_2 + ||C^T \text{prox}_{\nu h^*}(\nu C U_1 + \Lambda_1) - C^T \text{prox}_{\nu h^*}(\nu C U_2 + \Lambda_1)||_2 \\
+ || \left(\text{prox}_{\nu g^*}(\nu U_1 D + \Lambda_2)\right) D^T - \left(\text{prox}_{\nu g^*}(\nu U_2 D + \Lambda_2)\right) D^T ||_2.
\end{aligned}$$

By the definition of matrix 2-norm and the nonexpansiveness of the proximal operators [16, Lemma 2.4], we obtain

$$\begin{aligned}
||\nabla_U F(U_1) - \nabla_U F(U_2)||_2 &\leq ||U_1 - U_2||_2 + \sqrt{\lambda_{\max}(C^T C)} ||\nu C U_1 - \nu C U_2||_2 \\
&\quad + ||\nu U_1 D - \nu U_2 D||_2 \sqrt{\lambda_{\max}(D^T D)} \\
&\leq ||U_1 - U_2||_2 + \nu \lambda_{\max}(C^T C) ||U_1 - U_2||_2 + \nu \lambda_{\max}(D^T D) ||U_1 - U_2||_2 \\
&\leq \left(1 + \nu \lambda_{\max}(C^T C) + \nu \lambda_{\max}(D^T D)\right) ||U_1 - U_2||_2
\end{aligned}$$

□

## 4.4 Simulation

This subsection shows the performances of the proposed approach for estimating and assessing the biclusters by conducting experiments on both synthetic and real datasets. We executed the following algorithms in the experiments:

- COBRA: Dykstra-like proximal algorithm proposed by Chi et al. [15].
- ADMM: the ADMM proposed by Weylandt [91].
- G-ADMM (Generalized ADMM): the modified ADMM presented by Weylandt [91].
- Proposed method: the proposed algorithm showed in Algorithm 8.

They were all implemented by Rcpp on a Macbook Air with 1.6 GHz Intel Core i5 and 8 GB memory. We recorded the wall times for the four algorithms.

### 4.4.1 Artificial data analysis

We evaluate the performance of the proposed methods on synthetic data in terms of the number of iterations, the execution time, and the clustering quality.

We generate the artificial data $X \in \mathbb{R}^{N\times p}$ with a checkerboard bicluster structure similar to the way in [15]. We simulate $X_{ij} \sim N(\mu_{rc}, \sigma^2)$ (i.i.d.) as follows, where the indices $r$ and $c$ range in clusters $\{1, \cdots, R\}$ and $\{1, \cdots, C\}$, respectively, which means that the number of biclusters is $M := R \times C$. We assign each $x_{ij}$ randomly belongs to one of those $M$ biclusters. The mean $\mu_{rc}$ is chosen uniformly from an equally spaced sequence $\{-10, -9, \cdots, 9, 10\}$, and the $\sigma$ is chosen as 1.5 and 3.0 for different noise levels.

In our experiments, we consider the following stopping criteria for the four algorithms.

1. Relative error:
$$\frac{||U^{k+1} - U^k||_F}{\max\{||U^k||_F, 1\}} \leq \epsilon.$$
2. Objective function error:
$$||F(U^k) - F(U^*)||_F \leq \epsilon.$$

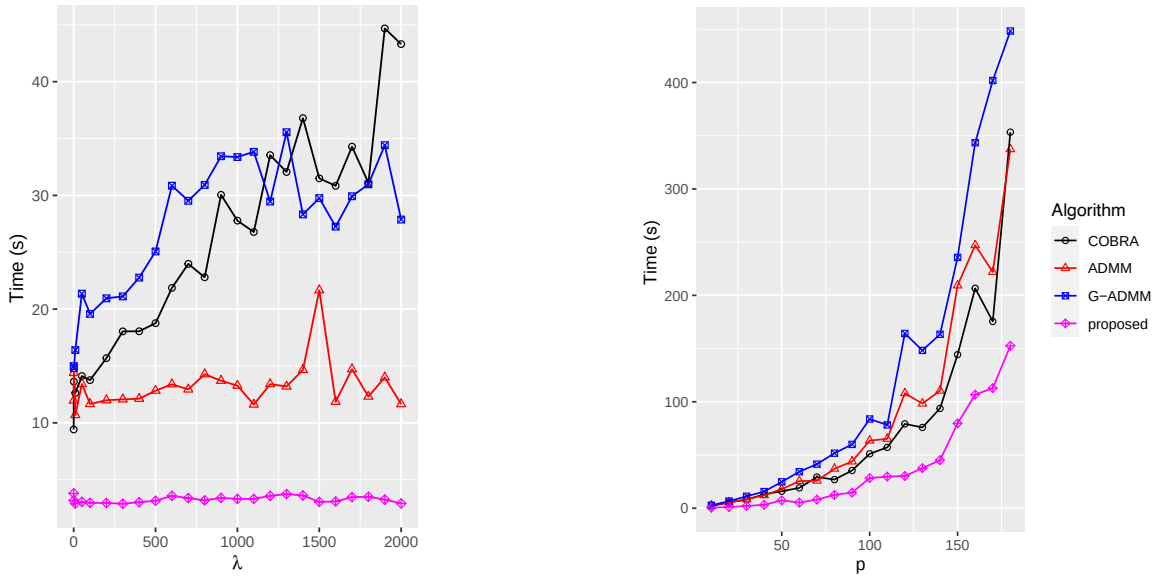
$\epsilon$ is a given accuracy tolerance, we terminate the algorithm if the above error is smaller than $\epsilon$ or the maximum number of iterations exceeds 10,000. We use the relative error for the time comparisons and quality assessment and the objective function error for convergence rate analysis.

### 4.4.2 Comparisons

We changed the sizes of $N, p$ of the data matrix $X$ and the tuning parameter $\lambda$ to test the performance of four algorithms and compared the performance among the algorithms. At first, we compare the execution time with different $\lambda$, ranging from 1 to $2,000$, and setting $R = 4$, $C = 4$, $\sigma = 1.5$. We obtained the results as in Figure 9a.

From Figure 9a, we observe that the execution time of the COBRA and G-ADMM increases rapidly as $\lambda$ varies. The execution time of COBRA is the largest when $\lambda > 1400$. Therefore, it will take a long time for COBRA to visualize the whole fusion process, particularly the single bicluster case. Our proposed method significantly outperforms the other three algorithms and offers high stability in a wide range of $\lambda$. Due to its low computational time, our proposed method will be a preferable choice to visualize the biclusters for various $\lambda$ values when applying biclustering.

Next, we compare the execution time with different $p$ (from 1 to 200). Here, we fix the number of column clusters and row clusters ($C = R = 4$). Figure 3 shows that the execution time of the algorithms increases as $p$ grows. The proposed shows better performance than the other




(a) Different $\lambda$, with $p = 40$.

(b) Different $p$, with $\lambda = 1$.

Figure 9: Execution time for various $\lambda$ and $p$ with $N = 100$ and $\epsilon = 1e^{-6}$.

three. In particular, the ADMM and G-ADMM are suffered from the feature dimension $p$ and the computations grow dramatically.

Then, we varied the sample size $N$ from 100 to $1,000$ and fixed the size of the feature $p = 40$, with $\lambda = 1$, $R = 4$, $C = 4$, $\sigma = 1.5$. Figure 10 shows the execution times of the three algorithms (COBRA, G-ADMM, and Proposed) for each $N$.

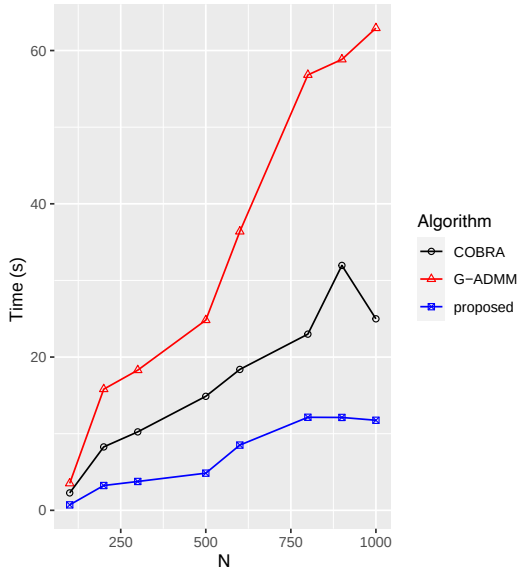



Figure 10: Execution times for each $N$ with $\lambda = 1, p = 40$ and $\epsilon = 1e^{-6}$.

The curves reveal that when the larger the sample size $N$, the more time the algorithms require. Moreover, the ADMM takes more than 500 seconds when $N > 500$ and takes around $1,800$ seconds when $N = 1,000$, which are much larger than the other three algorithms. Thus, we removed the result of ADMM from the figure. However, our proposed method only takes around 10 seconds even when $N = 1,000$, which is six times smaller than G-ADMM.

We evaluate the performance of the proposed methods on synthetic data in terms of the number of iterations, the execution time, and the clustering quality.

We generate the artificial data $X \in \mathbb{R}^{N \times p}$ with a checkerboard bicluster structure similar to the way in [15]. We simulate $X_{ij} \sim N(\mu_{rc}, \sigma^2)$ (i.i.d.) as follows, where the indices $r$ and $c$ range in clusters $\{1, \cdots, R\}$ and $\{1, \cdots, C\}$, respectively, which means that the number of biclusters is $M := R \times C$. We assign each $x_{ij}$ randomly belongs to one of those $M$ biclusters. The mean $\mu_{rc}$ is chosen uniformly from an equally spaced sequence $\{-10, -9, \cdots, 9, 10\}$, and the $\sigma$ is chosen as 1.5 and 3.0 for different noise levels.

We evaluate the clustering quality by a widely used criterion called Rand Index (RI) [66]. The value of RI ranges from 0 to 1, a higher value shows better performance, and 1 indicates the perfect quality of the clustering. Note that we can obtain the true bicluster labels in the data generation procedure. We generate the matrix data with $N = 100$ and $p = 100$, and set two noise levels, low ($\sigma = 1.5$) and high ($\sigma = 3.0$). We compare the clustering quality of our proposed method with

ADMM, G-ADMM, and COBRA under different settings. Setting 1: $R = 2$, $C = 4$, $\sigma = 1.5$; Setting 2: $R = 4$, $C = 4$, $\sigma = 1.5$; Setting 3: $R = 4$, $C = 8$, $\sigma = 1.5$; Setting 4: $R = 2$, $C = 4$, $\sigma = 3$; Setting 5: $R = 4$, $C = 4$, $\sigma = 3$; Setting 6: $R = 4$, $C = 8$, $\sigma = 3$.

Table 5 presents the result of the experiment. As the tuning parameter $\lambda$ increases, the biclusters tend to fuse and reduce noise interference in the raw data. While in some cases, for extremely high $\lambda$ like 10,000, the biclusters may be over-smoothed, and the value of the Rand Index decreased. For example, in the first case (Setting 1: the number of biclusters is $2 \times 4$ and $\sigma = 1.5$). The Rand Index in COBRA, ADMM, and our proposed method shows similar value in most cases because all the algorithms solve the same model. However, the G-ADMM exhibits the worst performance due to its slow convergence rate, and it cannot converge well when the tuning parameter $\lambda$ is large ($\lambda = 5,000$ and $\lambda = 10,000$). Overall, from the results in Table 5, our proposed method shows high accuracy and stability from low to high noise.

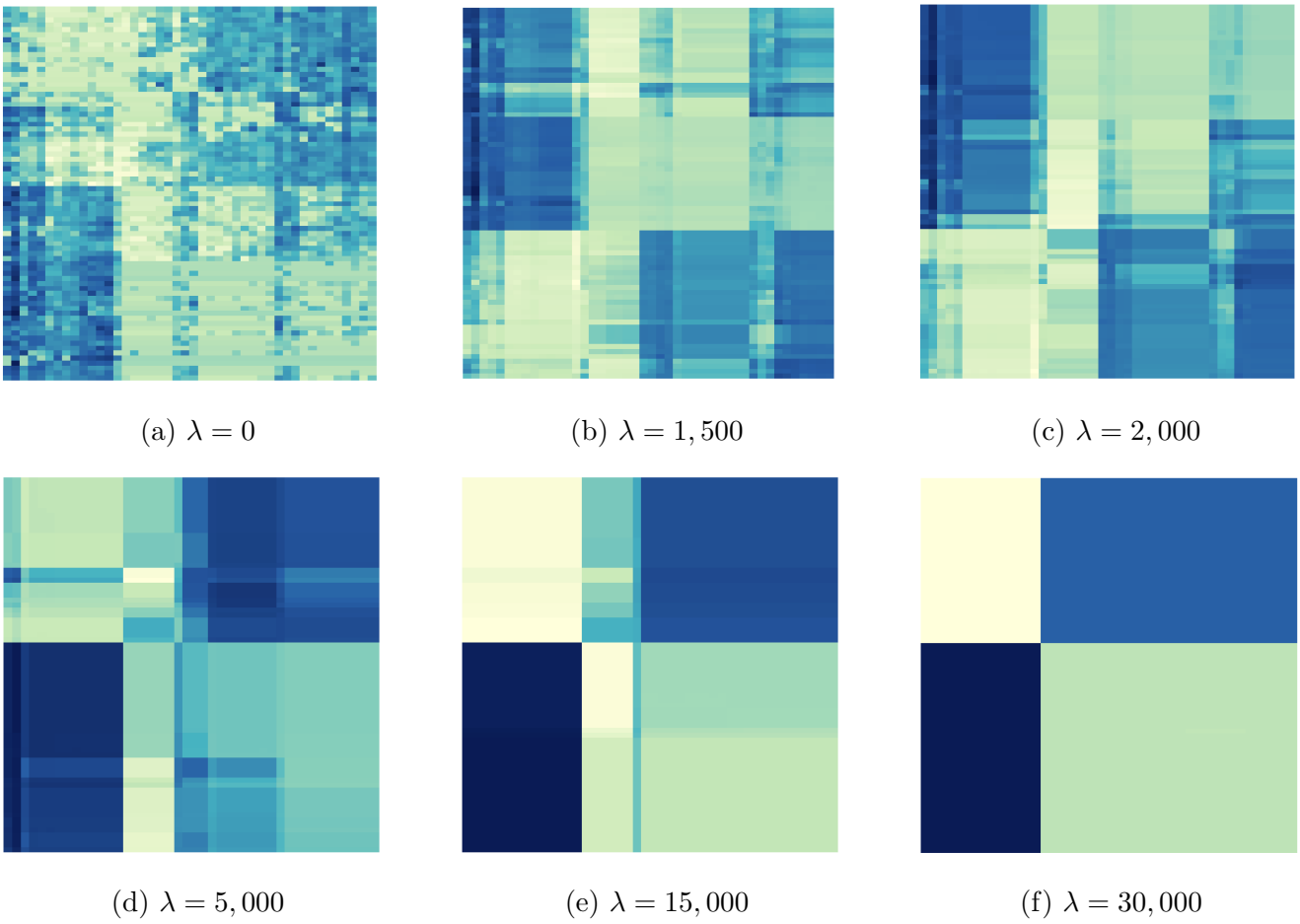
### 4.4.3 Real data analysis

In this section, we use three different real datasets to demonstrate the performance of our proposed method.

Firstly, we use the presidential speeches dataset preprocessed by Weylandt et al. [92] that contains 75 high-frequency words taken from the significant speeches of the 44 US presidents around the year 2018. We show the heatmaps in Figure 11 under a wide range of tuning parameters $\lambda$ to exhibit the fusion process of biclusters. We set the tolerance $\epsilon$ to be $1e^{-6}$, and use the relative error stopping criterion as described in Section 4.1. The columns represent the different presidents, and the rows represent the different words. When $\lambda = 0$, the heatmap is disordered, and there are no distinct subgroups. While we increase the $\lambda$, the biclusters begin to merge. We can further find out the common vocabulary used in some groups of the prime minister's speeches. Moreover, as shown in Figure 11f, the heatmap clearly shows four biclusters with two subgroups of presidents and two subgroups of words when $\lambda = 30,000$. It clusters the subgroups of presidents by different concerns of words in different periods. The result is the same as in Weylandt et al. [92], where the two subgroups of presidents represent two historical periods (modern and pre-modern presidents).

Table 5: Assessment result.

| Setting | Algorithm | Rand Index | | | |
|---|---|---|---|---|---|
| | | $\lambda = 100$ | $\lambda = 1,000$ | $\lambda = 5,000$ | $\lambda = 10,000$ |
| Setting 1 | COBRA | 0.874 | 0.875 | 0.999 | 0.931 |
| | ADMM | 0.872 | 0.875 | 0.999 | 0.931 |
| | G-ADMM | 0.874 | 0.875 | 0.874 | 0.872 |
| | Proposed | 0.875 | 0.875 | 0.999 | 0.931 |
| Setting 2 | COBRA | 0.928 | 0.932 | 0.994 | 0.999 |
| | ADMM | 0.928 | 0.935 | 0.994 | 0.999 |
| | G-ADMM | 0.928 | 0.934 | 0.981 | 0.936 |
| | Proposed | 0.928 | 0.934 | 0.994 | 0.999 |
| Setting 3 | COBRA | 0.959 | 0.962 | 0.962 | 0.999 |
| | ADMM | 0.961 | 0.962 | 0.962 | 0.998 |
| | G-ADMM | 0.959 | 0.962 | 0.967 | 0.967 |
| | Proposed | 0.961 | 0.962 | 0.962 | 0.999 |
| Setting 4 | COBRA | 0.870 | 0.870 | 0.870 | 0.935 |
| | ADMM | 0.870 | 0.868 | 0.871 | 0.933 |
| | G-ADMM | 0.870 | 0.870 | 0.871 | 0.871 |
| | Proposed | 0.870 | 0.870 | 0.871 | 0.935 |
| Setting 5 | COBRA | 0.934 | 0.934 | 0.934 | 0.964 |
| | ADMM | 0.934 | 0.932 | 0.934 | 0.964 |
| | G-ADMM | 0.934 | 0.934 | 0.932 | 0.932 |
| | Proposed | 0.934 | 0.934 | 0.934 | 0.964 |
| Setting 6 | COBRA | 0.960 | 0.960 | 0.962 | 0.962 |
| | ADMM | 0.961 | 0.962 | 0.962 | 0.962 |
| | G-ADMM | 0.960 | 0.962 | 0.962 | 0.960 |
| | Proposed | 0.961 | 0.962 | 0.962 | 0.962 |



(a) $\lambda = 0$ (b) $\lambda = 1,500$ (c) $\lambda = 2,000$

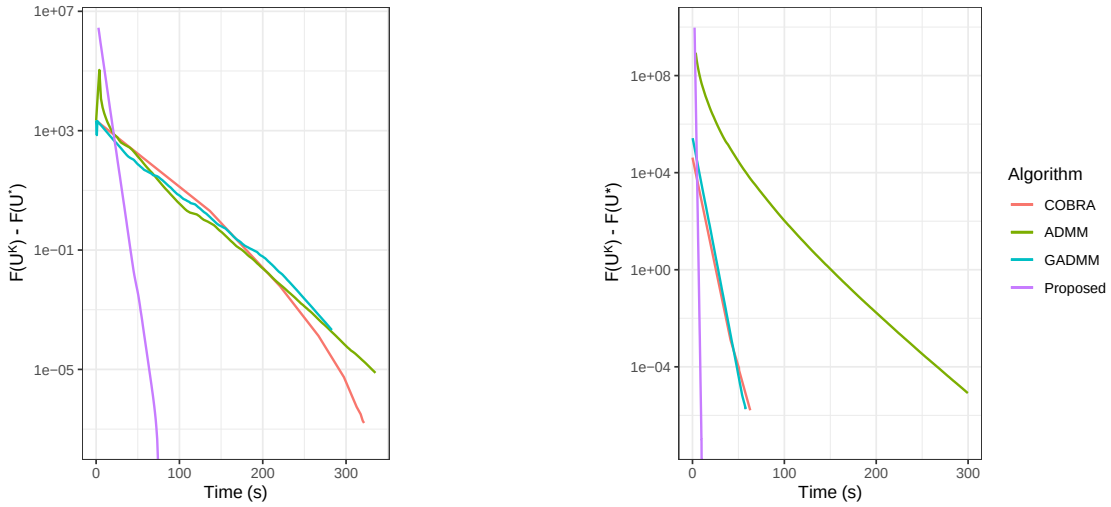
(d) $\lambda = 5,000$ (e) $\lambda = 15,000$ (f) $\lambda = 30,000$

Figure 11: The heatmap results of proposed method implementation on the presidential speeches dataset under a wide range of $\lambda$.

Secondly, we compare the computational time of four algorithms for two actual datasets. One is The Cancer Genome Atlas (TCGA) dataset [45], which contains 438 breast cancer patients (samples) and 353 genes (features), and the other one is the diffuse large-B-cell lymphoma (DLBCL) dataset [69] with 3,795 genes and 58 patient samples. In DLBCL, there are 32 samples from cured patients and 26 samples from sick individuals among the 58 samples. Furthermore, we extract 500 genes with the highest variances among the original genes.

Figures 12a and 12b depict the outcomes of the elapsed time comparison. From the curves, we observe that our proposed approach surpasses the other three methods. In contrast, ADMM shows the worst performance in the DLBCL dataset, and the case of tolerance $\epsilon < 10^{-3}$ requirement in the TCGA dataset.

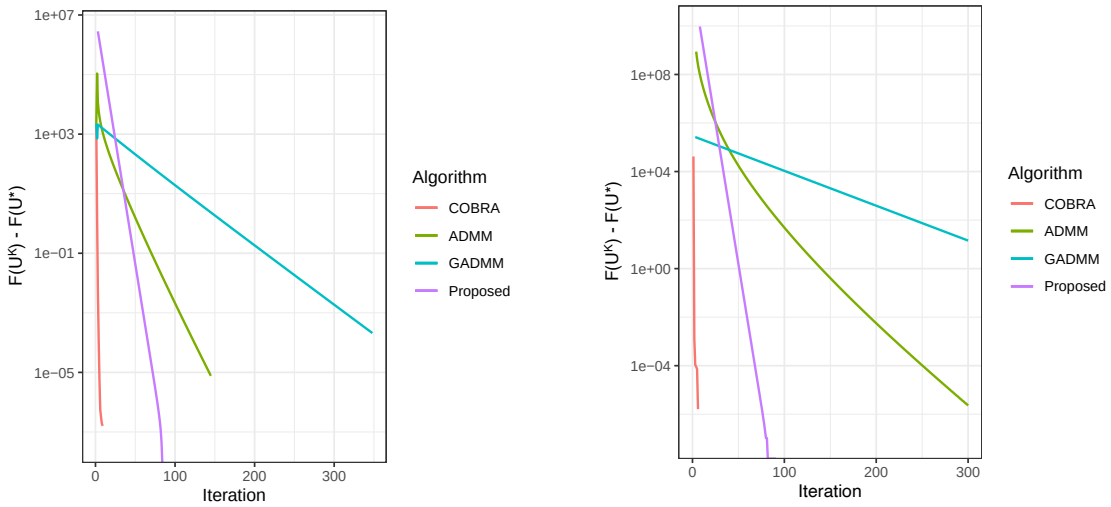
Lastly, we compare the number of iterations to achieve the specified tolerance of $F(U^k) - F(U^*)$ and run it on the TCGA and DLBCL datasets. Figure 13a and Figure 13b reveal that the COBRA




(a) TCGA (b) DLBCL

Figure 12: Plot of $\log(F(U^k) - F(U^*))$ vs. the elapsed time.

algorithm has the fastest convergence rate, whereas the Generalized ADMM is the slowest to converge. Our proposed method shows competitive performance in the convergence rate.




(a) TCGA (b) DLBCL

Figure 13: Plot of $\log(F(U^k) - F(U^*))$ vs. the number of iterations.

Overall, from the above experiment results of the artificial and real datasets, our proposed method has superior computational performance with high accuracy.

# 5 Proof

This section provides the proof of Proposition 1, Proposition 2 , and Proposition 3.

## 5.1 Proof of Proposition 1

We first introduce the Lagrange dual problem of (32). By introducing the auxiliary variables $\boldsymbol{Z} = \{\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(K)}\}$, we can rewrite the problem as follows:

$$\min_{\boldsymbol{\Theta}} f(\boldsymbol{\Theta}) + g(\boldsymbol{\Theta})$$
$$\text{subject to } \boldsymbol{Z} = \boldsymbol{\Theta}$$

Then, the Lagrange function of the above is given by:

$$L(\boldsymbol{\Theta}, \boldsymbol{Z}, \boldsymbol{\Lambda}) = f(\boldsymbol{\Theta}) + g(\boldsymbol{Z}) + \sum_{k=1}^{K} \langle \boldsymbol{\Lambda}^{(k)}, \boldsymbol{\Theta}^{(k)} - \boldsymbol{Z}^{(k)} \rangle,$$

where $\boldsymbol{\Lambda} = \{\boldsymbol{\Lambda}^{(1)}, \ldots, \boldsymbol{\Lambda}^{(K)}\}, \boldsymbol{\Lambda}^{(k)} \in \mathbb{R}^{p \times p}$ are dual variables. To get the dual problem, we minimize the primal variables as follows:

$$\begin{aligned}
\min_{\boldsymbol{\Lambda}, \boldsymbol{Z}} L(\boldsymbol{\Theta}, \boldsymbol{Z}, \boldsymbol{\Lambda}) &= \min_{\boldsymbol{\Theta}} \{ f(\boldsymbol{\Theta}) + \sum_{k=1}^{K} \langle \boldsymbol{\Lambda}^{(k)}, \boldsymbol{\Theta}^{(k)} \rangle \} - \max_{\boldsymbol{Z}} \{ -g(\boldsymbol{Z}) - \sum_{k=1}^{K} \langle \boldsymbol{\Lambda}^{(k)}, -\boldsymbol{Z}^{(k)} \rangle \} \\
&= \min_{\boldsymbol{\Theta}} \{ f(\boldsymbol{\Theta}) + \sum_{k=1}^{K} \langle \boldsymbol{\Lambda}^{(k)}, \boldsymbol{\Theta}^{(k)} \rangle \} - g^*(\boldsymbol{\Lambda}) \\
&= \min_{\boldsymbol{\Theta}} \left\{ \sum_{k=1}^{K} \langle \boldsymbol{\Lambda}^{(k)} + n_k \boldsymbol{S}^{(k)}, \boldsymbol{\Theta}^{(k)} \rangle - \sum_{k=1}^{K} n_k \text{logdet} \boldsymbol{\Theta}^{(k)} \right\} - g^*(\boldsymbol{\Lambda}).
\end{aligned}$$

Taking derivative of the function:

$$L_1 := \sum_{k=1}^{K} \langle \boldsymbol{\Lambda}^{(k)} + n_k \boldsymbol{S}^{(k)}, \boldsymbol{\Theta}^{(k)} \rangle - \sum_{k=1}^{K} n_k \text{logdet} \boldsymbol{\Theta}^{(k)},$$
$$\nabla_{\boldsymbol{\Theta}^{(k)}} L_1 = 0,$$

we get

$$n_k \boldsymbol{S}^{(k)} + \boldsymbol{\Lambda}^{(k)} = n_k (\boldsymbol{\Theta}^{(k)})^{-1} \tag{62}$$

for $k = 1, \cdots, K$. Then the dual problem becomes:

$$\min_{\boldsymbol{\Theta}, \boldsymbol{Z}} L(\boldsymbol{\Theta}, \boldsymbol{Z}, \boldsymbol{\Lambda}) = \sum_{k=1}^{K} n_k p + \sum_{k=1}^{K} n_k \text{logdet}(\boldsymbol{S}^{(k)} + \frac{1}{n_k} \boldsymbol{\Lambda}^{(k)}) - g^*(\boldsymbol{\Lambda}).$$

Hence, we can get the duality gap as follows:

$$f(\boldsymbol{\Theta}) + g(\boldsymbol{Z}) - \sum_{k=1}^{K} n_k p - \sum_{k=1}^{K} n_k \{\text{logdet}\boldsymbol{\Theta}^{(k)}\} + g^*(\boldsymbol{\Lambda})$$
$$= \sum_{k=1}^{K} n_k \text{trace}(\boldsymbol{S}^{(k)}\boldsymbol{\Theta}^{(k)}) + g(\boldsymbol{Z}) - \sum_{k=1}^{K} n_k p + g^*(\boldsymbol{\Lambda}),$$

when the gap value is 0, the optimal solution is found. Because the conjugate function $g^*(\boldsymbol{\Lambda})$ is the indicator function, is hence value 0 for the optimal solution.

Firstly, for the group penalty $P_G(\boldsymbol{\Theta})$, the duality gap is

$$\sum_{k=1}^{K}[n_k \text{trace}(\boldsymbol{S}^{(k)}\boldsymbol{\Theta}_*^{(k)}) - n_k p] + \lambda_1 \sum_{k=1}^{K}\sum_{i\neq j} |\theta_{*k,i,j}| + \lambda_2 \sum_{i\neq j} \sqrt{\sum_{k=1}^{K} {\theta_{*k,i,j}}^2} = 0. \tag{63}$$

From the equation (63), we get

$$\lambda_1 ||\boldsymbol{\Theta}_*||_1 = -\sum_{k=1}^{K} n_k \text{trace}(\boldsymbol{S}^{(k)}\boldsymbol{\Theta}_*^{(k)}) - \lambda_2 \sum_{i\neq j} \sqrt{\sum_{k=1}^{K} {\theta_{*k,i,j}}^2} + \sum_{k=1}^{K} n_k p + \sum_{k=1}^{K}\sum_{i=1}^{p} \lambda_1 |\theta_{*k,i,i}|$$
$$\leq \sum_{k=1}^{K} n_k p + \sum_{k=1}^{K}\sum_{i=1}^{p} \lambda_1 |\theta_{*k,i,i}|.$$

From the eqution (62), we have

$$\theta_{k,i,i}{}^* = diag\left(\boldsymbol{S}^{(k)} + \frac{1}{n_k}\boldsymbol{\Lambda}_*^{(k)}\right)^{-1},$$

and dual variable $\Lambda^*_{k,i,i} > 0$, for $k = 1, \ldots, K$. Hence,

$$||\boldsymbol{\Theta}_*||_1 \leq \frac{1}{\lambda_1}\sum_{k=1}^{K} n_k p + \sum_{k=1}^{K}\sum_{i=1}^{p} diag\left(\boldsymbol{S}^{(k)} + \frac{1}{n_k}\boldsymbol{\Lambda}_*^{(k)}\right)^{-1}$$
$$\leq \frac{1}{\lambda_1}\sum_{k=1}^{K} n_k p + \sum_{k=1}^{K}\sum_{i=1}^{p} diag\left(\boldsymbol{S}^{(k)}\right)^{-1}.$$

By $||\boldsymbol{\Theta}_*||_2 \leq ||\boldsymbol{\Theta}_*||_F \leq ||\boldsymbol{\Theta}_*||_1$, we get the upper bound:

$$||\boldsymbol{\Theta}_*||_2 \leq ||\boldsymbol{\Theta}_*||_F \leq \frac{1}{\lambda_1}\sum_{k=1}^{K} n_k p + \sum_{k=1}^{K}\sum_{i=1}^{p} {s_{k,i,i}}^{-1}. \tag{64}$$

The proof is similar for the fused penalty $P_F(\boldsymbol{\Theta})$, so we omit here. Next, we continue to prove the lower bound of $\boldsymbol{\Theta}_*$.

Firstly, for the group penalty $P_G(\boldsymbol{\Theta})$. Let $\boldsymbol{E}^{(k)}$ be non-negative $p \times p$ matrix satisfying $-E_{k,i,j} \leq \theta_{k,i,j} \leq E_{k,i,j}$. Introducing the Lagrange multipliers $\Gamma^{(k)}$ and $\Gamma_0^{(k)}$ for $k = 1, \ldots, K$. This procedure is similar to the way in [35].

Then, the new Lagrange problem becomes,

$$\max_{\Theta,E}\min_{\Gamma,\Gamma_0}\left\{f(\boldsymbol{\Theta})-\sum_{k=1}^{K}\sum_{i\neq j}\lambda_1 E_{k,i,j}-\lambda_2\sum_{i\neq j}\sqrt{\sum_{k=1}^{K}E_{k,i,j}^2}\right.$$
$$\left.-\sum_{k=1}^{K}tr(\boldsymbol{\Gamma}^{(k)}\boldsymbol{\Theta}^{(k)})-tr(abs(\boldsymbol{\Gamma}^{(k)})\boldsymbol{E}^{(k)})-tr(\boldsymbol{\Gamma}_0^{(k)}\boldsymbol{E}^{(k)})\right\},$$

Taking derivative w.r.t $\boldsymbol{\Theta}^{(k)}, E_{k,i,j}$, we get following equations:

$$n_k{\boldsymbol{\Theta}^{(k)}}^{-1}-n_k\boldsymbol{S}^{(k)}-\boldsymbol{\Gamma}^{(k)}=0, \tag{65}$$

$$-\lambda_1-\lambda_2\frac{E_{k,i,j}}{\sqrt{\sum_{k=1}^{K}E_{k,i,j}^2}}+|\Gamma_{k,i,j}|+\Gamma_0^{(k)}=0,\ \text{for } i\neq j, \tag{66}$$

$$|\Gamma_{k,i,j}|+\Gamma_0^{(k)}=0,\ \text{for } i=j. \tag{67}$$

When $i\neq j$, from equation (66),

$$|\Gamma_{k,i,j}|\leq\lambda_1+\lambda_2\frac{E_{k,i,j}}{\sqrt{\sum_{k=1}^{K}E_{k,i,j}^2}}$$

$$|\Gamma_{k,i,j}|^2\leq\left(\lambda_1+\lambda_2\frac{E_{k,i,j}}{\sqrt{\sum_{k=1}^{K}E_{k,i,j}^2}}\right)^2$$
$$=\lambda_1^2+2\lambda_1\lambda_2\frac{E_{k,i,j}}{\sqrt{\sum_{k=1}^{K}E_{k,i,j}^2}}+\lambda_2^2\frac{E_{k,i,j}^2}{\sum_{k=1}^{K}E_{k,i,j}^2}$$
$$\leq\lambda_1^2+2\lambda_1\lambda_2+\lambda_2^2\frac{E_{k,i,j}^2}{\sum_{k=1}^{K}E_{k,i,j}^2}.$$

Taking summation of each k,

$$\sum_{k=1}^{K}|\Gamma_{k,i,j}|^2\leq K\lambda_1^2+2K\lambda_1\lambda_2+\lambda_2^2.$$

Then,

$$\sqrt{\sum_{k=1}^{K}|\Gamma_{k,i,j}|^2}\leq\sqrt{K\lambda_1^2+2K\lambda_1\lambda_2+\lambda_2^2}.$$

From the equation (65),

$$\begin{aligned}
||\frac{1}{n_k}\boldsymbol{\Gamma}^{(k)} + \boldsymbol{S}^{(k)}||_2 &\leq \frac{1}{n_k}||\boldsymbol{\Gamma}^{(k)}||_2 + ||\boldsymbol{S}^{(k)}||_2 \\
&\leq \frac{p}{n_k}\max_{i,j}|\Gamma_{k,i,j}| + ||\boldsymbol{S}^{(k)}||_2 \\
&\leq \frac{p}{n_k}\max_{k}\max_{i,j}|\Gamma_{k,i,j}| + ||\boldsymbol{S}^{(k)}||_2 \\
&\leq \frac{p(\sqrt{K\lambda_1^2 + 2K\lambda_1\lambda_2 + \lambda_2^2})}{n_k} + ||\boldsymbol{S}^{(k)}||_2.
\end{aligned}$$

The last equation holds because

$$\max_{k}\max_{i,j}|\Gamma_{k,i,j}| \leq \sqrt{\sum_{k=1}^{K}|\Gamma_{k,i,j}|^2}.$$

We only consider the case when $i \neq j$ for $max_{i,j}|\Gamma_{ij}^{(k)}|$, because from equation (66) and (67), we know $|\Gamma_{ij}^{(k)}| > |\Gamma_{ii}^{(k)}|$. Overall, the lower bound is

$$\frac{n_k}{p\sqrt{K\lambda_1^2 + 2K\lambda_1\lambda_2 + \lambda_2^2} + n_k||\boldsymbol{S}^{(k)}||_2}.$$

The lower bound of fused penalty can be derived in similar way.

## 5.2 Proof of Proposition 2

By equation (41) and convexity of $F(\boldsymbol{\Theta})$, it is easy to get

$$||\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_*||_F \leq ||\boldsymbol{\Theta}_0 - \boldsymbol{\Theta}_*||_F.$$

Since $||\cdot||_2 \leq ||\cdot||_F$, then

$$\begin{aligned}
||\boldsymbol{\Theta}_t||_2 - ||\boldsymbol{\Theta}_*||_2 &\leq ||\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_*||_2 \\
&\leq ||\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_*||_F \\
&\leq ||\boldsymbol{\Theta}_0 - \boldsymbol{\Theta}_*||_F.
\end{aligned}$$

Hence,

$$\begin{aligned}
||\boldsymbol{\Theta}_t||_2 &\leq ||\boldsymbol{\Theta}_0 - \boldsymbol{\Theta}_*||_F + ||\boldsymbol{\Theta}_*||_2 \\
&\leq ||\boldsymbol{\Theta}_0||_F + 2||\boldsymbol{\Theta}_*||_F.
\end{aligned}$$

Then, by the equation (64), we can complete the proof of the upper bound.

In order to prove the lower bound, denote

$$a_t^{(k)} = \lambda_{\min}(\mathbf{\Theta}_t^{(k)})$$
$$(a_t)_l = \min_{k=1,\cdots,K} a_t^{(k)}.$$

By the definition of the matrix norm,

$$||\mathbf{\Theta}_t^{(k)}||_2 \geq a_t^{(k)} \geq (a_t)_l.$$

Denote the upper bound of $||\mathbf{\Theta}_t||_2$ as $M$, and that of $||\mathbf{\Theta}_t||_2$ as $M^{(k)}$, for $k = 1, \ldots, K$. By definition of tensor norm, we have $M \geq ||\mathbf{\Theta}_t||_2 \geq ||\mathbf{\Theta}_t^{(k)}||_2 \geq (a_t)_l$.

Let constant $C_1 := f(\mathbf{\Theta}_0) + g(\mathbf{\Theta}_0)$. By equation (41),

$$C_1 \geq f(\mathbf{\Theta}_t) + g(\mathbf{\Theta}_t).$$

Note that $\boldsymbol{S} \succeq 0, \mathbf{\Theta}_t \succ 0$ implies $tr(\boldsymbol{S}\mathbf{\Theta}_t) \geq 0$ and because $g(\mathbf{\Theta}_t) \geq 0$

$$\begin{aligned} C_1 &\geq -\sum_{k=1}^{K} n_k \text{logdet}\mathbf{\Theta}_t^{(k)} \\ &= -\sum_{k=1}^{K} n_k \log(\Pi_{i=1}^{p}\lambda_i). \end{aligned}$$

Let the eigenvalues of $\mathbf{\Theta}_t^{(k)}$ as $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p$. Then $a_t^{(k)} = \lambda_1 \leq \lambda_p \leq M^{(k)}$, hence,

$$\Pi_{i=1}^{p}(\lambda_i) = a_t^{(k)} \cdot \lambda_2 \cdot \cdots \cdot \lambda_p \leq a_t^{(k)} \cdot {M^{(k)}}^{(p-1)}.$$

Then,

$$\sum_{k=1}^{K} n_k \log(\Pi_{i=1}^{p}\lambda_i) \leq \sum_{k=1}^{K} n_k \left[ loga_t^{(k)} + (p-1)logM^{(k)} \right].$$

Let the coefficient $n_k$ of the term which contains $(a_t)_l$ in $-\sum_{k=1}^{K} n_k loga_t^{(k)}$ as $n_x$, then

$$\sum_{k=1}^{K} n_k loga_t^{(k)} = n_x log(a_t)_l + \sum_{k \neq x} n_k loga_t^{(k)}.$$

Because

$$M^{(k)} \leq M,$$

denote $n_m = \max_{i=1\cdots,K} n_k$ then,

$$\sum_{k=1}^{K} n_k loga_t^{(k)} \leq n_m log(a_t)_l + n_m(K-1)logM.$$

Hence,

$$\begin{aligned} C_1 &\geq -\sum_{k=1}^{K} n_k \log(\Pi_{i=1}^{p}(\lambda_i)) \\ &\geq -\sum_{k=1}^{K} n_k \left[ log a_t^{(k)} + (p-1) log M^{(k)} \right] \\ &\geq -n_m log(a_t)_l - n_m(K-1) log M \\ &\quad - K n_m (p-1) log M. \end{aligned}$$

Then, we can get

$$\begin{aligned} log(a_t)_l &\geq -K(p-1)logM - (K-1)logM - \frac{C_1}{n_m} \\ (a_t)_l &\geq e^{(1-Kp)logM - \frac{C_1}{n_m}}. \end{aligned}$$

Hence, the lower bound is proved:

$$||\mathbf{\Theta}_t||_2 \geq ||\mathbf{\Theta}_t^{(k)}||_2 \geq (a_t)_l \geq e^{-\frac{C_1}{n_m}} M^{(1-Kp)}.$$

## 5.3 Proof of Proposition 3

First, we define the following two functions,

$$r_1(V) := h(V) + \frac{\nu}{2}||V||_F^2,$$

and

$$r_2(Z) := g(Z) + \frac{\nu}{2}||Z||_F^2.$$

By the definition of $F(U)$ in (52),

$$\begin{aligned}
F(U) :&= f(U) + \min_{V,Z} \{L(U, V, Z, \Lambda_1, \Lambda_2)\} \\
&= f(U) + \min_{V,Z} \Bigg\{ h(V) + g(Z) + \sum_l \langle \Lambda_{1l}, C_{l,\cdot}U - V_l \rangle + \frac{\nu}{2} \sum_l ||C_{l,\cdot}U - V_l||_2^2 \\
&\qquad\qquad + \sum_l \langle \Lambda_{2l}, UD_{\cdot,l} - Z_l \rangle + \frac{\nu}{2} \sum_l ||UD_{\cdot,l} - Z_l||_2^2 \Bigg\} \\
&= f(U) + \min_V \left\{ h(V) + \frac{\nu}{2}||V||_F^2 - \sum_l \langle \Lambda_{1l} + \nu C_{l,\cdot}U, V_l \rangle \right\} + \frac{\nu}{2}||CU||_F^2 + \sum_l \langle \Lambda_{1l}, C_{l,\cdot}U \rangle \\
&\quad + \min_Z \left\{ g(Z) + \frac{\nu}{2}||Z||_F^2 - \sum_l \langle \Lambda_{2l} + \nu UD_{\cdot,l}, Z_l \rangle \right\} + \frac{\nu}{2}||UD||_F^2 + \sum_l \langle \Lambda_{2l}, UD_{\cdot,l} \rangle \\
&= -\max_V \left\{ \langle \nu CU + \Lambda_1, V \rangle - h(V) - \frac{\nu}{2}||V||_F^2 \right\} + f(U) + \frac{\nu}{2}||CU||_F^2 + \sum_l \langle \Lambda_{1l}, C_{l,\cdot}U \rangle \\
&\quad - \max_Z \left\{ \langle \nu UD + \Lambda_2, Z \rangle - g(Z) - \frac{\nu}{2}||Z||_F^2 \right\} + \frac{\nu}{2}||UD||_F^2 + \sum_l \langle \Lambda_{2l}, UD_{\cdot,l} \rangle \\
&= f(U) - r_1^*(\nu CU + \Lambda_1) - r_2^*(\nu UD + \Lambda_2) + \frac{\nu}{2}||CU||_F^2 + \langle \Lambda_1, CU \rangle \\
&\quad + \frac{\nu}{2}||UD||_F^2 + \langle \Lambda_2, UD \rangle.
\end{aligned}$$

By Theorem 26.3 in [68], if the function $r : \mathbb{R}^p \to \mathbb{R}$ is closed and strongly convex, then we have the differentiable conjugate function $r^*(v)$, and

$$\nabla r^*(v) = \arg\max_{u \in \mathbb{R}^p} \{\langle u, v \rangle - r(u)\}.$$

Hence, we can derive the following equations,

$$\begin{aligned}
\nabla r_1^*(v) &= \arg\max_u \left\{ \langle u, v \rangle - h(u) - \frac{\nu}{2}||u||_F^2 \right\} \\
&= \arg\max_u \left\{ -\frac{1}{2}||u||_F^2 + \frac{1}{\nu}\langle u, v \rangle - \frac{1}{\nu}h(u) \right\} \\
&= \arg\min_u \left\{ \frac{1}{2}||u - \frac{v}{\nu}||_F^2 + \frac{1}{\nu}h(u) \right\} \\
&= \text{prox}_{h/\nu}(\frac{v}{\nu}),
\end{aligned}$$

Then, we obtain

$$\nabla r_1^*(\nu CU + \Lambda_1) = \nu C^T \left( \text{prox}_{h/\nu}(CU + \frac{\Lambda_1}{\nu}) \right)$$

and, similarly,

$$\nabla r_2^*(\nu UD + \Lambda_2) = \nu \left( \text{prox}_{g/\nu}(UD + \frac{\Lambda_2}{\nu}) \right) D^T.$$

Next, take the derivative of $F(U)$ w.r.t $U$,

$$\begin{aligned}
\nabla_U F(U) &= \nabla f(U) - \nabla r_1^*(\nu CU + \Lambda_1) - \nabla r_2^*(\nu UD + \Lambda_2) \\
&\quad + \nu C^T CU + \nu UDD^T + C^T \Lambda_1 + \Lambda_2 D^T \\
&= \nabla f(U) - \nu C^T \left(\text{prox}_{h/\nu}(CU + \nu^{-1}\Lambda_1)\right) - \nu \left(\text{prox}_{g/\nu}(UD + \frac{\Lambda_2}{\nu})\right) D^T \\
&\quad + \nu C^T CU + C^T \Lambda_1 + \nu UDD^T + \Lambda_2 D^T \\
&= \nabla f(U) + C^T \left(\text{prox}_{\nu h^*}(\nu CU + \Lambda_1)\right) + \left(\text{prox}_{\nu g^*}(\nu UD + \Lambda_2)\right) D^T
\end{aligned}$$

, and we get the last equation by Moreau's decomposition.

# 6 Conclusions

Sparse estimation is often used in data science and machine learning problems. To find the solution of the optimization problem, penalty terms such as the $\ell_1$ and $\ell_2$ norms must be handled. They may not be either differentiable or decomposable. Moreover, to select the appropriate $\lambda$ in the penalty terms, we need to evaluate the performances of several $\lambda$ values and choose the optimum value.

This thesis has proposed efficient computational methods to solve the optimization problems in sparse estimation (joint graphical Lasso and convex biclustering). For the joint graphical Lasso, we propose proximal gradient methods with and without backtracking options. The method with backtracking (Algorithm 6) does not require extra variables. However, the ADMM needs to manually tune the Lagrangian penalty parameters [17] and handle dual variables. Moreover, we reduce the updated iterative step to subproblems that can be solved efficiently and accurately by Lasso-type problems. Then, we modified Algorithm 6 to one without backtracking (Algorithm 7), extending the strategy in [86] that did not rely on the Lipschitz assumption. This strategy does not require a backtracking line search, significantly reducing the computation needed when evaluating the expensive objective functions. From the theoretical perspective, we reach the linear convergence rate for the proximal gradient method, deriving the lower and upper bounds of the solution to the JGL problem and the iterates in the algorithms.

For convex biclustering, the penalty terms of the objective function are nondifferentiable and indecomposable. We transferred the objective function to the differentiable function and solve it by accelerated gradient descent method with the augmented Lagrangian method. We found that it outperformed the conventional algorithms, such as COBRA and ADMM-based procedures, in terms of efficiency. Our proposed method is more efficient than COBRA because the latter should solve two optimization problems containing nondifferentiable fused terms in each cycle. Moreover, the proposed method performed better than the ADMM-based procedures because ADMM spends more time computing the inverse matrix. Nevertheless, the proposed method uses the NAGM to update the variable $U$, which calculates only the function's gradient and does not contain the matrix inversion. Moreover, our proposed method is stable while varying the tuning parameters $\lambda$, which is convenient for identifying the appropriate $\lambda$ and visualizing the variation of the biclustering solutions under a wide range of $\lambda$.

We demonstrate the proposed methods by numerical experiments on simulated and real datasets to illustrate their high accuracy. Their efficiency is competitive with state-of-the-art algorithms.

## 6.1 Future work

For further computational improvements of solving JGL, the most expensive step in the proposed algorithms is to calculate the inversion of matrices required by the gradient of $f(\Theta)$ in JGL. Both algorithms have a complexity of $O(Kp^3)$ per iteration. We can solve the matrix inversion problem with more efficient techniques and lower complexity. In addition, we can also use the faster computation procedure in [17] to decompose the optimization problem for the proposed methods and regard it as preprocessing.

For solving CB, we can use ADMM as a warm start strategy to select an initial value for our proposed method. Moreover, it would be meaningful to derive the range of tuning parameters $\lambda$ that yield the nontrivial solutions of convex biclustering with more than one bicluster. The fusion process of the heatmap results in Figure 11 induces such an inference. Furthermore, we can extend the proposed method to solve other clustering problems, such as sparse singular value decomposition [48] and integrative generalized convex clustering models [89].

# 7 Appendix

## 7.1 Data generation

In experiments, $n_k$ i.i.d samples are generated from a normal distribution $N\{0, (\hat{\Theta}^{(k)})^{-1}\}$, and $\Theta^{(k)}$ is the $k$-th class's precision matrix. Specifically, we generate $p$ points randomly on a unit space and calculate their pairwise distances. Then, we find the $m$-nearest neighbors point using this distance. We connect any two points that are $m$-nearest neighbors of each other. The integer $m$ determines for the degree of sparsity of the data, and $m$ values vary from 4 to 9 in our experiments.

Moreover, the heterogeneity are added to the common structure by building extra individual connections: we randomly choose a pair of symmetric zero elements, $\theta_{k,i,j} = \theta_{k,j,i} = 0$, and substitute them with a value produced from the $[-1, -0,5] \cup [0.5, 1]$ interval uniformly. This operation is performed $M/2$ times, and $M$ is the number of off-diagonal nonzero elements in $\Theta^{(k)}$.

# 8 List of publications

Refereed papers

- Chen, J.; Suzuki, J. An Efficient Algorithm for Convex Biclustering. Mathematics 2021, 9, 3021. https://doi.org/10.3390/math9233021
- Chen, J.; Shimmura, R.; Suzuki, J. Efficient Proximal Gradient Algorithms for Joint Graphical Lasso. Entropy 2021, 23, 1623. https://doi.org/10.3390/e23121623

Unrefereed papers

- Zhang, B.; Chen, J.; Terada, Y. Dynamic Visualization for L1 Fusion Convex Clustering in Near-Linear Time. Uncertainty in Artificial Intelligence (UAI), 2021.

# Acknowledgements

First, I would like to express my deepest gratitude to my advisor Prof. Joe Suzuki for his outstanding support and patient mentorship throughout the entire year of my Ph.D. study. He offers me the freedom to explore my research interests and gives me helpful advice when I encounter problems. Numerous conversations with him have inspired me to conduct this research, and he is a role model in this pursuit. I am deeply thankful for his patience and encouragement.

Additionally, I would like to thank the thesis committee members, Yutaka Kano, Masayuki Uchida, and Shuichi Kawano, and Associate Professor Fuyuhiko Tanaka in our laboratory, for their insightful questions and constructive comments on my research content. I would like to thank them for the time they spent reading and attending the presentations.

In addition, I would like to thank my coauthors Yoshikazu Terada, Bingyuan Zhang, and Ryosuke Shimmura when I published journal and conference papers. They helped me with the research projects and provided many suggestions via discussion.

Last, I would like to thank my family for their unconditional love throughout all stages in my education. Without their unconditional support, I would never have gotten this far.

# References


[1] Onureena Banerjee, Laurent El Ghaoui, Alexandre d'Aspremont, and Georges Natsoulis. Convex optimization techniques for fitting sparse gaussian graphical models. In *Proceedings of the 23rd international conference on Machine learning*, pages 89–96, 2006.

[2] 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.

[3] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

[4] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[5] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[6] Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE transactions on image processing*, 18(11): 2419–2434, 2009.

[7] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[9] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[10] Florentina Bunea, Alexandre Tsybakov, and Marten Wegkamp. Sparsity oracle inequalities for the lasso. *Electronic journal of statistics*, 1:169–194, 2007.

[11] Jianhui Chen, Lei Tang, Jun Liu, and Jieping Ye. A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 137–144, 2009.

[12] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[13] Eric C Chi and Kenneth Lange. Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, 24(4):994–1013, 2015.

[14] Eric C Chi and Stefan Steinerberger. Recovering trees with convex clustering. *SIAM Journal on Mathematics of Data Science*, 1(3):383–407, 2019.

[15] Eric C Chi, Genevera I Allen, and Richard G Baraniuk. Convex biclustering. *Biometrics*, 73(1):10–19, 2017.

[16] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[17] 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*, 76(2):373, 2014.

[18] Alexandre d'Aspremont, Onureena Banerjee, and Laurent El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1): 56–66, 2008.

[19] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11): 1413–1457, 2004.

[20] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66(3):889–916, 2016.

[21] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[22] Jerome Friedman, Trevor Hastie, Holger Höfling, Robert Tibshirani, et al. Pathwise coordinate optimization. *Annals of applied statistics*, 1(2):302–332, 2007.

[23] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[24] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[25] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[26] Wenjiang Fu and Keith Knight. Asymptotics for lasso-type estimators. *The Annals of statistics*, 28(5):1356–1378, 2000.

[27] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1):17–40, 1976.

[28] Alexander J Gibberd and James DB Nelson. Regularized estimation of piecewise constant gaussian graphical models: The group-fused graphical lasso. *Journal of Computational and Graphical Statistics*, 26(3):623–634, 2017.

[29] Roland Glowinski and Americo Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 9(R2):41–76, 1975.

[30] Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.

[31] Dominique Guillot, Bala Rajaratnam, Benjamin T Rolfs, Arian Maleki, and Ian Wong. Iterative thresholding algorithm for sparse inverse covariance estimation. *arXiv preprint arXiv:1211.2532*, 2012.

[32] Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu. Joint estimation of multiple graphical models. *Biometrika*, 98(1):1–15, 2011.

[33] David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec. Network inference via the time-varying graphical lasso. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 205–213, 2017.

[34] Lei Han and Yu Zhang. Reduction techniques for graph-based convex clustering. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[35] Satoshi Hara and Takashi Washio. Learning a common substructure of multiple graphical gaussian models. *Neural Networks*, 38:23–38, 2013.

[36] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

[37] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[38] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

[39] Toby Dylan Hocking, Armand Joulin, Francis Bach, and Jean-Philippe Vert. Clusterpath an algorithm for clustering using convex fusion penalties. In *28th international conference on machine learning*, page 1, 2011.

[40] Holger Hoefling. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006, 2010.

[41] Jean Honorio and Dimitris Samaras. Multi-task learning of gaussian graphical models. In *ICML*, 2010.

[42] Cho-Jui Hsieh, Mátyás A Sustik, Inderjit S Dhillon, and Pradeep Ravikumar. Quic: quadratic approximation for sparse inverse covariance estimation. *J. Mach. Learn. Res.*, 15(1):2911–2947, 2014.

[43] Nicholas A Johnson. A dynamic programming algorithm for the fused lasso and l 0-segmentation. *Journal of Computational and Graphical Statistics*, 22(2):246–260, 2013.

[44] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[45] DCFR Koboldt, Robert Fulton, Michael McLellan, Heather Schmidt, Joelle Kalicki-Veizer, Joshua McMichael, Lucinda Fulton, David Dooling, Li Ding, Elaine Mardis, et al. Comprehensive molecular portraits of human breast tumours. *Nature*, 490(7418):61–70, 2012.

[46] Peter Langfelder and Steve Horvath. Wgcna: an r package for weighted correlation network analysis. *BMC bioinformatics*, 9(1):1–13, 2008.

[47] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[48] Mihee Lee, Haipeng Shen, Jianhua Z Huang, and JS Marron. Biclustering via sparse singular value decomposition. *Biometrics*, 66(4):1087–1095, 2010.

[49] Hongzhe Li and Jiang Gui. Gradient directed regularization for sparse gaussian concentration graphs, with applications to inference of genetic networks. *Biostatistics*, 7(2):302–317, 2006.

[50] Fredrik Lindsten, Henrik Ohlsson, and Lennart Ljung. Clustering using sum-of-norms regularization: With application to particle filter output computation. In *2011 IEEE Statistical Signal Processing Workshop (SSP)*, pages 201–204. IEEE, 2011.

[51] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM transactions on computational biology and bioinformatics*, 1(1):24–45, 2004.

[52] Rahul Mazumder and Trevor Hastie. The graphical lasso: New insights and alternatives. *Electronic journal of statistics*, 6:2125, 2012.

[53] Nicolai Meinshausen and Bin Yu. Lasso-type recovery of sparse representations for high-dimensional data. *The annals of statistics*, 37(1):246–270, 2009.

[54] Nicolai Meinshausen, Peter Bühlmann, et al. High-dimensional graphs and variable selection with the lasso. *Annals of statistics*, 34(3):1436–1462, 2006.

[55] Lance D Miller, Johanna Smeds, Joshy George, Vinsensius B Vega, Liza Vergara, Alexander Ploner, Yudi Pawitan, Per Hall, Sigrid Klaar, Edison T Liu, et al. An expression signature for p53 status in human breast cancer predicts mutation status, transcriptional effects, and patient survival. *Proceedings of the National Academy of Sciences*, 102(38):13550–13555, 2005.

[56] Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

[57] Arkadi Nemirovski. Interior point polynomial time methods in convex programming. *Lecture notes*, 42(16):3215–3224, 2004.

[58] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[59] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[60] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[61] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[62] Mee Young Park and Trevor Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4): 659–677, 2007.

[63] Kristiaan Pelckmans, Joseph De Brabanter, Johan AK Suykens, and B De Moor. Convex clustering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*, 2005.

[64] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[65] Peter Radchenko and Gourab Mukherjee. Convex clustering via l1 fusion penalization. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(5):1527–1546, 2017.

[66] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[67] James Renegar. *A mathematical view of interior-point methods in convex optimization*. SIAM, 2001.

[68] R Tyrell Rockafellar. The multiplier method of hestenes and powell applied to convex programming. *Journal of Optimization Theory and applications*, 12(6):555–562, 1973.

[69] Andreas Rosenwald, George Wright, Wing C Chan, Joseph M Connors, Elias Campo, Richard I Fisher, Randy D Gascoyne, H Konrad Muller-Hermelink, Erlend B Smeland, Jena M Giltnane, et al. The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. *New England Journal of Medicine*, 346(25):1937–1947, 2002.

[70] 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.

[71] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[72] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11): 2498–2504, 2003.

[73] Ryosuke Shimmura and Joe Suzuki. Converting admm to a proximal gradient for convex optimization problems. *arXiv preprint arXiv:2104.10911*, 2021.

[74] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2):231–245, 2013.

[75] Defeng Sun, Kim-Chuan Toh, and Yancheng Yuan. Convex clustering: Model, theoretical guarantee and efficient algorithm. *J. Mach. Learn. Res.*, 22:9–1, 2021.

[76] Joe Suzuki. *Sparse Estimation with Math and R: 100 Exercises for Building Logic*. Springer Nature, 2021.

[77] Kean Ming Tan and Daniela Witten. Statistical properties of convex clustering. *Electronic journal of statistics*, 9(2):2324, 2015.

[78] Kean Ming Tan and Daniela M Witten. Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics*, 23(4):985–1008, 2014.

[79] Kean Ming Tan, Zhaoran Wang, Tong Zhang, Han Liu, and R Dennis Cook. A convex formulation for high-dimensional sparse sliced inverse regression. *Biometrika*, 105(4):769–782, 2018.

[80] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl_1):S136–S144, 2002.

[81] Qingming Tang, Chao Yang, Jian Peng, and Jinbo Xu. Exact hybrid covariance thresholding for joint graphical lasso. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 593–607. Springer, 2015.

[82] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[83] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[84] Ryan J Tibshirani. The lasso problem and uniqueness. *Electronic Journal of statistics*, 7: 1456–1490, 2013.

[85] Ryan J Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *The annals of statistics*, 39(3):1335–1371, 2011.

[86] Quoc Tran-Dinh, Anastasios Kyrillidis, and Volkan Cevher. Composite self-concordant minimization. *J. Mach. Learn. Res.*, 16(1):371–416, 2015.

[87] Binhuan Wang, Yilong Zhang, Will Wei Sun, and Yixin Fang. Sparse convex clustering. *Journal of Computational and Graphical Statistics*, 27(2):393–403, 2018.

[88] Jie Wang, Peter Wonka, and Jieping Ye. Lasso screening rules via dual polytope projection. *J. Mach. Learn. Res.*, 16(1):1063–1101, 2015.

[89] Minjie Wang and Genevera I Allen. Integrative generalized convex clustering optimization and feature selection for mixed multi-view data. *Journal of Machine Learning Research*, 22: 1–73, 2021.

[90] Qi Wang, Pinghua Gong, Shiyu Chang, Thomas S Huang, and Jiayu Zhou. Robust convex clustering analysis. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1263–1268. IEEE, 2016.

[91] Michael Weylandt. Splitting methods for convex bi-clustering and co-clustering. In *2019 IEEE Data Science Workshop (DSW)*, pages 237–242. IEEE, 2019.

[92] Michael Weylandt, John Nagorski, and Genevera I Allen. Dynamic visualization and fast computation for convex clustering via algorithmic regularization. *Journal of Computational and Graphical Statistics*, 29(1):87–96, 2020.

[93] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

[94] Sen Yang, Zhaosong Lu, Xiaotong Shen, Peter Wonka, and Jieping Ye. Fused multiple graphical lasso. *SIAM Journal on Optimization*, 25(2):916–943, 2015.

[95] Lei Yuan, Jun Liu, and Jieping Ye. Efficient methods for overlapping group lasso. *Advances in neural information processing systems*, 24:352–360, 2011.

[96] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

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

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

[99] Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, 36(4):1567–1594, 2008.

[100] Peng Zhao and Bin Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

[101] Changbo Zhu, Huan Xu, Chenlei Leng, and Shuicheng Yan. Convex optimization procedure for clustering: Theoretical revisit. *Advances in Neural Information Processing Systems*, 27: 1619–1627, 2014.