Supplemental materials for this article are available online
http://pubs.amstat.org/doi/suppl/10.1198/106186008X382692/suppl_file/yuan.ZIP

Efficient Computation of ℓ_1 Regularized Estimates in Gaussian Graphical Models

Ming YUAN

In this article, I propose an efficient algorithm to compute ℓ_1 regularized maximum likelihood estimates in the Gaussian graphical model. These estimators, recently proposed in an earlier article by Yuan and Lin, conduct parameter estimation and model selection simultaneously and have been shown to enjoy nice properties in both large and finite samples. To compute the estimates, however, can be very challenging in practice because of the high dimensionality and positive definiteness constraint on the covariance matrix. Taking advantage of the recent advance in semidefinite programming, Yuan and Lin suggested a sophisticated interior-point algorithm to solve the optimization problem. Although a polynomial time algorithm, the optimization technique is known not to be scalable for high-dimensional problems. Alternatively, this article shows that the estimates can be computed by iteratively solving a sequence of ℓ_1 regularized quadratic programs. By effectively exploiting the sparsity of the graphical structure, I propose a new algorithm that can be applied to problems of larger scale. When combined with a path-following strategy, the new algorithm can be used to efficiently approximate the entire solution path of the ℓ_1 regularized maximum likelihood estimates, which also facilitates the choice of tuning parameter. I demonstrate the efficacy and usefulness of the proposed algorithm on a few simulations and real datasets.

Key Words: GraphGarrote, GraphLasso; Solution path.

1. INTRODUCTION

Let $X = (X^{(1)}, \dots, X^{(p)})'$ follow a multivariate normal distribution $\mathcal{N}_p(\mu, \Sigma)$ with unknown mean μ and nonsingular covariance matrix Σ. We are interested in the construction and estimation of Gaussian graphical models given an independent and identically distributed sample $X_1, \dots, X_n$ of X. This type of estimation problem naturally occurs in many statistical applications such as principal component analysis, linear discriminant analysis, and inferring relationship among multiple variables, to name a few. Most often,

Ming Yuan, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332 (E-mail: *myuan@isye.gatech.edu*).

Journal of Computational and Graphical Statistics, Volume 17, Number 4, Pages 809–826
DOI: 10.1198/106186008X382692

μ is estimated by the sample mean $\bar{X} = (\bar{X}^{(1)}, \ldots, \bar{X}^{(p)})'$ where

$$\bar{X}^{(i)} = \frac{1}{n} \sum_{j=1}^{n} X_j^{(i)}, \tag{1.1}$$

and Σ by the sample covariance matrix

$$\widehat{\Sigma} = \frac{1}{n-1} \sum_{i=1}^{n} \left(X_i - \bar{X}\right)' \left(X_i - \bar{X}\right). \tag{1.2}$$

A total of $p(p+1)/2$ parameters are needed to specify a completely unstructured covariance matrix. Due to the large number of unknown parameters to be estimated, $\widehat{\Sigma}$ is not a stable estimate for moderate or large p's, as often encountered in real applications. Even worse, when $p \geq n$, $\widehat{\Sigma}$ is not positive definite. In contrast, by effectively exploiting the conditional independence relationship, the dimension of the estimation problem can be greatly reduced in a Gaussian graphical model.

A graphical model for X is customarily represented by an undirected graph $G = (V, E)$, where V contains p vertices corresponding to each of the p coordinates and the edges $E = (e_{ij})_{1 \leq i < j \leq p}$ describe the conditional independence relationship among $X^{(1)}, \ldots, X^{(p)}$. The edge between $X^{(i)}$ and $X^{(j)}$ is absent if and only if $X^{(i)}$ and $X^{(j)}$ are independent conditional on the other variables. See Whittaker (1990), Lauritzen (1996), and Edwards (2000) for statistical properties of Gaussian graphical models and commonly used model selection and parameter estimation methods in such models. It is known that the graphical structure of (V, E) can be inferred from the zero pattern of the concentration matrix $C = \Sigma^{-1}$. A zero entry $c_{ij} = 0$ in the concentration matrix indicates the conditional independence between $X^{(i)}$ and $X^{(j)}$ given all other variables. Thus, parameter estimation and model selection in the Gaussian concentration graph model is equivalent to estimating parameters and identifying zeros in the concentration matrix.

The standard approach for constructing Gaussian graphical models, or often referred to as the covariance selection problem (Dempster 1972), is the greedy stepwise forward selection or backward deletion, and parameter estimation is based on the selected model. In each step the edge selection or deletion is typically done through hypothesis testing at some level α. It has long been recognized that this procedure does not correctly account for the multiple comparisons involved (see, e.g., Edwards 2000). Another drawback of the common stepwise procedure is its computational complexity. At each single step the edge selection or deletion requires fitting a large number of candidate models (of the order p^2). This is computationally infeasible for even moderate p's. A number of other covariance selection approaches have been introduced in recent years (Drton and Perlman 2004; Meinshausen and Bühlmann 2006; Li and Gui 2005; Yuan and Lin 2007a). In particular, Yuan and Lin (2007a) proposed an effective method that conducts parameter estimation and model selection simultaneously, and showed that their penalized likelihood estimates enjoy nice properties in both large and finite samples. Their so-called graphLasso estimate of the concentration matrix is a constrained maximum likelihood estimate defined as

$$\min_{C \succ 0} \left[-\ln|C| + \operatorname{tr}\left(C\bar{A}\right)\right], \quad \text{subject to} \quad \sum_{i \neq j} |c_{ij}| \leq M, \tag{1.3}$$

where $C \succ 0$ indicates that C is symmetric and positive definite,

$$\bar{A} = \frac{1}{n} \sum_{j=1}^{n} \left(X_j - \bar{X}\right)\left(X_j - \bar{X}\right)' \tag{1.4}$$

is the unrestricted maximum likelihood estimate of the covariance matrix, and $M > 0$ is a regularization parameter. Clearly when $M = +\infty$, it reduces to the unconstrained maximum likelihood estimate. As M decreases to 0, more and more c_{ij}'s will be shrunken to zero and therefore achieve the goal of model selection. The graphLasso estimate can also be equivalently written in the Lagrange form:

$$\widehat{C}^{\text{graphLasso}}(\lambda) = \arg\min_{C \succ 0} \left[-\ln|C| + \operatorname{tr}\left(C\bar{A}\right) + \lambda \sum_{i \neq j} |c_{ij}| \right], \tag{1.5}$$

for some $\lambda \geq 0$. A closely related estimate, referred to as graphGarrote, is given by

$$\min_{C \succ 0} \left[-\ln|C| + \operatorname{tr}\left(C\bar{A}\right) \right], \quad \text{subject to} \quad \sum_{i \neq j} \frac{c_{ij}}{\tilde{c}_{ij}} \leq M, \quad \text{and} \quad c_{ij}/\tilde{c}_{ij} \geq 0, \tag{1.6}$$

where $\tilde{C} = (\tilde{c}_{ij})_{p \times p}$ is a preliminary estimate of the concentration matrix. An obvious choice of $\tilde{C}$ is $\widehat{\Sigma}^{-1}$ given that $\widehat{\Sigma}$ is invertible. It can also be equivalently written as

$$\widehat{C}^{\text{graphGarrote}}(\lambda) = \arg \min_{C \succ 0, c_{ij}/\tilde{c}_{ij} \geq 0} \left[-\ln|C| + \operatorname{tr}\left(C\bar{A}\right) + \lambda \sum_{i \neq j} \frac{c_{ij}}{\tilde{c}_{ij}} \right]. \tag{1.7}$$

Although both estimates are given as the solution to convex optimization problems, to compute them can be very challenging in practice because of the high dimensionality and the positive definiteness constraint. Yuan and Lin (2007a) adopted an interior-point algorithm recently developed for the so-called maxdet problem (Vandenberghe, Boyd, and Wu 1998) to solve (1.5) and (1.7). Their algorithm has been demonstrated to be very effective in small-scale problems. But this algorithm is known not to be scalable for large-scale problems because it does not exploit the sparsity structure of high-dimensional graphs. For example, the maxdet program provided by Wu, Vandenberghe, and Boyd (1996) can only handle up to several hundred unknown parameters, which amounts to a covariance matrix with dimension up to about 25. This problem is exacerbated when a good regularization parameter needs to be selected.

In this article, I propose an alternative algorithm that more effectively exploits the structure of graphLasso and graphGarrote. I show that (1.5) or (1.7) can be solved by minimizing a sequence of ℓ_1 regularized least squares, which can be computed extremely fast in the same fashion as the recently proposed Lars for linear regression (Osborne, Presnell, and Turlach 2000; Efron, Hastie, Johnstone, and Tibshrani 2004). Taking advantage of the sparsity of the graphical structure, our algorithm can be applied to problems of higher dimension. Similar to other methods of regularization, graphLasso, and graphGarrote proceed in two steps in practice. First the solution path indexed by the tuning parameter λ (or equivalently M) is constructed. The second step, often referred to as tuning, selects the final estimate on the solution path. For most methods of regularization, it is very expensive to compute the exact solution path. One has to approximate the solution path by evaluating the estimate for a fine grid of tuning parameters and there is a tradeoff between the approximation accuracy and the computational cost in determining how fine a grid of tuning parameters to be considered. In particular, graphLasso or graphGarrote solution path can be approximated by solving (1.5) and (1.7) for a series of λ's, as done by Yuan and Lin (2007a). When combined with a path-following strategy similar to that of Park and Hastie (2007), the proposed algorithm can be used to efficiently approximate the entire solution path of the ℓ_1 regularized maximum likelihood estimates, which facilitates the choice of tuning parameter.

The rest of the article is organized as follows. We describe the new algorithm for solving (1.5) and (1.7) with a prespecified tuning parameter λ in the next section. Section 3 proposes a path-following algorithm to approximate the whole solution path of graphLasso or graphGarrote. Sections 4 and 5 demonstrate the efficacy and usefulness of the proposed algorithm on a few simulations and real datasets. Section 6 concludes with a discussion.

2. COMPUTING GRAPHLASSO AND GRAPHGARROTE

The objective function of (1.5) or (1.7) is strictly convex in the feasible region where C is restricted to be symmetric and positive definite. This guarantees that the solution is unique. Computationally, however, it is not immediately clear how these constraints, particularly the positive definiteness constraint, can be effectively enforced in solving (1.5) or (1.7).

We begin with the graphLasso estimate given by (1.5). First consider a subproblem where C is known up to its ith row and column. Suppose $C_{-i,-i}$ is positive definite, then C is also positive definite if and only if

$$c_{ii} - C_{i,-i}C_{-i,-i}^{-1}C_{-i,i} > 0, \tag{2.1}$$

where subscript $-i$ indicates that the ith column or row is removed from a matrix. Therefore it is very easy to ensure the positive definiteness of C if we want to update the ith row and column of C. Essentially, the original semidefinite program of (1.5) reduces to a constrained vector optimization problem

$$\min -\ln|C| + \operatorname{tr}\left(C\bar{A}\right) + \lambda\sum_{i\neq j}|c_{ij}| \quad \text{subject to} \quad c_{ii} - C_{i,-i}C_{-i,-i}^{-1}C_{-i,i} > 0, \tag{2.2}$$

where the minimization is taken over vector $C_{i,\cdot} = C'_{\cdot,i}$.

Note that

$$\det(C) = \det(C_{-i,-i})\left(c_{ii} - C_{i,-i}C_{-i,-i}^{-1}C_{-i,i}\right). \tag{2.3}$$

Therefore, up to a constant not depending on c_{ii} and $C_{-i,i}$, the objective function of (2.2) can be rewritten as

$$-\ln\left(c_{ii} - C'_{-i,i}C_{-i,-i}^{-1}C_{-i,i}\right) + c_{ii}\bar{A}_{ii} + 2\bar{A}_{i,-i}C_{-i,i} + 2\lambda\sum_{j\neq i}|c_{ij}|, \tag{2.4}$$

where we used the fact that $\bar{A}$ is symmetric and $C_{i,-i} = C'_{-i,i}$. A first-order condition implies that

$$c_{ii} = \frac{1}{\bar{a}_{ii}} + C'_{-i,i} C^{-1}_{-i,-i} C_{-i,i}. \tag{2.5}$$

Plugging (2.5) back to (2.4), $C_{-i,i}$ minimizes

$$\frac{1}{2} C'_{-i,i} \left(\bar{a}_{ii} C^{-1}_{-i,-i} \right) C_{-i,i} + A_{i,-i} C_{-i,i} + \lambda \sum_{j \neq i} |c_{ij}|, \tag{2.6}$$

which can be formulated as a quadratic program. We can compute (1.5) by solving (2.6) for each row and iterating until convergence. To sum up, the graphLasso estimate can be obtained using the iterative Algorithm 1.

Algorithm 1 Algorithm for Computing graphLasso Estimate

```
Input: $\bar{A}$, $\lambda \geq 0$ and an initial value for $C$
Output: The minimizer of (1.5)
Repeat
for $i = 1$ to $p$
Update $C_{-i,i}$, or equivalently $C_{i,-i}$ by solving
    $\min_{C_{-i,i}} \frac{1}{2} C'_{-i,i} \left( \bar{a}_{ii} C^{-1}_{-i,-i} \right) C_{-i,i} + \bar{A}_{i,-i} C_{-i,i} + \lambda \sum_{j \neq i} |c_{ij}|.$   (2.7)
Update $c_{ii} = 1/\bar{a}_{ii} + C'_{-i,i} C^{-1}_{-i,-i} C_{-i,i}$.
end
Until a certain convergence criterion is met
```

Using Algorithm 1, we solve (1.5) by iteratively solving quadratic programs described by (2.7). Convex quadratic programming can be done very efficiently, whereas nonconvex quadratic programming is also known to be NP-complete. Therefore it is crucial to ensure that $\bar{a}_{ii} C^{-1}_{-i,-i}$ is always positive definite during the iterations. The following lemma guarantees that this is the case whenever a strictly convex initial value for C is applied.

Lemma 1. *If the initial value of the concentration matrix is symmetric and positive definite, then (2.7) is convex at any stage of the iteration.*

Proof: If C is positive definite, then $C_{-i,-i}$ is also positive definite. Consequently $\bar{a}_{ii} C^{-1}_{-i,-i}$ is also positive definite. It is therefore sufficient to show that C's positive definiteness is preserved throughout the iterations. This can be proved by induction. Assume that C is positive definite prior to updating the ith row and column. Because we ensure (2.1) in updating $C_{i,\cdot}$ and $C_{\cdot,i}$, C remains to be positive definite after the update. □

Because (2.7) is convex, we can solve it using a path-following algorithm in a similar spirit to the Lars algorithm for the Lasso (Tibshirani 1996). For brevity, write $W =$

$\bar{a}_{ii}C^{-1}_{-i,-i}$ and $\mathbf{b} = \bar{A}_{-i,i}$. We can rewrite (2.7) as

$$\min_{\mathbf{u}} \frac{1}{2}\mathbf{u}'W\mathbf{u} + \mathbf{b}'\mathbf{u} + \lambda \sum_{i=1}^{p-1} |u_j|. \tag{2.8}$$

The Karush-Kuhn-Tucker conditions imply that

$$W_{i,\cdot}\mathbf{u} + b_i + \lambda \text{sign}(u_j) = 0 \quad \text{if} \quad u_j \neq 0 \tag{2.9}$$

$$\left|W_{i,\cdot}\mathbf{u} + b_i\right| \leq \lambda \quad \text{if} \quad u_j = 0. \tag{2.10}$$

Denote the active set $\mathcal{A} = \{i : u_i \neq 0\}$. Simple algebra leads to

$$\mathbf{u}_{\mathcal{A}} = W^{-1}_{\mathcal{A},\mathcal{A}}\left(\mathbf{b}_{\mathcal{A}} + \lambda \text{sign}(\mathbf{u}_{\mathcal{A}})\right), \tag{2.11}$$

which suggests that the minimizer of (2.8) is piecewise linear in λ. Taking advantage of this fact, we can sequentially construct the solution with the Lagrange parameter decreasing from $+\infty$ to λ. More precisely, we use Algorithm 2 to solve (2.7). The same idea is used to devise the popular Lars algorithm (Osborne et al. 2000; Efron et al. 2004). As pointed out by Efron et al. (2004), the complexity of Algorithm 2 is $O(p^3)$. Therefore each iteration of Algorithm 1 has complexity $O(p^4)$. In contrast, the complexity of one iteration in the interior point algorithm of Vandenberghe et al. (1998) is $O(p^6)$.

The graphGarrote estimate can also be computed using Algorithm 1. To update $C_{-i,i}$, replace (2.7) with

$$\min_{C_{-i,i}=C'_{i,-i}} \frac{1}{2}C'_{-i,i}\left(\bar{a}_{ii}C^{-1}_{-i,-i}\right)C_{-i,i} + A_{i,-i}C_{-i,i} + \lambda \sum_{j\neq i}\frac{c_{ij}}{\tilde{c}_{ij}} \quad \text{subject to} \quad c_{ij}/\tilde{c}_{ij} \geq 0. \tag{2.16}$$

Again, the solution of (2.16) is piecewise linear in λ. An algorithm similar to Algorithm 2 can be applied, and is stated as Algorithm 3

The following theorem shows that both Algorithms 2 and 3 are valid if $\bar{a}_{ii}C^{-1}_{-i,-i} \succ 0$.

Theorem 1. *Under the one-at-a-time condition, if* $\bar{a}_{ii}C^{-1}_{-i,-i} \succ 0$, *then Algorithms 2 and 3 are valid.*

Proof: The proof for Algorithm 2 follows in the same way as that for the Lasso (Efron et al. 2004), and Algorithm 3 as that of the nonnegative garrote (Yuan and Lin, 2007b). It is therefore omitted here. □

The "one-at-a-time" condition of Theorem 1 was first introduced by Efron et al. (2004) to derive the connection between the Lasso and the Lars, and later used by Yuan and Lin (2007b) to prove that the nonnegative garrote solution path is piecewise linear. With the current notation, the condition states that j^* in Step 6 of both algorithms is uniquely defined. This assumption basically means that (i) the addition occurs only for one edge at a time; (ii) no edge disappears at the time of addition; and (iii) no two edges disappear simultaneously. This is generally true in practice and can always be enforced by slightly perturbing the observations. For more detailed discussions, the readers are referred to Efron et al. (2004).

```
Algorithm 2 Algorithm for Minimizing (2.7)
Input: $W = A_{ii}C^{-1}_{-i,-i}$, $\mathbf{b} = A_{-i,i}$ and $\lambda \geq 0$
Output: The minimizer of (2.7)

  1. Start from $\mathbf{u}^{[0]} = 0$, $\lambda^{[0]} = \max_j |b_j|$ and $k = 1$.
  2. Compute the current "active set"
         $\mathcal{A}_1 = \arg\max_j |b_j|$.     (2.12)
  3. Compute the current direction $\nu$ which is a $p$ dimensional vector with $\gamma_{\mathcal{A}_k^c} = 0$ and
         $\gamma_{\mathcal{A}_k} = -W^{-1}_{\mathcal{A}_k,\mathcal{A}_k}\mathbf{b}'_{\mathcal{A}_k} - \mathbf{u}^{[k-1]}_{\mathcal{A}_k}$.     (2.13)
  4. For every $j \notin \mathcal{A}_k$, compute how far the algorithm will progress in the direction $\gamma$
     before a $j \notin \mathcal{A}_k$ enters the active set. This can be measured by a $\alpha_j \in [0,1]$ such
     that
         $\left|W_{j,\mathcal{A}_k}\left(\mathbf{u}^{[k-1]}_{\mathcal{A}_k} + \alpha\gamma_{\mathcal{A}_k}\right) + b_j\right| = (1-\alpha)\lambda^{[k-1]}$.     (2.14)
  5. For every $j \in \mathcal{A}_k$ such that $\gamma_j u^{[k-1]}_j < 0$, compute how far the algorithm will
     progress in the direction $\gamma$ before it leaves the active set. This can be measured by
     $\alpha_j = -u^{[k-1]}_j/\gamma_j$.
 (6) Let $\alpha^* = \min_j \alpha_j \equiv \alpha_{j^*}$. If $j^* \notin \mathcal{A}^{[k]}$, update $\mathcal{A}_{k+1} = \mathcal{A}^{[k]} \cup \{j^*\}$. Otherwise,
     update $\mathcal{A}^{[k+1]} = \{j \in \mathcal{A}^{[k]} : j \neq j^*\}$.
 (7) Update $\mathbf{u}^{[k]} = \mathbf{u}^{[k-1]} + \alpha^*\gamma$ and $\lambda^{[k]} = (1-\alpha^*)\lambda^{[k-1]}$. Set $k = k+1$ and Go back
     to step (3) until $\lambda^{[k]} \leq \lambda$.
 (8) Return
         $\mathbf{u} = \frac{(\lambda - \lambda^{[k]})\mathbf{u}^{[k-1]} + (\lambda^{[k-1]} - \lambda)\mathbf{u}^{[k]}}{\lambda^{[k-1]} - \lambda^{[k]}}$.     (2.15)
```

3. APPROXIMATING SOLUTION PATHS

The ℓ_1 regularization procedures are useful because they conduct parameter estimation and model selection simultaneously. They give a shrinkage estimate that is known to be useful when there are multiple parameters to be estimated, and select edges in a manner less greedy than stepwise methods. Results, however, depend on the amount of penalization. It is, therefore, critical to be able to compute a series of estimates for a fine grid of tuning parameters. In this section, we propose a strategy that, combined with the algorithm presented in the last section, can efficiently approximate the solution path for the ℓ_1 regularization.

To fix ideas, we focus on graphLasso. First note that when λ is large enough, $\widehat{C}^{\text{graphLasso}}(\lambda)$ is a diagonal matrix. The following lemma shows exactly how large λ needs to be.

Algorithm 3 Algorithm for Minimizing (2.16)

Input: $W = A_{ii}C^{-1}_{-i,-i}$, $\mathbf{b} = A_{-i,i}$, $\lambda \geq 0$, $\mathbf{w} = \tilde{C}_{-i,i}$ and $\mathbf{s} = (\text{sign}(w_1), \dots, \text{sign}(w_{p-1}))'$ **Output:** The minimizer of (2.16)

```
1. Start from $\mathbf{u}^{[0]} = 0$, $\lambda^{[0]} = -\min_j \left(w_j b_j\right)$ and $k = 1$.
2. Compute the current "active set"
       $$\mathcal{A}_1 = \arg\min_j \left(w_j b_j\right). \tag{2.17}$$
3. Compute the current direction $\nu$ which is a $p$-dimensional vector with $\gamma_{\mathcal{A}_k^c} = 0$ and
       $$\gamma_{\mathcal{A}_k} = -W^{-1}_{\mathcal{A}_k,\mathcal{A}_k}\mathbf{b}'_{\mathcal{A}_k} - \mathbf{u}^{[k-1]}_{\mathcal{A}_k}. \tag{2.18}$$
4. For every $j \notin \mathcal{A}_k$, compute how far the algorithm will progress in the direction $\gamma$
   before a $j \notin \mathcal{A}_k$ enters the active set. This can be measured by a $\alpha_j \in [0,1]$ such
   that
       $$\left[W_{j,\mathcal{A}_k}\left(\mathbf{u}^{[k-1]}_{\mathcal{A}_k} + \alpha\gamma_{\mathcal{A}_k}\right) + b_j\right] = -(1-\alpha)\lambda^{[k-1]}/w_j. \tag{2.19}$$
5. For every $j \in \mathcal{A}_k$ such that $\gamma_j w_j < 0$, compute how far the algorithm will progress
   in the direction $\gamma$ before it leaves the active set. This can be measured by $\alpha_j = -u^{[k-1]}_j/\gamma_j$.
6. Let $\alpha^* = \min_j \alpha_j \equiv \alpha_{j^*}$. If $j^* \notin \mathcal{A}^{[k]}$, update $\mathcal{A}_{k+1} = \mathcal{A}^{[k]} \cup \{j^*\}$. Otherwise,
   update $\mathcal{A}^{[k+1]} = \{j \in \mathcal{A}^{[k]} : j \neq j^*\}$.
7. Update $\mathbf{u}^{[k]} = \mathbf{u}^{[k-1]} + \alpha^*\gamma$ and $\lambda^{[k]} = (1-\alpha^*)\lambda^{[k-1]}$. Set $k = k+1$ and Go back
   to step (3) until $\lambda^{[k]} \leq \lambda$.
8. Return
       $$\mathbf{u} = \frac{(\lambda - \lambda^{[k]})\mathbf{u}^{[k-1]} + (\lambda^{[k-1]} - \lambda)\mathbf{u}^{[k]}}{\lambda^{[k-1]} - \lambda^{[k]}}. \tag{2.20}$$
```

Lemma 2. *The minimizer of (1.5) is* $\text{diag}(1/\bar{a}_{11}, \dots, 1/\bar{a}_{pp})$ *if and only if* $\lambda \geq \lambda_{\max} \equiv \max_{i \neq j} |\bar{a}_{ij}|$.

Proof: Let $l(C) = -\ln|C| + \text{trace}\left(C\bar{A}\right)$. Then, by the Karush-Kuhn-Tucker conditions (Boyd and Vandenberghe 2003), the minimizer of (1.5) also solves

$$\frac{\partial l}{\partial c_{ij}} = -\lambda \text{sign}(c_{ij}) \quad \text{for all} \quad c_{ij} \neq 0, \tag{3.1}$$

$$\left|\frac{\partial l}{\partial c_{ij}}\right| \leq \lambda \quad \text{for all} \quad c_{ij} = 0, \tag{3.2}$$

where $\partial l/\partial c_{ij} = -(C^{-1})_{ij} + \bar{a}_{ij}$. It is not hard to verify that $\text{diag}(1/\bar{a}_{11}, \dots, 1/\bar{a}_{pp})$ satisfies both (3.1) and (3.2) if and only if $\lambda \geq \lambda_{\max}$. □

The solution path for (1.5) is therefore determined as λ decreases from $\lambda_{\max}$ to 0. When $\lambda = \lambda_{\max}$, there are no edges in the graphical model and all nodes are independent. As λ decreases edges enter the graphical model, and occasionally, some existing edges may leave the model. A common approach to approximate the solution path is to evaluate $\widehat{C}(\lambda)$ for a fine grid of λ's and to use linear interpolate between them. One has to trade the approximation accuracy with the computational cost in determining how fine of a grid of tuning parameter is to be considered.

Here, we take an approach similar to that of Park and Hastie (2007). We capture the main structure of the solution path by computing the exact solution at the "turning points" when edge addition or deletion occurs, and then linearly interpolate between them. Starting with $\lambda_{\max}$, we compute $\widehat{C}^{\text{graphLasso}}(\lambda_{\max})$ and then make an intelligent guess for the next "turning point." A sensible choice actually comes as a byproduct of Algorithm 2. Note that in updating $C_{-i,i}$ using Algorithm 2, we compute a piecewise linear solution path where each change point corresponds to a change of the active set. Algorithm 2 not only solves (2.7), but also provides $\lambda_i^L < \lambda_{\max}$ where λ_i^L corresponds to the change point that is immediately next to $\lambda_{\max}$ in updating $C_{-i,i}$. A natural guess for the next "turning point" of the solution path $\widehat{C}^{\text{graphLasso}}(\lambda)$ is therefore $\lambda^{(1)} = \max \lambda_i^L$. We then compute the exact solution with this tuning parameter. This time, in using Algorithm 2, we may be able to find for each i, two tuning parameters $\lambda_i^L < \lambda^{(1)} < \lambda_i^U$ that correspond to the change points immediate next to $\lambda^{(1)}$.

Two possible scenarios may occur with our guess. If $\lambda^{(1)}$ is greater than or equal to the next actual "turning point," we update our guess with $\lambda^{(2)} = \max \lambda_i^L$ as before. If it is smaller than the next actual "turning point," we miss the actual "turning point" and need to increase $\lambda^{(1)}$. A good candidate is $\lambda^{(2)} = \min \lambda_i^U$. To distinguish between these two possible scenarios, we keep track of the so-called active set defined as

$$\mathcal{A}(\lambda) = \arg\max_{i \neq j} \left| \frac{\partial l}{\partial c_{ij}} \right|. \tag{3.3}$$

When $\lambda^{(1)}$ is greater than the next actual "turning point," $\mathcal{A}(\lambda^{(1)}) = \mathcal{A}(\lambda_{\max})$. Otherwise $\mathcal{A}(\lambda^{(1)})$ differs from $\mathcal{A}(\lambda_{\max})$ generally. At the actual "tuning point", $\mathcal{A}(\lambda^{(1)}) \neq \mathcal{A}(\lambda_{\max})$ but $\{(i,j) : i \neq j, \widehat{c}_{ij}(\lambda^{(1)}) \neq 0\} = \mathcal{A}(\lambda_{\max})$. We continue this process until the whole solution path is constructed.

Interestingly, in constructing the solution path, we are also able to speed up the computation for a fixed tuning parameter. Denote for a tuning parameter λ, $\mathcal{B}_i(\lambda) = \{j \neq i : \widehat{c}_{ij}(\lambda) \neq 0\}$. The solution to (2.7) can be given as $c_{ij} = 0$ if $j \notin \mathcal{B}_i(\lambda)$ and $C_{\mathcal{B}_i,i}$ is given by

$$\arg\min \left[\frac{1}{2} C'_{\mathcal{B}_i,i} \left(A_{ii} C^{-1}_{-i,-i} \right)_{\mathcal{B}_i,\mathcal{B}_i} C_{\mathcal{B}_i,i} + A_{i,\mathcal{B}_i} C_{\mathcal{B}_i,i} + \lambda \sum_{j \in \mathcal{B}_i} |c_{ij}| \right]. \tag{3.4}$$

For a sparse graph, the cardinality of $\mathcal{B}_j$ is much smaller than p. Therefore computing (3.4) should be much faster than minimizing (2.7). The difficulty, of course, is that $\mathcal{B}_i(\lambda)$ is not known apriori. Nevertheless, we can update C and $\mathcal{B}$'s in an iterative fashion.

Algorithm 4 An Alternative Algorithm for Minimizing (2.7)

```
Input: $W = A_{ii}C^{-1}_{-i,-i}$, $\mathbf{b} = A_{-i,i}$, $\lambda \geq 0$ and initial value for $\mathcal{B}_i$
Output: The minimizer of (2.7)

1. Using Algorithm 1 to compute (3.4). Update $C_{\mathcal{B}_i,i}$ with the solution. Set $C_{-\mathcal{B}_i,i} = \mathbf{0}$.
2. Compute the current "active set"
       $\mathcal{A}_i = \arg\max_j \left| W_{j,\mathcal{B}_i} C_{\mathcal{B}_i,i} + b_j \right|.$   (3.5)
3. If $\mathcal{A}_i \neq \mathcal{B}_i$, update $\mathcal{B}_i$ with $\mathcal{A}_i \cup \mathcal{B}_i$ and go back to step (1).
4. Return $C_{-i,i}$.
```

Lemma 3. *If $\bar{a}_{ii}C^{-1}_{-i,-i} \succ 0$, Algorithm 4 always converges and converges to the minimizer of (2.7).*

Proof: Clearly $\mathcal{B}_i$ is strictly increasing and the algorithm has to terminate within at most p iterations. At its convergence, it is evident that (3.2) is satisfied for any $j \notin \mathcal{B}_i$. On the other hand, by the Karush-Kuhn-Tucker conditions, (3.1) and (3.2) are satisfied for any $j \in \mathcal{B}_i$. The proof is then completed by the fact that (2.7) is strictly convex when $\bar{a}_{ii}C^{-1}_{-i,-i} \succ 0$.□

Depending on the initial value, Algorithm 4 may take up to p iterationsand could be computationally more demanding than Algorithm 2. On the other hand, if the true $\mathcal{B}_i$ is small and a good starting value is available, Algorithm 4 can be much faster than Algorithm 2. In constructing the solution path, we automatically have a very good initial value for $\mathcal{B}_i$. Denote by λ^* the previous "turning point". Then as discussed before, $\mathcal{B}_i(\lambda) = \{j : (i,j) \in \mathcal{A}(\lambda^*)\}$ unless λ is smaller than the next "turning point." Therefore, $\{j : (i,j) \in \mathcal{A}(\lambda^*)\}$ can be used as an initial value for $\mathcal{B}_i$ in Algorithm 4.

It is worth noting that in high-dimensional problems the graph is most often sparse, and we may not need to construct the whole solution path. The above procedure can therefore be terminated at an earlier stage. To determine when to stop the algorithm, one could adopt the Bayesian information criterion (BIC) of Yuan and Lin (2007a):

$$\mathrm{BIC}(\lambda) = -\ln|\widehat{C}(\lambda)| + \mathrm{trace}(\widehat{C}(\lambda)\bar{A}) + \frac{\ln n}{n}\sum_{i \leq j}\widehat{e}_{ij}(\lambda), \tag{3.6}$$

where $e_{ij}(\lambda) = 1$ if $\widehat{c}_{ij}(\lambda) \neq 0$ and 0 otherwise.

4. NUMERICAL EXAMPLES

4.1 MATHEMATICS MARKS DATA

To demonstrate the proposed algorithm, we first consider a relatively small-scale problem. We use the Mathematics marks dataset from Mardia, Kent and Bibby (1979). The

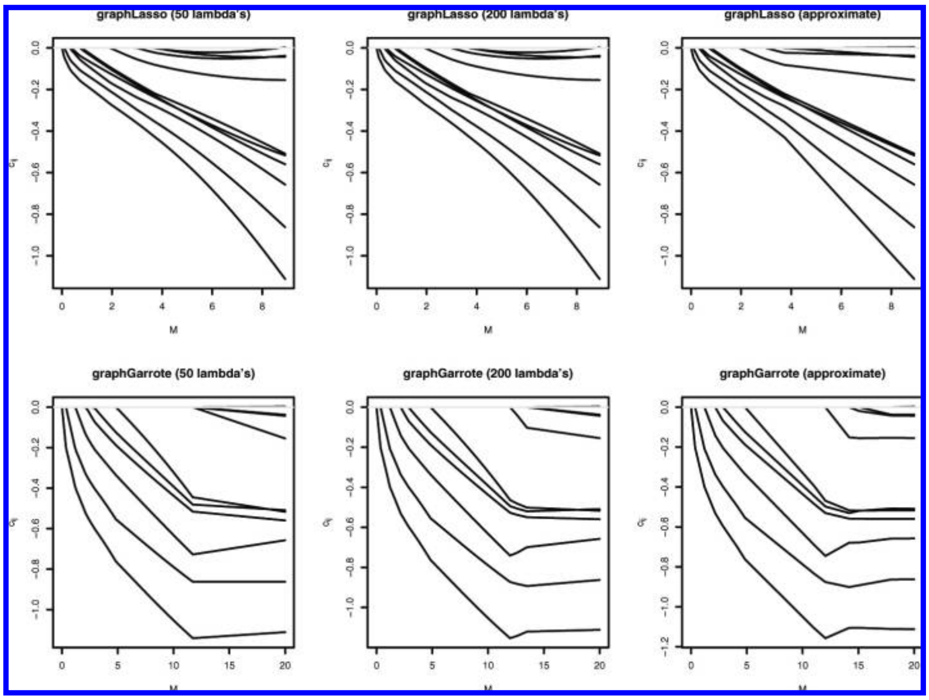

Figure 1. Mathematics marks dataset: solution paths constructed using equally spaced values for the tuning parameters, and the proposed algorithm. In each plot, the y-axis corresponds to the value of the off-diagonal entries of the estimated concentration matrix, and the x-axis represents the value of the tuning parameter $M = \sum_{i \neq j} |c_{ij}|$.

data contain the marks of $n = 88$ students in the $p = 5$ examinations in mechanics, vectors, algebra, analysis, and statistics. The data have previously been analyzed in Yuan and Lin (2007a). Yuan and Lin (2007a) computed the graphLasso and graphGarrote using their interior point algorithm for a grid of tuning parameters. For problem of this size, the estimates can be computed efficiently using both the proposed algorithms and those of Yuan and Lin (2007a). To appreciate the efficacy of the proposed algorithms for path construction, we plot the approximated solution path with 50 and 200 equally spaced values of the tuning parameters for graphLasso and graphGarrote in Figure 1. For the graphLasso, 50 values of the tuning parameters seem to be enough. But for the graphGarrote, some interesting characteristics are missing with only 50 values. In contrast, when using the proposed algorithm we are able to accurately recover the solution path with only ten different values of the tuning parameters.

The order that the edges enter the graphical model is presented in Table 1.

By constructing the whole solution path, our algorithm also facilitates the selection of the final graphical model. Yuan and Lin (2007a) proposed to use the BIC score defined in (3.6) to determine the final graphical model. The final model selected using the BIC score is presented in Figure 2. In particular, Yuan and Lin (2007a) showed that graphGarrote

Table 1. The order that each edge enters the graphical model for the math marks data.

Order	graphLasso	graphGarrote
Step 1	Algebra – Analysis	Algebra – Analysis
Step 2	Algebra – Statistics	Algebra – Statistics
Step 3	Vector – Algebra	Vector – Algebra
Step 4	Analysis – Statistics	Mechanics – Vector
Step 5	Mechanics – Vector	Mechanics – Algebra
Step 6	Mechanics – Algebra	Analysis – Statistics
Step 7	Vector – Analysis	Vector – Analysis
Step 8	Vector – Statistics	Mechanics – Statistics
Step 9	Mechanics – Analysis	Vector – Statistics
Step 10	Mechanics – Statistics	Mechanics – Analysis

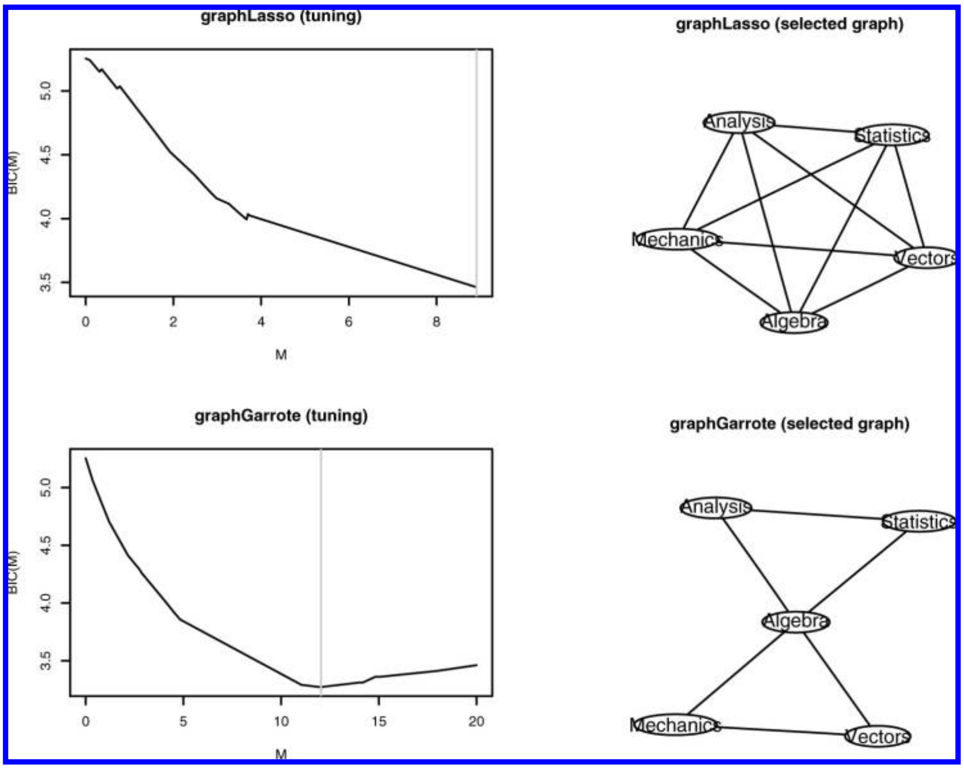

Figure 2. Math marks dataset: tuning with BIC, and the final graphical model selected. In the left plots, the x-axis represents the value of the tuning parameter M, and the y-axis represents the corresponding BIC score. The gray line in each plot is the value that minimizes the BIC score.

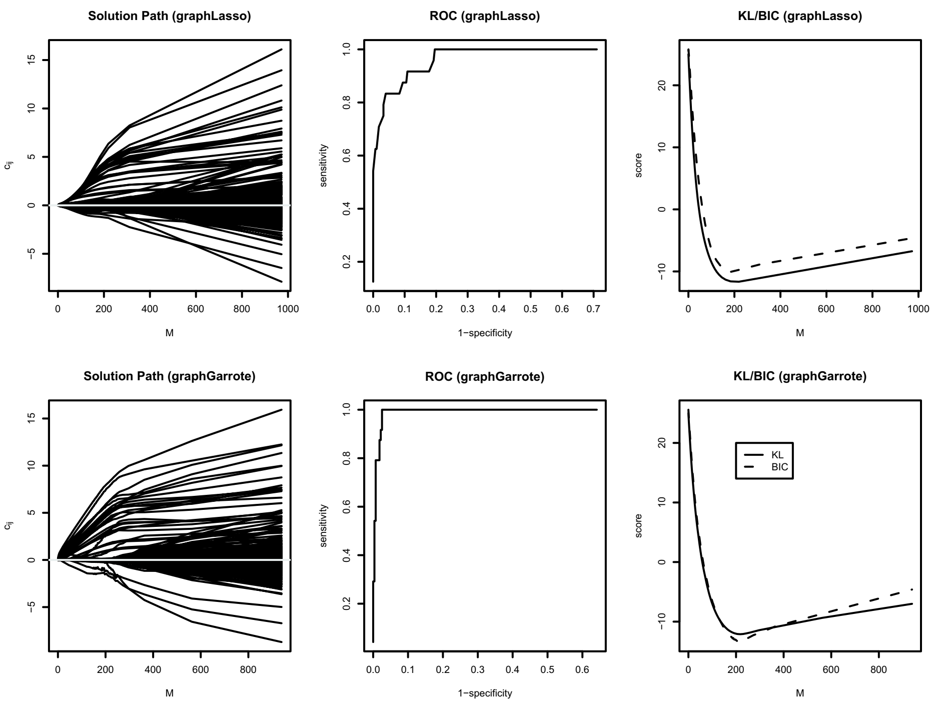

Figure 3. Simulated example ($p = 25$, $n = 100$): solution path, ROC curve, and KL loss and BIC scores. In each plot of the left column, the y-axis corresponds to the value of the off-diagonal entries of the estimated concentration matrix, and the x-axis represents the value of the tuning parameter M. The ROC curves of the plots in the middle column are produced with varying values of the tuning parameter. The two plots from the right column give the BIC and KL scores for different values of the tuning parameters.

is consistent in selecting the graphical model if p is held fixed as n goes to infinity whereas graphLasso may not be. In this example, one may suspect that this is the case. In fact, the model selected by graphGarrote has previously been confirmed using other methods (Edwards, 2000). All these findings agree with those from Yuan and Lin (2007a) obtained using a different but more expensive algorithm.

4.2 Simulated Example

Next we use a simulated example to illustrate the effect of sample size. Let the true concentration matrix be as follows: $c_{ii} = 1$, $c_{i,i-1} = 0.5$, $c_{i,i+1} = 0.5$, and $c_{ij} = 0$ for $|i - j| > 1$. Following Yuan and Lin (2007a), we standardize the population covariance matrix so that its diagonal elements are one. We consider a $p = 25$ dimensional problem. We first generated a sample of size $n = 100$. Using the same computer, our new algorithm took about 10 minutes to construct the whole solution path whereas the interior-point algorithm took nearly an hour to compute the estimate for a single value of the tuning parameter. The lack of efficiency of the existing algorithm makes it difficult to assess the merits of graphLasso and graphGarrote in these relatively high dimensional settings. Us-

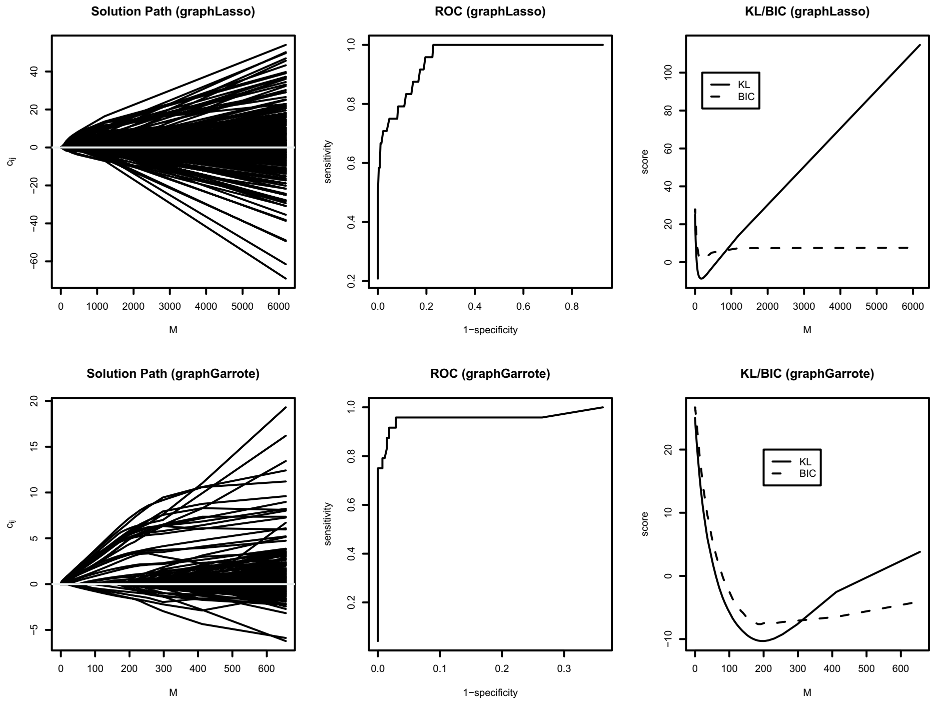

Figure 4. Simulated example ($p = n = 25$): solution path, ROC curve, and KL loss and BIC scores. In each plot of the left column, the y-axis corresponds to the value of the off-diagonal entries of the estimated concentration matrix, and the x-axis represents the value of the tuning parameter M. The ROC curves of the plots in the middle column are produced with varying values of the tuning parameter. The two plots from the right column give the BIC and KL scores for different values of the tuning parameters.

ing the proposed algorithm, we construct the solution paths, which are given in Figure 3. Also plotted are the receiver operating characteristics (ROC) curves, the BIC scores, and the Kullback-Leibler (KL) loss defined as

$$L(\widehat{C}) = -\ln|\widehat{C}| + \mathrm{tr}(\widehat{C}\Sigma). \tag{4.1}$$

In comparison, we simulated data with the same dimension $p = 25$ but only $n = 25$ observations. Clearly in this case, the sample covariance is not invertible and an alternative preliminary estimate is necessary. We first ran the graphLasso solution path. We chose the graphLasso estimate with λ determined by BIC as the initial estimate $\tilde{C}$ for the graphGarrote. Figure 4 provides the same details as Figure 3, but for the smaller dataset.

A few observations can be made from this exercise. First we note that BIC score is very efficient in picking the optimal tuning parameter that minimizes the true KL loss, particularly when the sample size is 100. Yuan and Lin (2007a) proved that in large sample situations, graphGarrote is consistent in model selection if p is held fixed. According to our experience, it also performs well when p is relatively large when compared with n, so long as a good initial estimate is used.

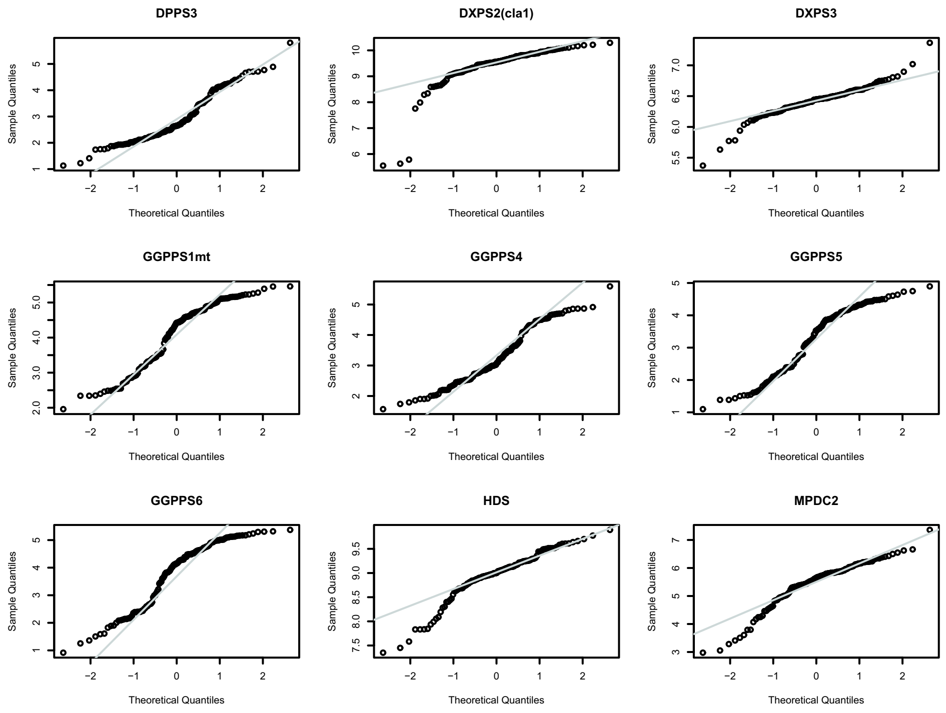

Figure 5. Gene Expression: qqplot for nine genes, each with 118 log transformed gene expression measurements.

5. APPLICATION

This section considers an application of the ℓ_1 regularization to construct the isoprenoid biosynthetic pathway. It is now known that plants contain two pathways for the synthesis of the structural precursors of isoprenoids: the mevalonate (MVA) pathway, located in the cytosol/ER, and the recently discovered methylerythritol 4-phosphate (MEP) pathway, located in the plastids. To better understand the pathway and gain insights into the crosslink between the two pathways, Wille et al. (2004) reported gene expression measurements obtained under various experimental conditions using 118 GeneChip microarrays. They focused on 40 genes, 16 of which were assigned to the cytosolic MVA pathway, 19 to the plastidal MEP pathway and five genes encoding proteins located in the mitochondria. Since the expression measurements are always positive, it is a common practice to assume normality for the log transformed expression. However, the qqplots in Figure 5 show that the normal assumption seems very problematic even after log-transformation for this data.

To avoid the model misspecification, we make a less restrictive assumption that the gene expression obtained from each GeneChip follows a Gaussian copula (Nelsen 1998), meaning that there exist a set of monotone transformation $g_1, \ldots, g_{40}$ so that $\big(g_1(X^{(1)}), \ldots, g_{40}(X^{(40)})\big)'$ follows a 40-dimensional multivariate normal distribution. A particular choice of g_i is $\Phi^{-1}(F_i(\cdot))$ where Φ and F_i are the cumulative distribution func-

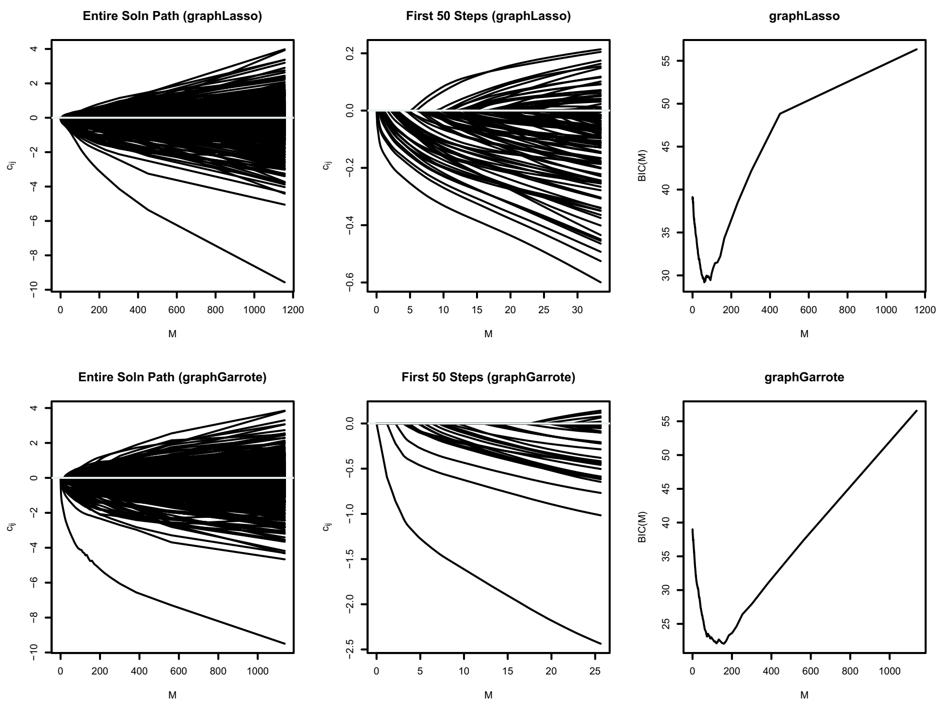

Figure 6. Gene Expression: solution paths and BIC scores. In each plot of the left or middle column, the y-axis corresponds to the value of the off-diagonal entries of the estimated concentration matrix, and the x-axis represents the value of the tuning parameter M. The two plots from the right column give the BIC scores for different values of the tuning parameters.

tion of a standard normal distribution and $X^{(i)}$ respectively. When F_i is unknown, one can replace it with its empirical version. Clearly, the methodology discussed before can be directly applied to $Z \equiv \big(g_1(X^{(1)}), \ldots, g_{40}(X^{(40)})\big)'$. Figure 6 presents the solution path and the trajectory of the BIC scores for graphLasso and graphGarrote.

The models that minimize the BIC criterion contain 271 and 201 edges, respectively, for graphLasso and graphGarrote. To gain more insight into the relationship among genes, we consider the first 25 edges selected by the graphGarrote. We favored graphGarrote for this task because it enjoys superior model-selection properties as shown before. The corresponding graphical model is given in Figure 7. MEP path is given on the left side of the plot and the MVA pathway on the right side. In the plot, the dashed lines and the dotted lines correspond to the known pathway, and the solid lines and the dotted lines correspond to the first 25 edges selected by graphGarrote. The dotted lines are on the known pathway and also selected by garroteGarrote. The result is very similar when compared with the previous analysis of Wille et al. (2004) and Li and Gui (2006).

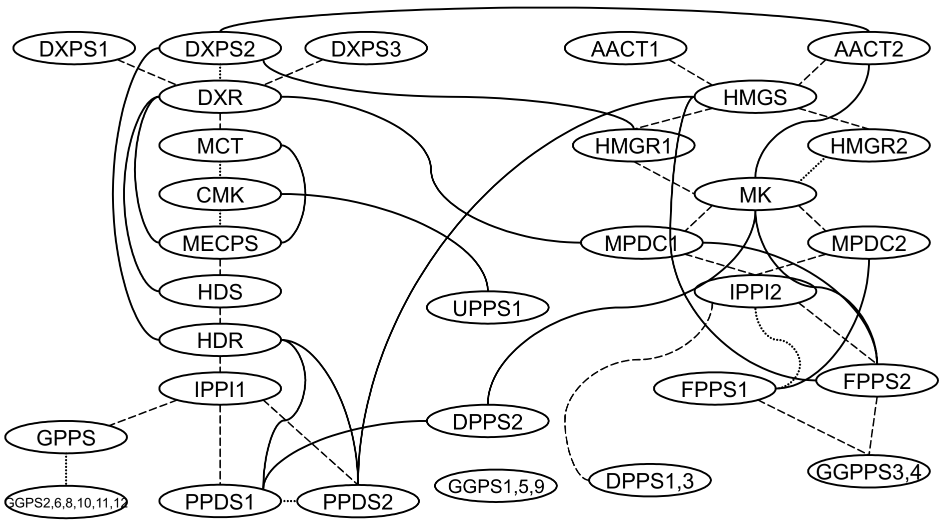

Figure 7. Gene Expression: known pathway and graphical model selected using graphGarrote. The dashed lines and the dotted lines correspond to the known pathway, and the solid lines and the dotted lines correspond to the first 25 edges selected by graphGarrote. The dotted lines are on the known pathway and also selected by garroteGarrote.

6. CONCLUSION

Covariance matrix estimation has always been an important problem in practice for its use in various statistical applications. Increasingly, statisticians face covariance matrix estimation problems in moderate or high dimensions. Traditional maximum likelihood estimate is known to perform poorly in such cases. Using the tool of Gaussian graphical models, graphLasso, and graphGarrote tackle this problem by effectively reducing the dimensionality of the estimation problem. This article considers the efficient computation of graphLasso and graphGarrote.

Despite its nice theoretical properties, these estimates encounter difficult optimization problems. Instead of using the sophisticated interior-point algorithm introduced by Yuan and Lin (2007a), we propose a faster algorithm that can potentially be applied to large scale problems. This algorithm can also be used to efficiently produce an approximate solution path of the ℓ_1 regularization.

ACKNOWLEDGMENTS

This research was supported in part by NSF grant DMS-0624841.

[Received October 2006. Revised October 2007.]

REFERENCES

Boyd, S., and Vandenberghe, L. (2003), *Convex Optimization*, Cambridge: Cambridge University Press.

Dempster, A. P. (1972), “Covariance Selection,” *Biometrika*, 32, 95–108.

Drton, M., and Perlman, M. (2004), “Model Selection for Gaussian Concentration Graphs,” *Biometrika*, 91, 591–602.

Edwards, D. M. (2000), *Introduction to Graphical Modelling*, New York: Springer.

Efron, B., Johnstone, I., Hastie, T., and Tibshirani, R. (2004), “Least Angle Regression,” *The Annals of Statistics*, 32, 407–499.

Lauritzen, S. L. (1996), *Graphical Models*, Oxford: Clarendon Press.

Li, H., and Gui, J. (2006), “Gradient Directed Regularization for Sparse Gaussian Concentration Graphs, With Applications to Inference of Genetic Networks,” *Biostatistcs*, 7, 302–317.

Mardia, K.V., Kent, J.T., and Bibby, J.M. (1979), *Multivariate Analysis*, London: Academic Press.

Meinshausen, N., and Bühlmann, P. (2006), “Consistent Neighbourhood Selection for High-Dimensional Graphs with the Lasso,” *The Annals of Statistics*, 34, 1436–1462.

Nelsen, R. (1998), *An Introduction to Copulas*, New York: Springer-Verlag.

Osborne, M.R., Presnell, B., and Turlach, B.A. (2000), “A New Approach to Variable Selection in Least Squares Problems,” *IMA Journal of Numerical Analysis*, 20, 389–403.

Park, M., and Hastie, T. (2007), “L_1-Regularized Path Algorithm for Generalized Linear Models,” *Journal of the Royal Statistical Society*, Series B, 69, 659–677.

Tibshirani, R. (1996), “Regression Shrinkage and Selection via the Lasso,” *Journal of the Royal Statistical Society*, Series B, 58, 267–288.

Vandenberghe, L., Boyd, S., and Wu, S.-P., (1998), “Determinant Maximization with Linear Matrix Inequality Constraints,” *SIAM Journal on Matrix Analysis and Applications*, 19, 499–533.

Whittaker, J. (1990), *Graphical Models in Applied Multivariate Statistics*, Chichester: Wiley.

Wille, A., Zimmermann, P., Vranova, E., Furholz, A., Laule, A., Bleuler, S., Hennig, L., Prelic, A., von Rohr, P., Thiele, L., Zitzler, E., Gruissem, W., and Buhlmann, P. (2004), “Sparse Gaussian Graphical Modelling of the Isoprenoid Gene Network in *Arabidopsis thaliana*,” *Genome Biology*, 5, R92.

Wu, S.-P., Vandenberghe, L., and Boyd, S. (1996), “Software for Determinant Maximization Problems—User’s Guild,” available online at *http://www.stanford.edu/~boyd/maxdet*.

Yuan, M., and Lin, Y. (2007a), “Model Selection and Estimation in the Gaussian Graphical Model,” *Biometrika*, 94, 19–35.

——— (2007b), “On the Nonnegative Garrote Estimator,” *Journal of the Royal Statistical Society*, Series B, 69, 143–161.