17 Optimization Methods for Sparse Inverse Covariance Selection

Katya Scheinberg katyas@lehigh.edu
Department of Industrial and Systems Engineering
Lehigh University
Bethlehem, PA 18015-1582

Shiqian Ma sm2756@columbia.edu
Department of Industrial Engineering and Operations Research
Columbia University
New York, NY 10027

17.1 Introduction

In many practical applications of statistical learning the objective is not simply to construct an accurate predictive model, but rather to discover meaningful interactions among the variables. For example, in applications such as reverse engineering of gene networks, discovery of functional brain connectivity patterns from brain-imaging data, and analysis of social interactions, the main focus is on reconstructing the network structure representing dependencies among multiple variables, such as genes, brain areas, and individuals. Probabilistic graphical models, such as Markov networks (or Markov random fields), provide a statistical tool for multivariate data analysis that allows the capture of interactions such as conditional independence relationships between variables. We focus on the task of learning the structure of a Markov network over Gaussian random variables, which is equivalent to learning the zero pattern of the inverse covariance matrix. A standard approach is to choose the sparsest network (inverse covariance matrix) that adequately explains the data. This can be achieved by solving a regularized maximum likelihood problem with the regularization term involving the number of nonzeros (ℓ_0-norm) in the inverse covariance matrix—

a generally intractable problem that is often solved approximately by greedy search (Heckerman, 1995). Recently, however, novel tractable approximations have been suggested that exploit the sparsity-enforcing property of ℓ_1-norm regularization and yield convex optimization problems (Meinshausen and Buhlmann, 2006; Wainwright et al., 2007; Yuan and Lin, 2007; Banerjee et al., 2008; Friedman et al., 2007). In this chapter we focus on one such convex formulation, often referred to as sparse inverse covariance selection (SICS), and describe several optimization approaches to this problem.

17.1.1 Problem Formulation

Let $\mathcal{S}$ be a set of p random variables with joint distribution $P(\mathcal{S})$. It is common to assume a multivariate Gaussian probability density function over $\mathcal{S}$, hence, if $\mathbf{s}$ is a p-dimensional vector which is a realization of p random variables, then

$$p(\mathbf{s}) = (2\pi)^{-p/2} \det(\Sigma)^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{s}-\mu)^T \Sigma^{-1}(\mathbf{s}-\mu)}, \tag{17.1}$$

where μ is the mean and Σ is the covariance matrix of the distribution, respectively. Without loss of generality we assume that the data are centered so that $\mu = 0$; hence the purpose is to estimate Σ. Introducing $X := \det(\Sigma)^{-1}$, we can rewrite (17.1) as

$$p(\mathbf{s}) = (2\pi)^{-p/2} \det(X)^{\frac{1}{2}} e^{-\frac{1}{2}\mathbf{s}^\top X \mathbf{s}}. \tag{17.2}$$

Missing edges in the above graphical model correspond to zero entries in the inverse covariance matrix X, and vice versa (Lauritzen, 1996), and thus the problem of structure learning for the above probabilistic graphical model is equivalent to the problem of learning the zero pattern of the inverse covariance matrix. Note that the maximum likelihood estimate of the covariance matrix Σ is the empirical covariance matrix $S = \frac{1}{n}\sum_{i=1}^{n} \mathbf{s}_i \mathbf{s}_i^\top$ where $\mathbf{s}_i$ is the ith sample, $i = 1, ..., n$. The inverse of S, even if it exists, does not typically contain any elements that are exactly zero. Therefore an explicit sparsity-enforcing constraint needs to be added to the estimation process.

A common approach is to include the (vector) ℓ_1-norm of X as a penalty term in the objective function, which is equivalent to imposing a Laplace prior on Σ^{-1} in a maximum likelihood framework (Friedman et al., 2007; Banerjee et al., 2008; Yuan and Lin, 2007). Formally, the entries X_{ij} of the inverse covariance matrix are assumed to be independent random variables, each following a Laplace distribution

$$p(X_{ij}) = \frac{\lambda}{2} e^{-\lambda |X_{ij} - \alpha_{ij}|} \tag{17.3}$$

with zero location parameter (mean) α_{ij}, yielding

$$p(X) = \prod_{i=1}^{p}\prod_{j=1}^{p} p(X_{ij}) = (\lambda/2)^{p^2} e^{-\lambda||X||_1}, \tag{17.4}$$

where $||X||_1 = \sum_{ij} |X_{ij}|$ is the (vector) ℓ_1-norm of X. Then the objective is to find the maximum log-likelihood solution $\arg\max_{X\succ 0} \log p(X|\mathbf{S})$, where $\mathbf{S}$ is the $n \times p$ data matrix whose rows are given by $\mathbf{s}^\top$, and $X \succ 0$ denotes that X is positive definite. Invoking Bayes's rule, $p(X|\mathbf{S}) = P(\mathbf{S}|X)P(X)/p(\mathbf{S})$, this max-likelihood estimate can be obtained by

$$\arg\max_{X\succ 0} \quad \log \prod_{i=1}^{n} [\frac{\det(X)^{\frac{1}{2}}}{(2\pi)^{p/2}} e^{-\frac{1}{2}\mathbf{s}_i^\top X \mathbf{s}_i}] + \log[(\lambda/2)^{p^2} e^{-\lambda||X||_1}]. \tag{17.5}$$

We write $\frac{1}{n}\sum_{i=1}^{n} \mathbf{s}_i^\top X \mathbf{s}_i = \langle S, X\rangle$. This yields the following optimization problem (also see Friedman et al. (2007); Banerjee et al. (2008); Yuan and Lin (2007)):

$$\max_{X\succ 0} \quad \log\det(X) - \langle S, X\rangle - \rho||X||_1, \tag{17.6}$$

where $\rho = \frac{2}{n}\lambda$.

More generally, one can consider the following formulation:

$$\max_{X\succ 0} \quad \log\det(X) - \langle S, X\rangle - \sum_{ij} M_{ij}|X_{ij}|. \tag{17.7}$$

If M is a product of $\rho = \frac{2}{n}\lambda$ and E is the matrix of all ones, then problem formulation (17.7) reduces to (17.6).

Note that by allowing the matrix M to have arbitrary nonnegative entries, we automatically include in the formulation the case where the diagonal elements of X are not penalized or the case when the absolute values of the entries of X are scaled by their estimated value, as was considered in Yuan and Lin (2007).

We will refer to (17.6) and (17.7) as the SICS problem. The two formulations are very similar in terms of optimization effort. In some cases, for brevity we present methods for (17.6) and explain its extension for (17.7) afterward.

The dual of (17.7) can be written as (Banerjee et al., 2008)

$$\max_{W\succ 0}\{\log\det(W) - p : \text{ s.t. } -M \leq W - S \leq M\}, \tag{17.8}$$

where the inequalities involving matrices W, S, and M are element-wise.

Problems (17.6) and (17.7) are both strictly convex, and as long as the off-diagonal elements of M are positive, the optimal solution is always attained.

The optimality conditions for this pair of primal and dual problems imply that $W = X^{-1}$ and that $W_{ij} - S_{ij} = M_{ij}$ if $X_{ij} > 0$ and $W_{ij} - S_{ij} = -M_{ij}$ if $X_{ij} < 0$. These optimality conditions are imperative when the primal sparsity structure needs to be recovered from the dual solution. Several existing methods solve (17.8) and obtain an approximate solution to (17.7) by inverting the solution of (17.8). Since the obtained dual solution is also approximate, the resulting primal approximate solution is not sparse. However, it is often the case that the dual solution is an accurate projection onto the dual feasible set. Hence, for the "true" nonzero elements of X_{ij} the appropriate complementarity conditions $W_{ij} - S_{ij} = M_{ij}$ and $W_{ij} - S_{ij} = -M_{ij}$ are observed from the solution obtained for (17.8).

17.1.2 Overview of Optimization Approaches

Problems (17.6) and (17.7) are special cases of a semidefinite programming problem (SDP) (Wolkowicz et al., 2000) which can be solved in polynomial time by interior-point methods (IPM). However, as is well known, each iteration of an IPM applied to a semidefinite programming problem of size p requires up to $O(p^6)$ operations and $O(p^4)$ memory space, which is very costly. Although an approximate IPM has recently been proposed for the SICS problem (Li and Toh, 2010), another reason that using an IPM is undesirable for our problem is that an IPM does not produce the sparsity pattern of the solution. The sparsity is, in theory, recovered in the limit. In practice it is recovered by thresholding the elements of an approximate solution; hence, numerical inaccuracy can interfere with the structure recovery.

As an alternative to IPMs, more efficient approaches, COVSEL and glasso, were developed for problem (17.6) in Banerjee et al. (2008) and (Friedman et al., 2007). The methods are similar in that they are based on applying a block coordinate descent (BCD) method to the dual of (17.6). At each iteration only one row (and the corresponding symmetric column) of the dual matrix is optimized while the rest of that matrix remains fixed. The resulting subproblem is a convex quadratic problem. The difference between the COVSEL method, described in Banerjee et al. (2008), and the glasso method in Friedman et al. (2007), is that COVSEL solves the subproblems via an interior-point approach, while glasso poses the subproblem as a dual of the Lasso problem (Tibshirani, 1996) and utilizes a coordinate descent (CD) approach (Tibshirani, 1996) to solve the resulting Lasso subproblem. Due to the use of the coordinate descent, the sparsity of the primal matrix is recovered more accurately than with the interior-point approach, and the glasso method (Friedman et al., 2007) is faster than COVSEL because it takes advantage of that sparsity. On the other hand, subproblems solved by glasso are solved by coordinate descent up to an unknown accuracy; hence, the current implementation of this method lacks rigor. A recent row-by-row (RBR) method for general SDP (Wen et al., 2009) is based on the same idea of updating one row and column at a time, like glasso and COVSEL, but can also be applied directly to the primal matrix. The resulting subproblem is in general a second-order cone problem (SOCP), while in the dual case it reduces to a convex quadratic problem (QP). It follows from the result in Wen et al. (2009) that the BCD approach converges to the optimal solution when applied to the primal and dual SICS formulations. The SINCO method proposed by Scheinberg and Rish (2009) is a greedy coordinate descent method applied to the primal problem. We will describe the approaches of glasso and SINCO in more detail below. Sun et al. (2009) propose solving the primal problem (17.6) by using a BCD method. They formulate the subproblem as a min-max problem and solve it using a prox-method proposed by Nemirovski (2005). All of these BCD and CD approaches lack iteration complexity bounds as of now.

As alternatives to block coordinate based approaches several gradient based approaches, for problem (17.6) have been suggested. A projected gradient method for solving the dual problem (17.8) was proposed by Duchi et al. (2008). Variants of Nesterov's method (Nesterov, 2005, 2004) were applied to solving the SICS problem by d'Aspremont et al. (2008) and Lu (2009, 2010). d'Aspremont et al. (2008) apply Nesterov's optimal first-order method to solve the primal problem (17.6) after smoothing the nonsmooth ℓ_1-term, obtaining an iteration complexity bound of $O(1/\varepsilon)$ for an ε-optimal solution. Lu solves the dual problem (17.8), which is a smooth problem, by Nesterov's algorithm, and improves the iteration complexity to $O(1/\sqrt{\varepsilon})$. However, since the practical performance of this algorithm is not attractive, Lu gives a variant of it (VSM) with unknown complexity that exhibits better performance. Yuan (2009) proposes an alternating direction method based on an augmented Lagrangian framework. Goldfarb et al. (2009) and Scheinberg et al. (2010) have developed an alternating linearization method with $O(1/\sqrt{\varepsilon})$ complexity which is similar to the augmented Lagrangian approach. A proximal point algorithm was proposed by Wang et al. (2009) which requires a reformulation which increases the size of the problem. The IPM in Li and Toh (2010) also requires such a reformulation.

In this chapter we give details of BCD approaches developed in Friedman et al. (2007) and Scheinberg and Rish (2009). The fundamental difference between the two approaches is in the choice of the next coordinates that are updated. From the class of the first-order methods we choose to present the alternating linearization method from Goldfarb et al. (2009) and Scheinberg et al. (2010), which currently appears to be the most efficient approach for SICS (see Scheinberg et al. (2010) for computational comparison).

17.1.3 Preliminaries

17.1.3.1 Properties of Matrix Determinant and Inverse

Let $X \in S^n_{++}$ be a positive definite matrix. We will list here a few useful properties from linear algebra.

Let X be partitioned as

$$X = \begin{pmatrix} \xi & y^\top \\ y & B \end{pmatrix}$$

where $\xi \in \mathbb{R}$, $y \in \mathbb{R}^{n-1}$, and $B \in S^{n-1}_{++}$. Then

$$\det X = (\xi - y^\top B^{-1} y) \det B. \tag{17.9}$$

Consider a matrix $X + uv^\top$, where $u, v \in \mathbb{R}^n$, then

$$\det(X + uv^\top) = \det(X)(1 + v^\top X^{-1} u) \tag{17.10}$$

and

$$(X + uv^\top)^{-1} = X^{-1} - X^{-1} uv^\top X^{-1}/(1 + v^\top X^{-1} u). \tag{17.11}$$

The last expression is well known as the Sherman-Morrison-Woodbury formula.

17.1.3.2 Soft Thresholding and Shrinkage

We will use the well-known fact that the solution to the following optimization problem,

$$\min_x \frac{1}{2}\|x - r\|_2^2 + \lambda \|x\|_1, \tag{17.12}$$

where x and r are vectors in $\mathbb{R}^n$, can be obtained in closed form by a shrinkage operator $x^* = \text{shrink}(r, \lambda)$, where

$$x_i^* = \text{shrink}(r_i, \lambda) = \begin{cases} r_i - \lambda & \text{if } r_i \geq \lambda \\ 0 & \text{if } -\lambda < r_i < \lambda \\ r_i + \lambda & \text{if } r_i \leq -\lambda. \end{cases} \tag{17.13}$$

17.1.3.3 Coordinate Descent for Lasso

Let us consider a general Lasso problem (Tibshirani, 1996),

$$\min_x \frac{1}{2}\|Ax-b\|_2^2 + \rho\|x\|_1, \tag{17.14}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$. The idea of a coordinate descent approach is at each step to fix all elements of x except for the ith element and to optimize the objective function for only one variable x_i. Let $\bar{x}$ denote the fixed part of vector x, and $\bar{A}$ the part of matrix A that corresponds to $\bar{x}$. Writing the reduced problem omitting the terms that do not depend on x_i, we get

$$\min_{x_i} \frac{1}{2}x_i^\top A_i^\top A_i x_i + \bar{x}^\top \bar{A}^\top A_i x_i - b^\top A_i x_i + \rho|x_i|, \tag{17.15}$$

where A_i is the ith column of A. Let $r_i = -A_i^\top(\bar{A}\bar{x} - b)/\|A_i\|^2$ and $\eta_i = \rho/\|A_i\|^2$; then problem (17.15) is equivalent to

$$\min_{x_i} \frac{1}{2}(x_i - r_i)^2 + \eta_i|x_i|, \tag{17.16}$$

which is solved by the soft thresholding step $x_i^* = \text{shrink}(r_i, \eta_i)$. Hence, each step of coordinate descent for Lasso requires computing r_i and η_i. The cost of this computation depends on the manner in which different parts of the expression for r are stored and updated. For instance, if the elements of $A^\top A$ are precomputed and stored, then at each step all r_i's can be updated in $O(n)$ operations. Alternatively, the residual $Ax - b$ can be stored and updated at the cost of $O(m)$ storage and operations, in which case each r_i can be computed in $O(m)$ operations whenever it is required.

17.2 Block Coordinate Descent Methods

17.2.1 Row-by-Row Method

For simplicity let us consider the case when $M = \rho E$, that is, formulation (17.6). Instead of solving (17.6) directly, the approaches in Banerjee et al. (2008) and Friedman et al. (2007) consider the dual

$$\max_{W\succ 0}\{\log\det(W) - p : \text{ s.t. } \|W - S\|_\infty \le \rho\}. \tag{17.17}$$

The subproblems solved at each iteration of the BCD methods in Banerjee et al. (2008) and Friedman et al. (2007) are constructed as follows. Given a positive definite matrix $W \succ 0$, W and S are partitioned conformally as

$$W = \begin{pmatrix} \xi & y^\top \\ y & B \end{pmatrix} \text{ and } S = \begin{pmatrix} \xi_S & y_S^\top \\ y_S & B_S \end{pmatrix},$$

where $\xi, \xi_S \in \mathbb{R}$, $y, y_S \in \mathbb{R}^{n-1}$, and $B, B_S \in S^{n-1}$. It follows from (17.9) that $\log\det W = \log(\xi - y^\top B^{-1} y) + \log\det B$, and B is fixed, so the BCD subproblem for (17.6) becomes the quadratic program

$$\min_{[\xi;y]} \; y^\top B^{-1} y - \xi, \quad \text{s.t.} \quad \|[\xi; y] - [\xi_S; y_S]\|_\infty \le \rho, \; \xi \ge 0. \tag{17.18}$$

Note that (17.18) is separable in y and ξ. The solution ξ is equal to $\xi_S + \rho$. In fact, the first-order optimality conditions of (17.6) and $X \succ 0$ imply that $W_{ii} = S_{ii} + \rho$ for $i = 1, \ldots, n$. Hence, problem (17.18) reduces to

$$\min_{y} \; y^\top B^{-1} y, \quad \text{s.t.} \quad \|y - y_S\|_\infty \le \rho. \tag{17.19}$$

The BCD method in Banerjee et al. (2008) solves a sequence of constrained problems (17.19). After each step the duality gap for (17.6) at the current iterate W^k can be obtained as $\langle (W^k)^{-1}, S\rangle - p + \rho\|(W^k)^{-1}\|_1$. The BCD method proposed in Banerjee et al. (2008) for solving (17.17) is outlined in algorithm 17.1.

Algorithm 17.1 Block coordinate descent method for (17.17)

```
1: Set $W^1 = S + \rho I$, $k := 1$, and $\varepsilon \ge 0$.
2: while $\langle (W^k)^{-1}, S\rangle - p + \rho\|(W^k)^{-1}\|_1 \ge \varepsilon$ do
3:     for $i = 1, \cdots, p$ do
4:         Set $B := W^k_{i^c,i^c}$, $y_S = S_{i^c,i}$, and $B_S = S_{i^c,i^c}$.
5:         Solve (17.19) to get $y$.
6:         Update $W^k_{i^c,i} := y$ and $W^k_{i,i^c} := y^\top$.
7:     end for
8:     Set $W^{k+1} := W^k$ and $k := k + 1$.
9: end while
```

Each instance of (17.19) is solved by applying an interior-point quadratic programming solver. Also, the inverse of W^k is computed at each iteration. This makes each iteration of the BCD costly. Moreover, the sparsity of the primal solution $X^k = (W^k)^{-1}$ is not exploited, and is obtained only in the limit. Since the BCD approach typically does not produce accurate solutions, this sparsity is hard to recover accurately.

The glasso method proposed in Friedman et al. (2007) is based on algorithm 17.1 except for the method of solving (17.19). Specifically, it can be easily verified that the dual of (17.19) is

$$\min_x \; x^\top Bx - y_S^\top x + \rho\|x\|_1, \tag{17.20}$$

which is also equivalent to

$$\min_x \; \left\|B^{\frac{1}{2}}x - \tfrac{1}{2}B^{-\frac{1}{2}}y_S\right\|_2^2 + \rho\|x\|_1. \tag{17.21}$$

If x solves (17.21), then $y = Bx$ solves (17.19).

Problem (17.21) is equivalent to the Lasso problem (Tibshirani, 1996). The Lasso problem is then solved using a coordinate descent algorithm, which does not require computation of either $B^{\frac{1}{2}}$ or $B^{-\frac{1}{2}}$. Indeed, if we recall the subproblem (17.16) which is generated and solved at each coordinate descent step for (17.14) we see that to compute the scalars r_i and η_i we require only elements of B and y, since $B_i^{\frac{1}{2}}B_i^{\frac{1}{2}} = B_i$ and $B_i^{\frac{1}{2}}B_i^{-\frac{1}{2}}y_i = y_i$.

For each BCD iteration, the resulting solution of the Lasso subproblem (17.21) has the same nonzero pattern as the corresponding row of $X^k = (W^k)^{-1}$, which can be viewed as the current estimate of the inverse of the covariance. Hence, the sparsity pattern is explicitly available and is exploited by the glasso algorithm. This makes glasso significantly more efficient than COVSEL (developed in Banerjee et al. (2008)). In terms of CPU time glasso is currently the most efficient block coordinate descent approach to the SICS problem. On the other hand, it is unclear if the convergence results of BCD apply to glasso due to the way the subproblems are addressed in the implementation.

It is easy to extend the BCD approach to the more general formulation (17.7). One needs to consider the appropriate partitioning of M,

$$M = \begin{pmatrix} \xi_M & y_M^\top \\ y_M & B_M \end{pmatrix}.$$

The bound constraints in (17.19) become $-y_M \le (y - y_S) \le y_M$ and $\xi = \xi_s + \xi_M$. Both glasso and COVSEL easily extend to this formulation.

The BCD approach cycles through each row/column of W one by one. Hence the nonzeros in the inverse covariance matrix generated by *glasso* are generated by rows. This may introduce small nonzero elements, which are undesirable in the solution. It is possible to modify this method so that only one element in each row is updated at a time, thus bringing the whole approach closer to the simple coordinate descent used for Lasso. However, it is unclear how to efficiently select the next working variable to be updated. A simple cycling rule would be equivalent to the approach in Friedman et al. (2007) which we just described. The method in the next subsection is a primal greedy coordinate descent, which addresses the issue of the working variable selection.

17.2.2 Primal Greedy Coordinate Descent

We now describe an algorithm which addresses the primal problem directly and also uses coordinate descent,[1] which naturally preserves the sparsity of the solution. The method is referred to as SINCO (Sparse INverse COvariance) and is introduced in Scheinberg and Rish (2009). Unlike the dual BCD approach of COVSEL, glasso, and the general RBR, SINCO optimizes only one diagonal or two (symmetric) off-diagonal entries of the matrix X at each step. The advantages of this approach are that only one nonzero entry (discounting the matrix symmetry) can be introduced at each step, and that the solution to each subproblem is available in closed form as a root of a quadratic equation. Computation at each step requires a constant number of arithmetic operations, independent of p. Hence, in $O(p^2)$ operations a potential step can be computed for *all pairs* of symmetric elements (i.e., for all pairs (i, j)). Then the step which provides the *best* objective function value improvement can be chosen, which is the essence of the greedy nature of this approach. Once the step is taken, the update of the gradient information requires $O(p^2)$ operations. Hence, overall, each iteration takes $O(p^2)$ operations. Note that each step is also suitable for massive parallelization.

In comparison, glasso and COVSEL (and RBR) require solution of a quadratic programming problem whose theoretical and empirical complexity varies depending on the method used, but always exceeds $O(p^2)$. These algorithms apply optimization to each row/column consecutively; hence the greedy nature is lacking. On the other hand, a whole row and a whole column are optimized at each step, thus reducing the overall number of steps. As is shown in Scheinberg and Rish (2009), SINCO, in a serial mode, is comparable to glasso, which is orders of magnitude faster than COVSEL, according to the results in Friedman et al. (2007). Also, SINCO may lead to a lower false-positive error than glasso since it introduces nonzero elements greedily. On the other hand, it may suffer from introducing too few nonzeros and thus misses some of the true positives, especially on dense networks.

Perhaps the most interesting consequence of SINCO's greedy nature is that it reproduces the regularization path behavior while using only one value of the regularization parameter ρ. We will discuss this property further

1. We should call it "coordinate ascent" since we are solving a maximization problem; however, we use the word "descent" to adhere to standard terminology.

after we describe the algorithm.

17.2.2.1 Algorithm Description

The main idea of the method is that at each iteration, the matrix X is updated by changing one element on the diagonal or two symmetric off-diagonal elements. This implies the change in X that can be written as $X + \theta(e_i e_j^\top + e_j e_i^\top)$, where i and j are the indices corresponding to the elements that are being changed. The key observation is that given the matrix $W = X^{-1}$, the exact line search that optimizes the objective function of problem (17.7) along the direction $e_i e_j^\top + e_j e_i^\top$ reduces to a solution of a quadratic equation, as we will show below. Hence, each such line search takes a constant number of operations. Moreover, given the starting objective value, the new function value on each step can be computed in a constant number of steps. This means that we can perform such line search for all (i, j) pairs in $O(p^2)$ time, which is linear in the number of unknown variables X_{ij}. We then can choose the step that gives the best improvement in the value of the objective function. After the step is chosen, the dual matrix $W = X^{-1}$ and, hence, the objective function gradient are updated in $O(p^2)$ operations.

We now describe the method. For a fixed pair (i, j) consider now the update of X of the form $X(\theta) = X + \theta(e_i e_j^\top + e_j e_i^\top)$, such that $X \geq 0$. Let us consider the objective function as the function of θ:

$$f(\theta) = \log\det(X + \theta e_i e_j^\top + \theta e_j e_i^\top) - \tag{17.22}$$

$$\langle S, X + \theta e_i e_j^\top + \theta e_j e_i^\top\rangle - \sum_{i,j=1}^{p} M_{ij}|X + \theta e_i e_j^\top + \theta e_j e_i^\top|. \tag{17.23}$$

We use the property of the determinant (17.10) and the Sherman-Morrison-Woodbury formula (17.11) to obtain

$$\begin{aligned} & \det(X + \theta e_i e_j^\top + \theta e_j e_i^\top) = \det(X + \theta e_j e_i^\top)(1 + \theta e_j^\top (X + \theta e_j e_i^\top)^{-1} e_i) \\ = \ & \det(X)(1 + \theta e_i^\top X^{-1} e_j)(1 + \theta e_j^\top X^{-1} e_i - \theta^2 e_j^\top X^{-1} e_j (1 + \theta e_i^\top X^{-1} e_j)^{-1} e_i^\top X^{-1} e_i) \\ = \ & \det(X)(1 + 2\theta e_i^\top X^{-1} e_j + (\theta e_j^\top X^{-1} e_i)^2 - \theta^2 e_i^\top X^{-1} e_i e_j^\top X^{-1} e_j). \end{aligned}$$

Given the dual solution $W = X^{-1}$, we can write the above as

$$\det(X + \theta e_i e_j^\top + \theta e_j e_i^\top) = \det(X)(1 + 2\theta W_{ij} + \theta^2(W_{ij}^2 - W_{ii}W_{jj})).$$

We define the function $g(\theta)$ as

$$g(\theta) := 1 + 2\theta W_{ij} + \theta^2(W_{ij}^2 - W_{ii}W_{jj}).$$

Recalling that W and S are symmetric, but M is not necessarily so, maximizing the objective function $f(\theta)$ over θ is equivalent to

$$\max_{\theta} \log(g(\theta)) - 2S_{ij}\theta - (M_{ij} + M_{ji})|X_{ij} + \theta|. \tag{17.24}$$

This problem can be rewritten as

$$\max_{\theta} \min_{u \in \mathbb{R}\colon |u| \leq M_{ij}+M_{ji}} \log(g(\theta)) - 2S_{ij}\theta - u(X_{ij} + \theta). \tag{17.25}$$

Swapping the min and the max, we have as the inner problem

$$\max_{\theta} f_u(\theta) = \log g(\theta) - 2S_{ij}\theta - u(X_{ij} + \theta), \tag{17.26}$$

which can be solved by setting the derivative of the objective with respect to θ to zero.

$$f'_u(\theta) = \frac{g'(\theta)}{g(\theta)} - 2S_{ij} - u = \frac{W_{ij} + \theta(W_{ij}^2 - W_{ii}W_{jj})}{\theta^2(W_{ij}^2 - W_{ii}W_{jj}) + 1 + 2\theta W_{ij}} - 2S_{ij} - u.$$

To find the maximum of $f(\theta)$, we need to find θ for which $f'(\theta) = 0$. Letting a denote $W_{ii}W_{jj} - W_{ij}^2$, this condition can be written as

$$W_{ij} - 2S_{ij} - u - (a + 2W_{ij}(2S_{ij} + u)\theta + a(2S_{ij} + u)\theta^2 = 0.$$

If we know the value of u, we can obtain the optimal θ from the above quadratic equation. From the optimality conditions for the primal-dual problem (17.25) we know that for optimal θ

$$\begin{array}{ll} X_{ij} + \theta \geq 0 & \text{if } u = M_{ij} + M_{ji} \\ X_{ij} + \theta \leq 0 & \text{if } u = -M_{ij} - M_{ji} \\ X_{ij} + \theta = 0 & \text{if } -M_{ij} - M_{ji} < u < M_{ij} + M_{ji}. \end{array}$$

Hence, considering the three different scenarios, one can find the optimal solution to (17.25) via solving quadratic equations in a constant number of steps, which does not depend on p. In Scheinberg and Rish (2009) the details of solving the quadratic equation are given along with the proof that the solution corresponding to the maximum of $f(\theta)$ always exists.

Once the optimal θ is computed, the objective function improvement is easily obtained from $f(\theta) - f(0)$, where $f(0)$ is the objective function value from the previous iteration. Since for each (i, j) pair the optimal θ and the objective function value improvement can be computed in a constant number of operations, then in $O(p^2)$ operations one can compute the (i, j) pair, which gives the largest objective function improvement. Once such a pair is determined, the appropriate update to X can be performed and the iteration is completed.

The inverse $\bar{W}$ of $\bar{X} = X + \theta e_i e_j + \theta e_j e_j$ is obtained, according to the Sherman-Morrison-Woodbury formula, in $O(p^2)$ operations as follows:

$$\begin{cases} \bar{W} &= W - \theta(\kappa_1 W_i W_j^\top + \kappa_2 W_i W_i^\top + \kappa_3 W_j W_j^\top + \kappa_1 W_j W_i^\top) \\ \kappa_1 &= -(1+\theta W_{ij})/\kappa \\ \kappa_2 &= \theta W_{jj}/\kappa \\ \kappa_3 &= \theta W_{ii}/\kappa \\ \kappa &= \theta^2(W_{ii}W_{jj} - W_{ij}^2) - 1 - 2\theta W_{ij} \end{cases} \tag{17.27}$$

We outline the steps of the SINCO method in algorithm 17.2.

Algorithm 17.2 Coordinate descent method for (17.7)

```
1: Set $X^1 = I$, set $k := 1$, $\theta^k = 0$, and $\varepsilon \geq 0$. Compute $f(X^1)$.
2: while $f(\theta^k) - f(X^k) \geq \varepsilon$ do
3:     for $i = 1, \cdots, p$, $j = i, \cdots, p$ do
4:         Compute $\theta_{ij} = \arg\max_\theta f(\theta)$ where $f(\theta)$ is defined by (17.22)
5:         If $f(\theta^k) > f(\theta_{ij})$, $\theta^k = \theta_{ij}$, $(i,j)^k = (i,j)$
6:
7:     end for
8:     Update $W^k$ according to (17.27) for $(i,j) = (i,j)^k$.
9: end while
```

The overall per-iteration complexity of SINCO is $O(p^2)$. Moreover, this algorithm lends itself readily to massive parallelization. Indeed, at each iteration of the algorithm the step computation for each (i, j) pair can be parallelized, and the procedure that updates W involves simply adding to each element of W a function that involves only two rows of W. Hence, the updates can be also done in parallel, and in very large-scale cases the matrix W can also be stored in a distributed manner. The same is true of the storage of matrices S and M (assuming that M needs to be stored, that is, not all elements of M are the same), while the best way to store the X matrix may be in sparse form.

The convergence of the method follows from the convergence of a block coordinate descent method on a strictly convex objective function, as is shown for the RBR method in Wen et al. (2009). The only constraints are box constraints (nonnegativity), and they do not hinder the convergence. In the case of SINCO we extensively use the fact that each coordinate descent step is cheap and, unlike the glasso algorithm, we select the next step based on the best function value improvement. On the other hand, we maintain both the primal matrix X and the inverse matrix W, while glasso method does not. However, none of these differences prevent the convergence result for RBR in Wen et al. (2009) to apply to both methods.

17.2.3 Regularization Path

One of the main challenges in sparse inverse covariance selection is the proper choice of the weight matrix M in (17.6). Typically M is chosen to be a multiple of the matrix of all ones as in the formulation (17.6). The multiplying coefficient ρ is called the regularization parameter. Clearly, for large values of ρ as $\rho \to \infty$, the solution to (17.6) is likely to be very sparse and eventually diagonal, which means that no structure recovery is achieved. On the other hand, if ρ is small as $\rho \to 0$, the solution X is likely to be dense and eventually approach S^{-1} and, again, no structure recovery occurs. Hence, exploration of a regularization path is an integral part of the sparse inverse covariance selection.

Typically, problem (17.6) is solved for several values of ρ in a predefined range and the best value, according to some criteria, is selected. The reason only a scalar parameter ρ is usually considered is that it is expensive to explore solutions along a multi-dimensional grid.

The work in Krishnamurthy and d'Aspremont (2009), also presented in this volume, addresses an algorithm for the SICS problem that computes the entire regularization path by a path-following method.

In this section, we concentrate on computing the regularization path (or some parts of it) by solving the SICS problem for a finite range of values of ρ rather than the complete path. All methods described in this chapter, including the alternating linearization method described below, are very well suited for the efficient computation of the regularization path, since it directly exploits warm starts. When ρ is relatively large, a very sparse solution can be obtained quickly. This solution can be used as a warm start to the problem with a smaller value of ρ and, if the new value of ρ is not too small compared with the previous value, then the new solution is typically obtained in just a few iterations, because the new solution has only a few extra nonzero elements.

The regularization path is evaluated via the ROC curves showing the trade-off between the number of true positive (TP) elements recovered and the number of false positive (FP) elements. Producing better curves (where the number of TPs rises fast relative to FPs) is usually an objective of any method that does not focus on specific ρ selection. An interesting property of SINCO is that it introduces nonzero entries into the matrix X as it progresses. Hence, if one uses looser tolerance and stops the algorithm early, then a sparser solution is obtained for any specific value of ρ. What we observe, as seen in figure 17.1, is that if we apply SINCO to problem (17.6) with ever tighter tolerance, the ROC curves obtained from the tolerance solution path match the ROC curves obtained from the regularization path. Here we show several examples of the matching ROC curves for various random networks (see Scheinberg and Rish (2009), for details of the experiments). The ROC curves of the regularization path computed by glasso are very similar to SINCO's ROC curves (Scheinberg and Rish, 2009). Note that changing tolerance does not have the same affect on glasso as it does on SINCO. The numbers of TP and FP do not change noticeably with increasing tolerance. This is due to the fact that the algorithm in glasso updates a whole row and a column of C at each iteration while it cycles through the rows and columns, rather than selecting the updates in a greedy manner.

Our observations imply that methods like SINCO can be used to greedily select the elements of the graphical model until the desired trade-off between FPs and TPs is achieved. In the limit SINCO solves the same problem as glasso and hence the limit numbers of the true and false positives are dictated by the choice of ρ. But since the real goal is to recover the true nonzero structure of the covariance matrix, it is not necessary to solve problem (17.6) accurately. For the purpose of recovering a good TP/FP ratio, one can apply the SINCO method without adjustments to ρ.

For details on the numerical experiments presented here, see Scheinberg and Rish (2009).

17.3 Alternating Linearization Method

Efficient alternatives to BCD methods are the gradient based methods which we discussed briefly in section 17.1.2. These methods have higher per-iteration complexity (typically $O(p^3)$), but they usually converge in fewer iterations than a BCD method. Moreover, several of these methods, including the one presented here, have variants with provable complexity bounds.

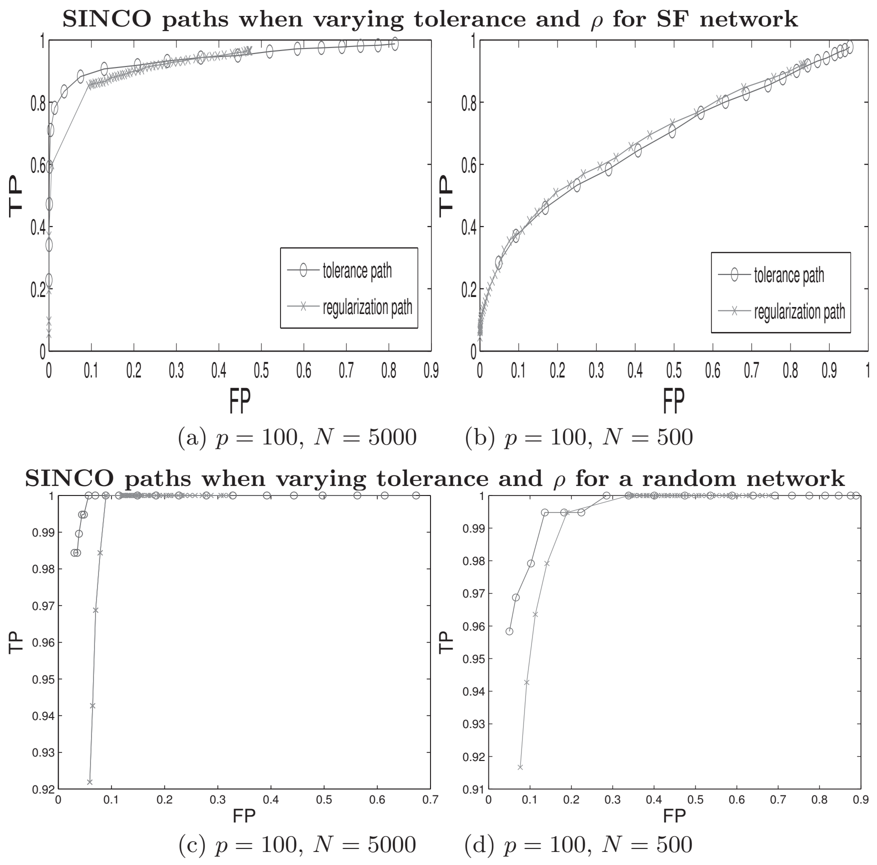
We discuss here the alternating linearization method (ALM) introduced in Goldfarb et al. (2009) for solving (17.6) which we write here as

$$\min_{X \in S^p_{++}} \quad F(X) \equiv f(X) + g(X), \tag{17.28}$$

where $f(X) = -\log\det(X) + \langle S, X\rangle$ and $g(X) = \rho\|X\|_1$. An effective way to approach an objective function of this form is to "split" f and g by introducing a new variable, that is, to rewrite (17.28) as

$$\min_{X,Y \in S^p_{++}} \{f(X) + g(Y) : X - Y = 0\}, \tag{17.29}$$

and to apply an alternating-direction augmented Lagrangian method to it.



(a) $p = 100$, $N = 5000$ (b) $p = 100$, $N = 500$

(c) $p = 100$, $N = 5000$ (d) $p = 100$, $N = 500$

Figure 17.1: Random and scale-free networks: SINCO paths when varying tolerance and ρ.

Given a penalty parameter $1/\mu$, and an estimate of the Lagrange multiplier Λ, at the kth iteration the augmented Lagrangian method minimizes the augmented Lagrangian function

$$\mathcal{L}(X, Y; \Lambda) := f(X) + g(Y) - \langle \Lambda, X - Y \rangle + \frac{1}{2\mu}\|X - Y\|_2^2,$$

with respect to X and Y, that is, it solves the subproblem

$$(X^k, Y^k) := \arg \min_{X,Y \in S^p_{++}} \mathcal{L}(X, Y; \Lambda^k) \tag{17.30}$$

and updates the Lagrange multiplier Λ via

$$\Lambda^{k+1} := \Lambda^k - (X^k - Y^k)/\mu. \tag{17.31}$$

Minimizing $\mathcal{L}(X, Y; \Lambda)$ with respect to X and Y jointly is difficult, while doing so with respect to X and Y alternatingly can be done efficiently, as we will show below. Moreover, minimization over Y does not have to include the constraint $Y \in S^p_{++}$.

The following alternating-direction version of the augmented Lagrangian method (ADAL) is often advocated (see, e.g., Fortin and Glowinski (1983); Glowinski and Le Tallec (1989)):

$$\begin{cases} X^{k+1} & := \arg\min_{X \in S^p_{++}} \mathcal{L}(X, Y^k; \Lambda^k) \\ Y^{k+1} & := \arg\min_Y \mathcal{L}(X^{k+1}, Y; \Lambda^k) \\ \Lambda^{k+1} & := \Lambda^k - (X^{k+1} - Y^{k+1})/\mu. \end{cases} \tag{17.32}$$

In Goldfarb et al. (2009) the following symmetric version of the ADAL method is considered:

$$\begin{cases} X^{k+1} & := \arg\min_X \mathcal{L}(X, Y^k; \Lambda^k_Y) \\ \Lambda^{k+1}_X & := \Lambda^k_Y - (X^{k+1} - Y^k)/\mu \\ Y^{k+1} & := \arg\min_Y \mathcal{L}(X^{k+1}, Y; \Lambda^{k+1}_X) \\ \Lambda^{k+1}_Y & := \Lambda^{k+1}_X - (X^{k+1} - Y^{k+1})/\mu. \end{cases} \tag{17.33}$$

Let us assume for the moment that $f(X)$ is in the class $C^{1,1}$ with Lipschitz constant $L(f)$,[2] while $g(X)$ is simply convex. In this case, from the first-order optimality conditions for the two subproblems in (17.33), we have

$$\Lambda^{k+1}_X = \nabla f(X^{k+1}) \quad \text{and} - \Lambda^{k+1}_Y \in \partial g(Y^{k+1}), \tag{17.34}$$

where $\partial g(Y^{k+1})$ is the subdifferential of $g(Y)$ at $Y = Y^{k+1}$. Substituting these relations into (17.33), we obtain the following algorithm for solving (17.28), which we refer to as an alternating linearization minimization algorithm.

Algorithm 17.3 Alternating linearization method (ALM)

```
1: Input: $X^0 = Y^0$
2: for $k = 0, 1, \cdots$ do
3:     1. Solve $X^{k+1} := \mathrm{argmin}_{X \in S^p_{++}} Q_g(X, Y^k) \equiv f(X) + g(Y^k) - \langle \Lambda^k, X - Y^k \rangle + \frac{1}{2\mu}\|X - Y^k\|_2^2$;
4:     2. Solve $Y^{k+1} := \mathrm{argmin}_Y Q_f(X^{k+1}, Y) \equiv f(X^{k+1}) + \langle \nabla f(X^{k+1}), Y - X^{k+1} \rangle + \frac{1}{2\mu}\|Y - X^{k+1}\|_2^2 + g(Y)$;
5:     3. $\Lambda^{k+1} = \nabla f(X^{k+1}) - (X^{k+1} - Y^{k+1})/\mu$.
6: end for
```

Algorithm 17.3 can be viewed in the following way: at each iteration we construct a quadratic approximation of the functions $g(X)$ at the current

2. This does not hold for our $f(X)$ if we consider $X \in S^p_{++}$, but it holds in a smaller feasible set.

iterates Y^k and minimize the sum of this approximation and $f(X)$. The approximation is based on linearizing $g(X)$ (hence the name "alternating linearization method") and adding a "prox"-term $\frac{1}{2\mu}\|X - Y^k\|_2^2$. Then the linearization step is applied to $f(X)$ at the next iterate X^{k+1}. The purpose of these linearizations is to replace one of the functions with a simple linear function with a prox-term to make optimization easy. It is important, however, that the resulting functions $Q_g(X, Y^k)$ and $Q_f(X^{k+1}, Y)$ provide a good approximation of $F(X)$. For theoretical purposes it is sufficient that at the obtained minimum $Q_g(X, Y^k) \le F(X)$ ($Q_f(X^{k+1}, Y) \le F(Y)$), which means that the reduction in the value of $F(X)$ achieved by minimizing $Q_g(X, Y^k)$ in step 1 and $Q_f(X^{k+1}, Y)$ in step 2 is not smaller than the reduction achieved in the value of $Q_g(X, Y^k)$ ($Q_f(X^{K+1}, Y)$) itself. For the case of $Q_f(X^{k+1}, Y)$, when μ is small enough ($\mu \le 1/L(f)$), the quadratic function $f(X^{k+1}) + \langle \nabla f(X^{k+1}), Y - X^{k+1}\rangle + \frac{1}{2\mu}\|Y - X^{k+1}\|_2^2$ is an upper approximation to $f(X)$, which means that condition $Q_f(X^{k+1}, Y) \le F(Y)$ is guaranteed to hold. For the case of $Q_g(X, Y^k)$, however, condition $Q_g(X, Y^k) \le F(X)$ may fail for any $\mu > 0$, since $g(X)$ is not smooth. In this case, if the condition fails, one can simply skip the step in X by assigning $X^{k+1} = Y^k$. This leads to the following ALM with skipping steps (algorithm 17.4).

Algorithm 17.4 Alternating linearization method with skipping step

```
1: Input: $X^0 = Y^0$
2: for $k = 0, 1, \cdots$ do
3:     1. Solve $X^{k+1} := \arg\min_X Q_g(X, Y^k) \equiv f(X) + g(Y^k) - \langle \Lambda^k, X - Y^k\rangle + \frac{1}{2\mu}\|X - Y^k\|_2^2$;
4:     2. If $F(X^{k+1}) > Q_g(X^{k+1}, Y^k)$, then $X^{k+1} := Y^k$.
5:     3. Solve $Y^{k+1} := \arg\min_Y Q_f(X^{k+1}, Y) \equiv f(X^{k+1}) + \langle \nabla f(X^{k+1}), Y - X^{k+1}\rangle + \frac{1}{2\mu}\|Y - X^{k+1}\|_2^2 + g(Y)$;
6:     4. $\Lambda^{k+1} = \nabla f(X^{k+1}) - (X^{k+1} - Y^{k+1})/\mu$.
7: end for
```

Algorithm 17.4 is identical to 17.3, and hence to the symmetric ADAL algorithm (17.33) as long as $F(X^{k+1}) \le Q_g(X^{k+1}, Y^k)$ at each iteration. If this condition fails, then the algorithm simply sets $X^{k+1} \leftarrow Y^k$. Algorithm 17.4 has the following convergence property and iteration complexity bound (Goldfarb et al., 2009).

Theorem 17.1. *Assume ∇f is Lipschitz continuous with Lipschitz constant*

$L(f)$. For $\mu \leq 1/L(f)$, algorithm 17.4 satisfies

$$F(y^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{2\mu(k + k_n)}, \forall k, \tag{17.35}$$

where x^* is an optimal solution of (17.28) and k_n is the number of iterations until the kth for which $F(x^{k+1}) \leq Q_g(x^{k+1}, y^k)$. Thus, algorithm 17.4 produces a sequence which converges to the optimal solution in function value, and the number of iterations needed is $O(1/\varepsilon)$ for an ε-optimal solution.

The iteration complexity bound in theorem 17.1 can be improved. Nesterov (1983, 2004) proved that one can obtain an optimal iteration complexity bound of $O(1/\sqrt{\varepsilon})$, using only the first-order information. His acceleration technique is based on using a linear combination of previous iterates to obtain a point where the approximation is built. This technique has been exploited and extended by Tseng (2008), Beck and Teboulle (2009), and many others. Specifically for SICS a method with complexity bound $O(1/\sqrt{\varepsilon})$ was introduced by Lu (2009). A similar technique can be adopted to derive a fast version of algorithm 17.4 that has an improved complexity bound of $O(1/\sqrt{\varepsilon})$, while keeping the computational effort in each iteration almost unchanged. However, we do not present this method here, since when it is applied to the SICS problem, it does not appear to work as well as algorithm 17.4.

Note that in our case $f(X) = -\log\det(X) + \langle S, X\rangle$ does not have a Lipschitz continuous gradient in general. Moreover, $f(X)$ is defined only for positive definite matrices while $g(X)$ is defined everywhere. These properties of the objective function make the SICS problem especially challenging for optimization methods. Nevertheless, we can still apply (17.33) to solve the problem directly. Moreover, we can apply algorithm 17.4 and obtain the complexity bound in theorem 17.1 as follows. As proved in Lu (2009), the optimal solution X^* of (17.28) satisfies $X \succeq \alpha I$, where $\alpha = \frac{1}{\|S\|+p\rho}$ (see proposition 3.1 in Lu (2009)). Therefore, the SICS problem (17.28) can be formulated as

$$\min_{X,Y}\{f(X) + g(Y) : X - Y = 0, X \in \mathcal{C}, Y \in \mathcal{C}\}, \tag{17.36}$$

where $\mathcal{C} := \{X \in S^n : X \succeq \frac{\alpha}{2} I\}$. We know that the constraint $X \succeq \frac{\alpha}{2} I$ is not tight at the solution. Hence, if we start the algorithm with $X \succeq \alpha I$ and restrict the stepsize μ to be sufficiently small, then the iterates of the method will remain in the domain where the gradient of $f(X)$ is Lipschitz continuous, and we can apply algorithm 17.3 and theorem 17.1.

Note, however, that the bound on the Lipschitz constant of the gradient of $f(X)$ is $1/\alpha^2$, and hence can be very large. It is not practical to restrict μ in the algorithm to be smaller than α^2, since μ determines the stepsize at each iteration. Below is a practical version of our algorithm applied to the SICS problem.

Algorithm 17.5 Alternating linearization method for SICS

```
1: Input: $X^0 = Y^0$, $\mu_0$.
2: for $k = 0, 1, \cdots$ do
3:     0. Pick $\mu_{k+1} \le \mu_k$.
4:     1. Solve $X^{k+1} := \arg\min_{X \in \mathcal{C}} Q_g(X, Y^k) \equiv f(X) + g(Y^k) - \langle \Lambda^k, X - Y^k \rangle + \frac{1}{2\mu_{k+1}} \|X - Y^k\|_F^2$;
5:     2. If $F(X^{k+1}) > Q_g(X^{k+1}, Y^k)$, then $X^{k+1} := Y^k$.
6:     3. Solve $Y^{k+1} := \arg\min_Y f(X^{k+1}) + \langle \nabla f(X^{k+1}), Y - X^{k+1} \rangle + \frac{1}{2\mu_{k+1}} \|Y - X^{k+1}\|_F^2 + g(Y)$;
7:     4. $\Lambda^{k+1} = \nabla f(X^{k+1}) - (X^{k+1} - Y^{k+1})/\mu_{k+1}$.
8: end for
```

17.3.1 Solving Subproblems of ALM for SICS

We now show how to solve the two optimization problems in algorithm 17.5. The first-order optimality conditions for step 1 in algorithm 17.5, ignoring the constraint $X \in \mathcal{C}$, are

$$\nabla f(X) - \Lambda^k + (X - Y^k)/\mu_{k+1} = 0. \tag{17.37}$$

Consider $V \operatorname{Diag}(d) V^\top$, the spectral decomposition of $Y^k + \mu_{k+1}(\Lambda^k - S)$, and let

$$\gamma_i = \left(d_i + \sqrt{d_i^2 + 4\mu_{k+1}} \right) /2, i = 1, \ldots, p. \tag{17.38}$$

Since $\nabla f(X) = -X^{-1} + S$, it is easy to verify that $X^{k+1} := V \operatorname{Diag}(\gamma) V^\top$ satisfies (17.37). When the constraint $X \in \mathcal{C}$ is imposed, the optimal solution changes to $X^{k+1} := V \operatorname{Diag}(\gamma) V^\top$ with

$$\gamma_i = \max \left\{ \alpha/2, \left(d_i + \sqrt{d_i^2 + 4\mu_{k+1}} \right) /2 \right\}, i = 1, \ldots, p.$$

We observe that solving (17.37) requires approximately the same effort ($O(p^3)$) as is required to compute $\nabla f(X^{k+1})$ itself. Moreover, from the solution to (17.37), $\nabla f(X^{k+1})$ is obtained with only a negligible amount of additional effort, since $(X^{k+1})^{-1} := V \operatorname{Diag}(\gamma)^{-1} V^\top$.

The first-order optimality conditions for step 2 in algorithm 17.5 are

$$0 \in \nabla f(X^{k+1}) + (Y - X^{k+1})/\mu_{k+1} + \partial g(Y). \tag{17.39}$$

Since $g(Y) = \rho\|Y\|_1$, the solution to (17.39) is given by

$$Y^{k+1} = \text{shrink}(X^{k+1} - \mu_{k+1}(S - (X^{k+1})^{-1}), \mu_{k+1}\rho),$$

where the "shrinkage operator" $\text{shrink}(Z, \rho)$ is defined as $\text{shrink}(Z, \rho)_{ij} = \text{shrink}(Z_{ij}, \rho)$. It is trivial to see that to apply algorithm 17.5 to the extended formulation (17.7), one only needs to modify the shrinkage step $\text{shrink}(Z, M)$, which is then defined as $[\text{shrink}(Z, M)]_{ij} = \text{shrink}(Z_{ij}, M_{ij})$.

Note that the $O(p^3)$ complexity of step 1 which requires a spectral decomposition, dominates the $O(p^2)$ complexity of step 2 which requires a simple shrinkage. There is no closed-form solution for the subproblem corresponding to Y when the constraint $Y \in \mathcal{C}$ is imposed. One can attempt to impose this by a line search on the value of μ_k. Imposing such a constraint in practice limits the stepsize too much, and the performance of the algorithm deteriorates substantially. Thus, resulting iterates Y^k may not be positive definite, while the iterates X^k remain so. Eventually, due to the convergence of Y^k and X^k, the Y^k iterates become positive definite and the constraint $Y \in \mathcal{C}$ is satisfied. Relaxing the positive definiteness constraint during the course of the algorithm appears to be desirable for the overall performance.

17.4 Remarks on Numerical Performance

A number of numerical comparisons of optimization methods for SICS have been presented in the literature. See, for instance, Duchi et al. (2008); Goldfarb et al. (2009); Scheinberg et al. (2010); Friedman et al. (2007); Lu (2009); Scheinberg and Rish (2009) for the comparison of methods discussed in this chapter. The comparison we discuss here is based solely on the time and iteration efficiency of the algorithms to achieve comparable solutions in terms of their objective function values. The sparsity patterns recovered by these methods (aside from COVSEL) appears to be comparable.

In summary, the first-order methods in Duchi et al. (2008); Goldfarb et al. (2009); Scheinberg et al. (2010); Lu (2009) outperform glasso in Friedman et al. (2007), which substantially outperforms [3] COVSEL in Banerjee et al. (2008) and somewhat outperforms SINCO in Scheinberg and Rish (2009). Most of the tests were performed on instances up to the size $p = 2000$. On very sparse structured large matrices SINCO outperforms glasso. In

3. The Fortran implementation of glasso has some faults and occasionally breaks down, especially in large-scale cases. We believe it is an issue with the implementation rather than the algorithm so we do not elaborate on this further.

all experiments in Scheinberg et al. (2010) ALM outperforms the projected gradient method in Duchi et al. (2008), and the smooth accelerated gradient method in Lu (2009), in terms of CPU time and accuracy of the solution. The first-order methods do not exploit the solution sparsity, in that regardless of the sparsity, the per-iteration complexity remains $O(p^3)$. The per-iteration complexity of glasso and SINCO is empirically smaller, but the number of iterations required to achieve comparable accuracy is larger than that of the first-order methods.

17.5 References

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian for binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Sciences*, 2(1):183–202, 2009.

A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and its Applications*, 30 (1):56–66, 2008.

J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse Gaussians. *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, 2008.

M. Fortin and R. Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. North-Holland, 1983.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2007.

R. Glowinski and P. Le Tallec. *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*. SIAM, Philadelphia, 1989.

D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. Technical report, Department of IEOR, Columbia University, 2009.

D. Heckerman. A tutorial on learning Bayesian networks. Technical report, Microsoft Research, 1995.

V. Krishnamurthy and A. d'Aspremont. A pathwise algorithm for covariance selection. *preprint*, 2009. arXiv:0908.0143.

S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

L. Li and K.-C. Toh. An inexact interior point method for l_1-regularized sparse covariance selection. *Mathematical Programming Computation*, 2(3–4):291–315, 2010.

Z. Lu. Smooth optimization approach for sparse covariance selection. *SIAM Journal on Optimization*, 19(4):1807–1827, 2009.

Z. Lu. Adaptive first-order methods for general sparse inverse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2000–2016, 2010.

N. Meinshausen and P. Buhlmann. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

A. Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2005.

Y. E. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$. *Doklady Akademia Nauk SSSR*, 269:543–547, 1983.

Y. E. Nesterov. Introductory lectures on convex optimization: A basic course. 87, 2004.

Y. E. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming, series A*, 103:127–152, 2005.

K. Scheinberg and I. Rish. SINCO - a greedy coordinate ascent method for sparse inverse covariance selection problem. 2009. Preprint available at http://www.optimization-online.org/DB_HTML/2009/07/2359.html.

K. Scheinberg, S. Ma, and D. Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In *Advances in Neural Information Processing Systems 23*, 2010.

L. Sun, R. Patel, J. Liu, K. Chen, T. Wu, J. Li, E. Reiman, and J. Ye. Mining brain region connectivity for alzheimer's disease study via sparse inverse covariance estimation. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2008.

M. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In *Advances in Neural Information Processing Systems 20*, pages 1465–1472. 2007.

C. Wang, D. Sun, and K.-C. Toh. Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm. *Preprint*, 2009.

Z. Wen, D. Goldfarb, S. Ma, and K. Scheinberg. Row by row methods for semidefinite programming. Technical report, Department of IEOR, Columbia University, 2009.

H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming*. Kluwer Academic, 2000.

M. Yuan and Y. Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

X. Yuan. Alternating direction methods for sparse covariance selection. 2009. Preprint available at http://www.optimization-online.org/DB_FILE/2009/09/2390.pdf.