arXiv:2112.10388v2 [stat.CO] 10 Jul 2023

On some algorithms for estimation in Gaussian graphical models

BY S. HØJSGAARD

Department of Mathematical Sciences, Aalborg University
Skjernvej 4A, DK-9220 Aalborg, Denmark
sorenh@math.aau.dk

S. LAURITZEN

Department of Mathematical Sciences, University of Copenhagen
Universitetsparken 5, DK-2100 Copemhagen, Denmark
lauritzen@math.ku.dk

SUMMARY

In Gaussian graphical models, the likelihood equations must typically be solved iteratively. We investigate two algorithms: A version of iterative proportional scaling which avoids inversion of large matrices, resulting in increased speed when graphs are sparse. We compare this to an algorithm based on convex duality and operating on the covariance matrix by neighbourhood coordinate descent, essentially corresponding to the graphical lasso with zero penalty. For large, sparse graphs, this version of the iterative proportional scaling algorithm appears feasible and has simple convergence properties. The algorithm based on neighbourhood coordinate descent is extremely fast and less dependent on sparsity, but needs a positive definite starting value to converge, which may be difficult to achieve when the number of variables exceeds the number of observations.

Some key words: Covariance selection; Convex duality; Maximum likelihood estimation.

1. INTRODUCTION

Maximum likelihood estimation in Gaussian graphical models can be carried out via generic optimization algorithms, Newton–Raphson iteration, iterative proportional scaling, other alternating algorithms (Speed and Kiiveri, 1986), and algorithms exploiting duality and operating on the covariance matrix, such as the algorithm given by Wermuth and Scheidt (1977) and neighbourhood coordinate descent (Hastie et al., 2016, p. 631 ff.). Neighbourhood coordinate descent may be seen as a special instance of the graphical lasso with zero penalty (Yuan and Lin, 2007; Banerjee et al., 2008; Friedman et al., 2008), or as a special instance of the GOLAZO algorithm (Lauritzen and Zwiernik, 2022).

Algorithms based on duality need a positive definite starting value to guarantee convergence and this may be difficult to find when the number of variables exceeds the number of observations. In addition, the concentration matrix returned by these algorithms after finitely many steps may not have exact zero values for entries corresponding to non-edges, and additional procedures are neccessary to ensure this.

In contrast, iterative proportional scaling is provably convergent from the identity matrix as a starting value when the maximum likelihood estimator exists and it satisfies the model restrictions at all times; but it may be slow when used with many variables.

The main contributions of this article are the following: We present a version of the iterative proportional scaling algorithm that updates the covariance and concentration matrices simultaneously and works on edges rather than cliques, so identification of cliques becomes unnecessary and inversion of large

matrices is avoided, resulting in much shorter computing times; further, we present a version of neighbourhood coordinate descent that may not need a positive definite starting value and is guaranteed to output a positive definite concentration matrix, if convergence is achieved.

Alternative ways of speeding up the iterative proportional scaling algorithm typically involves special methods for calculating appropriate marginals, for example using probability propagation as described for the analogous discrete case in Jiroušek and Přeučil (1995). Approaches along these lines have been used by Hara and Takemura (2010) and Xu et al. (2011, 2012). Xu et al. (2015) gives a thorough survey and comparison of the methods and also shows how to speed up the scaling algorithms by partitioning of the cliques and localized updating. The methods investigated in this article are based on simple and general matrix manipulations and avoid setting up more involved computational structures.

2. LIKELIHOOD EQUATIONS FOR GAUSSIAN GRAPHICAL MODELS

Let $X = (X_v, v \in V)$ be a d dimensional random vector, i.e. $|V| = d$, normally distributed with mean zero and covariance matrix Σ. The focus is on the pattern of zeros in the inverse covariance matrix, i.e. in the concentration matrix $K = \Sigma^{-1}$. If $K_{uv} = 0$ then X_u and X_v are conditionally independent given $X_{V\setminus\{u,v\}}$. The pattern of zeros in K may be represented by an undirected graph $\mathcal{G} = (V, E)$ with vertices V and edges E. A Gaussian graphical model is then defined by demanding $K_{uv} = 0$ unless there is an edge $uv \in E$. For further details, we refer to Lauritzen (1996, Ch. 4).

Let $\mathcal{G} = (V, E)$ be a simple, undirected graph and let S denote the empirical covariance matrix obtained from a sample $X^1 = x^1, \ldots, X^n = x^n$. The maximum likelihood estimate $\hat{\Sigma}$ of the covariance matrix Σ in an undirected Gaussian graphical model is the unique solution to the system of equations (Lauritzen, 1996, p. 133):

$$\hat{\Sigma}_{vv} - S_{vv} = 0 \ \text{ for all } v \in V, \tag{1}$$

$$\hat{\Sigma}_{uv} - S_{uv} = 0 \ \text{ for all } uv \in E, \tag{2}$$

$$\hat{K}_{uv} = (\hat{\Sigma}^{-1})_{uv} = 0 \ \text{ for all } uv \notin E, \tag{3}$$

provided such a solution exists.

3. ITERATIVE PROPORTIONAL SCALING

3.1. Computational issues of updating

Iterative proportional scaling cycles through subsets $c \subseteq V$ of variables which are complete in $\mathcal{G}$, i.e. all elements of c are mutual neighbours in the graph. The current estimate of Σ is then updated by keeping the parameters of the conditional distribution $X_{V\setminus c} \,|\, X_c$ fixed, whereas the parameters of the marginal distribution of X_c are updated to maximize the objective function under that restriction. The updates have the form

$$f(x;\Sigma) \ \leftarrow \ f(x)\frac{f(x_c;S_{cc})}{f(x_c;\Sigma_{cc})},$$

where S_{cc} and Σ_{cc} here and in the following indicate the corresponding marginals of the empirical covariance matrix and of the current value of Σ; hence the densities are scaled proportionally, whence the name of the algorithm. The algorithm is provably convergent when started in a point satisfying the model restrictions if the likelihood function is bounded, i.e. if the maximum likelihood estimate exists (Lauritzen, 1996, Thm. 5.4).

Let $c \subseteq V$ and $a = V \setminus c$ where c is a complete subset of V in $\mathcal{G}$. The update for c of the concentration matrix K takes the form (Lauritzen, 1996, p. 134)

$$K_{cc} \leftarrow (S_{cc})^{-1} + L, \tag{4}$$

whereas K_{ac}, K_{aa}, K_{ca} are unchanged. There are two alternatives for calculating L:

$$L = K_{ca}(K_{aa})^{-1}K_{ac} \tag{5}$$
$$= K_{cc} - (\Sigma_{cc})^{-1}. \tag{6}$$

Calculating L as in (5) gives what is referred to in this paper as the *concentration version* of the algorithm and has the advantage that $\Sigma = K^{-1}$ is not needed, so inversion of K is avoided and Σ need not be stored. This is efficient if a is small and c is large.

Calculating L as in (6) gives what is referred to in this paper as the *covariance version* of the algorithm. Expression (6) has the advantage that computation of $(K_{aa})^{-1}$ is not needed; this matrix inversion could be expensive if a is large. On the other hand, Σ needs to be stored and calculated.

Luckily it is possible to update Σ along with K, avoiding repeated and time consuming matrix inversions. This makes expression (6) feasible to use in practice and speeds up the computation considerably.

3.2. Updating Σ

The updated version $\tilde{\Sigma}$ of Σ can be calculated as

$$\tilde{\Sigma} = \begin{pmatrix} S_{cc} & S_{cc}(\Sigma_{cc})^{-1}\Sigma_{ca} \\ \Sigma_{ac}(\Sigma_{cc})^{-1}S_{cc} & \Sigma_{aa} - \Sigma_{ac}H\Sigma_{ca} \end{pmatrix}, \tag{7}$$

where H is equal to

$$H = (\Sigma_{cc})^{-1} - (\Sigma_{cc})^{-1}S_{cc}(\Sigma_{cc})^{-1} \tag{8}$$

which avoids inverting K. To see this is correct, we may establish that

$$\begin{pmatrix} S_{cc} & S_{cc}(\Sigma_{cc})^{-1}\Sigma_{ca} \\ \Sigma_{ac}(\Sigma_{cc})^{-1}S_{cc} & \Sigma_{aa} - \Sigma_{ac}H\Sigma_{ca} \end{pmatrix} = \begin{pmatrix} (S_{cc})^{-1} + K_{ca}(K_{aa})^{-1}K_{ac} & K_{ca} \\ K_{ac} & K_{aa} \end{pmatrix}^{-1},$$

which follows by direct matrix multiplication, using the identities

$$(\Sigma_{cc})^{-1}\Sigma_{ca} = -K_{ca}(K_{aa})^{-1}, \quad (K_{aa})^{-1} = \Sigma_{aa} - \Sigma_{ac}(\Sigma_{cc})^{-1}\Sigma_{ca}.$$

The update of Σ as in (7) is also given as formula (19) in Speed and Kiiveri (1986), on p. 185 of **?**, and as formula (2) of Xu et al. (2015), albeit in varying notations.

If the difference between Σ_{cc} and S_{cc} is small, the corresponding update may be ignored altogether, see further in Section 3.4.

There are two natural choices for the system of complete sets c: the set $\mathcal{C}$ of cliques of $\mathcal{G}$; or the set of edges E. Both choices are compared in our experiments, but we focus on the set of edges E, as this avoids the NP-complete task of determining the cliques of the graph.

3.3. Updating the likelihood function

Consider again a sample $X^1 = x^1, \ldots, X^n = x^n$ where $X^\nu \sim N_d(0, \Sigma)$ and let S denote the sample covariance matrix. The log-likelihood function (ignoring additive constants) is

$$l(K) = \frac{n}{2}\log\det(K) - \frac{n}{2}\operatorname{tr}(KS). \tag{9}$$

If we let Δ_{cc} denote the difference between the updated $\tilde{K}_{cc}$ and old K_{cc} i.e.

$$\Delta_{cc} = \{(S_{cc})^{-1} + L\} - K_{cc} = (S_{cc})^{-1} - (\Sigma_{cc})^{-1}, \tag{10}$$

we have

$$\det\tilde{K} = \det\{K_{cc} + \Delta_{cc} - K_{ca}(K_{aa})^{-1}K_{ac}\}\det K_{aa} = \det(S_{cc})^{-1}\det K_{aa}.$$

But $\det K = \det(\Sigma_{cc})^{-1}\det K_{aa}$. Hence, if we let $A = (\Sigma_{cc})^{-1}S_{cc}$ we have

$$\log\det\tilde{K} = \log\det K - \log\det A.$$

For the trace term we get from (10)

$$\begin{aligned}\mathrm{tr}(\tilde{K}S) &= \mathrm{tr}(KS) + \mathrm{tr}(\Delta_{cc}S_{cc}) \\ &= \mathrm{tr}(KS) + |c| - \mathrm{tr}\{(\Sigma_{cc})^{-1}S_{cc}\} = \mathrm{tr}(KS) + |c| - \mathrm{tr}(A),\end{aligned}$$

where $|c| = \mathrm{tr}\{(S_{cc})^{-1}S_{cc}\}$ is the size of c. We thus get the expression

$$\ell(\tilde{K}) = \ell(K) - \frac{n}{2}|c| - \frac{n}{2}\log\det A + \frac{n}{2}\mathrm{tr}(A)$$

and note that A has dimension $|c|$ so the adjustment is easily calculated if c is small.

3.4. Convergence issues

Convergence of the algorithms can be assessed by investigating whether the likelihood equations (1) and (2) are satisfied within a small numerical threshold since (3) remains exactly satisfied at all times. This requires that Σ is available and we may then express the deviation as $\|\Sigma(\mathcal{G}) - S(\mathcal{G})\|_\infty$, where

$$\|A\|_\infty = \max_{uv}|A_{uv}|$$

is the maximum absolute deviation norm, and $A(\mathcal{G})$ is obtained by replacing A_{uv} with zero for $uv \notin E$. The gradient of the log-likelihood function (9) is equal to

$$\nabla_K \ell(K) = \frac{n}{2}\{\Sigma(\mathcal{G}) - S(\mathcal{G})\}$$

so it seems appropriate to continue the iteration until the size of the gradient is small, hence until

$$\|\Sigma(\mathcal{G}) - S(\mathcal{G})\|_\infty \leq 2\varepsilon/n$$

where ε is a small threshold.

Commonly used, but less stringent, convergence criteria monitor whether changes in the log-likelihood or changes in parameter values between succesive iterations are small. However, one may find that such changes only indicates that the algorithm slows down even though the likelihood equations are far from being satisfied and the value of the likelihood function may be far from its maximum; so we warn against using such criteria.

3.5. Computational savings

To achieve a further speedup of the algorithm we may for each c check whether $\|\Sigma_{cc} - S_{cc}\|_\infty < 2\varepsilon/n$ before the update is executed. If this is the case we may ignore the update as this will be time consuming but ineffective; this allows the algorithm to move faster towards the limit when the algorithm is close but not close enough. If the algorithm is terminated when no local updates are needed, the likelihood equations (1) and (2) are still satisfied within the same threshold, since

$$\|\Sigma(\mathcal{G}) - S(\mathcal{G})\|_\infty = \max_{c\in\mathcal{A}}\|\Sigma_{cc} - S_{cc}\|_\infty,$$

where $\mathcal{A}$ is the chosen system of subsets.

4. ALGORITHMS BASED ON CONVEX DUALITY

4.1. The optimization problem and its dual

The problem of maximizing the log-likelihood function, to be referred to as the *primal problem*, may be formulated as follows, where as before S denotes the empirical covariance matrix and we have ignored the multiplicative constant $n/2$:

$$\begin{aligned}&\underset{K}{\text{maximize}} \quad \ell(K) = \log\det(K) - \mathrm{tr}(KS) \\ &\text{subject to} \quad K \in \mathbb{S}^{d\times d}_{\succ}(\mathcal{G}),\end{aligned} \tag{11}$$

where $\mathbb{S}_{\succ}^{d\times d}(\mathcal{G})$ denotes the set of positive definite matrices K with $K_{uv} = 0$ for all $uv \notin E(\mathcal{G})$. This is a convex optimization problem with a unique solution if and only if the maximum likelihood estimate exists. We shall for the moment assume that S is positive definite so that this is not a problem but later identify necessary modifications for a more general case. To exploit convex duality (**?**, Ch. 5), we consider the Lagrangian

$$\mathcal{L}(\Lambda, K) = \log\det(K) - \mathrm{tr}(KS) - \mathrm{tr}(K\Lambda),$$

where Λ is a symmetric matrix satisfying $\Lambda_{uv} = 0$ for all $uv \in E$ and $\Lambda_{uu} = 0$ for all $u \in V$. We now get the dual function

$$g(\Lambda) = \sup_K \mathcal{L}(\Lambda, K) = -\log\det(S + \Lambda) - d.$$

Since for $K \in \mathbb{S}^{d\times d}(\mathcal{G})$ we have $\mathcal{L}(\Lambda, K) = \ell(K)$, the dual function yields an upper bound on $\ell(K)$ which we now wish to minimize. Letting $\Sigma = S + \Lambda$ yields the dual optimization problem as

$$\begin{aligned} \underset{\Sigma}{\text{minimize}} \quad & g(\Lambda) = -\log\det(\Sigma) - d \\ \text{subject to} \quad & \Sigma \in \mathbb{S}_{\succ}^{d\times d}, \quad \Sigma_{uu} = S_{uu}, \Sigma_{uv} = S_{uv} \text{ for all } u \in V, uv \in E \end{aligned}$$

or, equivalently,

$$\begin{aligned} \underset{\Sigma}{\text{maximize}} \quad & \det(\Sigma) \\ \text{subject to} \quad & \Sigma \in \mathbb{S}_{\succ}^{d\times d}, \quad \Sigma_{uu} = S_{uu}, \Sigma_{uv} = S_{uv} \text{ for all } u \in V, uv \in E. \end{aligned} \tag{12}$$

A feasible point for (12) is often referred to as a *positive definite completion* (Grone et al., 1984) of the partial matrix

$$\Sigma_{\mathcal{G}} = \{\Sigma_{uu}, u \in V; \Sigma_{uv}, uv \in E\}.$$

The maximum likelihood estimator of Σ exists if and only if there is such a feasible point. It then holds that Σ is the unique optimizer of (12) if and only if $K = \Sigma^{-1}$ is the unique optimizer of (11).

4.2. Solving the dual problem

For a single vertex $u \in V$ we let $c = V \setminus \{u\}$, $b = \mathrm{bd}(u)$, and $r = \mathrm{cl}(u)^c$; so the variables X may be partitioned as

$$X = (X_u, X_{\mathrm{bd}(u)}, X_{\mathrm{cl}(u)^c})^\top = (X_u, X_b, X_r)^\top = (X_u, X_c)^\top.$$

Using Schur complements, we may then express the determinant as

$$\det\Sigma = \det\Sigma_{cc}\left\{\Sigma_{uu} - \Sigma_{uc}(\Sigma_{cc})^{-1}\Sigma_{cu}\right\}. \tag{13}$$

Keeping Σ_{cc} fixed and maximizing the Schur complement over feasible values of Σ_{uc} leads to changing the entries of non-neighbours r of u in the rows and columm of Σ as

$$\tilde{\Sigma}_{ru} = \Sigma_{rb}(\Sigma_{bb})^{-1}S_{bu} = \Sigma_{rb}(\Sigma_{bb})^{-1}\Sigma_{bu} = \Sigma_{rb}\beta_{bu}, \tag{14}$$

whereas all other entries of Σ are unchanged. Here and in the following we have let

$$\beta_{ub} = \beta_{bu}^\top = \Sigma_{ub}(\Sigma_{bb})^{-1}$$

denote the vector of regression coefficients for regressing X_u on X_b. To get the second equality in (14) we used that $\Sigma_{uv} = S_{uv}$ for all $uv \in E$.

If the graph is sparse, the boundary b is typically a small set and the expression (14) therefore avoids inversion of large matrices. Simple manipulations yield the updated value of the Schur complement

$$\tilde{\Sigma}_{uu} - \tilde{\Sigma}_{uc}(\tilde{\Sigma}_{cc})^{-1}\tilde{\Sigma}_{cu} = S_{uu} - S_{ub}(\Sigma_{bb})^{-1}S_{bu} = S_{uu} - S_{ub}\beta_{bu}. \tag{15}$$

This is also the optimization step in the more general GOLAZO algorithm (Lauritzen and Zwiernik, 2022) when specialized to estimation in Gaussian graphical models. The GOLAZO algorithm solves this step using quadratic programming, but in our special case the optimization is simple and explicit. Also, it is easy to verify that the update step in (14) is identical to the update step used in Hastie et al. (2016), Section 17.3.1; we refrain from giving the details as this is only a matter of comparing notations. We shall in the following refer to this algorithm as *neighbourhood coordinate descent*.

Another way is to consider non-edges $uv \notin E$ in turn and factor the determinant as

$$\det(\Sigma) = \det \Sigma_{AA} \det \left\{ \Sigma_{BB} - \Sigma_{BA}(\Sigma_{AA})^{-1}\Sigma_{AB} \right\},$$

where $A = V \setminus \{u, v\}$ and $B = \{u, v\}$, and then maximize the second factor in Σ_{AB} keeping all other elements of Σ fixed. This can easily be done explicitly; see for example Uhler (2019). This algorithm was implemented by Wermuth and Scheidt (1977) but we shall not investigate it further, as it will be slow when graphs are large and sparse, since then the number of non-edges then becomes huge.

4.3. When S is positive semidefinite

A problem with neighbourhood coordinate descent and other dual algorithms is that they need a feasible starting value Σ that satisfies the constraints of the dual problem i.e. $\Sigma_{uv} = S_{uv}$ for all $uv \in E$. If the starting value is not positive definite, the update might not be well defined, or the update might not increase the value of the determinant, and the algorithm may therefore not be convergent.

If the number of observations n used to form S does not exceed the dimension d, S is only positive semidefinite. We then still make a factorization as in (13)

$$\det \Sigma = \det \Sigma_{cc} \left\{ \Sigma_{uu} - \Sigma_{uc}(\Sigma_{cc})^{-}\Sigma_{cu} \right\}, \tag{16}$$

where now $(\Sigma_{cc})^{-}$ is any generalized inverse to Σ_{cc}. We may now proceed as in the positive definite case and maximize the value of the Schur complement, keeping Σ_{cc} fixed. As before, the Schur complement is maximized for

$$\tilde{\Sigma}_{ru} = \Sigma_{rb}(\Sigma_{bb})^{-} S_{bu}$$

and after the update we have

$$\det \tilde{\Sigma} = \left\{ \Sigma_{uu} - S_{ub}(\Sigma_{bb})^{-} S_{bu} \right\} \det \Sigma_{cc}.$$

If $\det \Sigma_{cc} = 0$, it also holds that $\det \tilde{\Sigma} = 0$. However, since

$$\operatorname{rank} \Sigma = \operatorname{rank} \Sigma_{cc} + \operatorname{rank} \left\{ \Sigma_{uu} - \Sigma_{uc}(\Sigma_{cc})^{-}\Sigma_{cu} \right\}$$

then if $\Sigma_{uu} - \Sigma_{uc}(\Sigma_{cc})^{-}\Sigma_{cu} = 0 \; < \; \Sigma_{uu} - S_{ub}(\Sigma_{bb})^{-} S_{bu}$, we would have $\operatorname{rank} \tilde{\Sigma} = \operatorname{rank} \Sigma_{cc} + 1$, so if $\operatorname{rank}(\Sigma_{cc}) = \operatorname{rank}(\Sigma)$, the rank will increase by one.

This may happen if the sample covariance is based on $n + 1$ observations and the graph is sparse so that $\deg(u) = |\operatorname{bd}(u)| < n < d$. In this way, Σ may be positive definite after a few rounds of iterations which will render the algorithm . But in general it may be difficult to find a positive definite starting value for the algorithm, see further in Section 4.7.

To illustrate the issue, consider the four-cycle with $\mathcal{G} = (\{1, 2, 3, 4\}, \{12, 23, 34, 14\})$ and let S be the empirical covariance matrix based on $n = 3$ observations, i.e. $\operatorname{rank}(S) = 2$ almost surely. Then, by **?**, the maximum likelihood estimate of Σ may or may not exist, depending on the exact configuration of the outcomes. However, updating the variable 1, leads to

$$\tilde{\Sigma}_{31} = \Sigma_{3,24}(\Sigma_{24,24})^{-} S_{24,1} = S_{3,24}(S_{24,24})^{-1} S_{24,1}$$

and

$$\operatorname{rank} \tilde{\Sigma} = \operatorname{rank} \left\{ S_{11} - S_{1,24}(S_{24,24})^{-1} S_{24,1} \right\} + \operatorname{rank} S_{234,234}.$$

But since also $\operatorname{rank} S_{24,24} = 2$, we have

$$\begin{aligned} 2 &= \operatorname{rank} S_{124,124} = \operatorname{rank}\left\{S_{11} - S_{1,24}(S_{24,24})^{-1}S_{24,1}\right\} + \operatorname{rank} S_{24,24} \\ &= \operatorname{rank}\left\{S_{11} - S_{1,24}(S_{24,24})^{-1}S_{24,1}\right\} + 2, \end{aligned}$$

so we must have $S_{11} - S_{1,24}(S_{24,24})^{-1}S_{24,1} = 0$ and thus $\operatorname{rank}\tilde{\Sigma} = \operatorname{rank} S_{234,234} = 2$. This argument may be repeated for subsequent updates. So the updates fail to produce a positive definite matrix no matter how the outcomes are configured, as opposed to the scaling algorithm that will converge for some configurations and diverge for others. In contrast, if we have $n = 4$ observations and $\operatorname{rank} S = 3$, we get

$$\begin{aligned} 3 &= \operatorname{rank} S_{124,124} = \operatorname{rank}\left\{S_{11} - S_{1,24}(S_{24,24})^{-1}S_{24,1}\right\} + \operatorname{rank} S_{24,24} \\ &= \operatorname{rank}\left\{S_{11} - S_{1,24}(S_{24,24})^{-1}S_{24,1}\right\} + 2, \end{aligned}$$

whence $\operatorname{rank}\left\{S_{11} - S_{1,24}(S_{24,24})^{-1}S_{24,1}\right\} = 1$ and the rank of the updated covariance matrix is

$$\operatorname{rank}\tilde{\Sigma} = \operatorname{rank}\left\{S_{11} - S_{1,24}(S_{24,24})^{-1}S_{24,1}\right\} + \operatorname{rank} S_{234,234} = 1 + 3 = 4.$$

Thus neighbourhood coordinate descent achieves a positive definite Σ after the first update and thus converges to the right value. Similar phenomena occur when dimensions are larger and graphs more complicated, but the general picture is not clear to us.

4.4. Monitoring convergence

Neighbourhood coordinate descent is implemented in the R package GGM (Marchetti et al., 2020). Since neighbourhood coordinate descent is identical to the graphical lasso algorithm with zero penalty, see p. 637 in Hastie et al. (2016), it is in effect also implemented in the GLASSO package (Friedman et al., 2019). In both of these implementations, convergence is monitored by changes in Σ after a full round of updating, so the algorithm is halted when consequtive values of Σ are identical up to a given tolerance. However, as mentioned earlier, this is unsatisfactory as it only indicates that the algorithm slows down. A more satisfactory indication of convergence is to check that the likelihood equations (3) are satisfied within a tolerance, as the equations (1) and (2) are satisfied exactly throughout the algorithm. However, this demands that the inverse $K = \Sigma^{-1}$ is calculated.

Since this inversion may be computationally demanding, we suggest that it is only done when the algorithm is terminated after slowing down as described above. If the likelihood equations still are not fulfilled to the desired tolerance, we continue the iteration, but now we do not need a full inversion of Σ at every step as there is a simple procedure for updating K after Σ has been updated to $\tilde{\Sigma}$ as shown below. Writing $\tilde{\Sigma}$ in block form with $r = \mathrm{bd}^{\mathrm{c}}(u)$ and $b = \mathrm{bd}(u)$, we have from (14) that

$$\tilde{\Sigma} = \begin{pmatrix} \Sigma_{rr} & \Sigma_{rb} & \tilde{\Sigma}_{ru} \\ \Sigma_{br} & \Sigma_{bb} & \Sigma_{bu} \\ \tilde{\Sigma}_{ur} & \Sigma_{ub} & \Sigma_{uu} \end{pmatrix} = \begin{pmatrix} \Sigma_{cc} & \tilde{\Sigma}_{cu} \\ \tilde{\Sigma}_{uc} & \Sigma_{uu} \end{pmatrix}$$

where $c = r \cup b$ and

$$\tilde{\Sigma}_{ru} = \Sigma_{rb}(\Sigma_{bb})^{-1}\Sigma_{bu}, \quad \tilde{\Sigma}_{bu} = \Sigma_{bu} = S_{bu}, \quad \tilde{\Sigma}_{uu} = \Sigma_{uu} = S_{uu}. \tag{17}$$

With a similar partitioning of $K = \Sigma^{-1}$ and $\tilde{K} = \tilde{\Sigma}^{-1}$ we then have

$$\tilde{K}_{cu} = -\tilde{K}_{cc}\tilde{\Sigma}_{cu}/\Sigma_{uu} = -(\Sigma_{cc})^{-1}\tilde{\Sigma}_{cu}\tilde{K}_{uu}.$$

Hence, using (17) and the fact that $\tilde{\Sigma}_{cc} = \Sigma_{cc}$ implies $\tilde{K}_{ru} = 0$ since

$$\begin{aligned} \tilde{K}_{ru} &= -(\tilde{K}_{rr}\tilde{\Sigma}_{ru} + \tilde{K}_{rb}\Sigma_{bu})/\Sigma_{uu} \\ &= -(\tilde{K}_{rr}\Sigma_{rb}(\Sigma_{bb})^{-1}\Sigma_{bu} + \tilde{K}_{rb}\Sigma_{bu})/\Sigma_{uu} \\ &= -(-\tilde{K}_{rr}(\tilde{K}_{rr})^{-1}\tilde{K}_{rb}\Sigma_{bu} + \tilde{K}_{rb}\Sigma_{bu})/\Sigma_{uu} = 0. \end{aligned}$$

This again implies

$$0_{bu} = \Sigma_{br}\tilde{K}_{ru} + \Sigma_{bb}\tilde{K}_{bu} + \Sigma_{bu}\tilde{K}_{uu} = \Sigma_{bb}\tilde{K}_{bu} + \Sigma_{bu}\tilde{K}_{uu}$$

whence, if we as before let $\beta_{bu} = (\Sigma_{bb})^{-1}\Sigma_{bu}$, we have

$$\tilde{K}_{bu} = -(\Sigma_{bb})^{-1}\Sigma_{bu}\tilde{K}_{uu} = -\beta_{bu}\tilde{K}_{uu}.$$

Also from the above we get

$$1 = \tilde{\Sigma}_{ur}\tilde{K}_{ru} + \Sigma_{ub}\tilde{K}_{ub} + \Sigma_{uu}\tilde{K}_{uu} = \Sigma_{ub}\tilde{K}_{ub} + \Sigma_{uu}\tilde{K}_{uu} = \Sigma_{uu}\tilde{K}_{uu} - \Sigma_{ub}(\Sigma_{bb})^{-1}\Sigma_{bu}\tilde{K}_{uu}$$

and thus

$$\tilde{K}_{uu} = \{\Sigma_{uu} - \Sigma_{ub}(\Sigma_{bb})^{-1}\Sigma_{bu}\}^{-1} = (\Sigma_{uu} - \Sigma_{ub}\beta_{bu})^{-1}.$$

Summarizing the above findings gives the following equations for determining $\tilde{K}_{cu}$:

$$\beta_{bu} = (\Sigma_{bb})^{-1}\Sigma_{bu}, \quad \tilde{K}_{ru} = 0, \tag{18}$$

$$\tilde{K}_{bu} = -\beta_{bu}\tilde{K}_{uu}, \text{ where } \tilde{K}_{uu} = (\Sigma_{uu} - \Sigma_{ub}\beta_{bu})^{-1}. \tag{19}$$

Further we have by standard results for block matrices that

$$(\tilde{\Sigma}_{cc})^{-1} = (\Sigma_{cc})^{-1} = K_{cc} - K_{cu}K_{uc}/K_{uu} = \tilde{K}_{cc} - \tilde{K}_{cu}\tilde{K}_{uc}/\tilde{K}_{uu},$$

whereby

$$\tilde{K}_{cc} = K_{cc} - K_{cu}K_{uc}/K_{uu} + \tilde{K}_{cu}\tilde{K}_{uc}/\tilde{K}_{uu}.$$

Exploiting that $\tilde{K}_{ru} = 0$ further yields

$$\tilde{K}_{rr} = K_{rr} - K_{ru}(K_{ur}/K_{uu}), \tag{20}$$

$$\tilde{K}_{rb} = K_{rb} - K_{ru}(K_{ub}/K_{uu}), \tag{21}$$

$$\tilde{K}_{bb} = K_{bb} - K_{bu}(K_{ub}/K_{uu}) + \tilde{K}_{bu}(\tilde{K}_{ub}/\tilde{K}_{uu}). \tag{22}$$

Combining (18)–(22) yields a procedure for updating K at every subsequent step in the iteration without inverting $\tilde{\Sigma}$. The most expensive operation could be the matrix inversion in (18) if $b = \mathrm{bd}(u)$ is large; but note from (14) that the update of Σ_{ru} may also be expressed as $\tilde{\Sigma}_{ru} = \Sigma_{rb}\beta_{bu}$. Hence β_{bu} has already been computed when updating Σ, so the additional work involved when updating K is not as computationally demanding as it could appear.

4.5. Computational savings

Still, the update of K will take some effort and also here a substantial saving may be obtained by ignoring unnecessary updates. Introduce the maximum column sum norm of a matrix $\Delta = \{\Delta_{\alpha\beta}\}$ as

$$|||\Delta|||_1 = \max_{\beta} \sum_{\alpha} |\Delta_{\alpha\beta}| \tag{23}$$

and let the algorithm terminate when $|||K(\mathcal{G}) - K|||_1 < 2\varepsilon/n$, since the gradient of the dual of the log-likelihood function is

$$\nabla_\Sigma\, \ell^*(\Sigma) = \nabla_\Sigma \left\{ \frac{n}{2}\left(-\log\det\Sigma - d\right)\right\} = \frac{n}{2}\{K(\mathcal{G}) - K\}.$$

This also corresponds to the equations (3) to be satisfied within that tolerance when properly scaled. Since

$$|||K(\mathcal{G}) - K|||_1 = \max_{u\in V} |||K_{ru}|||_1,$$

we may ignore the update corresponding to $u \in V$ if $|||K_{ru}|||_1 < 2\varepsilon/n$ and terminate the algorithm when all updates are ignorable.

4.6. Finding a feasible K

The inverse $K = \Sigma^{-1}$ obtained when the iteration terminates may not have exact zero values for non-edges, and some modification is necessary to achieve that $\check{K} \in \mathbb{S}_{\succ}^{d\times d}(\mathcal{G})$. For example, one might use the procedure described in Algorithm 17.1 of Hastie et al. (2016), which amounts to using only (18)–(19) in the final cycle, but ignoring (20)–(22), pretending that convergence has been achieved. This procedure does not ensure $\check{K}$ to be positive definite and if $\check{K}$ is determined in this way, the result will depend on the order in which the nodes $u \in V$ are visited. It seems difficult to control the outcome of this *ad hoc* procedure and when the dimension d was large, early experiments regularly encountered problems, whence it was abandoned.

Since K is available, it appears more direct to calculate $\check{K}$ from K by replacing elements K_{uv} for $uv \notin E$ with zero, i.e. letting $\check{K} = K(\mathcal{G})$. This also does not ensure that $\check{K}$ is positive definite, but if the algorithm has been run until the equations (3) are fulfilled up to a sufficiently small tolerance measured in a matrix norm (**?**, p. 340–41), it will be.

To see this, we argue as follows. First, to avoid scaling problems, we assume without loss of generality that S has been scaled as a correlation matrix. Let $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ denote the largest and smallest eigenvalue of a symmetric matrix A and let $\Delta = K - \check{K}$, so $\check{K} = K - \Delta$. Then we have

$$\begin{aligned}\lambda_{\min}(\check{K}) &= \inf_{\{x^\top x=1\}} x^\top (K-\Delta)\, x \;\geq\; \inf_{\{x^\top x=1\}} x^\top K\, x - \sup_{\{x^\top x=1\}} x^\top \Delta\, x \\ &= \lambda_{\min}(K) - \lambda_{\max}(\Delta) = \frac{1}{\lambda_{\max}(\Sigma)} - \lambda_{\max}(\Delta)\end{aligned}$$

and hence $\check{K}$ will be positive definite if $\lambda_{\max}(\Delta)\lambda_{\max(\Sigma)} < 1$. Further, since it holds throughout the iterative process that $\operatorname{tr}(S) = \operatorname{tr}(\Sigma) = d$, we have that $\lambda_{\max}(\Sigma) < d$. If $|||\cdot|||$ is any matrix norm, for example the maximum column sum norm in (23), **?**, Theorem 5.6.9 implies

$$\lambda_{\max}(\Delta) \leq |||\Delta|||.$$

Hence if we continue the iteration at least until $|||\Delta||| < d^{-1}$, $\check{K}$ will be positive definite.

For any pair (K, Σ) with $K \in \mathbb{S}_{\succ}^{d\times d}(\mathcal{G})$ and Σ positive definite and satisfying $\Sigma(\mathcal{G}) = S(\mathcal{G})$, the definition of the dual function yields

$$\log\det K - \operatorname{tr}(KS) \leq -\log\det\Sigma - d$$

with equality if and only if the pair (K, Σ) is optimal; hence

$$B = -\frac{n}{2}\left(\log\det\Sigma + d\right) \tag{24}$$

yields an upper bound on the log-likelihood function. Comparing the likelihood function for $\check{K}$ with the upper bound from the final value of Σ, gives a certificate to what extent the log-likelihood function is close to its maximum; the likelihood function is always at most γ from its optimum value where γ is the *duality gap*

$$\gamma = \frac{n}{2}\left\{\operatorname{tr}(\check{K}S) - \log\det(\check{K}\Sigma) - d\right\}.$$

In contrast to neighbourhood coordinate descent, K is feasible during the entire computational process under iterative proportional scaling; the price paid is that there is no easy certificate available, since K^{-1} is not dually feasible unless the optimum has been reached exactly.

4.7. Finding feasible starting values

The dual algorithms demand a dually feasible Σ as a starting value to guarantee convergence, i.e. a positive definite matrix Σ satisfying $\Sigma(\mathcal{G}) = S(\mathcal{G})$. The maximum likelihood estimator exists in the model if and only if such a feasible Σ exists. If S is positive definite, this is not an issue. But if S is based on $n + 1$ observations and $n < d$, some effort is needed to obtain a feasible Σ to initialize the algorithm.

If the graph is sparse so that the number of vertices of degree less than n is sufficiently large we may still run the algorithm and from time to time the rank of Σ may increase by one and after some rounds of updating the resulting Σ may have full rank. But we are not able to give simple conditions for this to happen, even when the maximum likelihood estimator is known to exist.

If there exists a chordal graph $\tilde{\mathcal{G}} \supset \mathcal{G}$ with clique size at most n, the maximum likelihood estimator $\tilde{K}$ in the graphical model determined by $\tilde{G}$ exists and $\tilde{K}$ can be explicitly calculated from formula (5.47) in Lauritzen (1996). Then $\tilde{\Sigma} = \tilde{K}^{-1}$ is dually feasible, and this may be used as a starting value for the algorithm. Also this condition ensures in itself the existence of the maximum likelihood estimator in the smaller model determined by $\mathcal{G}$.

But such a chordal cover might be difficult to find and the maximum likelihood estimator exists more widely than that, as demonstrated in Gross and Sullivant (2018). They establish, for example, existence of the maximum likelihood estimator when $n - 1 > \kappa$ if the κ-core of the graph is empty or, equivalently, if the *maximal coreness* of the graph is smaller than $n - 1$ (Theorem 3.5 loc. cit.). Thus the maximum likelihood estimator exists when $n - 1 \geq 3$ for the $r \times s$ grid (Corollary 3.8 loc. cit.), and also for $n - 1 \geq 4$ for any planar graph $\mathcal{G}$ (Corollary 3.9 loc. cit.). The maximal coreness of a graph may be calculated using the algorithm of **?**, as implemented in the R-package IGRAPH (**?**); hence potential non-existence may be flagged before the estimation algorithms are activated.

5. EMPIRICAL STUDY

5.1. Implementation of the algorithms

The algorithms have been implemented in R (R Core Team, 2021, version 4.3.0). We have made an implementation based on C++ using the RCPPARMADILLO package (Eddelbuettel and Sanderson, 2014, version 0.12.4.1.0). The experiments have been run on AMD EPYC 7302 16-core processors with 64 CPUs and 3 GHz clock frequency. The implementation is naive in the sense that we store the full matrices K and not just the non-zero elements. On the other hand, we store and calculate S_{cc} and its inverse $(S_{cc})^{-1}$ for all relevant subsets c once and for all to use for updating K in (4) so the empirical covariance matrix S is itself not needed. The code producing the results in this section as well as an .html file with a more detailed output is available as supplementary material.

The algorithms were applied to data representing 102 samples of the expression of 6033 genes associated with prostate cancer, originating from Singh et al. (2002) and published in the R package SPLS (Chung et al., 2019). In addition, artificial data were produced with 102 samples of the relevant number of variables, all entries simulated from the standard normal $N(0, 1)$ distribution. A default tolerance of $\varepsilon = 10^{-3}$ was used throughout.

For iterative proportional scaling, the algorithms were run until the likelihood equations were satisfied with an error less that $\varepsilon' = 2\varepsilon/102 = 10^{-3}/51$, when measured by the maximum absolute deviation norm $\|\Sigma(\mathcal{G}) - S(\mathcal{G})\|_\infty$, to match the tolerance to the gradient of the log-likelihood function. For the covariance based algorithm, updates were ignored if $\|\Sigma_{cc} - S_{cc}\|_\infty < \varepsilon'$, as described in Section 3.5.

Similarly, neighbourhood coordinate descent was first run until consequtive values of Σ differed less than ε' when measured by the maximum column sum norm (23). If the smallest eigenvalue of Σ then was larger than ε', a second series of cycles was run, updating K alongside Σ until $|||K(\mathcal{G}) - K|||_1 \leq \varepsilon''$, where $\varepsilon'' = \min(\varepsilon', d^{-1})$ as described in Section 4.6, thereby ensuring that $\check{K} = K(\mathcal{G})$ is positive definite. In this second cycle, updates of any vertex $u \in V$ was ignored if $|||K_{ru}|||_1 < \varepsilon''$ as described at the end of Section 4.5.

5.2. Comparing the algorithms for moderate size dense graphs

We first investigated the computing time for the iterative proportional scaling algorithms and neighbourhood coordinate descent for random graphs with 100 variables of varying density. Model fitting for the scaling algorithms was based on edges or cliques. The median computing time in seconds over five random graphs is displayed in Table 1.

Table 1. *Median computing time in seconds over five random graphs on 100 vertices for the covariance (COV) and concentration (CON) based iterative proportional scaling algorithms, applied cliquewise or edgewise, and for neighbourhood coordinate descent (NCD)*

		Simulated data					Prostate data				
	Expected	Cliquewise		Edgewise			Cliquewise		Edgewise		
Density	# of edges	CON	COV	CON	COV	NCD	CON	COV	CON	COV	NCD
10%	495	0.2	0.01	0.6	0.02	0.02	2	0.1	6	0.2	0.1
30%	1,485	2	0.1	3	0.1	0.03	10	0.3	89	2	0.1
50%	2,475	13	1	9	0.4	0.1	35	2	1,615	17	0.2
70%	3,465	300	16	47	1	0.1	552	14	20,360	232	0.2

The general picture in Table 1 is that the covariance based version is considerably faster than the concentration based algorithm and the effect is stronger when fitting a dense model and when models are updated edgewise. There are several reasons for this: Firstly, for sparse graphs, relatively many cliques would be pairs and hence there is little difference between edgewise or cliquewise updating. Secondly, when the model is dense, the cliques will be relatively large so updating Σ as in (7) will be time consuming. Thirdly, a random dense graph will typically have many large cliques sharing many variables. This means that the same edges are updated several times during each iteration. Finally, the speedup due to ignoring inefficient local updates has smaller effect when using cliques rather than edges.

The neighbourhood coordinate descent algorithm is obviously very fast for this type of models and is less sensitive to the density or sparsity of the graph. It fits the densest model for the prostate data in less than a second.

Computing times are systematically shorter when the algorithms are applied to simulated data. This is most likely a reflection of the fact that the empirical covariance matrix would tend to fit any of the models investigated and therefore be closer to the final estimate from the outset, hence demanding fewer iterations in the fitting procedures. For the densest model, the concentration based scaling algorithm sometimes failed to reach convergence within the specified limit when applied edgewise to the prostate data, even after 50,000 iterations. Hence the concentration based algorithm seems unfeasible when applied edgewise to dense graphs.

We also note that the algorithms in this and similar cases slow down much before it stops, but the likelihood function is still far from the correct value. This highlights the danger using slowness as convergence criterion.

5.3. Comparison with neighbourhood coordinate descent for large, sparse graphs

Next we shall compare computing times for the edgewise covariance based scaling algorithm and neighbourhood coordinate descent for larger dimensions. We use the same data as in the previous section for comparison.

For random graphs with more than 100 vertices, experiments may become complicated as there is a risk that the maximum likelihood estimate does not exist. So to extend the above comparisons to larger scale and a higher degree of sparsity without this difficulty, we first investigate the behaviour for rectangular grids. For thsuch grids, the maximum likelihood estimate exists with probability one for a sample covariance matrix with just three degrees of freedom (Gross and Sullivant, 2018, Corollary 3.8) and hence 102 observations are plenty. Although neighbourhood coordinate descent in principle needs a positive definite starting value to converge, this problem never emerged in any of the cases below. Indeed, the rank of Σ was in all cases catching up during the iteration as described in Section 4.3. For a rectangular grid, there is no difference between updating edgewise or cliquewise, as all cliques consist of exactly one edge. Computing times for the algorithms and various grid sizes are displayed in Table 2.

Table 2. *Computing time in seconds for covariance based scaling (COV) and neighbourhood coordinate descent (NCD) over a rectangular grid*

				Simulated data		Prostate data	
Grid size	# of variables	# of edges	Density	COV	NCD	COV	NCD
20×25	500	955	0.8%	0.3	0.2	1	2
40×25	1,000	1,935	0.4%	2	1	6	7
40×50	2,000	3,910	0.2%	20	7	84	43
80×50	4,000	7,870	0.1%	156	43	483	220

Table 2 indicates that the covariance based scaling algorithm is slightly slower than neighbourhood coordinate descent at this level of sparsity; it fits the model of an 80×50 grid to the prostate data in about eight minutes and to the simulated data in three minutes, whereas the neighbourhood coordinate descent algorithm uses about four minutes for the prostate data and about a minute for the simulated data.

Our final experiments are similar to those in Xu et al. (2015). We consider random graphs that are constructed by adding random edges with probabilities 0.001 and 0.005, and 0.010 to a random tree; the number of variables varying from 500 to 4,000, and up to 6,000 for the sparsest case. These results are displayed in Table 3.

Table 3. *Median computing time in seconds over five random trees with additional edges for covariance based iterative proportional scaling (COV) and neighbourhood coordinate descent (NCD)*

			Simulated data		Prostate data	
Density	# variables	Exp. # of edges	COV	NCD	COV	NCD
0.001	500	623	0.1	0.2	0.3	1
	1,000	1,498	1	1	4	7
	2,000	3,996	47	11	348	72
	4,000	11,993	707	135	5,960	438
	6,000	23,990	3,202	441	24,234	875
0.005	500	1,120	0.3	0.3	2	1
	1,000	3,491	3	1	157	16
	2,000	11,984	314	14	4,926	193
	4,000	43,969	4,728	157	73,883	705
0.010	500	1,742	0.8	0.5	5	2
	1,000	5,984	19	2	516	11
	2,000	21,969	799	31	16,191	522
	4,000	83,939	8,843	287	247,991	2,613

The computing times of the scaling algorithm is comparable to neighbourhood coordinate descent when the graph is sparse and the size is moderate, whereas the latter is considerably faster in higher dimension and higher densities. Neighbourhood coordinate descent is generally less sensitive to sparsity of the graph. Also, we note that the computing time for the algorithms as before are much higher for the prostate data than for the simulated data.

Xu et al. (2015) report best computing times for 4000 variables with simulated data to be 1,044 and 2,407 seconds for densities 0.005 and 0.010 respectively, which should be compared to our 4,682 and 9,681 seconds. However, these numbers are of the are not quite comparable since we have used a much stricter convergence criterion. We made experiments with a weaker criterion, but abandoned them; although the convergence was faster, the accuracy for the maximized likelihood function was not satisfactory.

It is possible that a partitioning of the edges as described in this reference might potentially also speed up the scaling algorithm. However, it is not clear to us whether the overhead of finding this partition and setting up the corresponding structure is included in the CPU times reported by Xu et al. (2015); since this involves simulated annealing, this might be a considerable bottleneck. We also note that when Xu et al. (2015) report results on real data with more than 6000 variables; the model has first been determined with the graphical lasso so tends to fit better than a random graph and as we have seen, this has a strong effect on the computing times. Also, the models used are extremely sparse, with a total of 11,000 to 15,000 edges and a maximal connected component between 4,000 and 5,000 variables. They then report computing times up to 6,000 seconds, which should be compared to our 738 seconds using scaling and 133 seconds using neighbourhood coordinate descent for a random tree with 4,000 variables and random edges added at density 0.001, since this has a similar number of edges.

6. Discussion

We have described a version of iterative proportional scaling for fitting Gaussian graphical models avoiding the NP-complete task of identifying the cliques and updating the concentration and covariance matrices simultationeously without inverting large matrices. The increase of speed is in particular noticeable when graphs are sparse.

Further, we have described a version of neighbourhood coordinate descent which is extremely fast, scales well with the size of the graph, is less sensitive to the density of the graph, and is guaranteed to provide a positive definite concentration matrix with zeros in the right places, when convergent. In addition, the algorithm provides a certificate that guarantees the accuracy of the value of the likelihood function. However, the algorithm might not always be convergent as it formally needs a positive definite starting value which might be difficult to provide. On the other hand, for sparse graphs, the algorithm seems to converge without problems although formal conditions for this to happen is unavailable at the moment.

As a consequence, we recommend to use neighbourhood coordinate descent by default and only use iterative proportional scaling if the former fails to converge.

References

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

Chung, D., Chun, H., and Keles, S. (2019). SPLS*: Sparse Partial Least Squares (SPLS) Regression and Classification*. R package version 2.2-3.

Eddelbuettel, D. and Sanderson, C. (2014). RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441.

Friedman, J., Hastie, T., and Tibshirani, R. (2019). *glasso: Graphical Lasso: Estimation of Gaussian Graphical Models*. R package version 1.11.

Grone, R., Johnson, C. R., de Sá, E. M., and Wolkowicz, H. (1984). Positive definite completions of partial Hermitian matrices. *Linear Algebra and its Applications*, 58:109–124.

Gross, E. and Sullivant, S. (2018). The maximum likelihood threshold of a graph. *Bernoulli*, 24:386–407.

Hara, H. and Takemura, A. (2010). A localization approach to improve iterative proportional scaling in Gaussian graphical models. *Communications in Statistics—Theory and Methods*, 39:1643–1654.

Hastie, T., Tibshirani, R., and Friedman, J. (2016). *The Elements of Statistical Learning*. Springer-Verlag, New York, 2nd edition. 11th printing.

Jiroušek, R. and Přeučil, R. (1995). On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics and Data Analysis*, 19:177–189.

Lauritzen, S., Uhler, C., and Zwiernik, P. (2019). Maximum likelihood estimation in Gaussian models under total positivity. *The Annals of Statistics*, 47:1835–1863.

Lauritzen, S. and Zwiernik, P. (2022). Locally associated graphical models and mixed convex exponential families. *The Annals of Statistics*, 50:3009–3038.

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

Marchetti, G. M., Drton, M., and Sadeghi, K. (2020). *ggm: Graphical Markov Models with Mixed Graphs*. R package version 2.5.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Singh, D., Febbo, P., Ross, K., Jackson, D., Manola, J., Ladd, C., Tamayo, P., Renshaw, A., DAmico, A., Richie, J., Lander, E., Loda, M., Kantoff, P., Golub, T., and Sellers, W. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:203–209.

Speed, T. P. and Kiiveri, H. (1986). Gaussian Markov distributions over finite graphs. *The Annals of Statistics*, 14:138–150.

Uhler, C. (2019). Gaussian graphical models. In Maathuis, M., Drton, M., Lauritzen, S., and Wainwright, M., editors, *Handbook of Graphical Models*, pages 217–238. CRC Press, Boca Raton, FL.

Wermuth, N. and Scheidt, E. (1977). Algorithm as 105: Fitting a covariance selection model to a matrix. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 26:88–92.

Xu, P.-F., Guo, J., and He, X. (2011). An improved iterative proportional scaling procedure for Gaussian graphical models. *Journal of Computational and Graphical Statistics*, 20:417–431.

Xu, P.-F., Guo, J., and Tang, M.-L. (2012). An improved Hara–Takamura procedure by sharing computations on junction tree in Gaussian graphical models. *Statistics and Computing*, 22:1125–1133.

Xu, P.-F., Guo, J., and Tang, M.-L. (2015). A localized implementation of the iterative proportional scaling procedure for Gaussian graphical models. *Journal of Computational and Graphical Statistics*, 24:205–229.

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