DOI: 10.1002/int.21984

WILEY

RESEARCH ARTICLE

Fast learning of scale-free networks based on Cholesky factorization

Vladisav Jelisavcic[1,2] | Ivan Stojkovic[1,3] | Veljko Milutinovic[1] | Zoran Obradovic[3]

[1]School of Electrical Engineering, University of Belgrade, Belgrade, Serbia
[2]Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, Serbia
[3]Center for Data Analytics & Biomedical Informatics, Temple University, Philadelphia, USA

Correspondence
Vladisav Jelisavcic, School of Electrical Engineering, University of Belgrade, Belgrade, Serbia.
E-mail: vladisav@mi.sanu.ac.rs

Funding information
NSF BIGDATA, Grant/Award Numbers: 1447670, 1659998, CNS-1625061; CURE

Abstract
Recovering network connectivity structure from high-dimensional observations is of increasing importance in statistical learning applications. A prominent approach is to learn a Sparse Gaussian Markov Random Field by optimizing regularized maximum likelihood, where the sparsity is induced by imposing L_1 norm on the entries of a precision matrix. In this article, we shed light on an alternative objective, where instead of precision, its Cholesky factor is penalized by the L_1 norm. We show that such an objective criterion possesses attractive properties that allowed us to develop a very fast Scale-Free Networks Estimation Through Cholesky factorization (SNETCH) optimization algorithm based on coordinate descent, which is highly parallelizable and can exploit an active set approach. The approach is particularly suited for problems with structures that allow sparse Cholesky factor, an important example being scale-free networks. Evaluation on synthetically generated examples and high-impact applications from a biomedical domain of up to more than 900,000 variables provides evidence that for such tasks the SNETCH algorithm can learn the underlying structure more accurately, and an order of magnitude faster than state-of-the-art approaches based on the L_1 penalized precision matrix.

1 | INTRODUCTION

A common theme across a wide variety of scientific domains is modeling the dependency structure among a large number of variables in some complex system of interest. Graphical models provide a convenient way to represent and analyze relations in such tasks. The preferred approach is to model the joint probability distribution by learning the Markov Random Field, an undirected instance of a Probabilistic Graphical Model.[1] Although representationally powerful, this approach is very computationally expensive and has intractable inference. Fortunately, additional assumptions and constraints on model expressiveness lead to models with very convenient and attractive properties.

Multivariate Normal (MVN) distribution suits the observation statistics of many real phenomena, and such an assumption in probabilistic models yields the Gaussian Markov Random Fields (GMRFs) model. MVN assumption allows reliable GMRF learning from the data, by finding maximum likelihood estimate of the parameters. When the observations are standardized to have a zero mean, learning the GMRF is equivalent to estimating the inverse covariance matrix (also precision or concentration matrix). Contemporary applications impose the need to mine increasingly larger high-dimensional data sets, therefore efficiency and scalability are crucial. One general way of achieving improvement in computational costs is exploiting the (apparent) structural regularities in the problem. The most commonly used assumption is sparsity, which results in a number of desirable properties: parsimony, increased generalizability, reduced computational and representational requirements. Various methods are proposed to solve the sparse precision matrix estimation problem,[2–8] and will be discussed in the Related Work section. Most of these approaches are based on optimizing the regularized maximum likelihood, where L_1 norm is imposed directly on the entries of a precision matrix.

However, sparsity in the precision matrix can be induced in a number of ways. In this work, we adopt an objective function that regularizes the entries of the Cholesky Factor instead of the precision matrix directly. Therefore, we are implicitly introducing the assumption that the Cholesky factor should be sparse. In the Method section, we show the convenient properties of such formulation based on which we propose an efficient coordinate descent optimization algorithm Scale-Free Networks Estimation Through Cholesky factorization (SNETCH). The approach is highly parallelizable, and with the use of an active set approach, can achieve substantial speedup. Another advantage of the approach is that as we work directly with the Cholesky factor, the precision matrix it recovers will always be Positive Definite (PD), while most other approaches need to regularly check that condition and correct for it (using Cholesky decomposition, Armijo backtrack search, Schur complement, etc.), which is a significant computational burden.

Another structural property often reported in complex systems is a power law degree distribution, resulting in scale-free networks. Scale-free networks possess a number of interesting properties, but the most interesting property regarding this approach is that their Cholesky factor can be made suitably sparse. In the Empirical Evaluation section, we show that finding the maximum likelihood estimation under the Laplace penalty on Cholesky factor is suitable for problems with the scale-free property. Using synthetic examples, we discovered that our method reconstructed the structure more accurately than the state-of-the-art approaches, QUIC[4] and BCDIC,[8] that optimize the common $\|\Sigma^{-1}\|_1$ objective. We have also empirically evaluated speed and scalability of the proposed SNETCH approach for large synthetic scale-free problems of up to 200,000 variables. Execution time results suggest that our approach is more than an order of magnitude faster compared to two state-of-the-art methods for large problems, Big & QUIC[9] and ML-BCDIC.[10] In addition, we also show the efficiency and speedup that may be achieved by the parallelization of our approach.

Finally, SNETCH was consistently fastest when applied on three real data sets, the largest having 904,739 variables. We found that uncovered Gene co-regulation, DNA methylation, and Brain EEG signal networks have scale-free properties, suggesting that our approach has a potential for broad applicability in real problems.

2 | RELATED WORK

Estimation of unknown covariance Σ (or inverse covariance Σ^{-1}) matrices is a problem of reconstructing (approximating) the actual matrix based on samples drawn from the multivariate distribution. The sample covariance matrix $(1/(N-1))\sum(y-\hat{y})(y-\hat{y})^T$ is an unbiased and efficient estimator of the covariance matrix (if the space of covariance matrices is viewed as an extrinsic convex cone in $R^{p\times p}$). However, unbiased empirical covariance and inverse covariance estimation is not stable for high-dimensional problems.

Under the additional assumptions on the underlying distribution (MVN), maximum likelihood can be assessed, and it is the preferred approach to learning the covariance matrix (and its inversion).

2.1 | Maximum-likelihood estimation

A random vector $x \in \mathbb{R}^{p\times 1}$ (a p-dimensional column vector of random variables) has a nondegenerate multivariate normal distribution with a covariance matrix Σ only if $\Sigma \in \mathbb{R}^{p\times p}$ is a PD matrix and the probability density function of x is

$$d(x) = (2\pi)^{-p/2} \det(\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{1}$$

where $\mu \in \mathbb{R}^{p\times 1}$ is the expected value of x. Assuming n independently and identically distributed observations $\{x_1, x_2, ..., x_n\} \in \mathbb{R}^{p\times 1}$, the likelihood is a product of individual density functions.

As the maximum-likelihood estimate of the mean vector μ is the "sample mean" vector $\overline{x} = (x_1 + \cdots + x_n)/n$, we centralize the data by subtracting it, and adopt matrix notation $X = [x_1 - \overline{x}, x_2 - \overline{x}, ..., x_n - \overline{x}] \in \mathbb{R}^{p\times n}$. After dropping the constant scaling factor $(2\pi)^{-np/2}$ the likelihood can be expressed as:

$$\mathcal{L}(\Sigma^{-1}) \propto \det(\Sigma^{-1})^{n/2} \exp\left(-\frac{1}{2}tr\left(X^T\Sigma^{-1}X\right)\right) \tag{2}$$

In practice, it is easier to work with the log-likelihood, and the problem boils down to minimization of negative log likelihood:

$$\hat{\Sigma}^{-1} = \underset{\Sigma^{-1}}{\operatorname{argmin}} \frac{1}{2}tr\left(X^T\Sigma^{-1}X\right) - \frac{n}{2}log|\Sigma^{-1}| \tag{3}$$

subjected to PD constraint $x^T\Sigma^{-1}x > 0, \forall x \in \mathbb{R}^{p\times 1}$.

It is easy to show that in this case, when a random variable has normal distribution, the sample covariance matrix has Wishart distribution and its maximum likelihood estimate S is: $S = XX^T/n$. However, if $n < p$, as is a common case in practice, the sample covariance matrix is not of full rank and its inverse is singular.

2.2 | Penalized maximum-likelihood estimation

That is why additional assumptions are introduced, usually through prior on the parameter values, which is just another perspective on regularization. Initially, it is observed that the precision matrix encodes the independence structure, and that it should be sparse.

2.2.1 | L_0 norm on precision

Sparsity was originally enforced[11] by optimizing log-likelihood with a minimal number of nonzero entries in the precision matrix: $min\|\Sigma^{-1}\|_0$. Since that objective was an expensive combinatorial optimization (e.g., greedy forward-backward search[12]), it did not scale well to the applications and there was necessity for more appropriate approaches.

2.2.2 | L_1 norm on precision

Subsequently, combinatorial L_0 regularization was replaced with its convex relaxation in terms of L_1 norm on precision matrix.[13] The introduced L_1 was suitable because it is convex, yet it still leads to a sparse solution. The neighborhood selection approach[5] used LASSO[14] formulation to select connected variables by solving a number of decoupled problems. Initial approaches[15] were computationally expensive, and suitable for only small problems. Graphical LASSO[3] improved efficiency by proposing a faster coordinate descent algorithm. Other optimization approaches based on the first order derivative followed, like a projected subgradient method,[2] greedy coordinate ascent[7] and proximal gradient (iterative thresholding algorithm).[6]

Among the first approaches that utilize the second order gradient information is QUIC,[4] which uses second order Newton method-based coordinate descent with Armijo search. The direction of descent is first found using the active set-based coordinate descent, which is followed by inexact line search to find the step with the sufficient decrease based on Armijo rule. Iterates are held within a PD cone by performing Cholesky decomposition, and finding the step size that ensures PD. Improving the QUIC approach for higher dimensional data[9] avoids some computational burden while searching for the direction of descent by clever use of a conjugate gradient method. The work[9] also recommends avoiding the Cholesky decomposition step, as with the increase in the problem size it becomes the most expensive part. That is because for the types of graphs investigated, the fill-in of the Cholesky factors grows nonlinearly with the problem size. Instead, positive definiteness is ensured by the Schur complement method. The BCDIC approach[8] states further advances in this direction by optimizing several variables at the same time, using the block coordinate descent approach. Its extension[10] goes one step further by employing a hierarchical multilevel framework. All three large-scale covariance estimation approaches[8–10] use the fast METIS graph partitioning algorithm in order to differentiate submatrices with higher and lower edge densities. This approach naturally fits the graphs generated through stochastic block model (community networks) and Erdos–Renyi type process (random binary networks).

2.2.3 | L_1 norm on Cholesky factor

Another formulation of the problem, CSEPSNL, was proposed in Ref. 16 There, the L_1 norm is applied on entries of a Cholesky factor L (where $\Sigma^{-1} = LL^T$), instead on the entries of the precision matrix Σ^{-1} itself.

In this work, we propose a novel optimization algorithm for the objective proposed in CSEPNL. We carefully analyze the setup, and show that it is well suited for the coordinate descent. It also allows an active set approach and belongs to a class of embarrassingly parallel algorithms, as optimization of the columns of a Cholesky factor is decomposable (totally independent of each other). We reconsider the effects of Cholesky factors fill-in for high-dimensional networks. As it turns out, scale-free networks, if ordered properly, are known to have sparse Cholesky factors. This is the main motivational force behind our approach, as having sparse Cholesky factors available at all times (without additional calculations) enables fast $O(p)$ computation of a log det term and ensures positive definiteness, thus enabling us to avoid step search and PD validation altogether. Also, as we will show, by exploiting the sparsity, a fast and efficient coordinate descent method can be derived based on the active set approach, which is orders of magnitude faster than the previous approaches. Furthermore, we will show in several examples that even if the data is not generated according to the scale-free graph assumption, our method is still applicable and its results are comparable to the previous approaches.

2.2.4 | Scale-free prior

There is also work oriented toward learning the scale-free graph structure from continuous data, mainly by enforcing the exponential degree distribution on the learned graph, by using additional regularization penalties.[17] Scale-free structure prior on precision and a novel Metropolis–Hastings based sampling method for learning the parameters has been proposed.[18] A ranking-based method to dynamically estimate the degree of a node, and a novel optimization method based on an alternating direction method of multipliers has been introduced.[19] Another approach based on a power law degree regularization with the minorize-maximize algorithm was demonstrated.[20] Although all approaches[17–20] reveal scale-free structures, methods where power law degree is enforced appear to be computationally inefficient, and do not scale as well as methods for more general inverse covariance estimation. Here, we do not claim that our formulation enforces scale-free structure in its solution, but just that scale-free structure is a very suitable problem for our approach. Therefore, approaches that enforce scale-free structure by using different regularization terms are out of the scope of this work.

Other approaches to inverse covariance estimation that impose different constraints like low-rank structure[21] and ridge type of penal[22] are also not in the focus of this study.

3 | METHOD

We adopt a parametrization based on Cholesky decomposition L, and generalize the scalar penalty λ into a nonnegative matrix Λ (in a way similarly proposed in[23]), where $*$ is Hadamard product:

$$f(L) = \frac{1}{2}tr\left(X^T LL^T X\right) - \frac{n}{2}log|LL^T| + \|\Lambda * L\|_1 \tag{4}$$

To derive optimization procedure, we analyze the objective (4) and follow the standard approach to separate it into differentiable $g(L)$ and nondifferentiable $h(L)$ (sparsity-inducing) parts. Let us first consider the differentiable part, where the trace is invariant to cyclic permutations, and the matrix product and the determinant are commutative operations (the determinant of a product is equivalent to the product of its determinants), allowing us to write:

$$g(L) = \frac{1}{2}tr(LL^T XX^T) - \frac{n}{2}log(|L||L^T|) \tag{5}$$

We can substitute empirical covariance estimate S in the first term, and conclude that the determinant of the Cholesky factor in the second term is the product of its diagonal elements, to obtain:

$$g(L) = \frac{n}{2}tr(LL^T S) - \frac{n}{2}log\left(\prod_{i=1}^{p} L_{ii}^2\right) = \frac{n}{2}tr(LL^T S) - n\sum_{i=1}^{p} log(L_{ii}). \tag{6}$$

In the new reparametrization of the objective function, the log det term can be efficiently computed in $O(p)$ time. Furthermore, the data-dependent part can easily be separated into several independent sub-objectives, which is the prerequisite for scaling up to high-dimensional data. In order to do so, L can be observed as a sum of p "rank 1" $p \times p$ matrices L'_j, where each matrix contains the j_{th} column of the L matrix and the rest of entries are set to zero (dropping the constant factor n):

$$g(L) = \frac{1}{2}\sum_{j=1}^{p} tr\left(L'_j {L'_j}^T S\right) - \sum_{j=1}^{p} log(L_{jj}). \tag{7}$$

Now we can derive the expressions for the derivatives of the differentiable function $g(L)$. There are two separate types of variables: off-diagonal (when $i = j$), and diagonal (when $i \neq j$), so we will have two sets of update equations. We continue by using the standard matrix calculus:

$$\frac{\partial tr(L'_j {L'_j}^T S)}{\partial L_{ij}} = \frac{1}{2} tr\left(\left(\frac{\partial L'_j}{\partial L_{ij}} {L'_j}^T + L'_j \frac{\partial {L'_j}^T}{\partial L_{ij}}\right) S\right) = tr\left(\frac{\partial L'_j}{\partial L_{ij}} {L'_j}^T S\right) = L_j^T S_i. \tag{8}$$

Here S_i denotes the ith row of matrix S, and L_j denotes the jth column vector of the matrix L. To get the expression (8) we used the fact that matrix $\frac{\partial L_j}{\partial L_{ij}}$ is actually a zero matrix with a single 1 on the ith row and jth column, and the cyclic property of the trace operator for symmetric matrices:

$$Tr\left(\frac{\partial L_j}{\partial L_{ij}} L_j^T S\right) = Tr\left(L_j \frac{\partial L_j^T}{\partial L_{ij}} S\right) \tag{9}$$

Finally, we get the derivative of the smooth part of the objective function:

$$\nabla g(L)_{ij} = \begin{cases} \sum_{k=1}^{p} L_{kj} S_{ik}, i \neq j \\ \sum_{k=1}^{p} L_{kj} S_{ik} - \frac{1}{L_{ii}}, i = j \end{cases} \tag{10}$$

Now we consider the penalty term. There is a rich body of literature for optimizing the L_1 penalized convex functions. For our problem, coordinate descent with the active set approach naturally imposes as an attractive method, since our penalty term is separable. Furthermore, the smooth part of function $g(L)$ can also be separated (as shown) into p independent tasks, one for each column, therefore enabling a high degree of parallelism, since p coordinates can be updated in parallel. In order to derive update equations, we first observe the sub-differential of $h(L)$, and its (i, j) component:

$$\nabla h(L)_{ij} = \begin{cases} \lambda_{ij}, L_{ij} > 0 \\ -\lambda_{ij}, L_{ij} < 0 \\ \in [-\lambda_{ij}, \lambda_{ij}], L_{ij} = 0 \end{cases} \tag{11}$$

The sub-gradient of the whole optimization function (4) is:

$$\nabla f(L)_{ij} = \begin{cases} \sum_k L_{kj} S_{ki} + sgn(L_{ij})\lambda_{ij}, L_{ij} \neq 0, i \neq j \\ \sum_k L_{kj} S_{kj} - \frac{1}{L_{ii}} + \lambda_{ij}, i = j \\ S\left(\sum_k L_{kj} S_{ki}, \lambda_{ij}\right), L_{ij} = 0, i \neq j \end{cases} \tag{12}$$

where S(z,r) is a soft thresholding function:

$$S(z,r) = sign(z)max(|z| - r, 0). \tag{13}$$

Now that we have expressions for sub-gradient we can optimize the objective (4) by using coordinate descent. Single variable updates can be found from optimality conditions (12). Again, we first show the single variable update equation for off-diagonal variables (13), which is simply a solution to the linear equation with a soft threshold:

$$L_{ij}^{t+1} = \begin{cases} -\frac{\sum\limits_{k\neq i} L_{kj}^t S_{ik} + sign(L_{ij}^t)\lambda_{ij}}{S_{jj}}, L_{ij}^t \neq 0 \\ -\frac{\sum\limits_{k\neq i} L_{kj}^t S_{ik} - sign(M)\lambda_{ij}}{S_{jj}}, L_{ij}^t = 0, M \geq \lambda_{ij} \\ 0, L_{ij}^t = 0, |M| < \lambda_{ij} \end{cases} \tag{14}$$

where $M = \sum\limits_{k\neq i} L_{kj}^t S_{ik}$.

If L is a diagonal matrix (which is the case for the first iteration, when we initialize the algorithm with identity matrix), update rule from Equation 14 is calculated in $O(1)$ time (a result analogous to one in[4]):

$$L_{ij}^1 = \begin{cases} -\frac{S_{ij} - sign(S_{ij})\lambda_{ij}}{S_{jj}}, |S_{ij}| > \lambda_{ij} \\ 0, |S_{ij}| < \lambda_{ij} \end{cases} \tag{15}$$

Starting set of nonzero values can be obtained from an update rule (15). This set constitutes the initial active set, and lower bounds the time complexity of our algorithm, since computing it requires $O(p^2)$ time.

Update for diagonal entries are solutions of the following quadratic equations:

$$L_{ii}^{t+1} = \frac{-\left(\sum\limits_{k\neq i} L_{ki}^t S_{ik} + \lambda_{ii}\right) + \sqrt{\left(\sum\limits_{k\neq i} L_{ki}^t S_{ik} + \lambda_{ii}\right)^2 + 4S_{ii}}}{2S_{ii}} \tag{16}$$

The second solution to the quadratic equation is always discarded, since L_{ii} must be positive in order for the Σ^{-1} matrix to be PD. By looking at the update Equations (14) and (16) we notice that if the column L_j is sparse with complexity $O(s_j)$, calculating updates will also have $O(s_j)$ time complexity.

Update Equations (14) and (16) represent an analytic solution for coordinate-wise optimization procedure for SNETCH described in Algorithm 1. After the initialization with identity matrix, the coordinates are iteratively updated until the convergence criteria is satisfied, that is the change in gradient drops below the threshold ϵ. Precision is obtained from optimized Cholesky factor $\Sigma_*^{-1} = L_*^T L_*$.

Algorithm 1 SNETCH algorithm

```
1: procedure $optimizeL(L, Y, \lambda, \epsilon)$
2:     for each column of L do
3:         $S_{active} \leftarrow initActiveSet(Y, \lambda)$
4:         while $|\nabla L_j| < \epsilon$ do
5:             for $i \in S_{active}$ do
6:                 $L_{ij} \leftarrow argmin(f_{ij}(L_j, \lambda))$          ▷ Defined in (14), (16)
7:             $t \leftarrow \lambda bound(L_j)$
8:             if $t < \lambda$ then
9:                 $S_{active} \leftarrow initActiveSet(Y, t)$
    return L
```

3.1 | Model interpretation

Since finding sparse Cholesky does not necessarily force a sparse precision matrix, additional investigation is needed to justify objective (4) and present a range of problems where it can fit into the sparse inverse covariance estimation framework. Let us point out the two well-known facts about Cholesky factorization. First, finding the sparsity pattern of the Cholesky factors corresponds to finding a chordal completion of the precision matrix. Second, fill-in of the Cholesky factors largely depends on the node ordering in the graph. Since the penalty of objective (4) tries to find the sparse chordal graph that completes (for a single chosen ordering) the precision matrix fitted to our data, this type of optimization largely depends on the type of structure hidden in the data. It has been suggested earlier[24,25] that scale-free graphs naturally fits this setting, as they are easily triangulated. As learning the scale-free GMRFs represent an important and still not enough explored problem, it is central in our experiments.

One of the biggest challenges is finding the right ordering of the variables, as this is known to be an NP-complete problem.[26] Several heuristics exist for solving this problem. In our experimental evaluation, we use Approximate Minimum Degree algorithm as a substep for our optimization method, which shows promising results. Further discussion regarding the model complexity and convexity can be found in the Appendices A and B, respectively.

4 | EMPIRICAL EVALUATION

To empirically test the characteristics of the proposed approach and its solution, we conducted extensive evaluations on the synthetically generated problems of interest. In such examples, we are comparing our SNETCH against two state-of-the-art approaches for learning sparse precision matrices of a medium size (i.e., all data can fit the memory), QUIC[4] and BCDIC.[8]

Samples from the desired distribution were generated by first designing an appropriate sparse precision matrix, in which the structure of interest is encoded. Subsequently, the samples were drawn from the MVN distribution with zero mean and the designed precision structure. In our experiments, several types of graphs were simulated, each posing a different type of challenge. As in evaluation approaches described in Refs. 4 and,8 we generated a linear chain and a random binary graph. Also, we analyzed two other synthetical problems of interest, a graph with sparse Cholesky factors (an "ideal" assumption for our model) and a scale-free graph, generated using the preferential attachment process.

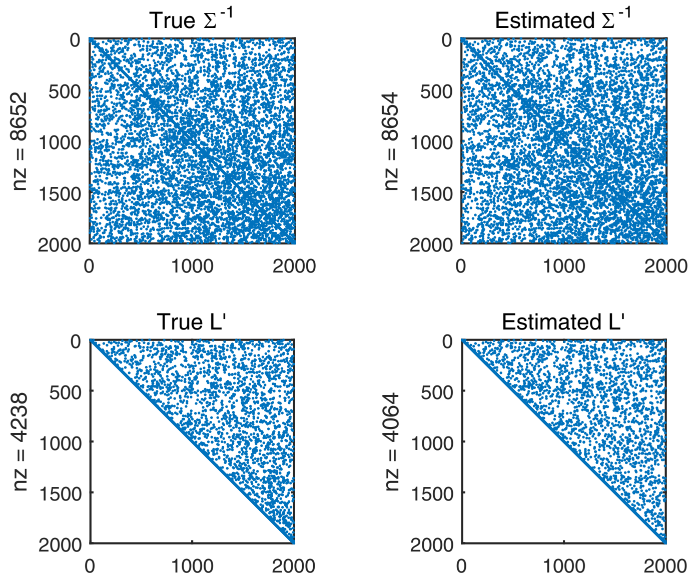

FIGURE 1 Synthetic examples obtained from a randomly sampled sparse Cholesky factor

TABLE 1 Performance on the random Cholesky problem

Random Cholesky factor (*p*=2,000; *n*=2,000; true *nz* = 8,652)						
Model	**Time (s)**	**Jaccard**	**Precision**	**Recall**	**λ**	***nz***
SNETCH	1.21	0.745	0.854	0.854	0.0818	8,654
BCDIC	9.04	0.500	0.667	0.667	0.0878	8,654
QUIC	20.2	0.501	0.668	0.668	0.0878	8,654

We start by generating a synthetic example based on our modeling assumption, that is, the Cholesky factor of a precision matrix is sparse. Random Cholesky graph structure was obtained by randomly assigning a number of nonzero entries in the Cholesky factor, and subsequently, a sparse precision matrix was obtained as a product of sparse Cholesky factors. Sparsity is quantified with a number of nonzero ("*nz*" in Figures and Tables) entries in precision matrix.

In Figure 1, on the left we see the sparsity patterns of true precision, and on the right, the precision estimated by our approach. At the bottom of the figure are the respective Cholesky factors, which are sparse by design. In Table 1, the results of all three competing algorithms are presented and it can be seen that our SNETCH approach achieves superior performance in terms of speed and quality of the obtained solution. The sparsity of the estimated solutions was tuned by hyperparameter λ to obtain the number of *nz* elements close to the underlying ground truth—true *nz*, but more importantly to be close to competing approaches, for the sake of comparison.

Quality of the solution we measured using Jaccard index, Precision and Recall, all suggesting our approach recovered true nonzero elements much better than competition (Table 1). However, in other works, the approaches were typically evaluated under different structural assumptions. Therefore, we also compared the performances on the most common synthetic examples: linear chain graph and randomly assigned sparse precision matrix.

Linear chain was created by specifying appropriate patterns of graph connections in the adjacency matrix. The adjacency matrix was subsequently turned into Laplacian and made diagonally dominant

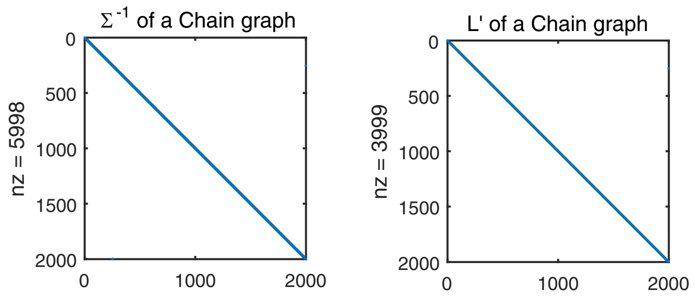

FIGURE 2 Synthetic example depicting linear chain sparsity structure

TABLE 2 Performance on the linear chain problem

Linear chain structure (p=2,000; n=2,000; true nz = 5,998)						
Model	**Time (s)**	**Jaccard**	**Precision**	**Recall**	λ	nz
SNETCH	0.69	1	1	1	0.2	5,998
BCDIC	5.45	1	1	1	0.2	5,998
QUIC	3.39	1	1	1	0.2	5,998

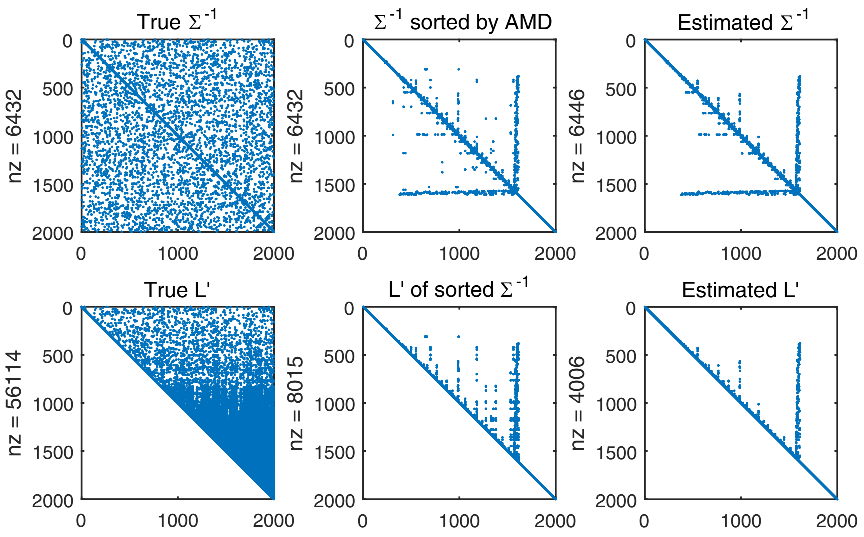

FIGURE 3 Synthetic example obtained by generating sparse random precision matrix

by adding a small value to the diagonal. The resulting matrix was taken as an underlying precision matrix for coloring the samples from plain MVN distribution.

In Figure 2, it is shown that the (ordered) linear chain example also has sparse precision and sparse Cholesky factor, and in Table 2, the results are showing that all approaches can perfectly reconstruct the underlying structure, but SNETCH did it much faster.

The Random Precision matrix was obtained by assigning a prespecified number of nonzero elements in the precision. A matrix is made symmetric by the addition of its transpose, and PD was assured by adding some small positive constant to the diagonal. This is the most common way of generating synthetic examples, and an example is shown in Figure 3.

TABLE 3 Performance on the random precision problem

Random Precision Matrix (p=2,000; n=2,000; true nz = 6,432)						
Model	**Time (s)**	**Jaccard**	**Precision**	**Recall**	**λ**	**nz**
SNETCH	0.62	0.840	0.911	0.914	0.162	6,446
BCDIC	7.78	0.986	0.992	0.994	0.11	6,440
QUIC	3.24	0.986	0.991	0.994	0.11	6,444

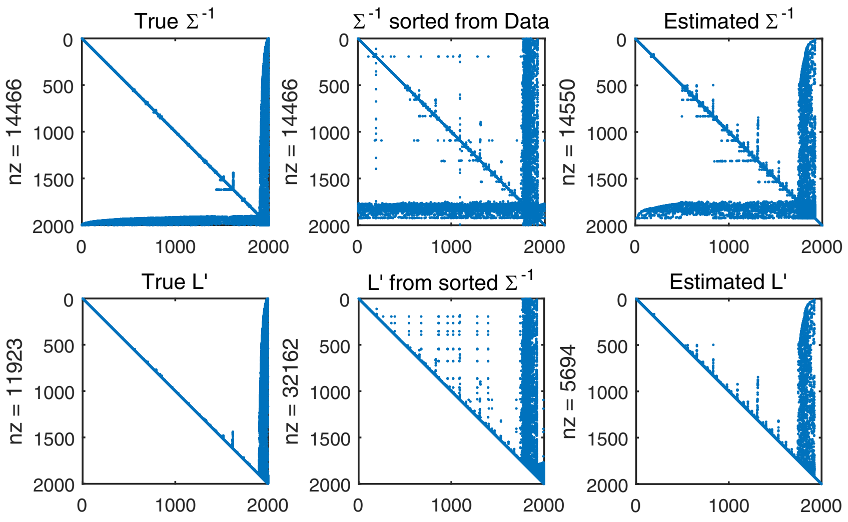

FIGURE 4 Synthetic example depicting scale-free type of structure

However, a Cholesky factor is dependent on ordering the variables (rows and columns), and it can be made sparse (up to a certain point) by appropriately rearranging the order. One known heuristic is Approximate Minimum Degree (AMD) sorting, which tends to result in sparse Cholesky factors. Such a property is visualized in the middle plots of Figure 3, where the true precision was sorted by AMD. After reordering, our approach was able to reconstruct the precision faster but slightly less accurate (for the given sample size) than the competing approaches as shown in Table 3.

Nevertheless, in practice, the most interesting structures lies somewhere in-between the two extremes of random (no-structure) and linear chain (total-order). We will therefore, from now on, turn our focus to such highly relevant structures.

4.0.1 | Scale-free networks

Scale-free networks have very intriguing characteristics,[27,28] and are found in a number of real-world phenomena, from paper citation networks to world wide web and networks that regulate processes in living cells.[29] We evaluated the appropriateness of our approach to discovery of structures with the scale-free property.

We generated scale-free problems by the process of preferential attachment with Poisson growth.[30] The graph is started with several (pre-specified) groups of densely connected nodes (cliques), and every new incoming node was added to the graph connecting to k of existing ones, where k was drawn from the Poisson distribution (based on the probability proportional to the node degree).

TABLE 4 Performance on a scale-free problem

Scale-free structure (p=2,000; n=2,000; true nz = 14,466)						
Model	**Time (s)**	**Jaccard**	**Precision**	**Recall**	λ	nz
SNETCH	0.78	0.407	0.577	0.580	0.0775	14,550
BCDIC	5.89	0.364	0.532	0.536	0.0717	14,548
QUIC	4.25	0.364	0.532	0.536	0.0717	14,550

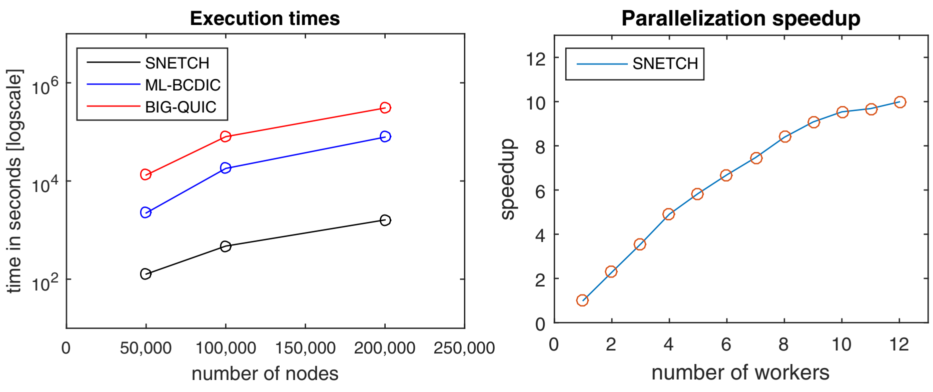

FIGURE 5 Left panel presents comparison of execution times for competing methods on scale-free problems of up to 200,000 variables, performed in a single threaded setup. On the right panel is parallelization speedup that can be achieved for SNETCH approach on the problem of 200,000

An example of precision with scale-free properties is depicted in Figure 4. On the left side is the true underlying precision, and its Cholesky factor is sparse. As typically, in applications, the variables are not sorted, the middle plots show how they can be appropriately sorted, by sorting the initial active set obtained from the observed data. On the right is the structure estimated by our approach, and results from Table 4 again point that our approach reconstructs the underlying dependencies better and faster than the alternatives.

4.0.2 | Scalability

In all those examples, our algorithm was the fastest. The next more involved benchmark aimed to characterize SNETCH against more scalable alternatives BIG & QUIC[9] and ML-BCDIC[10] on much larger problems where data is larger than the computer memory. Towards this objective, we generated synthetic problems with the scale-free structure of various sizes: 50,000; 100,000; and 200,000 variables.

In the left panel of Figure 5, for logarithmic y-axes, it can be seen that our approach achieves execution times that are sometimes two orders of magnitude faster than the state-of-the-art competitors. Table 5 presents the same results in greater details, and it should be noted that in all experiments SNETCH [1], BIG & QUIC, [2] and ML-BCDIC [3] were executed in a single threaded setup.

[1] https://github.com/vladisav/SNETCH

[2] http://bigdata.ices.utexas.edu/software/1035/

[3] https://github.com/erantreister/Multilevel-BCDIC.m

TABLE 5 Run time (sec) comparison for SNETCH versus BIG & QUIC and ML-BCDIC methods on synthetic scale-free data

Scale-free		SNETCH	ML-BCDIC	BIG & QUIC
p= 50,000	time	126.9	2,230	13,221
249,970	*nz*	281,260	284,260	286,696
p=100,000	time	539.4	18,031	74,425
499,930	*nz*	765,990	752,250	757,194
p=200,000	time	1,611.8	78,349	307,081
2,199,434	*nz*	2,061,680	2,070,664	2,070,740

4.0.3 | Parallelization

Moreover, as our approach is highly parallelizable, we show on the largest example how much speedup may be achieved. The right panel of Figure 5 shows how the optimization part can be speedup for up to 12 workers. Since each of the sub-objectives can be optimized in parallel, the linear speedup is expected. Experimental results are in agreement with the expected degree of parallelism and even show superlinear speedup for some configurations, which can be explained by better cache utilization and less communication overhead, since only a small fraction of data is needed to optimize each column. Parallelization was achieved by using the Matlab Parallel Computing Toolbox.

4.1 | Applications

Uncovering the connectivity pattern among a large number of variables is a common topic across a number of domains, and sparse GMRF learning was used for tackling the problems from the biological functioning of a cell, to brain connectivity patterns, up to interactions in the social networks.

In order to characterize the structural properties of some real data sets, we applied our SNETCH approach on three high-impact biomedical applications: human gene expression during respiratory infection,[31] DNA methylation in healthy people,[32] and brain electroencephalogram (EEG) signals.[33]

The gene expression data set was obtained from GEO [1], and it constitutes 12,023 probes (features, genes) measured in 2,886 blood samples drawn from human subjects infected with respiratory viruses (H3N2 Influenza type H3N2, Rhinovirus and Respiratory Syncytial Virus).[31] The data was standardized to have a zero mean and normalized to have unit values on the diagonal of sample covariance matrix. Extracting gene co-regulation network is of high biological interest as co-expressed genes are functionally related, often involved in the same pathways or controlled by the same regulatory process.[34]

The DNA methylation data set is also obtained from GEO [2] and has methylation levels measured in 473,034 genomic locations, for 656 healthy humans (samples) involved in the study on the utility of methylation markers as predictors of biological age.[32] The data were transformed with logit (inverse sigmoidal logistic) function, then standardized and normalized. Extracting DNA methylation correlation network is important as methylation is one of the most stable epigenomic mechanisms through which the environment affects the functioning on an organism.[35]

The brain EEG data set is obtained from an online repository [3]. It consists of 910,476 signals measured for 2 s by Emotive EPOC device. After removing 5,737 time-series that have corrupted readings

[1] GSE73072, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE73072

[2] GSE40279, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE40279

[3] http://www.mindbigdata.com/opendb

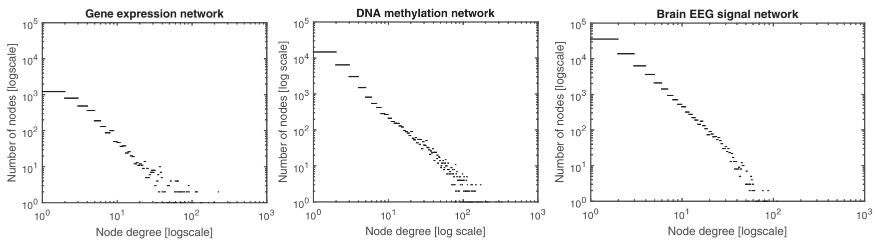

FIGURE 6 Degree distributions of extracted Gene co-expression Network, DNA methylation network and Brain EEG signal network exhibit strong scale-free property

TABLE 6 Network structure learning time

Data set	Size	SNETCH	ML-BCDIC	BIG & QUIC
Gene Expr	time	130.5	342.1	1,442.9
p=12,023	*nz*	35,123	33,677	35,921
DNA Meth	time	12,667	400,000+	400,000+
p=473,034	*nz*	613,868	–	–
Brain EEG	time	32,803	400,000+	400,000+
p=904,739	*nz*	1,090,607	–	–

(constant signal), the data was standardized and normalized. Dependencies among recorded events may provide important insight on how the brain responds to different situations.[33]

For certain sparsity levels, we have obtained a gene co-regulation network of 35,123 nonzero elements, a DNA methylation correlation network consisting of 613,868 nonzero elements, and a brain EEG correlation network of 1,090,607 elements. We have characterized their node degree patterns, which are shown in Figure 6. It can be seen that they all have a very distinct pattern of a power law, which on log–log scale appears as a straight line. These real examples, and a number of others, suggest the potential for wide applicability of the proposed SNETCH approach.

Table 6 shows that our approach estimated the structure in smallest amount of time for each of the data sets, while on two high-dimensional problems competing approaches have not finished optimization by 400,000 s (Big & QUIC reported finishing the second iteration at 104,596 s on DNA data, and the first iteration at 140,013 s on EEG data.)

5 | DISCUSSION AND FUTURE WORK

We presented a new method for learning large-scale GMRFs, and showed that it is a well-suited tool for problems with the scale-free structure. Comparing to previous work on large scale GMRF learning, our method is better fitted to problems where the chordal pattern can be fitted efficiently, a property suggested by others to be true for scale-free graphs. We also showed that the same method can be applied to several other sparse problems with different types of a structure with reasonably good performance. Results show significant speedup on several synthetical data sets. Finally, our method is applied to gene expression, DNA methylation, and brain EEG data, where we found scale-free structures. Since sparsity of a Cholesky factor largely depends on proper node labeling (which is known to be NP-complete problem), we proposed using the Approximate Minimum Degree algorithm as a substep in order to alleviate this problem, which proved to be a reasonably good solution in our experiments. Further investigation is needed in order to improve the model with more advanced node ordering strategy. Also, in our experiments, our first order method performed well, but implementing the presented approach into the quadratic solver could provide better convergence rates. Our approach might be further extended for learning a discriminative version of sparse GMRF named sparse Gaussian Conditional Random Field, which is suited for structured regression problems.[36]

ACKNOWLEDGEMENT

This research was supported in part by NSF BIGDATA grant 1447670, NSF 1659998, NSF CNS-1625061, and Pennsylvania Department of Health CURE grant.

REFERENCES

1. Koller D, Friedman N. Probabilistic Graphical Models: Principles and Techniques. Cambridge, MA: MIT Press; 2009.
2. Duchi J, Gould S, Koller D, et al. Projected subgradient methods for learning sparse Gaussians. In: *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*. Corvallis, OR: AUAI Press; 2008.
3. Friedman J, Hastie T, Tibshirani R. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*. 2008;9(3):432–441.
4. Hsieh C-J, Dhillon IS, Ravikumar PK, Sustik MA. Sparse inverse covariance matrix estimation using quadratic approximation. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ, eds. Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press; 2011:2300–2338.
5. Meinshausen N, Bühlmann P. High-dimensional graphs and variable selection with the lasso. *Ann Statist*. 2006;34:1436–1462.
6. Rolfs B, Rajaratnam B, Guillot D, Wong I, Maleki A. Iterative thresholding algorithm for sparse inverse covariance estimation. In: Bartlett P, ed. Advances in Neural Information Processing Systems. San Deigo, CA: NIPS; 2012:1574–1582.
7. Scheinberg K, Rish I. Learning sparse Gaussian Markov networks using a greedy coordinate ascent approach. In: Daelemans W, Morik K, Machine Learning and Knowledge Discovery in Databases. Berlin: Springer; 2010:196–212.
8. Treister E, Turek J. A block-coordinate descent approach for large-scale sparse inverse covariance estimation. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ. Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press; 2014:927–935.
9. Hsieh C-J, Sustik MA, Dhillon IS, Ravikumar PK, Poldrack R. Big & quic: Sparse inverse covariance estimation for a million variables. In: Burges CJC, Botton L, Welling M, Ghahramani Z, Weinberger KQ. Advances in Neural Information Processing Systems. San Deigo, CA: NIPS; 2013:3165–3173.
10. Treister E, Turek J, Yavneh I. A multilevel framework for sparse optimization with application to inverse covariance estimation and logistic regression. *SIAM J Sci Comput*. 2016;38(5):S566–S592.
11. Dempster AP. Covariance selection. *Biometrics*. 1972;28:157–175.
12. Lauritzen SL. Graphical Models. Oxford, UK: Clarendon Press; 1996.
13. Banerjee O, El Ghaoui L, d'Aspremont A, Natsoulis G. Convex optimization techniques for fitting sparse Gaussian graphical models. In: *Proceedings of the 23rd International Conference on Machine Learning*. New York, NY: ACM; 2006:89–96.
14. Tibshirani R. Regression shrinkage and selection via the lasso. *J Roy Stat Soc. B (Methodol)*. 1996;58:267–288.
15. Yuan M, Lin Y. Model selection and estimation in the Gaussian graphical model. *Biometrika*. 2007;94(1):19–35.
16. Huang JZ, Liu N, Pourahmadi M, Liu L. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*. 2006;93(1):85–98.

17. Defazio A, Caetano TS. A convex formulation for learning scale-free networks via submodular relaxation. In: Pereira F, Burges CJC, Botton L, Weinberger KQ. Advances in Neural Information Processing Systems. San Deigo, CA: NIPS; 2012:1250–1258.
18. Sheridan P, Kamimura T, Shimodaira H. A scale-free structure prior for graphical models with applications in functional genomics. *PLoS One*. 2010;5(11):e13580.
19. Tang Q, Sun S, Xu J. Learning scale-free networks by dynamic node specific degree prior. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. New York, NY: ACM; 2015:2247–2255.
20. Liu Q, Ihler AT. Learning scale free networks by reweighted l1 regularization. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. New York, NY: ACM; 2011:40–48.
21. Han L, Zhang Y, Zhang T. Fast component pursuit for large-scale inverse covariance estimation. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY: ACM; 2016:1585–1594.
22. Honorio J, Jaakkola TS. Inverse covariance estimation for high-dimensional data in linear time and space: spectral methods for riccati and sparse models. In: *Proceedings of the 29th Conference on Uncertainty in AI*. Cambridge, MA: MIT Press; 2013.
23. Bien J, Tibshirani RJ. Sparse estimation of a covariance matrix. *Biometrika*. 2011;98(4):807–820.
24. Golumbic MC. Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57). Amsterdam, the Netherlands: North-Holland Publishing Co.; 2004.
25. Sioutis M, Condotta J-F. Tackling Large Qualitative Spatial Networks of Scale-Free-Like Structure. Cham: Springer International Publishing; 2014.
26. George A, Liu WH. The evolution of the minimum degree ordering algorithm. *SIAM Rev*. 1989;31(1):1–19.
27. Barabási A-L, Albert R. Emergence of scaling in random networks. *Science*. 1999;286(5439):509–512.
28. Strogatz SH. Exploring complex networks. *Nature*. 2001;410(6825):268–276.
29. Barabási A-L. Scale-free networks: a decade and beyond. *Science*. 2009;325(5939):412–413.
30. Sheridan P, Yagahara Y, Shimodaira H. A preferential attachment model with poisson growth for scale-free networks. *Ann Inst Stat Math*. 2008;60(4):747–761.
31. Liu T-Y, Burke T, Park LP, et al. An individualized predictor of health and disease using paired reference and target samples. *BMC Bioinform*. 2016;17(1):1.
32. Hannum G, Guinney J, Zhao L, et al. Genome-wide methylation profiles reveal quantitative views of human aging rates. *Mol Cell*. 2013;49(2):359–367.
33. Wendling F, Ansari-Asl K, Bartolomei F, Senhadji L. From eeg signals to brain connectivity: a model-based evaluation of interdependence measures. *J Neurosci Meth*. 2009;183(1):9–18.
34. Stuart JM, Segal E, Koller D, Kim SK. A gene-coexpression network for global discovery of conserved genetic modules. *Science*. 2003;302(5643):249–255.
35. Bartlett TE, Olhede SC, Zaikin A. A dna methylation network interaction measure, and detection of network oncomarkers. *PloS One*. 2014;9(1):e84573.
36. Stojkovic I, Jelisavcic V, Milutinovic V, Obradovic Z. Distance based modeling of interactions in structured regression. In *Procedeengs of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*. Palo Alto, CA: AAAI; 2016:2032–2038.
37. Lam C, Fan J. Sparsistency and rates of convergence in large covariance matrix estimation. *Ann Stat*. 2009;37(6B):4254.

How to cite this article: Jelisavcic V, Stojkovic I, Milutinovic V, Obradovic Z. Fast learning of scale-free networks based on Cholesky factorization. *Int J Intell Syst*. 2018;33:1322–1339. https://doi.org/10.1002/int.21984

APPENDIX A: TIME AND MEMORY COMPLEXITY ANALYSIS

Several computational improvements contribute to the speedup achieved by the SNETCH algorithm. First, log det term can be calculated in linear time, as shown in (7). Second, feasible space coincides with the PD cone, and therefore no additional effort needs to be taken in order to ensure positive definiteness. Third, each column of Cholesky factors can be optimized independently, thus decomposing the problem into smaller, much easier to solve subproblems. Fourth, each subproblem is sparse, enabling us to keep only a subset of the covariance matrix in memory at a single time, as will be shown. This sparsity is then exploited using the active set based coordinate descent approach.

While selecting the active set for each of the subproblems we rely on the following observation: If the magnitude of the smooth part of the subgradient component (i, j) of the objective function is less than λ, then element L_{ij} will belong to the fixed set. This is a well-known trick when using a coordinate descent active set approach with L_1 penalty, and is already used in Refs. 4 and 8. Therefore (based on equation (14)), a simple condition can be derived for selecting the active set:

$$S_{active} = \{i \in [1..p]|| \sum_k L_{kj} S_{ki}| > \lambda\}. \tag{A.1}$$

Variables not belonging to the active set will belong to the fixed set, and can be skipped when updating. It should be noted that the diagonal elements will always be in the active set.

Let us define a set of active indices for this column in the current iteration:

$$S^t_{active} = \{k \in 1..P|L^{t-1}_{kj} \neq 0\}. \tag{A.2}$$

This set uniquely determines the variables that contribute to the active set equation (A.1) in the current iteration t, since all other variables in column j are zero. Initially, our starting point is $L^0 = I$, therefore S^1_{active} will contain a single element $S^1_{active} = \{j\}$.

We define c_i as the biggest element in the S^t_{active} subset of the ith row of S (infinity norm):

$$c_i = \|S_{ik}\|_\infty = max(S_{ik_1}, S_{ik_2}, ..., S_{ik_n}) \tag{A.3}$$

where the set of indices $\{k_1, k_2, ..., k_n\}$ corresponds to the active set S^t_{active} in the iteration t of the column j.

Using the triangle inequality and the previously defined c_i the following inequality can be derived:

$$\lambda \leq \Big| \sum_i L_{ij} S_{ij}| \leq \sum_i |L_{ij}||S_{ij}| \leq \sum_i |L_{ij}| c_j \leq c_j \sum |L_{ij}| \tag{A.4}$$

Now we can derive the following bound: If the element L_{ij} belongs to the active set, then the maximal element of the active subset of the ith row of the S matrix must be larger than $c_j \geq \frac{\lambda}{\sum_i |L_{ij}|}$. Initially, $\sum_i |L_{ij}|$ will be equal to one (which gives us the initial active set by thresholding the empirical covariance matrix S with λ[4,8]). If the l_∞ norm of the active subset of the ith row of empirical covariance matrix S is less than the derived bound, then element L_{ij} will not belong to the active set. This is convenient, because we can preselect some subset of rows/columns in advance, by thresholding the S matrix with $\lambda_{bound} = \frac{\lambda}{1+C_{tol}}$, where $C_{tol} > 0$ is a prespecified tolerance threshold. Since optimal value L^* is expected to be sparse, we can select a small threshold C_{tol} that is expected to be bigger than $\sum_i |L_{ij}|$, for each column independently.

If the C_{tol} is too big, a preselected subset of the S matrix will be too big, and will fail to fit in the memory, also increasing the time complexity of the optimization (which is directly proportional to the size of this subset). For C_{tol} too small, a recalculation of the subset for the column will be needed, which takes $O(p)$ time. Therefore, proper selection of this parameter is crucial in order for the algorithm to be executed quickly. In case the column L_j is sparse (which is true for most columns for the properly ordered scale-free graph), small tolerance will suffice. Ideally, since most of the nodes in the graph have a low degree, no recalculation will be needed in most cases, and a suitably small tolerance can be used, which also results in a small active set.

An approximate algorithm can also be derived, where the subset is never recalculated, regardless of whether the value of $\sum_i |L_{ij}|$ is exceeding the tolerance. In this case, optimization is equivalent to optimization with the approximated data matrix. The amount of approximation (in the sense of a distance induced by Frobenius norm) is bounded by the difference in the $\sum_i |L_{ij}|$ and $1 + C_{tol}$.

Pseudo-code of an overall coordinate gradient descent approach with active set update formula is presented in Algorithm 1. The time complexity for finding the initial active set is $O(p^2)$, and the complexity of the approximate minimal degree algorithm used for sorting the columns is $O(p)$. Time needed for optimizing a single column is $O(s_a s_s k)$, where s_s and s_a are the cardinality of the selected set of the column, and the cardinality of the active set of the column respectively ($s_a < s_s << p$), while k is the number of iterations needed to converge. The total time complexity is dominated by the $O(p^2)$, but compared to other methods for fast sparse inverse covariance estimation[4,8] the cost of each iteration is significantly lower $O(ps_s)$ compared to a full sweep $O(p^2)$ over the entire matrix. Recalculations of the selected set are done independently on demand for each column, although $O(p)$, it is executed for only a few of the columns, thus amortizing the cost.

Holding the whole empirical covariance matrix S in memory would require $O(p^2)$ memory, which is prohibitively expensive, when the dimension of the problem p is large. Therefore, only a subset of the covariance matrix can be available at a single time, and it has to be calculated on demand. This is handled naturally in our approach, since optimizing a single column requires only a small fraction of elements of the full covariance matrix once the initial active set has been calculated.

APPENDIX B: CONVEXITY PROOF

The proposed objective (4) is convex. Since $g(L)$ is separable across columns, it is sufficient to show that optimization function for a single column is convex, which can be done by observing that the Hessian matrix of the smooth part is equal to sum of the submatrix of the empirical covariance matrix and a diagonal matrix with a single positive element, which is always PD:

$$H_{g(L_j)} = S_{jj} + D_{jj} \geq 0 \tag{B.1}$$

Penalty term is also convex, as it is the standard L_1 norm over the optimization variables, weighted by the nonnegative factors.[37]