PLOS ONE

RESEARCH ARTICLE

DC algorithm for estimation of sparse Gaussian graphical models

Tomokaze Shiratori[1]*, **Yuichi Takano**[2]

1 Graduate School of Science and Technology, University of Tsukuba, Tsukuba, Ibaraki, Japan, **2** Institute of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki, Japan

* cygnus.2xl@gmail.com

OPEN ACCESS

Citation: Shiratori T, Takano Y (2024) DC algorithm for estimation of sparse Gaussian graphical models. PLoS ONE 19(12): e0315740. https://doi.org/10.1371/journal.pone.0315740

Editor: Jianchao Bai, Northwestern Polytechnical University, CHINA

Received: September 20, 2024

Accepted: December 1, 2024

Published: December 23, 2024

Data Availability Statement: All synthetic datasets are available by running the code in our GitHub repository (https://github.com/torikaze/DC-GGM). Data used for creating the figures are also available in the same GitHub repository.

Funding: One of the authors, Tomokaze Shiratori (TS) is employed by Nissan Motor Co., Ltd. The founder provided support in the form of salaries for TS, but did not have any additional role in the study design, data collection and analysis.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Sparse estimation of a Gaussian graphical model (GGM) is an important technique for making relationships between observed variables more interpretable. Various methods have been proposed for sparse GGM estimation, including the graphical lasso that uses the ℓ_1 norm regularization term, and other methods that use nonconvex regularization terms. Most of these methods approximate the ℓ_0 (pseudo) norm by more tractable functions; however, to estimate more accurate solutions, it is preferable to directly use the ℓ_0 norm for counting the number of nonzero elements. To this end, we focus on sparse estimation of GGM with the cardinality constraint based on the ℓ_0 norm. Specifically, we convert the cardinality constraint into an equivalent constraint based on the largest-K norm, and reformulate the resultant constrained optimization problem into an unconstrained penalty form with a DC (difference of convex functions) representation. To solve this problem efficiently, we design a DC algorithm in which the graphical lasso algorithm is repeatedly executed to solve convex optimization subproblems. Experimental results using two synthetic datasets show that our method achieves results that are comparable to or better than conventional methods for sparse GGM estimation. Our method is particularly advantageous for selecting true edges when cross-validation is used to determine the number of edges. Moreover, our DC algorithm converges within a practical time frame compared to the graphical lasso.

Introduction

Background

Quantifying structural relationships between variables from observed data is a fundamental task in data mining. One commonly used measure is Pearson's product-moment correlation coefficient, defined as the covariance of standardized variables. However, this measure has obvious limitations, such as its inability to deal with spurious correlations. In contrast, the Gaussian graphical model (GGM) involves learning partial correlations that correspond to elements of the precision matrix (i.e., the inverse of the covariance matrix). This approach provides a conditional independence graph, which graphically represents the relationships between variables while taking into account the influence of other variables. Such structural

estimation has been effectively used in various fields, including analysis of brain activity patterns [1], anomaly detection [2], and sentiment analysis on social networks [3].

Since most variables usually have some relationship between them, the direct application of GGM often produces dense graphs, which have edges for many pairs of variables. For this reason, sparse estimation methods for GGM have been actively studied to estimate simple and essential relationships between variables [4–6]. The sparse estimation of GGM aims to create a conditional independence graph in a sparse form by reducing the number of nonzero elements in the estimated precision matrix. This approach allows us to estimate an interpretable graph even when the number of variables is larger than the sample size. However, sparse estimation of GGM faces several technical challenges, such as reducing the computational complexity and ensuring the positive definiteness of the precision matrix.

Related work

Methods for estimating sparse precision matrices have long existed, including statistical testing methods [7] and threshold-based methods [8] for selecting nonzero elements. The lasso [9], a least-squares regression model with the ℓ_1 norm regularization term, has also been used to estimate relationships between variables [10, 11]. However, these methods face computational challenges, such as the enormous computation time required for high-dimensional data and the inability to guarantee the positive definiteness of the precision matrix.

We focus on the method of adding regularization terms to the negative log-likelihood, which has become mainstream in recent years. Sparse GGM estimation was formulated as a convex optimization problem by adding the ℓ_1 norm of elements of the precision matrix to the negative log-likelihood [12, 13]. The graphical lasso [14] is widely used to solve this optimization problem because it works quickly and stably even when the number of variables is larger than the sample size or when correlations between variables are high. It is also known that upon (asymptotic) convergence, the graphical lasso provides a positive definite precision matrix [15]. The graphical lasso is an iterative algorithm that minimizes the negative log-likelihood and the ℓ_1 norm regularization term for GGM, where the strength of sparsity is adjusted by a regularization parameter. Various methods for sparse GGM estimation have been derived from the graphical lasso [15–17].

Methods for tuning regularization parameters include using information criteria, performing cross-validation, and analyzing the stability of the estimation results. Regularization parameters tuned using information criteria such as AIC and BIC work well for low-dimensional data, but tend to estimate graphs with high false positive rates for high-dimensional data [18]. The extended BIC is more effective at reproducing the true graph than the original BIC when the number of true edges is small [19, 20]. Cross-validation allows for more accurate selection of true edges than the use of information criteria, but suffers from high model variability [19]. Methods for analyzing the stability of the estimation results (e.g., by subsampling) have shown high accuracy in reproducing the true graph for high-dimensional data [18, 20]. Recently proposed methods include minimizing a network-characteristic-based function with respect to the regularization parameter [21], and assuming multivariate probability distributions other than the normal [22, 23].

While there are many successful methods based on the lasso for sparse estimation, it is well known that estimators with the ℓ_1 norm regularization term are biased. A desirable property of estimators, known as the oracle property [24], has led to methods that compensate for the shortcomings of the lasso. Such methods include SCAD [25] and MCP [26], which use continuous nonconvex functions as regularization terms, and the adaptive lasso [27], which gives different regularization weights to each element of the precision matrix. SELO [28] was designed

with a regularization term that closely approximates the ℓ_0 norm, which represents the number of nonzero elements. A nonconvex regularization term was also proposed using an inverse trigonometric function that converges to the ℓ_0 norm [29]. Although these approaches aim to approximate the ℓ_0 norm by more tractable functions, it is more preferable to directly use the ℓ_0 norm for counting the number of nonzero elements. A different approach is to solve the Lagrangian dual problem for estimating cardinality-constrained graphical models [30]. However, since the ℓ_0 norm is a discontinuous nonconvex function, the associated sparse estimation is known to be NP-hard [31] and involves a positive duality gap. To the best of our knowledge, there is no sparse estimation method that directly uses the cardinality constraint based on the ℓ_0 norm for GGM.

The DC (difference of convex functions) algorithm has been used to solve sparse optimization problems with the ℓ_0 norm [32–34]. This method expresses a nonconvex objective function as the difference of two convex functions and repeatedly solves a convex optimization problem based on a linear approximation of the concave function to find a high-quality solution to the original nonconvex optimization problem [35, 36]. The DC algorithm have been applied to a variety of problem classes, including quadratic and bilevel optimization [37]. Phan et al. [38] designed a DC algorithm based on approximated DC representations for sparse estimation of the covariance matrix, whereas we focus on sparse estimation of the precision matrix based on the ℓ_0 norm. Recently, Gotoh et al. [34] proposed new DC formulations and algorithms for sparse optimization problems, reporting favorable experimental results compared to the lasso. This DC optimization approach also allows us to estimate regularization parameter values that guarantee optimality for specific problems, avoiding the use of excessively large regularization parameter values.

Our contribution

The main goal of this paper is to propose a high-performance algorithm for sparse GGM estimation with the ℓ_0 norm. To this end, we apply the DC optimization framework proposed by Gotoh et al. [34] to sparse GGM estimation. Specifically, we first equivalently rewrite the cardinality constraint based on the ℓ_0 norm by using the largest-K norm defined by Gotoh et al. [34]. We then reformulate this constrained optimization problem into an unconstrained penalty form with a DC representation, which is the difference of two convex functions. To solve this problem efficiently, we design a DC algorithm, which repeatedly executes the graphical lasso algorithm to solve convex optimization subproblems.

The effectiveness of our method is validated through computational experiments using two types of synthetic datasets. We investigate the results when the number of edges is determined by 5-fold cross-validation and when it is given in common to all methods. Experimental results show that our method can generate true graphs with accuracy comparable to or better than conventional methods for sparse GGM estimation. In particular, our method provides superior accuracy when estimating the number of edges through cross-validation. Furthermore, the computation time of our DC algorithm is only a few times longer than the graphical lasso, confirming that the algorithm converges within a practical time frame.

Methods

In this section, we first give an overview of conventional models for sparse GGM estimation, then describe our method for sparse GGM estimation using the DC algorithm. Throughout this paper, we denote the set of consecutive integers as $[n] \coloneqq \{1, 2, \ldots, n\}$.

Sparse estimation of Gaussian graphical models

Gaussian graphical model. Let $\boldsymbol{x} := (x_1, x_2, \ldots, x_p)^\top \in \mathbb{R}^p$ be a vector composed of p random variables that follow a multivariate normal distribution. A Gaussian graphical model (GGM) is a method for estimating a graph of the relationships between variables. Let $\mathcal{N}(\mu, \sigma^2)$ denote a normal distribution with mean μ and variance σ^2, and $\boldsymbol{\Omega} := (\omega_{jk})_{(j,k)\in[p]\times[p]} \in \mathbb{R}^{p\times p}$ denote the precision matrix, which is the inverse of the covariance matrix $\boldsymbol{\Sigma} := (\sigma_{jk})_{(j,k)\in[p]\times[p]} \in \mathbb{R}^{p\times p}$ of random vector $\boldsymbol{x}$. Then, the conditional distribution of x_j given the other variables $\boldsymbol{x}_{-j} := (x_k)_{k\neq j}$ can be written as follows:

$$\Pr(x_j \mid \boldsymbol{x}_{-j}) = \mathcal{N}\left(-\frac{1}{\omega_{jj}}\sum_{k\neq j}\omega_{jk}x_k,\ \frac{1}{\omega_{jj}}\right). \tag{1}$$

Note here that the relationship between x_j and x_k can be determined from the corresponding element ω_{jk} of the precision matrix.

Typically, the precision matrix is estimated through maximum likelihood estimation. Given n observed data points $\boldsymbol{x}_i \in \mathbb{R}^p$ $(i \in [n])$, the sample mean vector and the sample covariance matrix are defined as

$$\boldsymbol{m} := \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \quad \text{and} \quad \boldsymbol{S} := \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{m})(\boldsymbol{x}_i - \boldsymbol{m})^\top,$$

respectively. Then, the log-likelihood function of the precision matrix $\boldsymbol{\Omega}$ is written as

$$\begin{aligned}
\ell(\boldsymbol{\Omega}) &:= \log\left(\prod_{i=1}^{n}(2\pi)^{-\frac{p}{2}}\det(\boldsymbol{\Omega})^{\frac{1}{2}}\exp\left[-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{m})^\top\boldsymbol{\Omega}(\boldsymbol{x}_i - \boldsymbol{m})\right]\right) \\
&= -\frac{pn}{2}\log(2\pi) + \frac{n}{2}\log\det(\boldsymbol{\Omega}) - \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{m})^\top\boldsymbol{\Omega}(\boldsymbol{x}_i - \boldsymbol{m}) \\
&= -\frac{pn}{2}\log(2\pi) + \frac{n}{2}\log\det(\boldsymbol{\Omega}) - \frac{n}{2}\mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{S}), \quad \because \boldsymbol{x}^\top\boldsymbol{\Omega}\boldsymbol{x} = \mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{x}\boldsymbol{x}^\top)
\end{aligned}$$

where $\det(\cdot)$ and $\mathrm{tr}(\cdot)$ are the determinant and the trace (i.e., the sum of diagonal elements) for a square matrix, respectively. By removing from the log-likelihood function the constant terms and coefficients that are irrelevant to the optimization and multiplying it by (−1), we obtain the following loss function (i.e., the negative log-likelihood) to be minimized:

$$-\log\det(\boldsymbol{\Omega}) + \mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{S}). \tag{2}$$

After differentiation, we can derive the maximum likelihood estimator $\hat{\boldsymbol{\Omega}}$ of the precision matrix as

$$-\boldsymbol{\Omega}^{-1} + \boldsymbol{S} = \boldsymbol{O} \quad \Rightarrow \quad \hat{\boldsymbol{\Omega}} = \boldsymbol{S}^{-1},$$

where $\boldsymbol{O}$ is the zero matrix of appropriate size.

Regularization. If $\omega_{jk} = 0$ $(j \neq k)$ in Eq (1), x_k does not influence x_j given $\boldsymbol{x}_{-j}$, and this situation is called conditional independence. Therefore, a conditional independence graph, which connects only the variables that are not conditionally independent, is made sparse by assuming that ω_{jk} is exactly zero for many $(j, k) \in [p] \times [p]$. To estimate such a sparse graph (or sparse precision matrix), we add a regularization term $p_\lambda(\boldsymbol{\Omega})$ to the loss function (2) to penalize the

absolute values of elements of the precision matrix as

$$-\log\det(\boldsymbol{\Omega}) + \mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{S}) + p_\lambda(\boldsymbol{\Omega}), \tag{3}$$

where $\lambda > 0$ is the regularization parameter for adjusting the strength of the penalty. As λ gets larger, more elements of $\boldsymbol{\Omega}$ are estimated to be zero.

Various types of sparse estimators can be represented by the choice of the regularization term $p_\lambda(\boldsymbol{\Omega})$. For example, the regularization term for the graphical lasso [14] is defined based on the ℓ_1 norm as

$$p_\lambda(\boldsymbol{\Omega}) = \lambda\|\mathrm{vec}(\boldsymbol{\Omega})\|_1, \tag{4}$$

where the $\mathrm{vec}(\cdot)$ operator rearranges the elements of a matrix into a vector as follows:

$$\mathrm{vec}(\boldsymbol{\Omega}) := (\omega_{11}, \omega_{21}, \ldots, \omega_{pp})^\top \in \mathbb{R}^{n^2}.$$

Next, let us define for $x \in \mathbb{R}$,

$$\mathrm{SCAD}_{\lambda,a}(x) := \begin{cases} \lambda|x| & \text{if } |x| \le \lambda, \\ \dfrac{a\lambda|x| - (x^2+\lambda^2)/2}{a-1} & \text{if } \lambda < |x| \le a\lambda, \\ \dfrac{(a+1)\lambda^2}{2} & \text{if } |x| > a\lambda, \end{cases}$$

with a parameter $a > 2$. Then, the SCAD regularization term [25] is defined as

$$p_\lambda(\boldsymbol{\Omega}) = \sum_{j=1}^{p}\sum_{k=1}^{p}\mathrm{SCAD}_{\lambda,a}(\omega_{jk}). \tag{5}$$

Additionally, let $\tilde{\boldsymbol{\Omega}} := (\tilde{\omega}_{jk})_{(j,k)\in[p]\times[p]}$ be a consistent estimator of $\boldsymbol{\Omega}$. Then, the regularization term for the adaptive lasso [27], a weighted version of lasso, is written as

$$p_\lambda(\boldsymbol{\Omega}) = \lambda\sum_{j=1}^{p}\sum_{k=1}^{p}\frac{1}{|\tilde{\omega}_{jk}|^\gamma}|\omega_{jk}|, \tag{6}$$

with a parameter $\gamma > 0$.

Fig 1 illustrates graphs of $p_\lambda(x)$ of the graphical lasso, SCAD, and the adaptive lasso for $x \in [-2, 2]$ with parameters $\lambda = 0.5$, $a = 3.7$, $\tilde{x} = 0.5$, and $\gamma = 0.5$.

Graphical lasso. The graphical lasso [14], which is closely related to our algorithm, uses the regularization term (4) based on the ℓ_1 norm. Let us define the sign function of $x \in \mathbb{R}$ as

$$\mathrm{sign}(x) := \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases} \tag{7}$$

Then, the following optimality condition is derived by differentiating Eq (3) with respect to $\boldsymbol{\Omega}$ as

$$-\boldsymbol{\Omega}^{-1} + \boldsymbol{S} + \lambda\boldsymbol{\Gamma}(\boldsymbol{\Omega}) = \boldsymbol{O}, \tag{8}$$

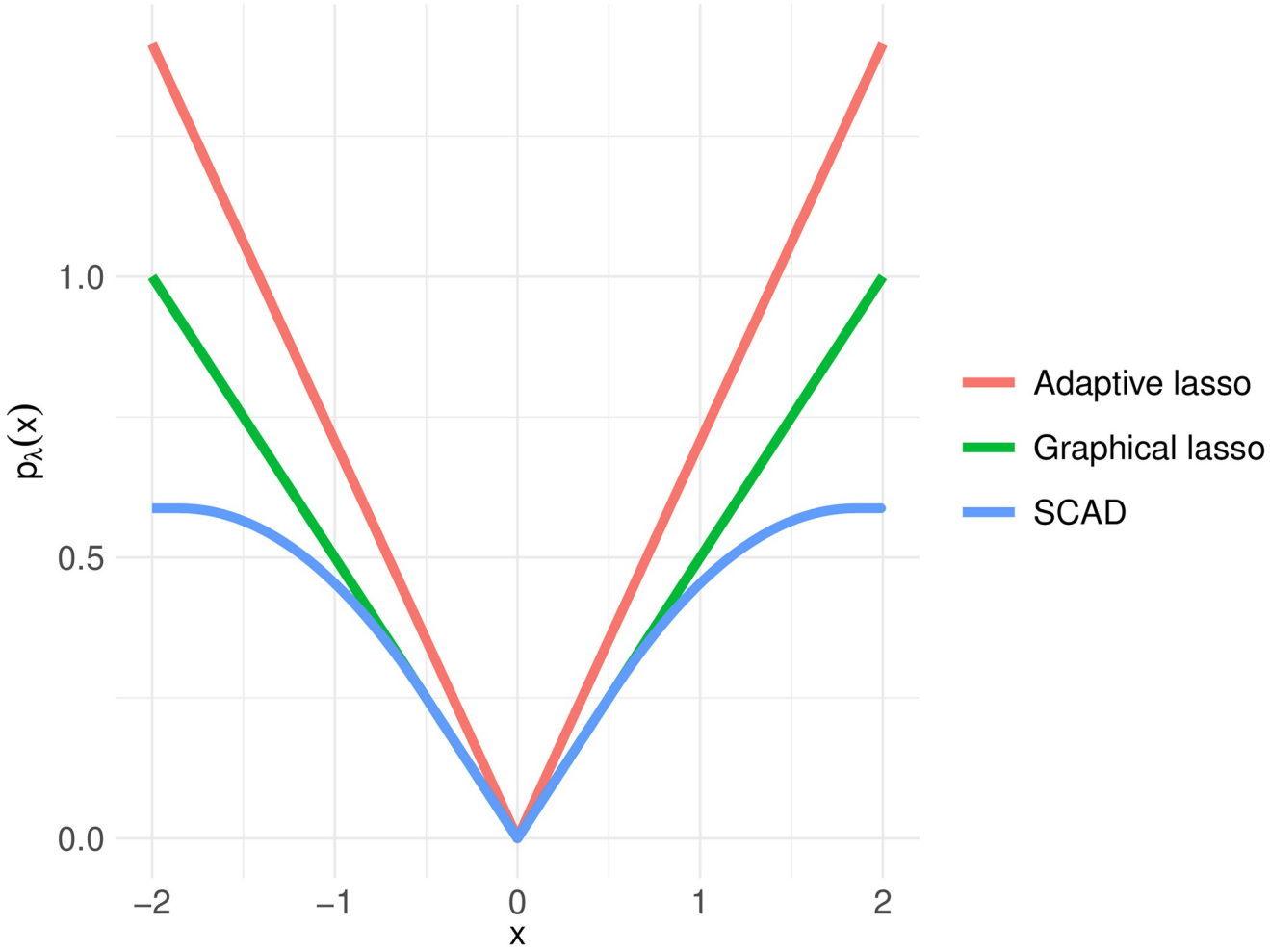

Fig 1. Graphs of the regularization terms.

https://doi.org/10.1371/journal.pone.0315740.g001

where

$$\Gamma(\Omega) \coloneqq (\gamma_{jk}(\omega_{jk}))_{(j,k)\in[p]\times[p]} \in \mathbb{R}^{p\times p},$$
$$\gamma_{jk}(\omega_{jk}) \in \begin{cases} \{\mathrm{sign}(\omega_{jk})\} & \text{if } \omega_{jk} \neq 0, \\ [-1,1] & \text{if } \omega_{jk} = 0. \end{cases} \tag{9}$$

The graphical lasso simultaneously searches for solutions Ω and $\Sigma = \Omega^{-1}$ to the nonlinear Eq (8) by sequentially updating each column $j \in [p]$ of the matrices. For this purpose, the matrices are decomposed into blocks (after row and column rearrangements) as

$$\Omega = \begin{bmatrix} \Omega_{-j} & \boldsymbol{\omega}_j \\ \boldsymbol{\omega}_j^\top & \omega_{jj} \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{-j} & \boldsymbol{\sigma}_j \\ \boldsymbol{\sigma}_j^\top & \sigma_{jj} \end{bmatrix}, \tag{10}$$

where $\Omega_{-j}, \Sigma_{-j} \in \mathbb{R}^{(p-1)\times(p-1)}$; $\boldsymbol{\omega}_j, \boldsymbol{\sigma}_j \in \mathbb{R}^{p-1}$; and $\omega_{jj}, \sigma_{jj} \in \mathbb{R}$. Then, the nonlinear Eq (8) with respect to the j-th column can be reduced to the lasso regression [9], and thus, each column can be computed efficiently using the coordinate descent method [14].

The procedure of the graphical lasso is summarized in Algorithm 1. The covariance matrix is initialized as $\Sigma_0 = \boldsymbol{S} + \lambda \boldsymbol{I}$, which is derived from the diagonal elements determined from Eq (8) and the off-diagonal elements obtained by maximum likelihood estimation, where $\boldsymbol{I}$ is the identity matrix of appropriate size. The algorithm terminates when the update of the precision matrix becomes smaller than a threshold parameter $\varepsilon > 0$ in terms of the Frobenius norm $\|\cdot\|_{\mathrm{F}}$. Note also that since this algorithm has been criticized for the fact that the objective function does not decrease monotonically, several methods have been proposed to accelerate the convergence [15].

Algorithm 1 Graphical Lasso for Sparse GGM Estimation

Input: Sample covariance matrix $\boldsymbol{S}$, regularization parameter $\lambda > 0$, convergence threshold $\varepsilon > 0$.
Output: Precision matrix $\boldsymbol{\Omega}$.
Initialize: Iteration number $t \leftarrow 0$, covariance matrix $\boldsymbol{\Sigma}_0 = \boldsymbol{S} + \lambda \boldsymbol{I}$, precision matrix $\boldsymbol{\Omega}_0 = \boldsymbol{\Sigma}_0^{-1}$.
$(\boldsymbol{\Omega}, \boldsymbol{\Sigma}) \leftarrow (\boldsymbol{\Omega}_0, \boldsymbol{\Sigma}_0)$.
repeat
  for $j \in [p]$ **do**
    Decompose $\boldsymbol{\Omega}$ and $\boldsymbol{\Sigma}$ into block matrices (after rearrangement) as in Eq (10).
    Update $\boldsymbol{\omega}_j$, ω_{jj}, $\boldsymbol{\sigma}_j$, σ_{jj} using the lasso regression [14].
    Rearrange the elements of $\boldsymbol{\Omega}$ and $\boldsymbol{\Sigma}$ back into the original matrices.
  end for
  $(\boldsymbol{\Omega}_{t+1}, \boldsymbol{\Sigma}_{t+1}) = (\boldsymbol{\Omega}, \boldsymbol{\Sigma})$.
  $t \leftarrow t + 1$.
until $\|\boldsymbol{\Omega}_t - \boldsymbol{\Omega}_{t-1}\|_{\mathrm{F}}^2 < \varepsilon$.
return $\boldsymbol{\Omega}_t$.

DC algorithm for sparse GGM estimation

Formulation. For $\boldsymbol{w} := (w_i)_{i \in [m]} \in \mathbb{R}^m$, we denote the ℓ_0 (pseudo) norm by

$$\|\boldsymbol{w}\|_0 := |\{i \in [m] \mid w_i \neq 0\}|,$$

which counts the number of nonzero elements of $\boldsymbol{w}$. To find a positive definite precision matrix $\Omega \succ \boldsymbol{O}$, we impose the constraint $\Omega \succeq \delta \boldsymbol{I}$ (i.e., $\Omega - \delta \boldsymbol{I}$ is positive semidefinite) with a small positve constant $\delta > 0$. Then, sparse GGM estimation can be naturally posed as the following cardinality-constrained optimization problem:

$$\underset{\boldsymbol{\Omega} \succeq \delta \boldsymbol{I}}{\text{minimize}} \quad -\log\det(\boldsymbol{\Omega}) + \mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{S}) \tag{11}$$

$$\text{subject to} \quad \|\mathrm{vec}(\boldsymbol{\Omega})\|_0 \leq K, \tag{12}$$

where $K \in [p^2]$ is a cardinality parameter for limiting the number of nonzero elements of the precision matrix.

Following Gotoh et al. [34], we now define the largest-K norm as follows.

Definition 1. For $\boldsymbol{w} := (w_i)_{i \in [m]} \in \mathbb{R}^m$, let π be a permutation of $[m]$ satisfying $|w_{\pi(1)}| \geq |w_{\pi(2)}| \geq \cdots \geq |w_{\pi(m)}|$. Then, the largest-$K$ norm is defined as the sum of the K largest absolute

values as

$$|||\boldsymbol{w}|||_K := \sum_{i=1}^{K} |w_{\pi(i)}|. \tag{13}$$

Note here that

$$\|\boldsymbol{w}\|_0 \leq K \Longleftrightarrow \sum_{i=K+1}^{m} |w_{\pi(i)}| = 0 \Longleftrightarrow \|\boldsymbol{w}\|_1 - |||\boldsymbol{w}|||_K = 0.$$

Therefore, problem (11) and (12) can be equivalently rewritten as

$$\underset{\boldsymbol{\Omega} \succeq \delta \boldsymbol{I}}{\text{minimize}} \quad -\log\det(\boldsymbol{\Omega}) + \mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{S}) \tag{14}$$

$$\text{subject to} \quad \|\mathrm{vec}(\boldsymbol{\Omega})\|_1 - |||\mathrm{vec}(\boldsymbol{\Omega})|||_K = 0. \tag{15}$$

Although the ℓ_0 norm in Eq (12) is a discontinuous function, Eq (15) is represented by the difference of two convex continuous functions and defines the same feasible region as the original problem (11) and (12).

In what follows, we focus on the following penalized version of problem (14) and (15):

$$\underset{\boldsymbol{\Omega} \succeq \delta \boldsymbol{I}}{\text{minimize}} \quad -\log\det(\boldsymbol{\Omega}) + \mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{S}) + \eta\Big(\|\mathrm{vec}(\boldsymbol{\Omega})\|_1 - |||\mathrm{vec}(\boldsymbol{\Omega})|||_K\Big), \tag{16}$$

or equivalently,

$$\underset{\boldsymbol{\Omega} \succeq \delta \boldsymbol{I}}{\text{minimize}} \quad \Big(-\log\det(\boldsymbol{\Omega}) + \mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{S}) + \eta\|\mathrm{vec}(\boldsymbol{\Omega})\|_1\Big) - \eta|||\mathrm{vec}(\boldsymbol{\Omega})|||_K, \tag{17}$$

where $\eta > 0$ is a penalty parameter. Problem (17) is called a DC optimization problem [35] because its objective is the difference of two convex functions.

Algorithm. Each iteration of the DC algorithm constructs a linear approximation of the concave function and solves the resultant convex optimization problem to update the solution.

Following Gotoh et al. [34], we calculate a subgradient of the largest-K norm based on the sign function (7) as

$$\boldsymbol{s}(\boldsymbol{w}) := (s_i(\boldsymbol{w}))_{i\in[m]} \in \partial|||\boldsymbol{w}|||_K, \tag{18}$$

where

$$s_i(\boldsymbol{w}) := \begin{cases} \mathrm{sign}(w_i) & \text{if } \pi^{-1}(i) \in [K], \\ 0 & \text{otherwise} \end{cases} \qquad (i \in [m]). \tag{19}$$

Let $\boldsymbol{\Omega}_t$ be an incumbent solution at the t-th iteration of the DC algorithm. By introducing a linear approximation of the largest-K norm, a surrogate objective function is given by

$$g_t(\boldsymbol{\Omega}) := -\log\det(\boldsymbol{\Omega}) + \mathrm{tr}(\boldsymbol{\Omega}\boldsymbol{S}) + \eta\|\mathrm{vec}(\boldsymbol{\Omega})\|_1 - \eta\boldsymbol{s}(\mathrm{vec}(\boldsymbol{\Omega}_t))^\top \mathrm{vec}(\boldsymbol{\Omega}). \tag{20}$$

By differentiating $g_t(\boldsymbol{\Omega})$, we obtain the following optimalitiy condition based on Eq (9):

$$\frac{\partial g_t}{\partial \boldsymbol{\Omega}} = -\boldsymbol{\Omega}^{-1} + (\boldsymbol{S} - \eta\boldsymbol{V}(\boldsymbol{\Omega}_t)) + \eta\boldsymbol{\Gamma}(\boldsymbol{\Omega}) = \boldsymbol{O}, \tag{21}$$

where $\boldsymbol{V}(\boldsymbol{\Omega}_t) := \mathrm{vec}^{-1}(\boldsymbol{s}(\mathrm{vec}(\boldsymbol{\Omega}_t))) \in \mathbb{R}^{p\times p}$. Note that this nonlinear equation corresponds to Eq (8), where $\boldsymbol{S}$ is replaced by $\boldsymbol{S} - \eta\boldsymbol{V}(\boldsymbol{\Omega}_t)$. Accordingly, the graphical lasso algorithm can be

applied to Eq (21) and gives a solution Ω, which is positive definite upon (asymptotic) convergence.

Our DC algorithm for estimating a sparse precision matrix is described in Algorithm 2. Although the graphical lasso assumes that the sample covariance matrix is positive definite (i.e., $\boldsymbol{S} \succ \boldsymbol{O}$), the corresponding matrix $\boldsymbol{S} - \eta \boldsymbol{V}(\Omega_t)$ in Eq (21) may not be positive definite depending on the value of the penalty parameter η. Note here that if $\eta \approx 0$, then $\boldsymbol{S} - \eta \boldsymbol{V}(\Omega_t) \approx \boldsymbol{S} \succ \boldsymbol{O}$. In addition, all diagonal elements of $\boldsymbol{V}(\Omega_t)$ are equal to 1 due to the positive definiteness of the precision matrix; therefore, if $\eta > \lambda_{\min}(\boldsymbol{S})$, then $\boldsymbol{S} - \eta \boldsymbol{V}(\Omega_t) \not\succ \boldsymbol{O}$, where $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue of a matrix. For this reason, our algorithm adaptively searches for the largest possible $\eta \in [0, \lambda_{\min}(\boldsymbol{S})]$ such that $\boldsymbol{S} - \eta \boldsymbol{V}(\Omega_t) \succ \boldsymbol{O}$.

Algorithm 2 DC Algorithm for Sparse GGM Estimation

Input: Sample covariance matrix $\boldsymbol{S}$, cardinality parameter $K \in [p^2]$, convergence threshold $\varepsilon > 0$, shrinking parameter $\alpha \in (0, 1)$.
Output: Precision matrix $\boldsymbol{\Omega}$.
Initialize: Iteration number $t \leftarrow 0$, precision matrix $\boldsymbol{\Omega}_0 \succ \boldsymbol{O}$.
repeat
  Compute the subgradient $\boldsymbol{s}(\mathrm{vec}(\boldsymbol{\Omega}_t)) \in \partial |||\mathrm{vec}(\boldsymbol{\Omega}_t)|||_K$ as in Eqs (18) and (19).
  $\eta \leftarrow \lambda_{\min}(\boldsymbol{S})$.
  repeat
    $\eta \leftarrow \alpha\eta$.
  until $\boldsymbol{S} - \eta V(\boldsymbol{\Omega}_t) \succ \boldsymbol{O}$.
  Solve Eq (21) using Algorithm 1 to compute $\boldsymbol{\Omega}_{t+1}$.
  $t \leftarrow t + 1$.
until $\|\boldsymbol{\Omega}_t - \boldsymbol{\Omega}_{t-1}\|_F^2 < \varepsilon$.
return $\boldsymbol{\Omega}_t$.

Experimental results and discussion

In this section, we report experimental results on two types of synthetic datasets to validate the effectiveness of our method for sparse GGM estimation (The source code of the experiments is available at https://github.com/torikaze/DC-GGM).

Synthetic datasets

Following Mazumder and Hastie [15], and Yuan and Lin [13], we prepared two types of synthetic datasets based on random and chain graphs. For each dataset, we begin by defining a ground-truth precision matrix as follows.

Random graph: Create a symmetric matrix $\boldsymbol{A}_2 := (\boldsymbol{A}_1 + \boldsymbol{A}_1^\top)/2 \in \mathbb{R}^{p \times p}$, where each element of $\boldsymbol{A}_1 \in \mathbb{R}^{p \times p}$ is independently generated from the standard normal distribution. Randomly set some of the off-diagonal elements of $\boldsymbol{A}_2$ to zeros while maintaining symmetry of the matrix. Define $\Omega_{\mathrm{rnd}} := \boldsymbol{A}_2 + \eta_{\mathrm{rnd}} \boldsymbol{I}$, with η_{rnd} being set such that $\lambda_{\min}(\Omega_{\mathrm{rnd}}) = 1$.

Chain graph: Set up a tridiagonal matrix as follows:

$$\omega_{jk} := \begin{cases} 1 & \text{if } j = k, \\ 0.5 & \text{if } |j - k| = 1, \\ 0.25 & \text{if } |j - k| = 2, \\ 0 & \text{otherwise} \end{cases} \quad ((j, k) \in [p] \times [p]).$$

Randomly set some of the nonzero off-diagonal elements to zeros while maintaining symmetry of the matrix to obtain a precision matrix $\Omega_{\mathrm{chn}} := (\omega_{jk})_{(j,k) \in [p] \times [p]}$.

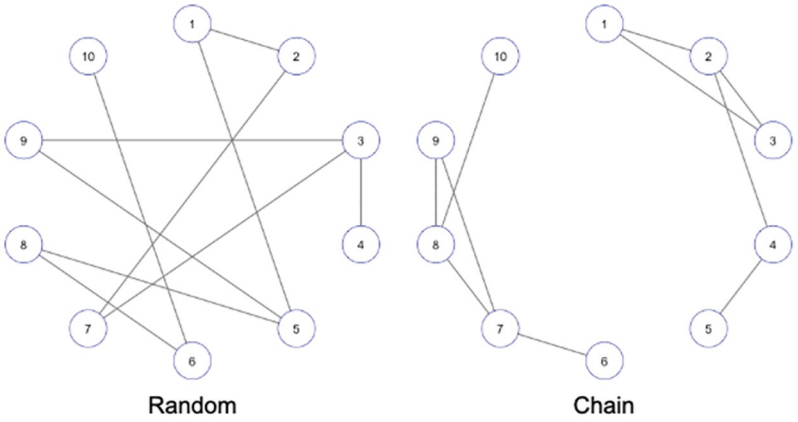

Fig 2. Examples of ground-truth graph structures with $(p, n_{\neq 0}) = (10, 10)$.

https://doi.org/10.1371/journal.pone.0315740.g002

Fig 2 shows examples of graph structures based on the precision matrices $\mathbf{\Omega}_{\mathrm{rnd}}$ and $\mathbf{\Omega}_{\mathrm{chn}}$. Let $n_{\neq 0}$ be the number of true edges (i.e., half the number of nonzero off-diagonal elements of the precision matrix). The procedure for creating synthetic datasets is described as follows:

1. Generate a ground-truth precision matrix $\mathbf{\Omega}^{\star} \coloneqq (\omega^{\star}_{jk})_{(j,k)\in[p]\times[p]} \in \{\mathbf{\Omega}_{\mathrm{rnd}}, \mathbf{\Omega}_{\mathrm{chn}}\}$ with $2 \cdot n_{\neq 0}$ nonzero off-diagonal elements, and create the corresponding covariance matrix as $\Sigma^{\star} \coloneqq (\mathbf{\Omega}^{\star})^{-1}$.
2. Generate $\boldsymbol{x}_i \in \mathbb{R}^p \ (i \in [n])$ independently from a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^{\star})$, and compute the sample covariance matrix $\boldsymbol{S}$.
3. Compute $\boldsymbol{S} \leftarrow \zeta \boldsymbol{D}_{\boldsymbol{S}} + (1 - \zeta)\boldsymbol{S}$ based on the shrinkage estimation [39], where $\boldsymbol{D}_{\boldsymbol{S}}$ is the diagonalized matrix of $\boldsymbol{S}$, and $\zeta \in [0, 1]$ is a shrinkage parameter.

For generation of synthetic datasets, we set the number of variables, the sample size, and the number of true edges as follows:

$$p \in \{50, 100, 200, 400\}, \quad n \in \{p/2, p, 2p\}, \quad \text{and} \quad n_{\neq 0} = 30.$$

Due to the randomness of dataset generation, we created 30 precision matrices for each case and show average results with 95% confidence intervals.

Experimental setup

To validate the effectiveness of our method, we compared the estimation accuracy and characteristics of the following methods for sparse GGM estimation:

DC: Our DC algorithm (Algorithm 2);

glasso: Graphical lasso (Algorithm 1) [14];

SCAD: SCAD regularized estimation [25];

adapt: Adaptive lasso [27].

All experiments were conducted using the R programming language. We used the `glasso` package [14] to implement the graphical lasso, and the `GGMncv` package [40] to implement the SCAD regularized estimation and the adaptive lasso. In the DC algorithm, we set $\alpha = 0.5$ as

the shrinking parameter, and $\Omega_0 = (\boldsymbol{S} + \boldsymbol{I})^{-1}$ as the initial solution. Following Fan and Li [24], we set $a = 3.7$ in Eq (5) for the SCAD regularized estimation. In Eq (6) for the adaptive lasso, we set $\gamma = 0.5$ by following Fan et al. [25], and $\tilde{\boldsymbol{\Omega}} = \boldsymbol{S}^{-1}$ according to default configuration of the `GGMncv` package. We set $\varepsilon = 10^{-4}$ as the convergence threshold.

To evaluate the accuracy of the estimated precision matrix $\hat{\boldsymbol{\Omega}} \coloneqq (\hat{\omega}_{jk})_{(j,k)\in[p]\times[p]} \in \mathbb{R}^{p\times p}$, we first define the true positive (TP), false positive (FP), and false negative (FN) rates as

$$\mathrm{TP} \coloneqq \sum_{j=1}^{p}\sum_{k=j+1}^{p} \mathrm{I}(\hat{\omega}_{jk} \neq 0 \text{ and } \omega^{\star}_{jk} \neq 0),$$

$$\mathrm{FP} \coloneqq \sum_{j=1}^{p}\sum_{k=j+1}^{p} \mathrm{I}(\hat{\omega}_{jk} \neq 0 \text{ and } \omega^{\star}_{jk} = 0),$$

$$\mathrm{FN} \coloneqq \sum_{j=1}^{p}\sum_{k=j+1}^{p} \mathrm{I}(\hat{\omega}_{jk} = 0 \text{ and } \omega^{\star}_{jk} \neq 0),$$

where $\mathrm{I}(Q)$ is an indicator function that returns 1 if the proposition Q is true, and 0 otherwise. The F1 score is then defined as

$$\text{F1 score} \coloneqq \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}},$$

where

$$\text{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \qquad \text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}.$$

The F1 score is an appropriate evaluation metric for imbalanced datasets such as those used in our experiments. The F1 score was also used for evaluation of regularized graphical models [18] and subset selection for linear regression [41].

Results with number of edges determined by cross-validation

We will now investigate the results where the number of edges in an estimated graph was determined through 5-fold cross-validation of the loss function (2). Here, the cardinality parameter K for the DC algorithm was chosen from 100 equally spaced values between $p + 2$ and p^2. The regularization parameter λ for the other methods was chosen from 100 equally spaced values in the range $[0, \lambda_{\max}]$, where $\lambda_{\max}$ was set such that the number of selected edges was zero.

Figs 3 and 4 respectively show the F1 scores and the numbers of selected edges for the random graph dataset, where the number of variables is $p \in \{50, 100, 200, 400\}$, and the sample size is $n \in \{p/2, p, 2p\}$. In Fig 3, our DC method often outperformed the other methods in terms of the F1 score, except when $p = 400$. Additionally, the estimation accuracy of our DC method tended to improve as the sample size increased. Fig 4 shows that the glasso, SCAD, and adapt methods often selected too many edges, resulting in low F1 scores. In contrast, our DC method showed relatively small variations in the number of selected edges, indicating that it is possible for our DC algorithm to produce estimates that are robust to changes in the data.

To examine the number of edges selected through cross-validation in more detail, Fig 5 shows the relationship between the average number of selected edges and the average log-likelihood in cross-validation on the random graph dataset. Note that this figure shows the result of one of 30 trials, and that each method selected the number of edges that maximizes the log-likelihood. As a general trend, fewer edges were selected when $p > n$, whereas more edges

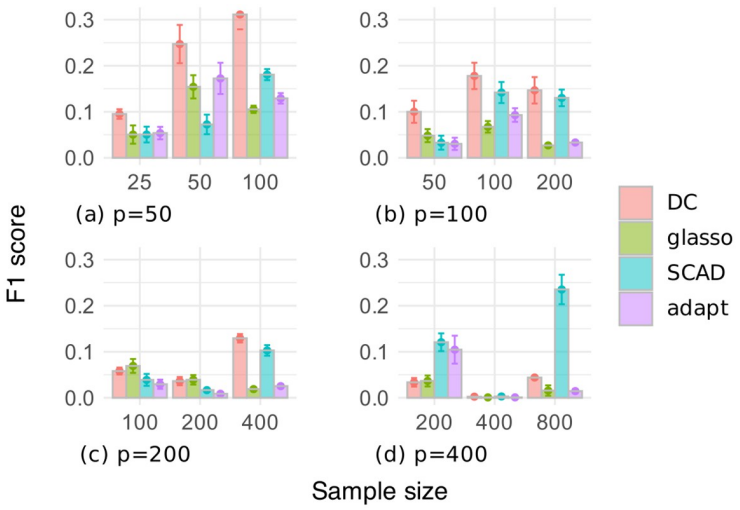

Fig 3. F1 score of edges selected through cross-validation on the random graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g003

were selected when $p < n$. Our DC method often maximized the log-likelihood at close to the true number of edges compared to the other methods. However, with our DC method, the relationship between the number of selected edges and the log-likelihood was not as smooth as with the other methods.

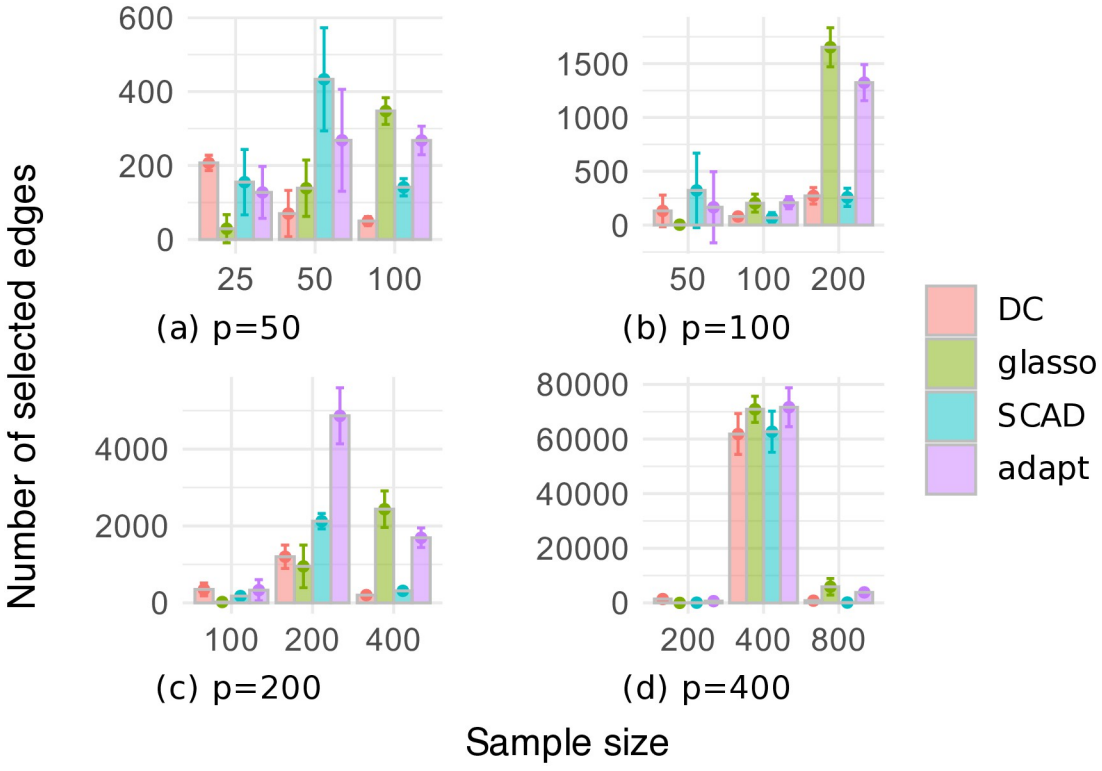

Fig 4. Number of edges selected through cross-validation on the random graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g004

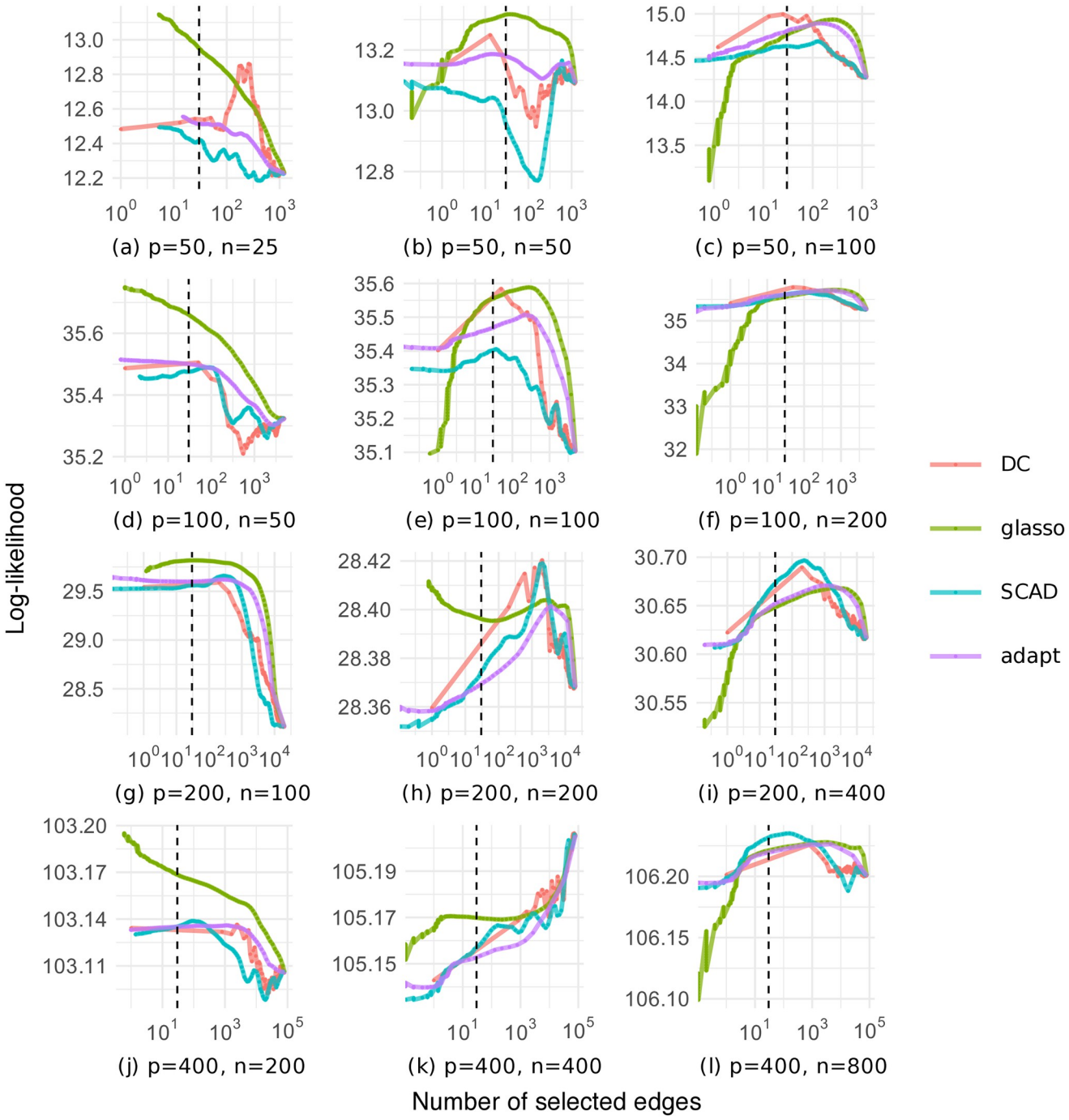

Fig 5. Log-likelihood as a function of the number of selected edges on the random graph dataset (black dashed line: The true number of edges).

https://doi.org/10.1371/journal.pone.0315740.g005

Figs 6 and 7 respectively show the F1 scores and the numbers of selected edges for the chain graph dataset. In Fig 6, our DC method significantly outperformed the other methods in terms of the F1 score. Fig 7 implies that the glasso, SCAD, and adapt methods had low F1 scores because they produced very dense graphs. In contrast, our DC method selected a relatively small and stable number of edges, consistent with the trends observed in the random graph dataset.

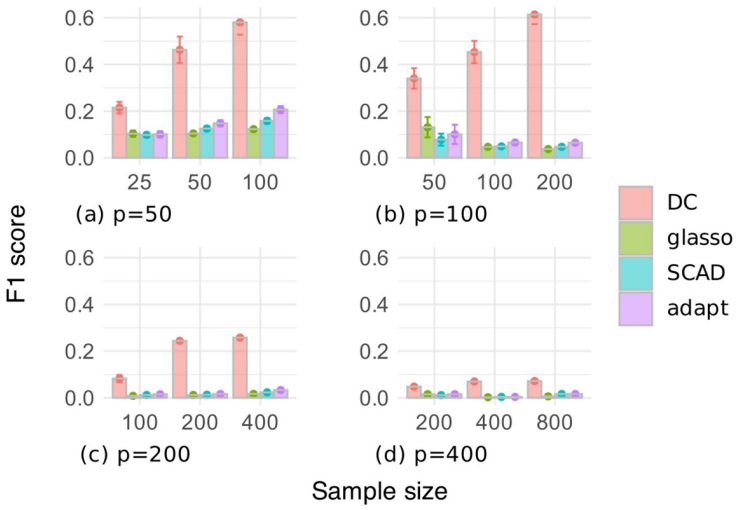

Fig 6. F1 score of edges selected through cross-validation on the chain graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g006

Fig 8 shows the relationship between the average number of selected edges and the average log-likelihood in cross-validation on the chain graph dataset. Our DC method often maximized the log-likelihood at close to the true number of edges compared to the other methods; however, as with the random graph dataset, the relationship between the number of selected

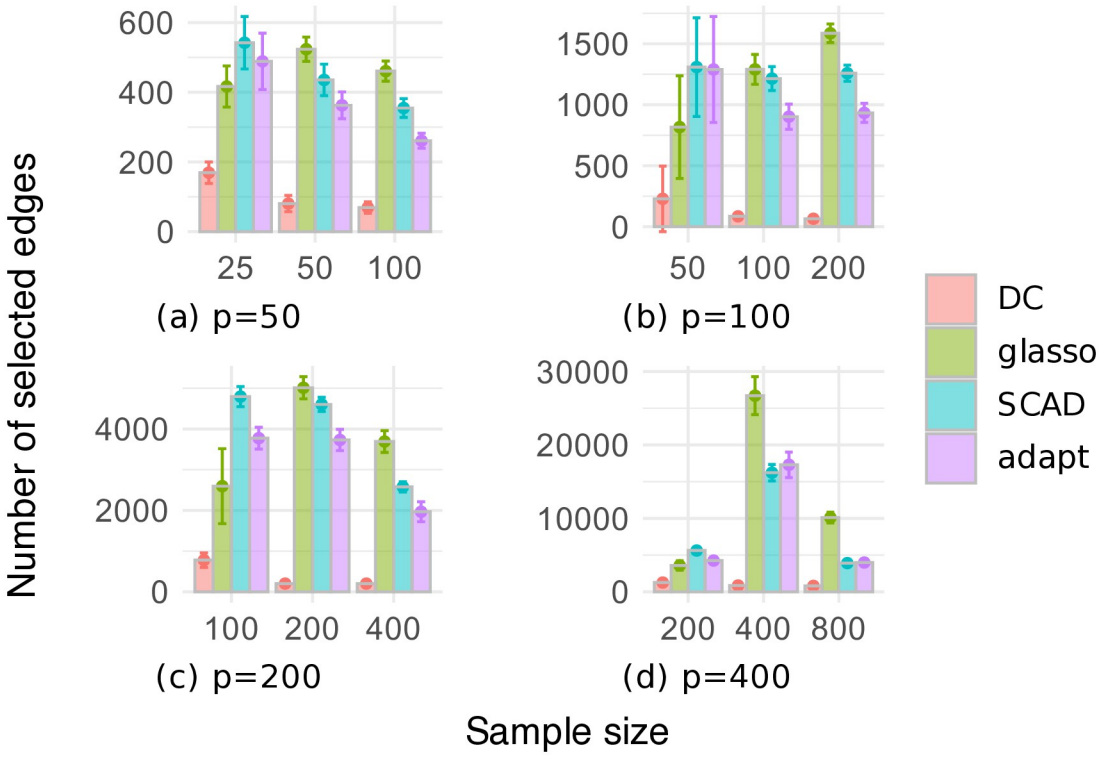

Fig 7. Number of edges selected through cross-validation on the chain graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g007

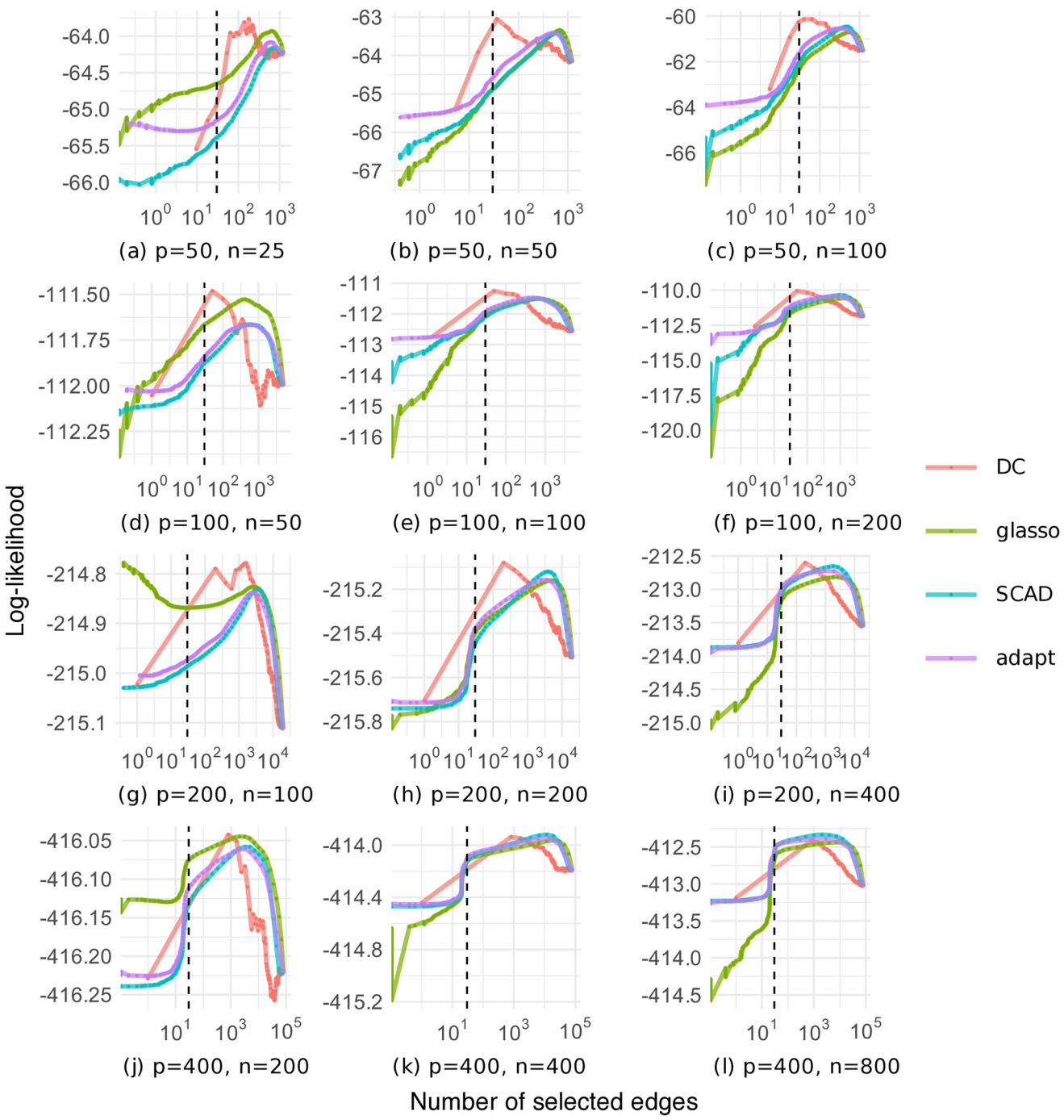

Fig 8. Log-likelihood as a function of the number of selected edges on the chain graph dataset (black dashed line: The true number of edges).

https://doi.org/10.1371/journal.pone.0315740.g008

edges and the log-likelihood was not very smooth, and the number of selected edges was biased relative to the true number of edges.

These results confirm that our method was very accurate in edge selection when cross-validation was used to determine the number of edges. In contrast, other methods often selected an excessively large number of edges, resulting in low F1 scores.

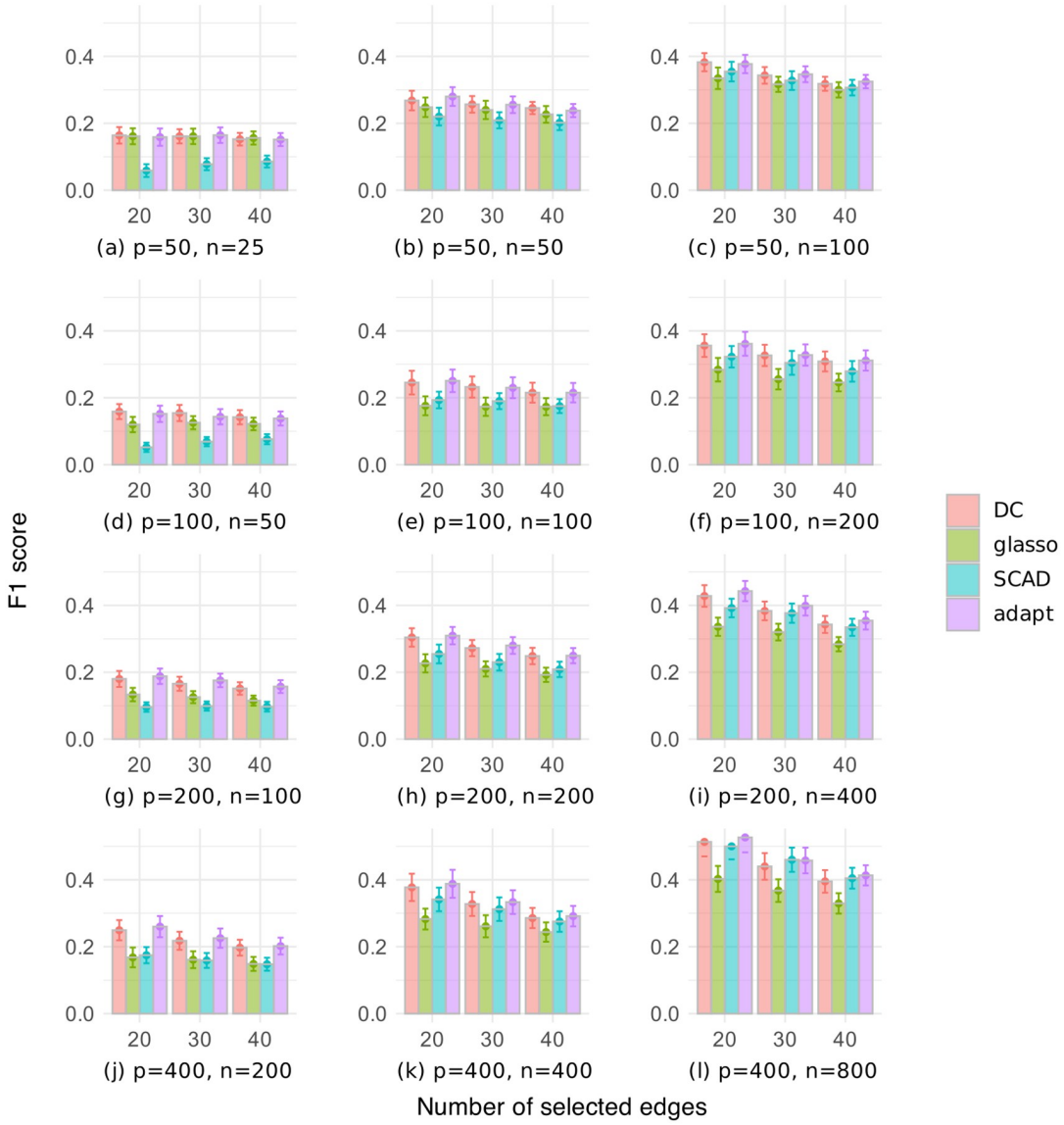

Fig 9. F1 score of a given number of selected edges on the random graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g009

Results with a given number of edges

We will now investigate the results where the number of edges in an estimated graph was given as 20, 30, and 40 commonly for all methods.

Fig 9 shows the F1 scores with different numbers of selected edges for the random graph dataset, where the number of variables is $p \in \{50, 100, 200, 400\}$, and the sample size is $n \in \{p/2, p, 2p\}$. Overall, the F1 scores were better for Fig 9 than for Fig 3, with the DC and adapt methods performing particularly well in Fig 9. Conversely, the glasso and SCAD methods generally had low F1 scores. As the sample size increased, the F1 scores of all methods improved, possibly due to more accurate estimation of the sample covariance matrix. Additionally, as the number of selected edges increased, the F1 scores of all methods tended to decrease, likely due to an increase in the number of false positive edges.

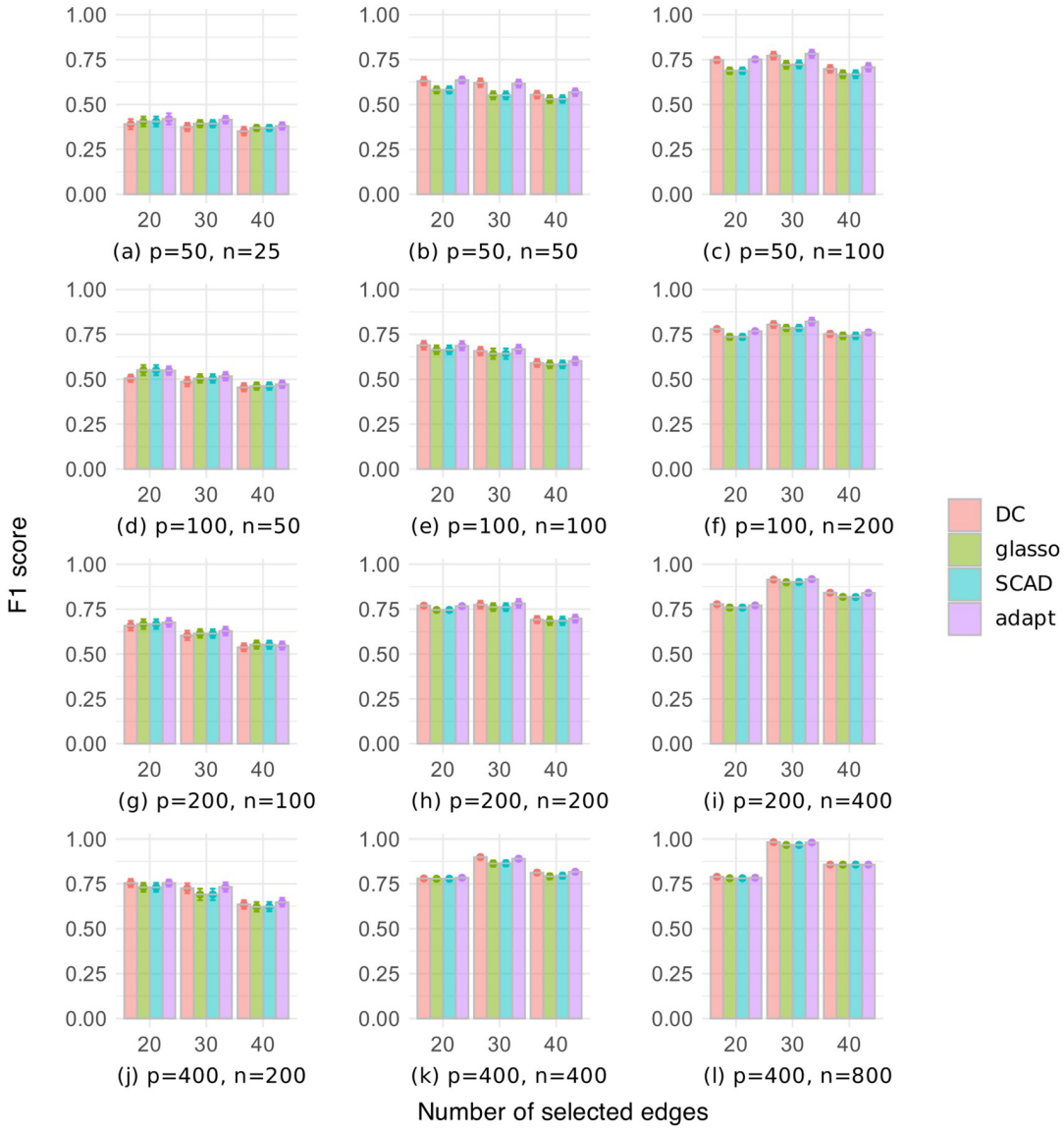

Fig 10. F1 score of a given number of selected edges on the chain graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g010

Fig 10 shows the F1 scores with different numbers of selected edges for the chain graph dataset. The F1 scores were generally high compared to the random graph dataset, with the DC and adapt methods showing slight superiority. Although the F1 scores of our DC method were comparable to or lower than those of the other methods when $p > n$, our DC method performed relatively well when $p \leq n$. As with the random graph dataset, when $p \geq n$, increasing the number of selected edges tended to decrease the F1 score. When $p < n$, setting the number of edges to 30, which is equal to the number of true edges, often yielded the best results. These results show that it was easier to select true edges in the chain graph dataset than in the random graph dataset, and that setting the number of edges to the true number resulted in fewer false positive and false negative edges when the sample size was large enough.

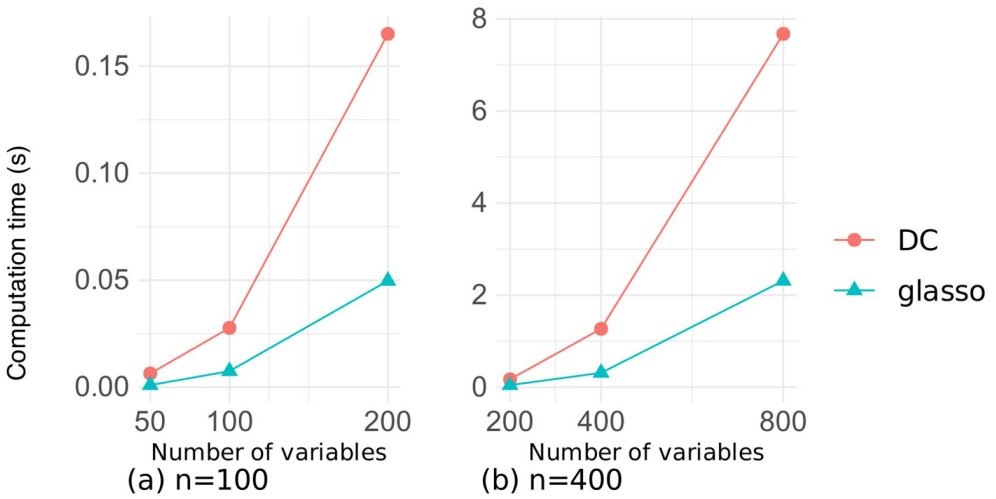

Fig 11. Computation time as a function of the number of variables on the random graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g011

These results confirm that for the random graph dataset, the DC and adapt methods performed better than the other methods when selecting a given number of edges. On the other hand, for the chain graph dataset, all methods showed very high scores, with small differences.

Computation time

We will now investigate the computation time required by our DC algorithm for estimating sparse precision matrices. Here, the cardinality parameter K in our DC method was set to half the total number of edges (i.e., $K = p(p-1)/4$), and the regularization parameter λ in the glasso method was set to the median of the absolute values of off-diagonal elements of the sample covariance matrix. Since there were minor differences among the glasso, SCAD and adapt methods, only the results for the glasso method are shown.

Figs 11 and 12 illustrate the relationship between the number of variables and the computation time for estimation on the datasets of random and chain graphs, respectively, with sample

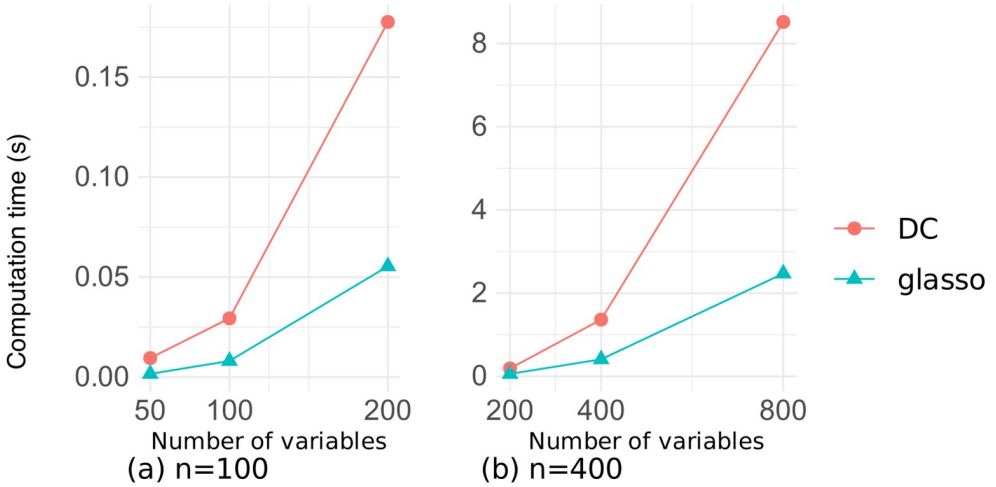

Fig 12. Computation time as a function of the number of variables on the chain graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g012

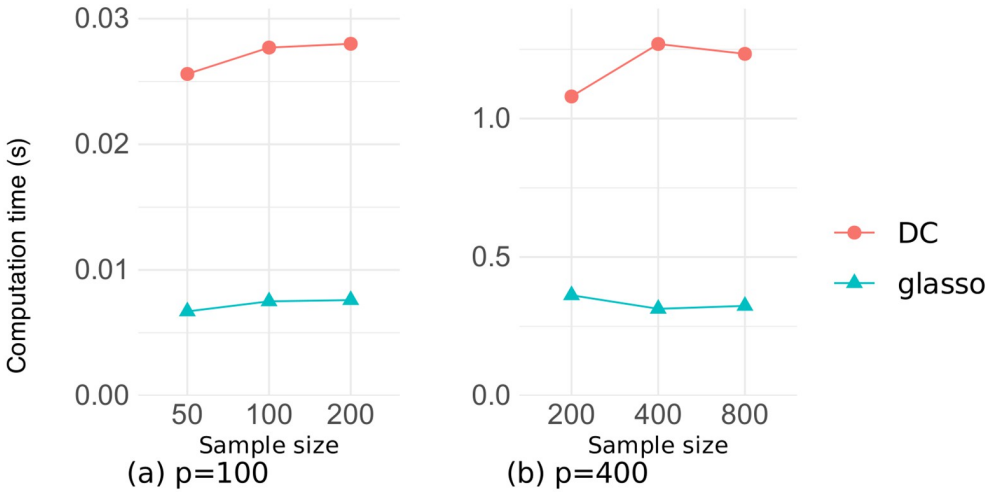

Fig 13. Computation time as a function of the sample size on the random graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g013

sizes $n \in \{100, 400\}$. There was a little difference in the computation time between the two datasets, and our DC method took about four times longer than did the glasso method. This is due to the two reasons, namely the repeated execution of the graphical lasso algorithm, and the repeated eigenvalue calculations in tuning the penalty parameter η in Algorithm 2. However, both methods took less than 1.5 seconds for $p \leq 400$, and our DC method converged in approximately 8 seconds even for $p = 800$, demonstrating that our algorithm was sufficiently fast.

Figs 13 and 14 illustrate the relationship between the sample size and the computation time for estimation on the datasets of random and chain graphs, respectively, where the number of variables is $p \in \{100, 400\}$. These figures confirm that the computation time for both methods was strongly dependent on the number of variables and changed very little even when the sample size was increased several times.

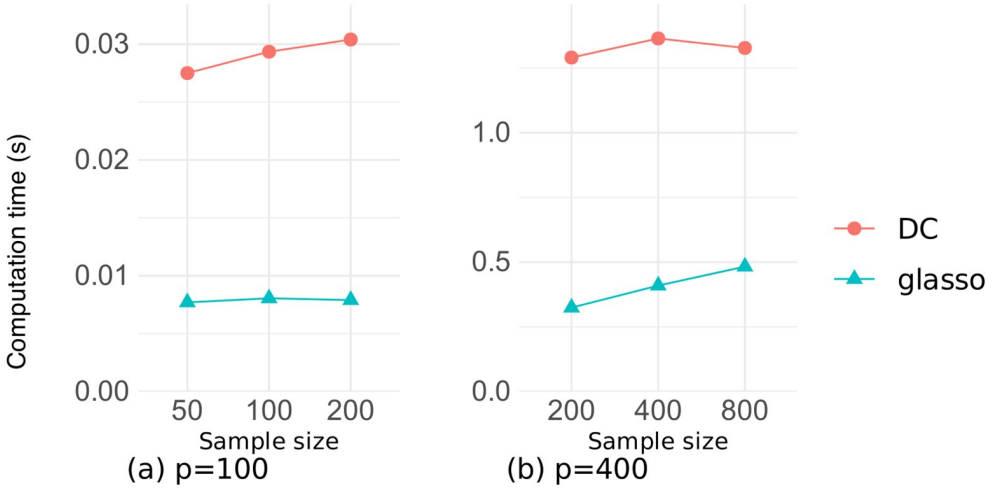

Fig 14. Computation time as a function of the sample size on the chain graph dataset.

https://doi.org/10.1371/journal.pone.0315740.g014

Table 1. Numbers of iterations (#Ite) and eigenvalue calculations (#Eig) in the DC algorithm on the random and chain graph datasets.

p	n	Random		Chain	
		#Ite	#Eig	#Ite	#Eig
100	50	2.0	8.0	2.0	8.0
	100	2.0	8.0	2.0	10.0
	200	2.0	10.0	2.0	10.0
400	200	2.0	10.0	2.0	10.0
	400	2.0	10.0	2.0	12.0
	800	2.0	10.0	2.0	12.0

https://doi.org/10.1371/journal.pone.0315740.t001

Table 1 lists the average numbers of iterations and eigenvalue calculations required by our DC algorithm. Recall here that the DC algorithm executes the graphical lasso algorithm at each iteration and repeatedly calculates the eigenvalues to tune the penalty parameter η. We can see from Table 1 that the DC algorithm terminated in only two iterations and calculated the eigenvalues around ten times.

Conclusion

We considered estimation of sparse Gaussian graphical models using the cardinality constraint based on the ℓ_0 norm. We reformulated the sparse estimation problem with the cardinality constraint as an unconstrained penalty form using the largest-K norm. To solve this problem efficiently, we designed a DC algorithm that repeatedly executes the graphical lasso algorithm.

To verify the performance of our method, we conducted computational experiments using two types of synthetic datasets. In the experiments where the number of edges was selected through cross-validation, our method estimated conditional independence graphs more accurately than did other conventional methods. In the experiments where the number of selected edges was given, our method outperformed the graphical lasso and SCAD regularization and was comparable to the adaptive lasso in terms of the edge selection accuracy. In addition, our method took only about four times as long as the graphical lasso, indicating that the computation of our algorithm is fast enough for practical use.

A future direction of study will be to overcome computational challenges of our algorithm for sparse GGM estimation. As for the computational efficiency, Nakayama and Gotoh [42] reported that proximal gradient methods outperformed DC algorithms in some aspects of sparse regression, and Zhou et al. [43] proposed a proximal alternating direction method of multipliers for DC optimization problems. Additionally, since our method solves a penalized form of the problem, the obtained solutions do not always satisfy the original cardinality constraint. Another direction of future research will be to extend our method to multivariate time series analysis [44–46].

Author Contributions

Conceptualization: Tomokaze Shiratori, Yuichi Takano.

Data curation: Tomokaze Shiratori.

Formal analysis: Tomokaze Shiratori.

Funding acquisition: Yuichi Takano.

Investigation: Tomokaze Shiratori, Yuichi Takano.

Methodology: Tomokaze Shiratori, Yuichi Takano.

Project administration: Tomokaze Shiratori.

Resources: Tomokaze Shiratori, Yuichi Takano.

Software: Tomokaze Shiratori.

Supervision: Yuichi Takano.

Validation: Tomokaze Shiratori.

Visualization: Tomokaze Shiratori.

Writing – original draft: Tomokaze Shiratori.

Writing – review & editing: Yuichi Takano.

References

1. Ortiz A, Munilla J, Álvarez-Illán I, Górriz JM, Ramírez J, Alzheimer's Disease Neuroimaging Initiative. Exploratory graphical models of functional and structural connectivity patterns for Alzheimer's Disease diagnosis. Frontiers in Computational Neuroscience. 2015; 9:132. https://doi.org/10.3389/fncom.2015.00132 PMID: 26578945
2. Idé T, Lozano AC, Abe N, Liu Y. Proximity-based anomaly detection using sparse structure learning. In: Proceedings of the 2009 SIAM International Conference on Data Mining; 2009. p. 97–108.
3. Tan C, Lee L, Tang J, Jiang L, Zhou M, Li P. User-level sentiment analysis incorporating social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2011. p. 1397–1405.
4. Fan J, Liao Y, Liu H. An overview of the estimation of large covariance and precision matrices. The Econometrics Journal. 2016; 19(1):C1–C32. https://doi.org/10.1016/j.jeconom.2018.04.002
5. Drton M, Maathuis MH. Structure learning in graphical modeling. Annual Review of Statistics and Its Application. 2017; 4(1):365–393. https://doi.org/10.1146/annurev-statistics-060116-053803
6. Chen LP. Estimation of graphical models: An overview of selected topics. International Statistical Review. 2024; 92(2):194–245. https://doi.org/10.1111/insr.12552
7. Dempster AP. Covariance selection. Biometrics. 1972; 28(1):157–175. https://doi.org/10.2307/2528966
8. Bickel PJ, Levina E. Covariance regularization by thresholding. The Annals of Statistics. 2008; 36(6):2577–2604. https://doi.org/10.1214/08-AOS600
9. Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B: Statistical Methodology. 1996; 58(1):267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x
10. Meinshausen N, Bühlmann P. High-dimensional graphs and variable selection with the Lasso. The Annals of Statistics. 2006; 34(3):1436–1462. https://doi.org/10.1214/009053606000000281
11. Peng J, Wang P, Zhou N, Zhu J. Partial correlation estimation by joint sparse regression models. Journal of the American Statistical Association. 2009; 104(486):735–746. https://doi.org/10.1198/jasa.2009.0126 PMID: 19881892
12. Banerjee O, Ghaoui LE, d'Aspremont A, Natsoulis G. Convex optimization techniques for fitting sparse Gaussian graphical models. In: Proceedings of the 23rd International Conference on Machine learning; 2006. p. 89–96.
13. Yuan M, Lin Y. Model selection and estimation in the Gaussian graphical model. Biometrika. 2007; 94(1):19–35. https://doi.org/10.1093/biomet/asm018
14. Friedman J, Hastie T, Tibshirani R. Sparse inverse covariance estimation with the graphical lasso. Biostatistics. 2008; 9(3):432–441. https://doi.org/10.1093/biostatistics/kxm045 PMID: 18079126
15. Mazumder R, Hastie T. The graphical lasso: New insights and alternatives. Electronic Journal of Statistics. 2012; 6:2125–2149. https://doi.org/10.1214/12-EJS740 PMID: 25558297
16. Cai T, Liu W, Luo X. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. Journal of the American Statistical Association. 2011; 106(494):594–607. https://doi.org/10.1198/jasa.2011.tm10155

17. Rolfs B, Rajaratnam B, Guillot D, Wong I, Maleki A. Iterative thresholding algorithm for sparse inverse covariance estimation. Advances in Neural Information Processing Systems. 2012; 25.

18. Liu H, Roeder K, Wasserman L. Stability approach to regularization selection (StARS) for high dimensional graphical models. Advances in Neural Information Processing Systems. 2010; 23. PMID: 25152607

19. Foygel R, Drton M. Extended Bayesian information criteria for Gaussian graphical models. Advances in Neural Information Processing Systems. 2010; 23.

20. Meinshausen N, Bühlmann P. Stability selection. Journal of the Royal Statistical Society Series B: Statistical Methodology. 2010; 72(4):417–473. https://doi.org/10.1111/j.1467-9868.2010.00740.x

21. Mestres AC, Bochkina N, Mayer C. Selection of the regularization parameter in graphical models using network characteristics. Journal of Computational and Graphical Statistics. 2018; 27(2):323–333. https://doi.org/10.1080/10618600.2017.1366910

22. Avella-Medina M, Battey HS, Fan J, Li Q. Robust estimation of high-dimensional covariance and precision matrices. Biometrika. 2018; 105(2):271–284. https://doi.org/10.1093/biomet/asy011 PMID: 30337763

23. Chun H, Lee MH, Kim SH, Oh J. Robust precision matrix estimation via weighted median regression with regularization. Canadian Journal of Statistics. 2018; 46(2):265–278. https://doi.org/10.1002/cjs.11356

24. Fan J, Li R. Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American statistical Association. 2001; 96(456):1348–1360. https://doi.org/10.1198/016214501753382273

25. Fan J, Feng Y, Wu Y. Network exploration via the adaptive LASSO and SCAD penalties. The Annals of Applied Statistics. 2009; 3(2):521–541. https://doi.org/10.1214/08-AOAS215SUPP PMID: 21643444

26. Zhang CH. Nearly unbiased variable selection under minimax concave penalty. Annals of Statistics. 2010; 38(2):894–942. https://doi.org/10.1214/09-AOS729

27. Zou H. The adaptive lasso and its oracle properties. Journal of the American Statistical Association. 2006; 101(476):1418–1429. https://doi.org/10.1198/016214506000000735

28. Dicker L, Huang B, Lin X. Variable selection and estimation with the seamless-L_0 penalty. Statistica Sinica. 2013; 23:929–962.

29. Wang Y, Zhu L. Variable selection and parameter estimation with the Atan regularization method. Journal of Probability and Statistics. 2016; 2016(1):6495417.

30. Fang EX, Liu H, Wang M. Blessing of massive scale: Spatial graphical model estimation with a total cardinality constraint approach. Mathematical Programming. 2019; 176(1):175–205. https://doi.org/10.1007/s10107-018-1331-z

31. Natarajan BK. Sparse approximate solutions to linear systems. SIAM Journal on Computing. 1995; 24 (2):227–234. https://doi.org/10.1137/S0097539792240406

32. Neumann J, Schnörr C, Steidl G. Combined SVM-based feature selection and classification. Machine Learning. 2005; 61:129–150. https://doi.org/10.1007/s10994-005-1505-9

33. Le Thi HA, Dinh TP, Le HM, Vo XT. DC approximation approaches for sparse optimization. European Journal of Operational Research. 2015; 244(1):26–46. https://doi.org/10.1016/j.ejor.2014.11.031

34. Gotoh Jy, Takeda A, Tono K. DC formulations and algorithms for sparse optimization problems. Mathematical Programming. 2018; 169:141–176. https://doi.org/10.1007/s10107-017-1181-0

35. Tao PD, et al. Algorithms for solving a class of nonconvex optimization problems. Methods of subgradients. In: North-Holland Mathematics Studies. vol. 129. Elsevier; 1986. p. 249–271.

36. Tao PD, An LH. Convex analysis approach to DC programming: Theory, algorithms and applications. Acta Mathematica Vietnamica. 1997; 22(1):289–355.

37. Le Thi HA, Pham Dinh T. DC programming and DCA: Thirty years of developments. Mathematical Programming. 2018; 169(1):5–68. https://doi.org/10.1007/s10107-018-1235-y

38. Phan DN, Le Thi HA, Dinh TP. Sparse covariance matrix estimation by DCA-based algorithms. Neural Computation. 2017; 29(11):3040–3077. https://doi.org/10.1162/neco_a_01012 PMID: 28957024

39. Touloumis A. Nonparametric Stein-type shrinkage covariance matrix estimators in high-dimensional settings. Computational Statistics & Data Analysis. 2015; 83:251–261. https://doi.org/10.1016/j.csda.2014.10.018

40. Williams DR. Beyond lasso: A survey of nonconvex regularization in Gaussian graphical models. PsyArXiv. 2020;.

41. Hastie T, Tibshirani R, Tibshirani R. Best subset, forward stepwise or lasso? Analysis and recommendations based on extensive comparisons. Statistical Science. 2020; 35(4):579–592. https://doi.org/10.1214/19-STS733

42. Nakayama S, Gotoh Jy. On the superiority of PGMs to PDCAs in nonsmooth nonconvex sparse regression. Optimization Letters. 2021; 15(8):2831–2860. https://doi.org/10.1007/s11590-021-01716-1

43. Zhou Y, He H, Zhang L. A proximal alternating direction method of multipliers for DC programming with structured constraints. Journal of Scientific Computing. 2024; 99(3):89. https://doi.org/10.1007/s10915-024-02550-0

44. Hallac D, Park Y, Boyd S, Leskovec J. Network inference via the time-varying graphical lasso. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2017. p. 205–213.

45. Hyndman R. Forecasting: Principles and practice. OTexts; 2018.

46. Shiratori T, Kobayashi K, Takano Y. Prediction of hierarchical time series using structured regularization and its application to artificial neural networks. Plos One. 2020; 15(11):e0242099. https://doi.org/10.1371/journal.pone.0242099 PMID: 33180811