arXiv:2307.00127v1 [stat.ME] 30 Jun 2023

High-Dimensional Bayesian Structure Learning in Gaussian Graphical Models using Marginal Pseudo-Likelihood

Reza Mohammadi, Marit Schoonhoven, Lucas Vogels, S. Ilker Birbil
Department of Business Analytics, University of Amsterdam

July 4, 2023

Abstract

Gaussian graphical models depict the conditional dependencies between variables within a multivariate normal distribution in a graphical format. The identification of these graph structures is an area known as structure learning. However, when utilizing Bayesian methodologies in structure learning, computational complexities can arise, especially with high-dimensional graphs surpassing 250 nodes. This paper introduces two innovative search algorithms that employ marginal pseudo-likelihood to address this computational challenge. These methods can swiftly generate reliable estimations for problems encompassing 1000 variables in just a few minutes on standard computers. For those interested in practical applications, the code supporting this new approach is made available through the R package BDgraph.

Keywords: Bayesian structure learning; High dimensionality; Gaussian; Graphical model; Marginal pseudo-likelihood; Birth-death process; Markov Chain Monte Carlo approach.

1 Introduction

Graphical models have been used in many application areas for analyzing conditional dependencies in a network with a large number of variables. Partial correlation is the correlation between two variables with the effect of the other variables removed; two variables are called conditionally independent if they are partially uncorrelated. The conditional dependencies can be depicted in a graph where each node corresponds to a random variable (see Lauritzen (1996)). For an undirected graph, a missing edge between two nodes indicates that the two variables are conditionally independent (Rue and Held (2005)). Estimating the structure of the graph is called structure learning.

Consider the undirected graph $G = (V, E)$, where $V = \{1, ..., p\}$ is the set of nodes, E is the set of edges and where we use if $(i, j) \in E$ then $i < j$. The nodes correspond to the random variables and form a p-dimensional vector $\boldsymbol{x}$. In this article, we focus on the Gaussian graphical model (GGM), in which case $\boldsymbol{x} = (x_1, ..., x_p)^T \sim \mathcal{N}_p(\mathbf{0}, \Sigma)$ with Σ the covariance matrix. The precision matrix is denoted by $K = \Sigma^{-1}$. If $K_{ij} = 0$, then $(i, j) \notin E$ and x_i and x_j are conditionally independent (see Lauritzen (1996)).

For the purpose of structure learning in GGMs, both frequentist (see e.g. Friedman et al. (2008)) and Bayesian approaches are available. In this paper, we focus on Bayesian methods in graphical models, which is known as Bayesian structure learning (BSL). BSL uses knowledgde about prior beliefs, which can increase the accuracy of the output. Another advantage of BSL is that it outputs the full posterior distribution of the parameters. However, the disadvantage of existing BSL methods is that they require considerable computational time, especially for high dimensions, i.e. when the number of variables (p) is high.

Suppose we have a random sample $\boldsymbol{X} = (\boldsymbol{X}^1, ..., \boldsymbol{X}^n)^T$ from $\mathcal{N}_p(\mathbf{0}, \Sigma)$ and the depen-

dence structure is represented by an undirected graph G^*. Structure learning aims at identifying G^* based on $\boldsymbol{X}$ and a Bayesian approach achieves this by obtaining the posterior distribution of a graph G given the data $\boldsymbol{X}$, which is denoted by $P(G|\boldsymbol{X})$. Obtaining $P(G|\boldsymbol{X})$ requires the calculation of $P(\boldsymbol{X}|G)$, which in turn requires the calculation of a complex integral. For this reason, most BSL methods obtain the joint posterior distribution of G and K given the data $\boldsymbol{X}$, which is denoted by $P(G, K|\boldsymbol{X})$.

Calculating $P(G, K|\boldsymbol{X})$ is computationally impossible because of the number of possible graphs and precision matrices. Therefore, most BSL methods rely on a search algorithm over the graph space. Most BSL methods use Markov Chain Monte Carlo (MCMC) sampling. Green (1995) proposed a so-called reversible jump MCMC, which is based on a discrete-time Markov chain. The derivation of $P(G, K|\boldsymbol{X})$ requires the prior distribution of the precision matrix K given the graph G, denoted by $P(K|G)$. Most BSL methods use the G-Wishart prior for $P(K|G)$, which was introduced by Roverato (2008). A computationally expensive step in the search algorithm is to determine the ratio of prior normalizing constants for the G-Wishart distribution. To reduce the computational time, approximations for the ratio of normalizing constants have been proposed by Wang and Li (2012), Lenkoski (2013) and Hinne et al. (2014) among others. In addition, Mohammadi and Wit (2015) proposed a search algorithm known as the birth-death MCMC algorithm, which is based on a continuous-time Markov chain, to explore the graph space more efficiently. Subsequently, Mohammadi et al. (2021) proposed a closed-form approximation to obtain the ratio of the normalizing constants, which further reduces the computational time of the reversible jump and birth-death search algorithms. Wang (2015) introduced a new type of MCMC-based search algorithm, namely the spike-and-slab algorithm. His algorithm uses a new prior on the precision matrix K and therefore avoids the computationally costly

sample from the G-Whisart distribution and also allows changing the graph by more than one edge leading for a faster convergence of the MCMC chain. However, a disadvantage of the existing MCMC algorithms is that they evaluate $P(G, K|\boldsymbol{X})$ instead of $P(G|\boldsymbol{X})$. This means that, when a graph is updated during an iteration of the algorithm, the precision matrix K needs to be updated, which is a computationally expensive step. Existing MCMC-based BSL methods are therefore computationally expensive from 100 variables upward for the reversible jump and birth-death MCMC algorithms, or 250 variables for the spike-and-slab algorithm introduced by Wang (2015).

The main contribution of this paper is to introduce a new MCMC-based BSL method. This method evaluates $P(G|\boldsymbol{X})$ instead of $P(G, K|\boldsymbol{X})$ and thus avoids having to sample the precision matrix at each iteration of the search algorithm. With this method, we approximate $P(\boldsymbol{X}|G)$ by using the MPL approach. The MPL has been used by Pensar et al. (2017) to learn undirected graphical models with discrete variables. Leppä-Aho et al. (2017) applied the MPL approach to approximate $P(\boldsymbol{X}|G)$ but used a hill-climbing search algorithm that outputs single graph estimation. In this paper, we will use the MPL approach in combination with the MCMC search algorithm, allowing statistical inference.

The rest of the paper is structured as follows. Section 2 reviews the basic concepts related to BSL, Section 3 outlines the new BSL method together with the MCMC-based search algorithm and Section 4 evaluates the proposed BSL method as well as existing state-of-the-art BSL methods. We end with concluding remarks and a description of some issues for future research.

2 Bayesian Structure Learning

This section starts by reviewing the basic concepts related to Bayesian Structure Learning (BSL).

Recall from Section 1 that $\boldsymbol{X} = (\boldsymbol{X}^1, ..., \boldsymbol{X}^n)^T$ denotes the sample used to identify G^*. We assume that $\boldsymbol{X}$ has an $\mathcal{N}_p$ distribution whose conditional dependence structure is represented by the undirected graph G^*, an unknown quantity. A BSL approach aims at identifying G^* based on $\boldsymbol{X}$ by obtaining the posterior probability of the graph conditional on the data

$$P(G|\boldsymbol{X}) \propto P(\boldsymbol{X}|G)P(G) \tag{1}$$

where $P(G)$ is the likelihood of a graph G, for which a non-informative prior is often used such as the uniform distribution over all graphs with p nodes (see Mohammadi et al. (2021)), and $P(\boldsymbol{X}|G)$ is the marginal likelihood of G, which is given by

$$P(\boldsymbol{X}|G) = \int_K P(\boldsymbol{X}|G, K)P(K|G)dK \tag{2}$$

where $P(K|G)$ denotes the prior for K given G and $P(\boldsymbol{X}|G, K)$ is the likelihood function. The G-Wishart distribution (Roverato (2008) and Letac and Massam (2007)) is the conjugate prior for $P(K|G)$. The G-Wishart density is given by

$$P(K|G) = \frac{1}{I_G(b, D)}|K|^{\frac{b-2}{2}} exp\{\frac{-1}{2}tr(KD)\}\mathbf{1}_{P_G}(K) \tag{3}$$

where $|K|$ denotes the determinant of K, $tr(A)$ is the trace of a square matrix A, P_G is the set of positive definite matrices K with $k_{ij} = 0$ if $(i, j) \notin E$ and $\mathbf{1}_{P_G}(K)$ is an indicator function that equals 1 if $K \in P_G$ and 0 otherwise. The symmetric positive definite matrix D and the scalar $b > 2$ are the scale and the shape parameters of the G-Wishart distribution. They are often set to $b = 3$ and $D = I$. $I_G(b, D)$ is the normalizing constant, which is given

by

$$I_G(b, D) = \int_{K \in P_G} |K|^{\frac{b-2}{2}} exp\{\frac{-1}{2} tr(KD)\} dK. \tag{4}$$

Substituting the G-Wishart distribution into Equation 2 gives

$$P(\boldsymbol{X}|G) = (2\pi)^{-np/2} \frac{I_G(b+n, D+\boldsymbol{X}^T\boldsymbol{X})}{I_G(b, D)}. \tag{5}$$

This ratio of normalizing constants is hard to obtain. Most BSL methods therefore sample over the joint space of graphs and precision matrices. The joint posterior distribution of the graph G and the precision matrix K is given by

$$\begin{aligned} P(G, K|\boldsymbol{X}) &\propto P(\boldsymbol{X}|K, G)P(K|G)P(G) \\ &\propto P(G)\frac{1}{I_G(b, D)}|K|^{\frac{b+n-2}{2}} exp\{\frac{-1}{2} tr(K(D+\boldsymbol{X}^T\boldsymbol{X}))\}. \end{aligned} \tag{6}$$

Calculating $P(G, K|\boldsymbol{X})$ is not possible because the set of all possible graphs is too big. To obtain $P(G, K|\boldsymbol{X})$, most BSL methods therefore use Markov Chain Monte Carlo (MCMC) sampling. A well-known sampling algorithm for GGMs is the reversible jump MCMC algorithm based on a discrete-time Markov chain, which was first used for Gaussian graphical models by Dobra et al. (2011). During each iteration $s = 1, ..., S$, the state of the Markov chain is denoted by $(G^{(s)}, K^{(s)})$ and the chain jumps to state $(G^{(s+1)}, K^{(s+1)})$. For a sufficiently large number of iterations, the distribution of the sample pairs $\{(G^{(1)}, K^{(1)}), ..., (G^{(S)}, K^{(S)})\}$ approximates the posterior distribution $P(G, K|\boldsymbol{X})$. The reversible jump MCMC algorithm applies the Metropolis Hastings (MH) algorithm that was introduced by Green (1995). Each iteration first proposes a new graph $G^{(s+1)}$ by deleting or adding one edge from $G^{(s)}$; the move to the new graph is accepted with probability α. To get the probability α, the following, computationally expensive, ratio of normalizing constants needs to be determined

$$\frac{I_{G^{(s+1)}}(b, D)}{I_{G^{(s)}}(b, D)}. \tag{7}$$

Given the new graph $G^{(s+1)}$, a new precision matrix $K^{(s+1)}$ needs to be derived by sampling from the $G^{(s+1)}$-Wishart distribution. This step is also computationally expensive. After the introduction of the reversible jump MCMC for BSL, several improvements have been proposed to reduce its computational time. Wang (2012), Cheng and Lenkoski (2012) and Mohammadi et al. (2021) introduced approximations for the ratio of the normalizing constants in Equation 7. Second, Lenkoski (2013) proposed a more efficient sampling algorithms to reduce the computational time needed to sample from the G-Wishart distribution. Hinne et al. (2014) used existing techniques based on conditional Bayes factors and the continuous-time Markov chain to reduce the computational time.

The reversible jump algorithms are based on a discrete-time Markov chain. As a consequence, these algorithms often have low acceptance rates and need more time for MCMC chain convergence. To overcome this issue, Mohammadi and Wit (2015) proposed a new algorithm based on the continuous-time Markov chain. To move from a state $(G^{(s)}, K^{(s)})$ with edge set $E^{(s)}$ to a new state $(G^{(s+1)}, K^{(s+1)})$, the algorithm first calculates the so-called birth rates for all edges not in $E^{(s)}$ and death rates for all edges in $E^{(s)}$. The birth (death) rate of an edge (not) in $E^{(s)}$ follows an independent Poisson process with rate equal to the corresponding birth (death) rate, and the time between two events is exponentially distributed. In every state $(G^{(s)}, K^{(s)})$, the chain spends a waiting time $W^{(s)}$ in this state before it jumps to a new state $(G^{(s+1)}, K^{(s+1)})$. After a sufficiently large number of jumps, the samples of $(G^{(s)}, K^{(s)})$ weighted by their waiting time $W^{(s)}$ are an approximation of the posterior distribution $P(G, K|\boldsymbol{X})$.

Despite the above improvements, reversible jump and birth-death search algorithms need a lot of computational time because, during each iteration, a new precision matrix needs to be sampled from the G-Wishart distribution and, during each iteration, the graph

changes by at most one edge. To overcome these problems, Wang (2015) introduced the so-called spike-and slab-prior that replaces the G-Wishart prior.

3 Bayesian Structure Learning With Marginal Pseudo-Likelihood

In this section, we derive new BSL methods based on the marginal pseudo-likelihood (MPL) approximation. We describe the approach in combination with the reversible jump and birth-death MCMC search algorithms. Section 3.1 and 3.2 describe the birth-death and reversible jump search algorithms respectively, and Section 3.3, derives the birth and death probabilities for the birth-death algorithm and α for the reversible jump algorithm, using the MPL approximation.

3.1 Birth-Death MCMC Algorithm

Recall that, for this new method, we aim to save computational time by obtaining $P(G|\boldsymbol{X})$ from Equation 1 (instead of $P(G, K|\boldsymbol{X})$ from Equation 6) and searching over the graph space instead of the joint space of graphs and precision matrices. Calculating $P(G|\boldsymbol{X})$ for all possible graphs G directly is not possible because the graph space is too big. We therefore use MCMC sampling. Mohammadi and Wit (2015) introduced birth-death MCMC sampling to search over the joint space of graphs and precision matrices in order to approximate $P(G, K|\boldsymbol{X})$. Since we approximate $P(G, K|\boldsymbol{X})$ by the MPL approximation, the birth-death MCMC search algorithm (BMCMC) that we describe below searches only over the graph space.

The birth-death algorithm is based on a continuous-time Markov chain (Preston (1975))

and was first applied to Gaussian graphical models by Mohammadi and Wit (2015). During iteration s, with $s = 1, ..., S$, the state of the Markov chain is a certain graph $G^{(s)}$ and it jumps to a new state $G^{(s+1)}$ by adding or removing one edge. These events of adding (removing) one edge are called birth (death) processes and are modeled as independent Poisson processes. Each edge $e \notin E$ is born independently of other edges that are not in E as a Poisson process with rate $B_e(G)$. If the birth of edge e occurs, the process jumps to $G^{+e} = (V, E \cup \{e\})$. Similarly, each edge $e \notin E$ dies independently of other edges in E as a Poisson process with rate $D_e(G)$. If the death of edge e occurs, the process jumps to $G^{-e} = (V, E \backslash \{e\})$. Since the birth and death processes are modeled as independent Poisson processes, the time between two consecutive events is exponentially distributed with mean

$$W(G) = \frac{1}{\sum_{e \in \bar{E}} B_e(G) + \sum_{e \in E} D_e(G)}, \tag{8}$$

where $W(G)$ is called the waiting time and the birth/death probabilities are

$$P(\text{birth of edge } e) = \frac{B_e(G)}{\sum_{e \in \bar{E}} B_e(G) + \sum_{e \in E} D_e(G)}, \text{ for each } e \in \bar{E} \tag{9}$$

$$P(\text{death of edge } e) = \frac{D_e(G)}{\sum_{e \in \bar{E}} B_e(G) + \sum_{e \in E} D_e(G)}, \text{ for each } e \in E. \tag{10}$$

Dobra and Mohammadi (2018) proved that the following is a sufficient condition for the stationary distribution of the birth-death process to equal the posterior distribution $P(G|\boldsymbol{X})$

$$B_e(G)P(G|\boldsymbol{X}) = D_e(G^{+e})P(G^{+e}|\boldsymbol{X}). \tag{11}$$

Based on Equation 11, the birth and death rates are defined as:

$$B_e(G) = min\left\{\frac{P(G^{+e}|\boldsymbol{X})}{P(G|\boldsymbol{X})}, 1\right\} \text{ for each } e \in \bar{E} \tag{12}$$

and

$$D_e(G) = min\left\{\frac{P(G^{-e}|\boldsymbol{X})}{P(G|\boldsymbol{X})}, 1\right\} \text{ for each } e \in E \tag{13}$$

and more compactly as

$$R_e(G) = min\left\{\frac{P(G'|\boldsymbol{X})}{P(G|\boldsymbol{X})}, 1\right\} \tag{14}$$

with G' either G^{+e} or G^{-e}. This birth-death algorithm that searches over the graph space only is denoted by BD-G and described in Algorithm 1.

Algorithm 1: BD-G search algorithm

Input: A graph $G = (V, E)$ with data $\boldsymbol{X}$.

for *S iterations* **do**

  for *all the possible moves in parallel* **do**

    Calculate the birth and death rates by Equation 14;

  Calculate the waiting time by Equation 8;

  Update the graph based on the birth/death probabilities in Equations 9 and 10.

Output: Samples from the posterior distribution (1).

The output of Algorithm 1 consists of a set of S graphs $G^{(1)}, ..., G^{(S)}$ and a set of S waiting times $W^{(1)}, ..., W^{(S)}$. Based on the Blackwellized estimator given by Cappé et al. (2003), the estimated posterior probability of each graph is proportional to the expectation of the waiting time of that graph. The posterior probability of an edge e can be estimated by

$$\hat{P}(\text{edge } e|\boldsymbol{X}) = \frac{\sum_{s=1}^{S} \mathbf{1}(e \in G^{(s)})W^{(s)}}{\sum_{s=1}^{S} W^{(s)}}, \tag{15}$$

where S denotes the number of MCMC iterations and $\mathbf{1}(e \in G^{(s)})$ is the indicator function that equals 1 if $e \in G^{(s)}$ and 0 otherwise.

3.2 Reversible Jump MCMC Algorithm

The reversible jump MCMC searach algorithm uses the Metropolis Hastings (MH) algorithm introduced by Green (1995). Within each step of the search algorithm, a new graph

G' is proposed by adding or deleting an edge from G, and the acceptance probability $\alpha(G)$ is calculated, which is used to decide whether or not the graph should be changed. We use the random distribution as the proposal distribution for the new graph, as has been done by Dobra et al. (2011) and Lenkoski and Dobra (2011). The acceptance probability $\alpha(G)$ can be calculated as

$$\alpha(G) = min\left\{\frac{P(G'|\boldsymbol{X})}{P(G|\boldsymbol{X})}, 1\right\} \tag{16}$$

as shown by Green (1995). Note that $\alpha(G)$ is equal to $R_e(G)$ given by Equation 14. The reversible jump MCMC search algorithm that we use within our new method only moves over the graph space and is abbreviated to RJ-G. The procedure is described by Algorithm 2.

Algorithm 2: RJ-G search algorithm

Input: A graph $G = (V, E)$ with data $\boldsymbol{X}$.

for S *iterations* **do**

Select a proposal graph;

Calculate $\alpha(G)$ according to 16 and update the graph.

Output: Samples from the posterior distribution (1).

3.3 Marginal Pseudo-Likelihood

In this subsection, we approximate the ratio in 14 to obtain $R_e(G)$. To do so, we rewrite the fraction in 14 as

$$\frac{P(G'|\boldsymbol{X})}{P(G|\boldsymbol{X})} = \frac{P(G')P(\boldsymbol{X}|G')}{P(G)P(\boldsymbol{X}|G)}, \tag{17}$$

$\forall\ G' = (V, E \cup e = (i, j))$ or $G' = (V, E \backslash e = (i, j))$ and $G = (V, E)$. To approximate $P(\boldsymbol{X}|G)$, we use the MPL approximation:

$$P(\boldsymbol{X}|G) \approx \prod_{h=1}^{p} P_G(\boldsymbol{X}_h|\boldsymbol{X}_{nb(h)}) \tag{18}$$

where $\boldsymbol{X}_{nb(h)}$ denotes the data belonging to $x_{nb(h)}$ with $nb(h)$ the neigbors of node h i.e. nodes i for which $(i,j) \in E$, and P_G is the probability based on graph G. We then have:

$$\frac{P(\boldsymbol{X}|G')}{P(\boldsymbol{X}|G)} = \frac{\prod_{h=1}^{p} P_{G'}(\boldsymbol{X}_h|\boldsymbol{X}_{nb(h)})}{\prod_{h=1}^{p} P_G(\boldsymbol{X}_h|\boldsymbol{X}_{nb(h)})} = \frac{P_{G'}(\boldsymbol{X}_i|\boldsymbol{X}_{nb(i)})P_{G'}(\boldsymbol{X}_j|\boldsymbol{X}_{nb(j)})}{P_G(\boldsymbol{X}_i|\boldsymbol{X}_{nb(i)})P_G(\boldsymbol{X}_j|\boldsymbol{X}_{nb(j)})}. \tag{19}$$

For this last step, we use the fact that the graphs G and G' are the same except that one link $e = (i,j)$ is added or removed in moving from G to G'. As a result, the probabilities of all nodes except x_i and x_j are the same and can be removed from the fraction; only the probabilities that relate to x_i and x_j remain. Finally, from Leppä-Aho et al. (2017) and Consonni and Rocca (2012), we know that

$$P_G(\boldsymbol{X}_h|\boldsymbol{X}_{nb(h)}) = \pi^{\frac{-(n-1)}{2}} \frac{\Gamma\left(\frac{n+p_h}{2}\right)}{\Gamma\left(\frac{p_h+1}{2}\right)} n^{\frac{-1(2p_h+1)}{2}} \left(\frac{|S_{fa(h)}|}{|S_{nb(h)}|}\right)^{\frac{-(n-1)}{2}}, \tag{20}$$

where $fa(h) = nb(h) \cup h$ is the family of variable x_h, p_h is the size of the set $nb(h)$, $S = \boldsymbol{X}^T\boldsymbol{X}$ is the unscaled sample covariance matrix, S_A denotes the submatrix of S corresponding to the variables in set A, and matrices $S_{fa(h)}$ and $S_{nb(h)}$ should be positive definite for every h, which is the case if $n \geq max\{p_h + 1 | h = 1, ..., p\}$. Using the result from Equation 20 to obtain the probabilities in the right-hand side of Equation 19, we can easily calculate the fraction $\frac{P(G'|\boldsymbol{X})}{P(G|\boldsymbol{X})}$ in Equation 17 to obtain $R_e(G)$ in 14.

4 Performance Evaluation

In this section, we compare the performances of the newly proposed methods based on the MPL approximation combined with the reversible jump and birth-death MCMC search algorithms (MPL-BD and MPL-RJ respectively) with two state-of-the-art methods, namely the birth-death MCMC method (BD) proposed by Mohammadi et al. (2021), and the method proposed by Wang (2015) based on the spike-and-slab prior (SS). We use simulation to evaluate the methods in terms of quality and efficiency. Section 4.1 describes

the performance metrics used, Section 4.1 gives the simulation settings and Section 4.3 summarizes the results.

4.1 Performance Metrics

The methods will be evaluated in terms of accuracy and computational time. We start our explanation with graph accuracy. Recall that the MCMC methods do not produce a single graph estimation G but, for each MCMC iteration, a graph resulting in a collection of graphs $\{G^{(1)}, ..., G^{(S)}\}$ with S the number of MCMC iterations. The graph collection $\{G^{(1)}, ..., G^{(S)}\}$ is used to calculate the accuracy metrics namely the Area Under Curve (AUC) (see e.g. Hanley and Mcneil (1982)), the average edge inclusion probability for all edges in the underlying graph G^* (p^+) and the average edge inclusion probability for all the non-edges (p^-).

To define the AUC, we first have to calculate the edge inclusion probability p_{ij}, the true positive ratio (TPR) and the false positive ratio (FPR). The edge inclusion probability p_{ij} is defined by Equation 15. Note that $W^{(s)} = 1$ for the MPL-RJ and RJ methods. The TPR (the ratio between the number of incorrectly identified edges and the number of true edges) can then be calculated as

$$TPR = \frac{1}{|E^*|} \sum_{\forall (i,j) \in G^*} I[p(i,j) > \alpha], \tag{21}$$

with $|E^*|$ the number of edges in the true graph G^* and α the threshold value. Similary, the FPR (the ratio between the incorrectly identified edges and the number of non-edges) is calculated by

$$TPR = \frac{1}{|\bar{E}^*|} \sum_{\forall (i,j) \notin G^*} I[p(i,j) > \alpha], \tag{22}$$

with $|\bar{E}^*|$ the number of non-edges in the true graph G^*. The Receiver Operating Characteristic (ROC) shows the TPR and FPR for different values of α. For a method that ranks

perfectly, it is possible to find a threshold value α for which the FPR value is still 0 and the TPR value is 1. In that case, the area under the ROC curve (AUC) is 1. Thus, the AUC metric evaluates the quality of the ranking based on the outputted p_{ij} values.

The AUC does not evaluate the level of the p_{ij} values and in practice it is not known what the ideal value for α is. This is why we will also evaluate the absolute level p_{ij} values, by calculating the average p_{ij} for the edges in G^*, resulting in

$$p^+ = \frac{\sum_{\forall (i,j) \in G^*} p_{ij}}{|E^*|}, \tag{23}$$

and the average p_{ij} for the non-edges in G^*, resulting in

$$p^- = \frac{\sum_{\forall (i,j) \notin G^*} p_{ij}}{|\bar{E}^*|}. \tag{24}$$

The metric p^+ ranges from 0 (worst) to 1 (best); p^- ranges from 0 (best) to 1 (worst).

Apart from accuracy, we evaluate the efficiency of the methods by measuring the wall clock time using the following procedure. We run the methods for at least XXX iterations and then determine whether the AUC has stabilized based on visual inspection. If the AUC has not stabilized, we run the method again for XXXX iterations. If the AUC has converged after those iterations, we determine the minimum time t for which the AUC differs no more than 0.01 from the AUC value in the last iteration.

4.2 Simulation Settings

In the following, we set out the simulation settings, namely the graph types, values for n and p, the number of simulations and the method-specific settings.

We consider three graph types:

1. *Random:* A graph in which every edge is randomly generated from independent Bernoulli distributions with probability 0.2.

2. *Cluster:* A graph in which the number of clusters is $\max(2, [p/20])$. Each cluster has the same structure as the random graph.

3. *Scale-free:* A tree graph generated with the B-A algorithm provided by Albert and Barabási (2002).

The expected number of edges depends on the value of p and the graph type, and is given by Table 1.

p	50	100	500	1000
Random	20%	20%	20%	20%
Cluster	8%	3.3%	0.8%	0.4%
Scale-free	4%	2%	0.4%	0.2%

Table 1: *Expected number of edges, expressed as a percentage of the number of pairwise combinations of p.*

We used $p = 10, 50, 100, 500, 1000$ and $n \in p, 2p, 10p$. For each sampled graph G^*, the true precision matrix K^* was sampled from the G-Wishart distribution $W_G(3, I_p)$. For $p = 10, 50, 100$, we sampled 50 true graphs and corresponding precision matrices; for $p = 500, 1000$, we sampled 10 true graphs and precision matrices. Subsequently, for each combination of G^* and K^*, we sampled n observations from the $p-$variate normal distribution with covariance matrix $\Sigma^* = K^{*-1}$. As a prior for G, we used $P(G) = 0.2^{|E|}0.8^{p(p-1)/2-|E|}$. For each MCMC search algorithm, we initialized the Markov chain with an empty graph with p nodes. The number of iterations depended on AUC stabilization, varying between 5000 and 200 million. All methods were implemented using the BDgraph R package by Mohammadi et al. (2021) and the ssgraph R package by Mohammadi (Mohammadi). For the

SS algorithm, we set $\epsilon = 0.02, \upsilon = 2$ and $\lambda = 2$ because these result in optimal performance, as shown by Wang (2015). We did not use burn-in iterations.

4.3 Results

In this subsecton, we report the results of the simulation. As mentioned before, we evaluate the discriminative power with the AUC metric, the level of the edge inclusion probabilities with metrics p^+ and p^- and the efficiency of the methods in terms of wall clock time until AUC convergence.

Table 2 shows the AUC values. We can learn from the table that, for problems with a moderate number of variables ($50 \leq p \leq 100$), the AUC values are comparable. When both the number of variables and observations are low ($p = 10$ in combination with $n = 10$ or $n = 20$), the MPL-based methods achieve a slightly lower AUC value. When the number of variables is high, i.e. $p = 500$ or $p = 1000$, the AUC value is considerably lower for the BD methods than for the other methods. This is due to the fact that there are a lot of true edges in G^*: 99,900. The Markov chain starts with an empty graph and the MPL-BD method changes only one edge at a time so that too many iterations are necessary to approach the true graph.

The level of the edge inclusion probabilities is evaluated by means of p^+ and p^-. Table 3 shows the p^+ values; these values should be close to 1 because then the edges in the true graph G^* have a high edge inclusion probability p_{ij}. When the number of variables is moderate ($p = 50$ or $p = 100$), the BD method has the highest p^+ value. For all three graph types, a high number of variables ($p = 500$ or $p = 1000$) raises the p^+ value of the MPL-based methods considerably. It is surprising that the SS method performs poorly in terms of p_{ij}'s with its p^+ value is considerably lower in all situations.

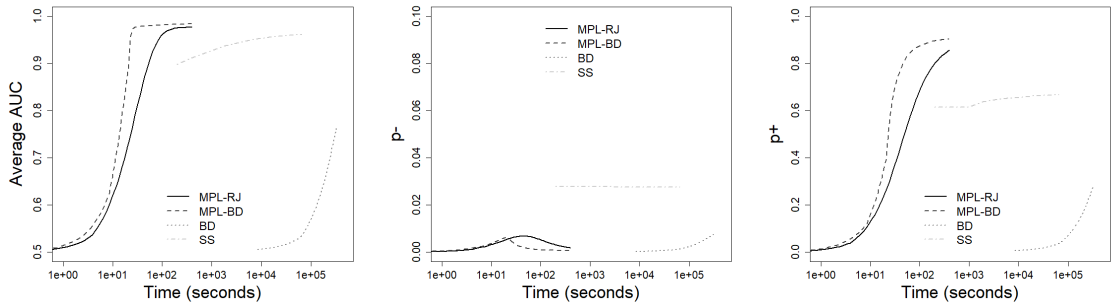
We also evaluate the p^- metric, whose values are shown in Table 4. p^- should be close to 0 because it means that the p_{ij} values of the non-edges of G^* are close to 0. For a moderate or high number of variables ($p \geq 50$), the p^- values of the SS method are higher than those of the other methods, while the MPL-based methods have the lowest p^- values. Only in the case of the random graph does the MPL-BD method show worse performance, which is due to the huge number of edges in this situation.

The wall clock time until AUC convergence is shown in Table 5. For the random graph, the MPL-RJ and SS methods are the fastest when the number of variables is high, i.e. $p = 500, 1000$. However, that is an exceptional case because the number of possible edges in the true graph G^* is enormous. For the cluster graph, when the number of variables is high, i.e. $p = 500, 1000$, the MPL-based methods are the fastest.

We also depict the differences in results by selecting two situations. Figure 1 shows plots of the AUC, $p+$ and $p-$ as functions of the wall clock time for the cluster graph given a high number of variables and observations, i.e. $p = 500$ and $n = 1000$. We can see that the AUCs of the MPL-based methods converge the fastest and that the resulting p^+ and p^- values are better. Figure 2 shows the results for the random graph given a relatively small problem size, i.e. $p = 50$ and $n = 500$. The plots show that the AUC of the MPL-based methods converges the fastest but not much faster than the other methods and that the p^+ and p^- values of the SS method are not as good.

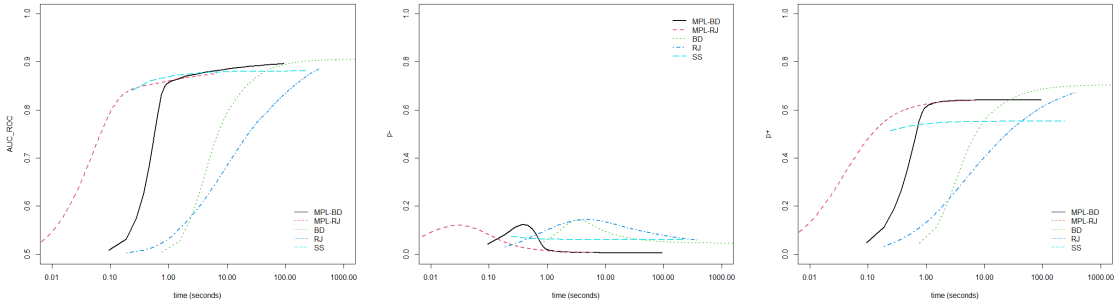
To summarize, the simulation shows that for large problems, i.e. $p = 500, 1000$, the MPL-based methods are of similar quality in terms of discriminative power and calibration, and show outstanding performance in terms of computational efficiency. The only exception is that, when the number of links is extremely high (the probability of a link is 0.2), the MPL-BD method takes up considerable computational time. However, we can speed up

the algorithm by using parallelization (i.e. calculating the birth and death rates in parallel) or by considering multiple jumps in each iteration of the algorithm.



(a) AUC (b) $p-$ (c) $p+$

Figure 1: Convergence of AUC (left), $p-$ (middle) and $p+$ (right) for the case of $p = 500$, $n = 5000$ on the cluster graph



(a) AUC (b) $p-$ (c) $p+$

Figure 2: Convergence of AUC (left), $p-$ (middle) and $p+$ (right) for the case of $p = 50$, $n = 500$ on the random graph

5 Concluding Remarks and Issues for Future Research

In this paper, we have proposed a new method that is based on a combination of MCMC search algorithms, namely the reversible jump and birth-death MCMC, using the MPL to

graph	p	n	MPL-RJ	MPL-BD	BD	SS
random	10	10	0.62 (0.63)	0.61 (0.61)	0.7 (0.71)	0.7 (0.7)
		20	0.73 (0.74)	0.73 (0.75)	0.76 (0.76)	0.76 (0.77)
		100	0.86 (0.89)	0.86 (0.88)	0.88 (0.9)	0.87 (0.88)
	50	50	0.71 (0.71)	0.72 (0.72)	0.73 (0.73)	0.72 (0.72)
		100	0.77 (0.77)	0.78 (0.79)	0.79 (0.79)	0.78 (0.78)
		500	0.88 (0.88)	0.9 (0.89)	0.91 (0.91)	0.88 (0.88)
	100	100	0.69 (0.69)	0.72 (0.72)	0.73 (0.72)	0.73 (0.73)
		200	0.75 (0.75)	0.78 (0.78)	0.79 (0.79)	0.78 (0.78)
		1000	0.86 (0.86)	0.89 (0.88)	0.9 (0.9)	0.88 (0.88)
	500	5000	0.84 (0.84)	0.77 (0.84)	0.5* (0.5)	0.81 (0.82)
	1000	10000	0.78 (0.77)	0.5* (0.5)	0.5* (0.5)	0.79 (0.79)
cluster	10	10	0.66 (0.69)	0.66 (0.69)	0.72 (0.73)	0.73 (0.74)
		20	0.76 (0.83)	0.78 (0.81)	0.79 (0.84)	0.79 (0.82)
		100	0.92 (0.95)	0.92 (0.93)	0.93 (0.95)	0.92 (0.95)
	50	50	0.82 (0.83)	0.83 (0.83)	0.85 (0.86)	0.84 (0.85)
		100	0.87 (0.88)	0.88 (0.88)	0.89 (0.9)	0.88 (0.88)
		500	0.94 (0.94)	0.94 (0.94)	0.95 (0.96)	0.93 (0.93)
	100	100	0.85 (0.86)	0.87 (0.87)	0.88 (0.88)	0.87 (0.86)
		200	0.89 (0.89)	0.91 (0.91)	0.92 (0.92)	0.89 (0.89)
		1000	0.94 (0.94)	0.95 (0.96)	0.95 (0.95)	0.93 (0.93)
	500	5000	0.98 (0.98)	0.98 (0.98)	0.77* (0.77)	0.96 (0.96)
	1000	10000	0.98 (0.98)	0.99 (0.99)	0.52* (0.52)	0.97 (0.96)
scale-free	10	10	0.65 (0.66)	0.65 (0.66)	0.71 (0.71)	0.72 (0.73)
		20	0.75 (0.77)	0.75 (0.78)	0.77 (0.78)	0.77 (0.77)
		100	0.88 (0.9)	0.89 (0.9)	0.9 (0.92)	0.89 (0.89)
	50	50	0.8 (0.8)	0.8 (0.81)	0.84 (0.84)	0.84 (0.84)
		100	0.85 (0.86)	0.86 (0.87)	0.88 (0.89)	0.89 (0.89)
		500	0.93 (0.93)	0.94 (0.94)	0.95 (0.95)	0.94 (0.94)
	100	100	0.83 (0.82)	0.85 (0.87)	0.88 (0.87)	0.89 (0.89)
		200	0.87 (0.86)	0.89 (0.89)	0.91 (0.92)	0.91 (0.91)
		1000	0.93 (0.93)	0.95 (0.95)	0.95 (0.96)	0.95 (0.95)

Table 2: *AUC mean (median) over several replications. Number of replications differs across cases (see excel file "simulations-overview") . Entries with a * indicate that the AUC had not converged after 5 days.*

graph	p	n	MPL-RJ	MPL-BD	BD	SS
random	10	10	0.42 (0.38)	0.46 (0.39)	0.31 (0.29)	0.23 (0.21)
		20	0.39 (0.36)	0.39 (0.36)	0.4 (0.38)	0.32 (0.32)
		100	0.62 (0.61)	0.62 (0.61)	0.61 (0.61)	0.48 (0.43)
	50	50	0.28 (0.28)	0.28 (0.28)	0.36 (0.36)	0.31 (0.31)
		100	0.38 (0.38)	0.38 (0.38)	0.46 (0.46)	0.39 (0.39)
		500	0.64 (0.64)	0.64 (0.64)	0.7 (0.7)	0.55 (0.54)
	100	100	0.26 (0.26)	0.26 (0.26)	0.37 (0.36)	0.33 (0.33)
		200	0.37 (0.37)	0.37 (0.37)	0.47 (0.46)	0.42 (0.41)
		1000	0.62 (0.62)	0.63 (0.63)	0.71 (0.7)	0.58 (0.56)
	500	5000	0.57 (0.57)	0.47 (0.5)	0.02* (0.02)	0.58 (0.59)
	1000	10000	0.46 (0.44)	0.16* (0.16)	0* (0)	0.53 (0.54)
cluster	10	10	0.48 (0.45)	0.5 (0.46)	0.35 (0.36)	0.26 (0.25)
		20	0.47 (0.47)	0.47 (0.47)	0.44 (0.45)	0.37 (0.34)
		100	0.68 (0.68)	0.68 (0.68)	0.67 (0.67)	0.53 (0.5)
	50	50	0.47 (0.46)	0.47 (0.46)	0.5 (0.5)	0.44 (0.43)
		100	0.58 (0.57)	0.58 (0.58)	0.61 (0.61)	0.51 (0.5)
		500	0.77 (0.77)	0.78 (0.77)	0.79 (0.8)	0.59 (0.59)
	100	100	0.53 (0.53)	0.53 (0.54)	0.57 (0.58)	0.51 (0.52)
		200	0.63 (0.63)	0.64 (0.63)	0.67 (0.67)	0.56 (0.57)
		1000	0.8 (0.81)	0.81 (0.82)	0.83 (0.83)	0.6 (0.62)
	500	5000	0.86 (0.86)	0.9 (0.9)	0.29* (0.28)	0.67 (0.66)
	1000	10000	0.79 (0.79)	0.92 (0.92)	0.02* (0.02)	0.69 (0.68)
scale-free	10	10	0.49 (0.43)	0.52 (0.43)	0.34 (0.32)	0.26 (0.24)
		20	0.39 (0.36)	0.39 (0.36)	0.4 (0.38)	0.33 (0.31)
		100	0.66 (0.67)	0.66 (0.67)	0.66 (0.67)	0.5 (0.46)
	50	50	0.45 (0.45)	0.46 (0.45)	0.5 (0.49)	0.43 (0.42)
		100	0.57 (0.57)	0.58 (0.58)	0.6 (0.6)	0.49 (0.48)
		500	0.79 (0.78)	0.79 (0.79)	0.81 (0.81)	0.56 (0.56)
	100	100	0.53 (0.55)	0.54 (0.56)	0.59 (0.61)	0.51 (0.53)
		200	0.65 (0.65)	0.66 (0.66)	0.69 (0.69)	0.55 (0.56)
		1000	0.82 (0.83)	0.83 (0.84)	0.85 (0.86)	0.58 (0.61)

Table 3: *p+ mean (median) over several replications. Number of replications differs across cases (see excel file "simulations-overview") . Entries with a * indicate that the AUC had not converged after 5 days.*

graph	p	n	MPL-RJ	MPL-BD	BD	SS
random	10	10	0.29 (0.18)	0.33 (0.18)	0.15 (0.15)	0.12 (0.12)
		20	0.09 (0.09)	0.09 (0.09)	0.12 (0.12)	0.1 (0.09)
		100	0.03 (0.03)	0.03 (0.03)	0.06 (0.06)	0.05 (0.04)
	50	50	0.03 (0.03)	0.03 (0.03)	0.11 (0.11)	0.11 (0.11)
		100	0.02 (0.02)	0.02 (0.02)	0.09 (0.08)	0.09 (0.09)
		500	0.01 (0.01)	0.01 (0.01)	0.05 (0.04)	0.06 (0.05)
	100	100	0.02 (0.02)	0.02 (0.02)	0.11 (0.1)	0.11 (0.11)
		200	0.01 (0.01)	0.01 (0.01)	0.08 (0.08)	0.1 (0.09)
		1000	0.01 (0.01)	0 (0)	0.04 (0.04)	0.06 (0.05)
	500	5000	0.02 (0.02)	0.12 (0.05)	0.02* (0.02)	0.18 (0.18)
	1000	10000	0.02 (0.02)	0.16* (0.16)	0* (0)	0.19 (0.19)
cluster	10	10	0.31 (0.16)	0.33 (0.15)	0.14 (0.13)	0.11 (0.11)
		20	0.08 (0.08)	0.08 (0.07)	0.1 (0.1)	0.09 (0.08)
		100	0.03 (0.02)	0.03 (0.02)	0.04 (0.04)	0.04 (0.04)
	50	50	0.02 (0.02)	0.02 (0.03)	0.07 (0.07)	0.07 (0.07)
		100	0.01 (0.01)	0.01 (0.01)	0.05 (0.05)	0.05 (0.05)
		500	0 (0)	0 (0)	0.02 (0.02)	0.03 (0.03)
	100	100	0.01 (0.01)	0.01 (0.01)	0.05 (0.05)	0.06 (0.06)
		200	0.01 (0.01)	0.01 (0.01)	0.03 (0.03)	0.05 (0.05)
		1000	0 (0)	0 (0)	0.02 (0.02)	0.03 (0.03)
	500	5000	0 (0)	0 (0)	0.01* (0.01)	0.03 (0.03)
	1000	10000	0 (0)	0 (0)	0* (0)	0.03 (0.03)
scale-free	10	10	0.33 (0.22)	0.37 (0.22)	0.16 (0.16)	0.13 (0.12)
		20	0.08 (0.07)	0.08 (0.07)	0.12 (0.11)	0.1 (0.09)
		100	0.02 (0.02)	0.02 (0.02)	0.06 (0.05)	0.06 (0.05)
	50	50	0.03 (0.03)	0.03 (0.03)	0.07 (0.07)	0.07 (0.07)
		100	0.02 (0.02)	0.02 (0.02)	0.05 (0.05)	0.05 (0.05)
		500	0.01 (0.01)	0.01 (0.01)	0.02 (0.02)	0.03 (0.03)
	100	100	0.01 (0.01)	0.01 (0.01)	0.05 (0.05)	0.06 (0.06)
		200	0.01 (0.01)	0.01 (0.01)	0.03 (0.03)	0.04 (0.04)
		1000	0 (0)	0 (0)	0.02 (0.02)	0.03 (0.03)

Table 4: *p- mean (median) over several replications. Number of replications differs across cases (see excel file "simulations-overview"). Entries with a * indicate that the AUC had not converged after 5 days.*

graph	p	n	MPL-RJ	MPL-BD	BD	SS
random	10	10	0.03 (0)	0.11 (0.01)	0.58 (0.1)	0.22 (0.05)
		20	0.01 (0)	0.05 (0.01)	0.49 (0.1)	0.2 (0.04)
		100	0.03 (0)	0.01 (0)	0.14 (0.02)	0.19 (0.02)
	50	50	3.17 (2.28)	4.59 (2.43)	234.25 (169.84)	0.98 (0.71)
		100	2.78 (2.73)	7.45 (4)	163.34 (131.15)	1.02 (0.71)
		500	1.88 (1.53)	13.65 (9.27)	117.79 (106.31)	1.5 (1.19)
	100	100	6.27 (5.64)	14.97 (14.91)	5608.53 (5600.37)	7.17 (8.25)
		200	5.85 (5.96)	24.03 (21.87)	5715.78 (5819.88)	9.37 (8.26)
		1000	4.57 (3.6)	39.97 (37.15)	3810.87 (3703.09)	12.55 (13.36)
	500	5000	415.93 (412.85)	13919.92 (10944.59)	$>$ 5 days	26823.1 (26850.71)
	1000	10000	228873.8 (235717.9)	$>$ 5 days	$>$ 5 days	292573.2 (294132.5)
cluster	10	10	0.02 (0)	0.09 (0.01)	0.55 (0.07)	0.17 (0.03)
		20	0.01 (0)	0.05 (0.01)	0.44 (0.06)	0.05 (0.02)
		100	0.01 (0)	0.01 (0)	0.22 (0.02)	0.07 (0.02)
	50	50	2.68 (1.71)	4.76 (1.7)	214.11 (116.75)	1.11 (0.7)
		100	1.81 (1.33)	3 (1.11)	157.55 (75.3)	1.1 (0.59)
		500	0.65 (0.35)	3.24 (0.52)	43.55 (19.76)	2.51 (0.98)
	100	100	3.33 (2.2)	5.88 (3.93)	3578.04 (2881.66)	6.44 (3.3)
		200	2.66 (3.08)	6.82 (6.7)	2180.34 (1629.79)	6.81 (3.33)
		1000	0.71 (0.65)	5.62 (4.75)	424.29 (296.87)	41.87 (6.66)
	500	5000	119.09 (119.18)	27.05 (26.44)	$>$ 5 days	9456.11 (7515.02)
	1000	10000	12917.69 (12500.31)	131.04 (126.85)	$>$ 5 days	84294.99 (82907.87)
scale-free	10	10	0.06 (0)	0.14 (0.01)	1.01 (0.11)	0.3 (0.08)
		20	0.02 (0.01)	0.06 (0.01)	0.27 (0.08)	0.1 (0.05)
		100	0.02 (0)	0.02 (0)	0.13 (0.02)	0.07 (0.02)
	50	50	2.47 (1.24)	3.35 (0.98)	128.92 (53.73)	1.39 (0.69)
		100	1.81 (0.99)	4.46 (1.2)	86.14 (32.17)	2.64 (1.18)
		500	1.15 (0.54)	4.37 (1.04)	63.5 (33.8)	4.22 (1.19)
	100	100	5.6 (5.42)	12.06 (10.93)	2583.78 (3260.38)	8.84 (5.01)
		200	2.97 (2.59)	27.16 (8.93)	2481.84 (1769.74)	9.93 (6.63)
		1000	1.22 (1.13)	21.37 (1.73)	636.93 (578.35)	70.66 (19.81)

Table 5: *Time in seconds until AUC convergence: mean (median) over several replications. Number of replications differs across cases. (see excel file "simulations-overview").*

approximate the marginal likelihood $P(\boldsymbol{X}|G)$. This allows us to obtain the posterior distribution $P(G|\boldsymbol{X})$ instead of $P(G, K|\boldsymbol{X})$, making it possible to search over the graph space rather than the joint space of graphs and precision matrices and thus save computional time.

The newly proposed MPL-based methods behave as follows. For relatively small-sized problems ($p = 10, 50, 100$), the new methods generate a similar or slightly worse AUC than state-of-the-art BSL methods (BD and SS). For high dimensions, i.e. $p = 500, 1000$, the MPL methods can solve problems with excellent accuracy within minutes. For the cluster graph with settings $p = 500$ and $n = 5000$, the MPL-based methods outputted an excellent AUC within 2 minutes while the SS and BD methods needed 2.6 hours and more than 5 days, respectively. For the cluster graph with $p = 1000$ and $n = 10000$, the MPL-BD method had an excellent AUC within 2 minutes, the MPL-RJ method within 3.6 hours, the SS method within 1 day and the DB method after more 5 days.

There are some promising lines of future research that could improve the performance of the newly proposed methods even further. One of these involves the calculation in parallel of multiple birth and death rates in order to reduce the computational time needed by the MPL-BD method. And it might also pay to investigate whether computational time can be reduced if the methods use a suitable burn, another prior $P(G)$ or multiple jumps during a single iteration.

Software

The BD-MPL and RJ-MPL methods are implemented in the R package BDgraph, which is freely available from the Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/packages=BDgraph`.

References

Albert, R. and A.-L. Barabási (2002). Statistical mechanics of complex networks. *Rev. Mod. Phys. 74*(1), 47–97.

Cappé, O., C. Robert, and T. Rydén (2003). Reversible jump, birth-and-death and more general continuous time markov chain monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 65*(3), 679–700.

Cheng, Y. and A. Lenkoski (2012). Hierarchical Gaussian graphical models: Beyond reversible jump. *Electronic Journal of Statistics 6*, 2309–2331.

Consonni, G. and L. L. Rocca (2012). Objective bayes factors for gaussian directed acyclic graphical models. *Scandinavian Journal of Statistics 39*(4), 743–756.

Dobra, A., A. Lenkoski, and A. Rodriguez (2011). Bayesian inference for general gaussian graphical models with application to multivariate lattice data. *Journal of the American Statistical Association 106*(496), 1418–1433.

Dobra, A. and R. Mohammadi (2018). Loglinear model selection and human mobility. *The Annals of Applied Statistics 12*(2), 815–845.

Friedman, J., T. Hastie, and R. Tibshirani (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics 9*(3), 432–441.

Green, P. (1995). Reversible jump markov chain Monte carlo computation and Bayesian model determination. *Biometrika 82*(4), 711–732.

Hanley, J. and B. Mcneil (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology 143*(1), 29–36.

Hinne, M., A. Lenkoski, T. Heskes, and M. van Gerven (2014). Efficient sampling of gaussian graphical models using conditional bayes factors. *Stat 3*(1), 326–336.

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

Lenkoski, A. (2013). A direct sampler for G-Wishart variates. *Stat 2*(1), 119–128.

Lenkoski, A. and A. Dobra (2011). Computational aspects related to inference in Gaussian graphical models with the G-Wishart prior. *Journal of Computational and Graphical Statistics 20*, 140–157.

Leppä-Aho, J., J. , T. Roos, and J. Corander (2017). Learning gaussian graphical models with fractional marginal pseudo-likelihood. *International Journal of Approximate Reasoning 83*, 21–42.

Letac, G. and H. Massam (2007). Wishart distributions for decomposable graphs. *The Annals of Statistics 35*(3), 1278–1323.

Mohammadi, A. and E. Wit (2015). Bayesian structure learning in sparse Gaussian graphical models. *Bayesian Analysis 10*(1), 109–138.

Mohammadi, R. ssgraph: Bayesian graph structure learning using spike-and-slab priors. *R package version 1.15.*.

Mohammadi, R., H. Massam, and G. Letac (2021). Accelerating bayesian structure learning in sparse gaussian graphical models. *Journal of the American Statistical Association*, 1–14.

Mohammadi, R., E. Wit, and A. Dobra (2021). *BDgraph: Bayesian Structure Learning in Graphical Models using Birth-Death MCMC*. R package version 2.64.

Pensar, J., H. Nyman, J. Niiranen, and J. Corander (2017). Marginal pseudo-likelihood learning of discrete markov network structures. *Bayesian Analysis 12*(4), 1195–1215.

Preston, C. (1975). Spatial birth-and-death processes. *Bull. Inst. Inst. Stat. 46*(2), 371–391.

Roverato, A. (2008). Hyper inverse wishart distribution for non decomposable graphs and its application to bayesian inference for gaussian graphical models. *Scandinavian Journal of Statistics 29*(3), 432–441.

Rue, H. and L. Held (2005). *Gaussian Markov Random Fields: Theory and Applications*. london: Chapman and Hall-CRC Press.

Wang, H. (2012). The Bayesian graphical Lasso and efficient posterior computation. *Bayesian Analysis 7*, 771–790.

Wang, H. (2015). Scaling it up: Stochastic search structure learning in graphical models. *Bayesian Analysis 10*, 351–377.

Wang, H. and S. Li (2012). Efficient Gaussian graphical model determination under G-Wishart prior distributions. *Electronic Journal of Statistics 6*, 168–198.