12-2022

# Segment-wise time-varying dynamic Bayesian network with graph regularization

Xing YANG

Chen ZHANG

Baihua ZHENG
*Singapore Management University*, bhzheng@smu.edu.sg

# Segment-Wise Time-Varying Dynamic Bayesian Network with Graph Regularization

XING YANG and CHEN ZHANG, Tsinghua University, China
BAIHUA ZHENG, Singapore Management University, Singapore

Time-varying dynamic Bayesian network (TVDBN) is essential for describing time-evolving directed conditional dependence structures in complex multivariate systems. In this article, we construct a TVDBN model, together with a score-based method for its structure learning. The model adopts a vector autoregressive (VAR) model to describe inter-slice and intra-slice relations between variables. By allowing VAR parameters to change segment-wisely over time, the time-varying dynamics of the network structure can be described. Furthermore, considering some external information can provide additional similarity information of variables. Graph Laplacian is further imposed to regularize similar nodes to have similar network structures. The regularized maximum a posterior estimation in the Bayesian inference framework is used as a score function for TVDBN structure evaluation, and the alternating direction method of multipliers (ADMM) with L-BFGS-B algorithm is used for optimal structure learning. Thorough simulation studies and a real case study are carried out to verify our proposed method's efficacy and efficiency.

CCS Concepts: • **Mathematics of computing → Bayesian networks**; • **Computing methodologies → Bayesian network models;**

Additional Key Words and Phrases: Time-varying dynamic Bayesian network, structure learning, segment-wise change, acyclic property, graph Laplacian, ADMM, directed acyclic graph

**ACM Reference format:**
Xing Yang, Chen Zhang, and Baihua Zheng. 2022. Segment-Wise Time-Varying Dynamic Bayesian Network with Graph Regularization. *ACM Trans. Knowl. Discov. Data.* 16, 6, Article 113 (September 2022), 23 pages.
https://doi.org/10.1145/3522589



## 1 INTRODUCTION

**Directed acyclic graphs** (**DAGs**), also known as **Bayesian networks** (**BNs**), get rapid development in the past few decades. BN has shown promise for explainable models and causal insights about an underlying process with multiple variables. Moreover, by denoting different variables as nodes, and their interactions as directed edges, BN can clearly describe the directed conditional dependence structure between different variables. BNs have been generally applied in various areas, such as operation and risk analysis [23, 36, 49], biology [14, 35], economics and finance [43, 47]. Pioneer works focus on **static BN** (**SBN**) modeling, assuming the values of variables will not change over time. Later, SBNs are extended to **dynamic Bayesian networks** (**DBNs**), which consider

Authors' addresses: X. Yang and C. Zhang, Tsinghua University, Beijing, 100084, China; emails: x-yang16@mails.tsinghua.edu.cn, zhangchen01@mail.tsinghua.edu.cn; B. Zheng, Singapore Management University, Singapore, 178902, Singapore; email: bhzheng@smu.edu.sg.


1556-4681/2022/09-ART113
https://doi.org/10.1145/3522589

the variables' values can evolve and their directed conditional dependence relationships exist both within and across different time points [13]. Variable interactions within the same time point is called intra-slice directed conditional dependence relationships. Interactions across different time points with a time "lag" are called inter-slice directed conditional dependence relationships. Therefore, DBN provides a flexible mechanism for identifying directed conditional dependencies in multivariate time-series data, and DBN has also been generally applied to various areas. For example, they derive both intra-slice and inter-slice causal relationships among different elements such as genes, proteins, metabolites, and neurons in computational biology based upon multi-dimensional temporal data. Kim et al. [27] proposed to adopt DBN to analyze the metabolic pathway gene expression data, which could get better performance (detecting more correct relationships and reducing the number of false positives) and could estimate gene regulations more effectively compared with other network estimation methods. Besides, Vinh et al. [48] conducted experiments on large-scale cyanobacterial genetic networks containing 733 genes on 34 time points, which successfully reconstructed biologically plausible network structures and managed to produce more enriched hubs. However, an essential assumption of traditional DBNs is that stationary processes generate data, i.e., the conditional probabilities of different variables do not change with time. Hence DBN is also called **time-homogeneous DBN** (**THDBN**). Unfortunately, this assumption yet may not be valid in many critical settings.

In reality, the network structure may evolve over time [41]. For example, DBN has been used in reconstructing transcriptional regulatory networks for gene expression data in biology. These regulatory networks evolve during the development of genes, with specific directed conditional dependencies between gene products being created as the organism develops and others being destroyed. DBN has been used to model traffic flow patterns of different roads in transportation. The dynamic utilization of those roads changes daily during the morning rush, lunch, evening rush, and weekends. Though in these cases, DBN is still applicable by learning several DBNs for each subset time interval of the data, with the assumption that in each subset, the data is stationary, yet in many cases, subsets are not easy to be well defined and depend on researcher's bias, which may lead to inaccurate and misleading DBN results.

As such, further integrating temporal variation on the directed conditional dependence structures of THDBN and relaxing its data stationarity assumption would be more reasonable. This is the idea of **time-varying DBN** (**TVDBN**) [45] or **non-homogeneous DBN** (**NHDBN**) [41]. TVDBN shows particularly potential application for non-stationary time-series analysis, such as biological networks [9], human brain connectivity [53], and transportation networks [5]. A big advantage of TVDBN is its flexibility to describe directed conditional dependence dynamics of multivariate data streams. However, this flexibility increases modeling difficulty. On the one hand, designing the evolving mechanism of TVDBN determines the model's generality and applicability in real cases. The unsuitable design may fail to catch the actual evolving mechanism of the system or lead to underfitting or overfitting. On the other hand, the designed evolving mechanism will also influence its structure learning process, which aims at estimating the network structure, i.e., whether there is a directed edge between two variables and what the edge weights are.

Structure learning for TVDBN is incredibly challenging compared with SBN and THDBN. First, the learning algorithm should correspond to the time-varying mechanism of structure to ensure that the learned structure is consistent with the temporal dynamics. Second, since the network structure is time-varying, the number of parameters to be inferred is enormous, and meanwhile, the acyclic property should be guaranteed, adding the difficulty of parameter learning. Last, there can be some external knowledge about the similarity of different nodes, which may provide certain information about the structure. Take the biology example above, Dondelinger et al. [9] concluded gene relationships from a biological perspective. These static relationships between genes and also

implies that there exists a similarity between genes. Then how to incorporate this knowledge into structure learning to increase credibility should be carefully considered.

In this article, we propose a TVDBN model and a score-based structure learning algorithm. The model is built upon the **vector autoregressive** (**VAR**) model to describe the temporal dynamics of different variables. By allowing the parameters of VAR models to change over time, we can describe its time-varying dynamics. Considering that network structure may only change at certain discrete time points but keep consistent between these neighboring changepoints, we further impose segment-wise assumptions on the time-varying VAR structure. Then we propose a Bayesian learning framework for structure learning. Considering the external information can provide additional similarity information of variables. In the learning algorithm, we further regularize similar nodes to have similar network structures with the graph Laplacian method. Finally, the regularized maximum a posterior estimation of the network structure is solved by **alternating direction method of multipliers** (**ADMM**) with L-BFGS-B algorithm [56]. Thorough simulation studies and comparisons with baselines are carried out to verify the efficacy and efficiency of our proposed method.

The main contributions of this article are the following:

— We propose a TVDBN model based on the segmental VAR model. The model can capture both intra-slice and inter-slice interactions of variables and in the meanwhile guarantee the acyclic property of the network. Furthermore, the model particularly aims at describing segmental-varying network structure by imposing sparsity prior to the number of changes for each time point.
— The model also incorporates external similarity information of variables. By formulating an additional similarity matrix of variables based on external information, graph Laplacian regularization is imposed to ensure similar nodes have similar interactions with others in the whole network.
— We develop a Bayesian estimation framework for structure learning. The regularized maximum a posterior is regarded as a score function, with the acyclic constraint formulated in an algebraic characterization. Then a search algorithm combining ADMM with L-BFGS-B is developed to efficiently get the maximum solution of the score function.

The remainder of this article is organized as follows. Section 2 reviews related models and algorithms for time-varying networks, including score-based structure learning methods for different BNs. Section 3 introduces our proposed TVDBN model in detail. Section 4 introduces the structure learning procedure. Section 5 uses some numerical studies and a case study to demonstrate the efficiency and superiority of the proposed model compared with some other state-of-the-art methods. Finally, Section 6 concludes this article with remarks and prospects possible future directions.

## 2 RELATED WORK

We firstly review TVDBN models together with some representative undirected time-varying network models in Section 2.1. Then we review score-based algorithms that are most widely used for general BN structure learning problems in Section 2.2.

### 2.1 Time-Varying Dynamic Network

The concept of TVDBN [45] is almost equivalent to the concept of NHDBN [41]. Both aim to model the time-varying directed conditional dependence structure of DBN (we hereafter use TVDBN for unification). Since this concept has been proposed, increasing works have been concentrated on developing various TVDBN models based on statistical and machine learning algorithms

considering different time-varying patterns in DBN models and the corresponding structure learning algorithms.

The current TVDBN models can be mainly categorized into two groups: (i) models that only allow the network parameters to vary with time [19, 20], and (ii) models that enable the network structure to change over time as well [24, 41]. We focus on the second group since it offers more model flexibility than the first. Two types of changes are commonly considered in the literature. The first is segment-wise change. In particular, Robinson and Hartemink [41] considered node variables following multinomial distributions. The distribution parameters are allowed to change over time for dynamics modeling. Sparse regularization on the change-points is imposed by adding an exponential distribution as prior on the total number of parameter changes to avoid overfitting. Then a Bayesian **Markov chain Monte Carlo** (**MCMC**) estimation framework is used for structure learning. Dondelinger et al. [9] also considered modeling the number of change-points as a Poisson distribution. These two models assume the lag of time is fixed. Later Jia and Huan [25] considered releasing this limitation by proposing a **reverse jump MCMC** (**RJMCMC**) estimation algorithm to learn the most suitable lags adaptively. However, their regularization formulas still render much more change-points than existed and lead to overfitting. Also, the above models only focus on binary networks whose edges can only be zero or one, yet cannot model TVDBN with weighted edges. The second type is smooth change. In particular, Song et al. [45] proposed a linear VAR model to describe the interrelationship between different nodes. The VAR model parameters are assumed to change smoothly following a Gaussian **radial basis function** (**RBF**) kernel function. To further guarantee causal Markov assumption, Chu et al. [5] extended the model by using asymmetric kernel for causal boundary correction. However, the above TVDBN models evade dealing with the acyclicity of BNs by assuming the causal relationship is time-lagged, i.e., each node can only influence other nodes or itself in the future but cannot influence other nodes concurrently. In this way, the structure learning is much simplified, with no need to be concerned about acyclic property. However, this assumption also limits the models' applicability in solving many real-world problems where both intra-slice(concurrent) and inter-slice(lagged) causal relationships exist.

Besides TVDBN, time-varying undirected network models are also well studied in the literature. Consider these models may shed light upon our modeling of TVDBN. We review some state-of-art works as well. Specifically, Kolar and Xing [29] extended graphical LASSO to dynamic cases and regularized the precision matrix's dynamics overtime via kernel smoothing. Hallac et al. [21] discussed different types of **time-varying graphical LASSO** (**TVGL**) by adding various penalties on the change of precision matrix. Tomasi et al. [46] further developed TVGL with latent variables considering the influence of hidden factors, which provided several patterns for regularization terms for describing dynamic mechanisms. Besides, Hallac et al. [21] and Tomasi et al. [46] targeted an undirected network, where the relationships are not directed conditional dependence relationships. These models may enlighten potential extensions to TVDBN by incorporating edge direction and acyclic properties. However, all the above models only focus on intra-slice correlation modeling but do not address inter-slice correlations. Furthermore, none of them take extra similarity information of nodes into account.

### 2.2 Bayesian Network Structure Learning

There are mainly two types of approaches for general BN structure learning: constraint-based and score-based methods. Constraint-based methods employ statistical hypothesis tests to identify directed conditional independence relationships from the data and construct a Bayesian network structure that best fits those directed conditional independence relationships. However, constraint-based methods are built upon two basic assumptions [3]: the directed conditional independence

model should be faithful to a DAG, and the directed conditional independence tests performed on data should accurately reflect the independence model, which is generally difficult to be satisfied in reality. As a result, constraint-based methods are pretty unstable and prone to cascade effects where a single early error in the learning process can result in a very different DAG structure. For score-based methods, a scoring function is used to evaluate the goodness of a network structure for training data fitting, and a searching procedure for the best structure with the highest score is formulated as a combinatorial optimization problem. In this article, we use the score-based method for structure learning, and hence mainly review score-based BN learning literature as below. Furthermore, based on the searching mechanism, score-based methods can be further divided into two categories: exact(global) and approximate(local) search algorithms.

Approximate algorithms are based on stochastic local research, i.e., heuristic methods, such as genetic algorithm [30], particle swarm [17], ant colony [7], and so on. They travel through the solution space using non-deterministic transition among structures and choose the solution with the highest score. However, these stochastic moves only guarantee a certain probability to reach the optimum. Consequently, the solution's accuracy cannot always be ensured. Yet these methods require a short searching time and can be extended to solve large-scale problems. In contrast, as the name indicates, exact methods can guarantee the global optimum. However, they require intensive searching time and are a bit unfriendly to large-scale networks with many nodes. Pioneering exact algorithms [28, 44] are based on **dynamic programming** (**DP**). Their basic idea is the segment-wise search method, where nodes are sequentially added into sub-networks with acyclic property guaranteed until obtaining a global network structure. Later many methods [32] were further proposed to reduce time and memory costs. The above methods are for SBN structure learning. Recently, Dang et al. [8] also extended this principle to high-order DBNs. Besides DP, integer linear programming (ILP, Bartlett and Cussens [2]) and linear program (LP, Cussens et al. [6]) are also widely used, by encoding the acyclic constraint as a linear inequality. They can handle relatively larger networks than DPs. However, the practical usage of these algorithms is still very restricted for large-scale problems.

Some other exact algorithms for BN structure learning have been developed to reduce computation costs. For example, by casting the structure learning as a shortest path finding problem, Yuan et al. [52] proposed the A$^\star$ search algorithm. Built upon it, Fan et al. [11] further proposed to calculate the potentially optimal parent sets of each node, to prune large portions of the search space, and consequently improved the searching time and space. Zheng et al. [55] represented acyclic constraints by an algebraic characterization, i.e., a nonlinear constraint of the score function, and used limited-memory quasi-Newton algorithm [56] for structure learning of SBN. It leads to a significant reduction in computation complexity to cubic in the number of nodes. Later Pamfil et al. [38] extended it to THDBN with lag. Our article also adopts this algebraic characterization algorithm for acyclic constraint formulation in constructing the score function.

## 3 PROBLEM DEFINITION AND MODEL

We firstly introduce the formulation of THDBN by the VAR model, as the prior knowledge in Section 3.1. Then we extend it to TVDBN in Section 3.2. Under the Bayesian framework with maximum a posterior estimation, a score function for structure learning is constructed in Section 3.3, where external information is also incorporated into the score function via graph Laplacian constraint.

### 3.1 Dynamic Bayesian Network

A BN represents a joint probability distribution over a collection of variables $\mathscr{X} = \{X_1, \ldots, X_n\}$, by formulating it as a set of directed conditional (in)dependence relationships between variables. Here, variables are abstracted from stationary processes. A BN is determined by its graph structure

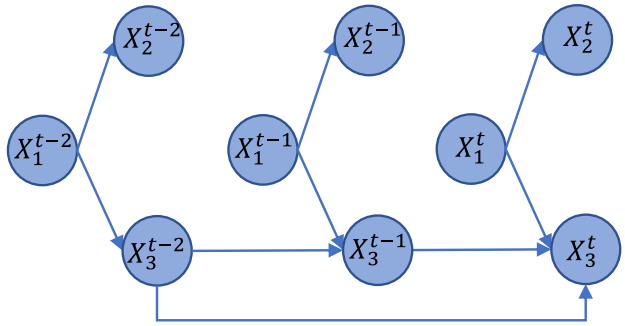



Fig. 1. THDBN with $n = 3$ nodes and lag $m = 2$.

$\mathscr{G}$ and a set of directed conditional distribution functions with a parameter set $\mathscr{B}$. $\mathscr{G}$ is a directed acyclic graph with nodes corresponding to random variables in $\mathscr{X}$. An edge in $\mathscr{G}$ directed from node $i$ to $j$ encodes $X_i$'s directed conditional dependence on $X_j$, and hence $X_i$ is called a parent of $X_j$. A variable $X_i$ is independent of its non-descendants given all its parents set $\pi_i$. Therefore, the joint probability distribution over $\mathscr{X}$ can be decomposed by the chain rule:

$$p(\mathscr{X}) = \prod_{i=1}^{n} p(X_i|\pi_i). \tag{1}$$

In this article we consider linear dependence relationship for a BN model: $X_i = \sum_{j\in\pi_i} b_{ij} \cdot X_j + \varepsilon_i$, where $\varepsilon_i$ is independent noise with mean $\mathrm{E}[\varepsilon_i] = 0$ and homoscedastic variance $\mathrm{Var}[\varepsilon_i] = \sigma_0^2$ for $i = 1, \ldots, n$.

A DBN is the extension of a BN to model temporal processes. In DBN, a set of random processes are represented by $\mathscr{X}$, and $X_i(t)$ is the random variable of process $X_i$ at discrete time point $t$. The network structure $\mathscr{G}$ now defines the intra-slice and inter-slice dependency among these variables over a period of time, i.e., $\pi_i \subset \{\mathscr{X}(t), \mathscr{X}(t-1), \ldots, \mathscr{X}(t-m)\}$ where $m$ is the largest lag considered. All the conditional distributions are assumed to be stationary. For linear conditions, we have

$$X_i(t) = \sum_{k=0}^{m} \sum_{j\in\pi_i^k} b_{ij}^k \cdot X_j(t-k) + \varepsilon_i(t), t = 1, \ldots, L, \tag{2}$$

where $\pi_i^k$ indicates the $k$-lag parent set of variable $i$, with its total parent set as $\pi_i = \cup_{k=0}^{m} \pi_i^k$. For variable $j \in \pi_i^k$, $b_{ij}^k$ captures the directional relationship of variable $j$ to variable $i$ with $lag = k$, i.e., the influence coefficient of variable $j$ at time $t-k$ on variable $i$ at time $t$. $\varepsilon_i(t)$ is independent Gaussian noise with $\mathrm{E}[\varepsilon_i] = 0$ and homoscedastic variance $\mathrm{Var}[\varepsilon_i] = \sigma_0^2$ for $i = 1, \ldots, n$ and for all the time points among variables. It is worth mentioning that for every $t$ belongs to the discrete-time set $\{1, \ldots, L\}$, $b_{ij}^k$ stays the same due to the time homogeneousness assumption of THDBN.

An example of THDBN based on VAR with lage order $m = 2$ and node number $n = 3$ is shown in Figure 1, where each node is a variable and directed edges represent directed conditional dependencies.

### 3.2 Time Varying Network Representation

The time-homogeneous assumption is not always valid in real-time. We introduce a TVDBN model whose network structure and parameters can vary over time. Consider extending Equation (2) to time-varying cases. A straightforward way is to allow $b_{ij}^k$ to evolve as a variable of $t$, i.e., $b_{ij}^k(t)$ can

be different for different $t$. Then the data evolving mechanism can be modeled as

$$X_i(t) = \sum_{k=0}^{m} \sum_{j \in \pi_i^k(t)} b_{ij}^k(t) \cdot X_j(t-k) + \varepsilon_i(t), t = 1, \ldots, L. \tag{3}$$

Denote the time-varying directed conditional dependence relationship as $\mathscr{B}_t = \{b_{ij}^k(t), k = 0, \ldots, m, i, j = 1, \ldots, n\}$. In reality, we only have the node data $\mathscr{X}_t = \{X_i(t), i = 1, \ldots, n\}$, and $\mathscr{B}_t$ are unknown. Our goal is to construct a learning method for $\mathscr{B}_t$ as follows.

First, the above model gives too much model flexibility and maybe overfitting. The extra-large variable space also increases the difficulty of structure learning. As such, we propose to add additional prior structures on $\mathscr{B}(t)$ to regularize its change.

Assumption 1. *Considering that the directed conditional dependence relationships would be very sparse in a large-scale network with many nodes. Hence we set the prior distribution of $b_{ij}^k(t)$ as a Laplace distribution, i.e.,*

$$p_{01}\left(b_{ij}^k(t)\right) = \frac{\lambda_1}{4} \exp\left(-\frac{\lambda_1}{2}\left|b_{ij}^k(t)\right|\right), \quad \forall j \in \pi_i^k(t), k = 1, \ldots, m, i = 1, \ldots, n. \tag{4}$$

It is more prone to end up with many zero-valued coefficients, some moderate-sized coefficients, and some large-sized coefficients using such prior. Jing et al. [26] noticed that Lasso regression under Bayesian setting is equivalent to using Laplace prior.

Assumption 2. *Consider that local structure change may only occur at discrete time points and yet keep the same for a time duration between two neighbor changepoints. $b_{ij}^k(t) - b_{ij}^k(t-1)$ would be zero for most $t \in 1, \ldots, L$. Hence we further regularize the change $b_{ij}^k(t) - b_{ij}^k(t-1)$ by assuming it's every component follows a Laplace distribution, i.e.,*

$$p_{02}\left(b_{ij}^k(t) - b_{ij}^k(t-1)\right) = \frac{\lambda_2}{4} \exp\left(-\frac{\lambda_2}{2}\left|b_{ij}^k(t) - b_{ij}^k(t-1)\right|\right), \forall j \in \pi_i^k(t), k = 1, \ldots, m, \ i = 1, \ldots, n. \tag{5}$$

An example of this TVDBN with $m = 2$ and $n = 3$ is illustrated in Figure 2. Solid lines play the same role as edges in Figure 1, and yellow dashed lines are used to emphasize that this directed conditional dependence will disappear at the next time point of $t$ (we no longer see it in $t + 1$). Similarly, orange dashed lines are used to emphasize that this directed conditional dependence will appear at the next time point of $t$, and since then, we can always observe it until its vanishment at the next change-point.

### 3.3 Score Function

Based on the above modeling framework, we can develop the score function for structure learning using the posterior distribution of $\mathscr{B}_{1:L}$. In particular, the posterior distribution of $\mathscr{B}_{1:L}$ given data $\mathscr{X}_{1:L}$ can be expressed as

$$p(\mathscr{B}_{1:L} \mid \mathscr{X}_{1:L}) \propto p(\mathscr{X}_{1:L} \mid \mathscr{B}_{1:L}) p_0(\mathscr{B}_{1:L}), \tag{6}$$

where

$$p(\mathscr{X}_{1:L} \mid \mathscr{B}_{1:L}) = \prod_{t=1}^{L} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(X_i(t) - \sum_{k=0}^{m} \sum_{j \in \pi_i^k(t)} b_{ij}^k(t) \cdot X_j(t-k)\right)^2}{2\sigma^2}\right), \tag{7}$$

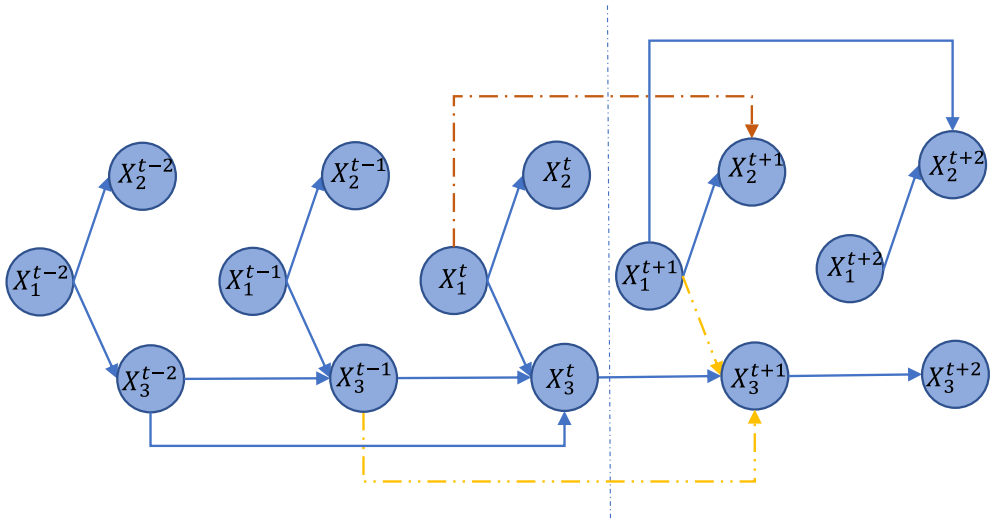



Fig. 2. TVDBN with $n = 3$ nodes and lag $m = 2$.

and

$$p_0(\mathscr{B}_{1:L}) = \prod_{t=1}^{L} p_{01}(\mathscr{B}_t) \prod_{t=2}^{L} p_{02}(\mathscr{B}(t) - \mathscr{B}(t-1)) \tag{8}$$
$$\propto \prod_{i=1}^{n} \exp\left(-\frac{\lambda_1}{2} \|\mathbf{b}_i(1)\|_1\right) \prod_{t=2}^{L} \prod_{i=1}^{n} \exp\left(-\frac{\lambda_2}{2} \|\mathbf{b}_i(t) - \mathbf{b}_i(t-1)\|_1\right) \exp\left(-\frac{\lambda_1}{2} \|\mathbf{b}_i(t)\|_1\right),$$

where $\mathbf{b}_i(t) = [b_{i1}^m(t), \ldots, b_{in}^m(t), \ldots, b_{i1}^1(t), \ldots, b_{in}^1(t), b_{i1}^0(t), \ldots, b_{in}^0(t)] \in \mathbb{R}^{(m+1)n}$ with $b_{ii}^0(t) = 0$, $t = 1, \ldots L$. The negative log-posterior can thus be calculated as

$$-\log p(\mathscr{B}_{1:L} \mid \mathscr{X}_{1:L}) \propto \sum_{t=1}^{L} \sum_{i=1}^{n} \frac{\left(X_i(t) - \sum_{k=0}^{m} \sum_{j \in \pi_i^k(t)} b_{ij}^k(t) \cdot X_j(t-k)\right)^2}{2\sigma^2} + \sum_{t=1}^{L} \sum_{i=1}^{n} \frac{\lambda_1}{2} \|\mathbf{b}_i(t)\|_1 \tag{9}$$
$$+ \sum_{t=2}^{L} \sum_{i=1}^{n} \frac{\lambda_2}{2} \|\mathbf{b}_i(t) - \mathbf{b}_i(t-1)\|_1 .$$

A larger log-posterior indicates a better structure. As such, we would like to minimize Equation (9) for structure pursuit.

Furthermore, in many real-case problems, we have some additional information on variables' similarities. Then we can use this prior knowledge of their similar structure for better structure learning. For example, if $X_i(t)$ and $X_j(t)$ are similar to each other, their topological positions, in the network would be similar, which indicates that $b_{ui}^k(t)$ and $b_{uj}^k(t)$ tend to have similar values for $k = 0, \ldots, m, u = 1, \ldots, n$. Of course, this prior similarity information can also evolve with time. Consequently, we can define $W_{ij}^t$ as the external similarity metric between $X_i(t)$ and variable $X_j(t)$ at time $t$, and a higher $W_{ij}^t$ indicates these two variables are more similar. Then we add another regularization term $\lambda_3(b_{ui}^k(t) - b_{uj}^k(t))^2 W_{i,j}^t$ in Equation (9). This regularization forces $|b_{ui}^k(t) - b_{uj}^k(t)|$ to be very small when $W_{ij}^t$ is large and vice versa. Formulating the similarity matrix as $\mathbf{W}^t$ whose

$(i, j)$ component is $W_{ij}^t$, actually this regularization is equivalent to $\lambda_3 tr(\mathbf{B}^k(t)\boldsymbol{L}(t)\mathbf{B}^k(t)')$. Here $\mathbf{b}_u^k(t) = [b_{u1}^k(t), \ldots, b_{un}^k(t)] \in \mathbb{R}^n$, $\mathbf{B}^k(t) = [\mathbf{b}_1^k(t)', \ldots, \mathbf{b}_n^k(t)'] \in \mathbb{R}^{n\times n}$. $\boldsymbol{L}(t) = \mathbf{D}^t - \mathbf{W}^t$, and $\mathbf{D}^t \in \mathbb{R}^{n\times n}$ with its $(i, i)$ component as $D_{i,i}^t = \sum_{j=1}^n W_{ij}^t$ and other components as 0. This penalization term is graph Laplacian [15] and we adopt it to integrate the additional information to further improve model accuracy. Besides, we define $\mathcal{B}(t) = [\mathbf{B}^m(t); \ldots; \mathbf{B}^0(t)] \in \mathbb{R}^{n(m+1)\times n}$. Then, the final score function for structure learning can be formulated as

$$\begin{aligned}\min_{\mathcal{B}(t), t=1,\ldots,L} &\sum_{t=m+1}^{L}\sum_{i=1}^{n}\left(X_i(t) - \sum_{j=1}^{n}\sum_{k=0}^{m} b_{ij}^k(t)X_j(t-k)\right)^2 + \lambda_1\sum_{t=1}^{L}\sum_{i=1}^{n}\|\mathbf{b}_i(t)\|_1 \\ &+ \lambda_2\sum_{t=2}^{L}\sum_{i=1}^{n}\|\mathbf{b}_i(t) - \mathbf{b}_i(t-1)\|_1 + \lambda_3\sum_{t=1}^{L}\sum_{k=0}^{m} tr\left(\mathbf{B}^k(t)\boldsymbol{L}(t)\mathbf{B}^k(t)'\right), \\ &\text{s.t. } \mathcal{B}(t) \text{ is acyclic.}\end{aligned} \tag{10}$$

Note that here we remove the notation $\pi_i^k(t)$, since its information can be fully expressed by the values of $\pi_i^k(t) = \{\forall j = 1, \ldots, n, b_{ij}^k(t) \neq 0\}$. Now we need to design an optimization algorithm to get the best $\mathcal{B}(t), t = 1, \ldots, L$ for Equation (10), which will be described as follows.

## 4 ALGORITHM

We first introduce a proposition to simplify our score function and translate constraint into algebraic characterization in Section 4.1. Then we develop a search algorithm to solve this problem in Section 4.2.

### 4.1 Formulation Simplification

Equation (10) consists of four parts: the sum of square errors of VAR, the $\ell_1$-norm of $\mathbf{b}_i(t)$ for network sparsity, the $\ell_1$-norm of $\mathbf{b}_i(t) - \mathbf{b}_i(t-1)$ for segment-wise structure change, and the graph Laplacian to capture variables' additional information. We first simplify this formulation with the following proposition.

Proposition 4.1. *Define* $\mathbf{x}(t) = [X_1(t-m), \ldots, X_n(t-m), \ldots, X_1(t), \ldots, X_n(t)] \in \mathbb{R}^{(m+1)n}$ *when* $t \geqslant m+1$, *and* $\mathbf{x}(t) \in \mathbb{R}^{(m+1)n}$ *are zeros when* $t \leqslant m$, *to represent all possible observation variables that affect* $X_i(t)$ *with the maximum lag* $m$. *Furthermore, define* $\mathbf{X} = diag(\overbrace{\mathbf{x}(t), \ldots, \mathbf{x}(t)}^{L}) \in \mathbb{R}^{L\times L(m+1)n}$ *diagonally composed by* $L$ *number of same* $\mathbf{x}(t)$, *and* $\tilde{\mathbf{X}} = diag(\overbrace{\mathbf{X}, \ldots, \mathbf{X}}^{n}) \in \mathbb{R}^{Ln\times L(m+1)n^2}$. *We further define* $\mathbf{Y} \in \mathbb{R}^{Ln}$ *composed by* $\mathbf{Y}_{(i-1)L:iL} = [\mathbf{0}_{1\times m}, X_i(m+1), X_i(m+2), \ldots, X_i(L)] \in \mathbb{R}^L$. *Set* $\mathbf{D} \in \mathbb{R}^{L(m+1)n\times L(m+1)n}$ *where its* $(j, j)$ *component is* $(\mathbf{D})_{jj} = -1$, *and* $(j, j + (m+1)n)$ *component is* $(\mathbf{D})_{j(j+(m+1)n)} = 1$ *for* $j = 1, 2, \ldots, (L-1)(m+1)n$. $\tilde{\mathbf{D}} = diag(\overbrace{\mathbf{D}, \ldots, \mathbf{D}}^{m+1}) \in \mathbb{R}^{L(m+1)n^2\times L(m+1)n^2}$ *is the first order difference matrix. Define* $\boldsymbol{L}^t = diag(\overbrace{\boldsymbol{L}(t), \ldots, \boldsymbol{L}(t)}^{n}) \in \mathbb{R}^{(m+1)n\times(m+1)n}$ *and* $\tilde{\boldsymbol{L}} = diag(\boldsymbol{L}^1, \ldots, \boldsymbol{L}^L) \in \mathbb{R}^{L(m+1)n\times L(m+1)n}$. $\mathbf{C}^t = diag(\mathbf{B}^m(t), \ldots, \mathbf{B}^0(t)) \in \mathbb{R}^{(m+1)n\times(m+1)n}$ *is parameter matrix, and* $\tilde{\mathbf{B}} = diag(\mathbf{C}^1, \ldots, \mathbf{C}^L) \in \mathbb{R}^{L(m+1)n\times L(m+1)n}$. *Then the optimization of Equation (10) is equivalent to a mixed-LASSO problem, i.e.,*

$$\begin{aligned}\min_{\tilde{\mathbf{b}}} &\|\mathbf{Y} - \tilde{\mathbf{X}}\tilde{\mathbf{b}}\|_2^2 + \lambda_1\|\tilde{\mathbf{b}}\|_1 + \lambda_2\|\tilde{\mathbf{D}}\tilde{\mathbf{b}}\|_1 + \lambda_3 tr(\tilde{\mathbf{B}}\tilde{\boldsymbol{L}}\tilde{\mathbf{B}}'), \\ &s.t. \quad \mathcal{B}(t) \text{ is acyclic}, \quad t = 1, \ldots, L,\end{aligned} \tag{11}$$

*where* $\mathbf{b}^i = [\mathbf{b}_i(1), \ldots, \mathbf{b}_i(L)] \in \mathbb{R}^{L(m+1)n}$ *and* $\tilde{\mathbf{b}} = [\mathbf{b}^1, \ldots, \mathbf{b}^n] \in \mathbb{R}^{L(m+1)n^2}$. *It is to be noted that, in Equation (11) we abuse the notations of* $\mathcal{B}(t)$, $\tilde{\mathbf{b}}$ *and* $\tilde{\mathbf{B}}$ *to represent different forms of* $b_{ij}^k(t)$ *for notation convenience, but they can be converted to each other.*

As to the acyclic constraint, for a binary adjacency matrix $\mathbf{B} \in \{0,1\}^{n\times n}$ whose largest eigenvalue has an absolute value smaller than 1, Zheng et al. [55] prove that a matrix exponential form, i.e., $tr(\mathbf{B}^k) = \sum_{i=1}^{d}(\mathbf{B}^k)_{ii}$, where $(\mathbf{B}^k)_{ii}$ represents the $(i,i)$ component of $\mathbf{B}^k$, actually counts the number of length-$k$ closed walks in a directed graph. Then we can identify that $\mathbf{B}$ has no cycles if and only if $(\mathbf{B}^k)_{ii} = 0$ for all $k \geqslant 1$ and all $i$. This is equivalent to $\sum_{k=1}^{\infty}\sum_{i=1}^{d}(\mathbf{B}^k)_{ii}/k! = tr(e^{\mathbf{B}}) - d = 0$. For general weighted $\mathbf{B}$, following Zheng et al. [55], the following proposition is also satisfied.

Proposition 4.2. *Set* $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{Ln\times Ln}$ *with* $\boldsymbol{\mathcal{B}}_{:,\,tn:(t+1)n} = [\mathbf{0}_{(t-m-1)n\times n}; \mathcal{B}(t); \mathbf{0}_{(L-t)n\times n}]$ *for* $t \geqslant m+1$ *and* $\boldsymbol{\mathcal{B}}_{:,\,tn:(t+1)n} = [\mathbf{0}_{Ln\times Ln}]$ *for* $t \leqslant m$. *Then if the following algebraic characterization is satisfied:*

$$tr(e^{\boldsymbol{\mathcal{B}}\circ\boldsymbol{\mathcal{B}}}) - Ln = 0, \tag{12}$$

*where* $\circ$ *is the Hadamard product of* $\boldsymbol{\mathcal{B}}$ *to ensure a non-negative weighted matrix, we can guarantee* $\mathcal{B}(t), t = 1, \ldots, L$ *is acyclic.*

### 4.2 Optimization via Mixed Algorithm

As a powerful solver for separable convex optimization problems with linear/nonlinear constraints, the ADMM algorithm has been widely used in many areas. It splits the original optimization problem into a separable form by replacing some parts of the original objective function with new variables' states and constraints, further transforming into an augmented Lagrangian problem afterward. Then closed-form solutions for each variable's sub-problem are derived for the augmented Lagrangian problem in an iterative way. The global convergence of the ADMM algorithm was established in the early 1990s by Eckstein and Bertsekas [10]. In particular, in our scenario, we introduce two consensus variables $\mathbf{Z}$ and $\boldsymbol{\theta}$ for $\tilde{\mathbf{D}}\tilde{\mathbf{b}}$ and $\tilde{\mathbf{B}}$ respectively. Based on them, Equation (11) with the algebraic characterization can be rewritten as Equation (13) equivalently:

$$\begin{aligned} &\min_{\tilde{\mathbf{b}},\boldsymbol{\theta},Z} \|\mathbf{Y} - \tilde{\mathbf{X}}\tilde{\mathbf{b}}\|_2^2 + \lambda_1\|\tilde{\mathbf{b}}\|_1 + \lambda_2\|Z\|_1 + \lambda_3 tr(\boldsymbol{\theta}\tilde{\mathbf{L}}\boldsymbol{\theta}'), \\ &\text{s.t. } tr(e^{\boldsymbol{\mathcal{B}}\circ\boldsymbol{\mathcal{B}}}) - Ln = 0, \mathbf{Z} = \tilde{\mathbf{D}}\tilde{\mathbf{b}}, \tilde{\mathbf{b}} = g(\tilde{\mathbf{B}}) = g(\boldsymbol{\theta}), \end{aligned} \tag{13}$$

where $g(\cdot)$ is the transform function between $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{B}}$. The corresponding augmented Lagrangian is expressed as Equation (14):

$$\begin{aligned} \mathcal{L}_\rho(\tilde{\mathbf{b}}, \boldsymbol{\theta}, Z, \mathbf{U}) = &\|\mathbf{Y} - \tilde{\mathbf{X}}\tilde{\mathbf{b}}\|_2^2 + \lambda_1\|\tilde{\mathbf{b}}\|_1 + \lambda_2\|Z\|_1 + \lambda_3 tr(\boldsymbol{\theta}\tilde{\mathbf{L}}\boldsymbol{\theta}') + \alpha|h(\boldsymbol{\mathcal{B}})| + \frac{\rho'}{2}|h(\boldsymbol{\mathcal{B}})|^2 \\ &+ \frac{\rho}{2}(\|\tilde{\mathbf{b}} - g(\boldsymbol{\theta}) + \mathbf{U}_1\|_2^2 - \|\mathbf{U}_1\|_2^2) + \frac{\rho}{2}(\|\tilde{\mathbf{D}}\tilde{\mathbf{b}} - Z + \mathbf{U}_2\|_2^2 - \|\mathbf{U}_2\|_2^2), \end{aligned} \tag{14}$$

where $h(\boldsymbol{\mathcal{B}}) = tr(e^{\boldsymbol{\mathcal{B}}\circ\boldsymbol{\mathcal{B}}}) - Ln$, $\mathbf{U} = \{\mathbf{U}_1, \mathbf{U}_2\}$ is the scaled dual variable, $\rho'$ and $\rho$ are augmented Lagrangian parameters. Denote $k$ as the iteration number, each updating step of ADMM is shown as follows.

**(a)** $\tilde{\mathbf{b}}^{k+1} = \arg\min_{\tilde{\mathbf{b}}} \mathcal{L}_\rho(\tilde{\mathbf{b}}, \boldsymbol{\theta}^k, Z^k, \mathbf{U}^k)$.

In Equation (14), the part related with $\tilde{\mathbf{b}}$ is as follows:

$$\begin{aligned} \mathcal{L}_\rho(\tilde{\mathbf{b}}, \boldsymbol{\theta}, Z, \mathbf{U}) = &\|\mathbf{Y} - \tilde{\mathbf{X}}\tilde{\mathbf{b}}\|_2^2 + \lambda_1\|\tilde{\mathbf{b}}\|_1 + \alpha|h(\boldsymbol{\mathcal{B}})| + \frac{\rho'}{2}|h(\boldsymbol{\mathcal{B}})|^2 \\ &+ \frac{\rho}{2}(\|\tilde{\mathbf{b}} - g(\boldsymbol{\theta}) + \mathbf{U}_1\|_2^2 - \|\mathbf{U}_1\|_2^2) + \frac{\rho}{2}(\|\tilde{\mathbf{D}}\tilde{\mathbf{b}} - Z + \mathbf{U}_2\|_2^2 - \|\mathbf{U}_2\|_2^2). \end{aligned} \tag{15}$$

The minimization over Equation (15) has no closed-form solution. Hence we apply the L-BFGS-B method [55] to find the minimum of $\tilde{\mathbf{b}}$. L-BFGS-B algorithm is a proximal quasi-Newton algorithm, which constructs the proximal Hessian matrix and only stores curvature information of the most recent iterations in calculating the Hessian matrix. It is both time and memory saving in solving the convex optimization problem with constraints. In particular, the derivative of smooth part $\mathcal{F}_\rho(\tilde{\mathbf{b}}, \boldsymbol{\theta}, Z, \mathrm{U}) = \|\mathrm{Y} - \tilde{\mathrm{X}}\tilde{\mathbf{b}}\|_2^2 + \alpha|h(\boldsymbol{\mathcal{B}})| + \frac{\rho'}{2}|h(\boldsymbol{\mathcal{B}})|^2 + \frac{\rho}{2}(\|\tilde{\mathbf{b}} - g(\boldsymbol{\theta}) + \mathrm{U}_1\|_2^2 - \|\mathrm{U}_1\|_2^2) + \frac{\rho}{2}(\|\tilde{\mathrm{D}}\tilde{\mathbf{b}} - Z + \mathrm{U}_2\|_2^2 - \|\mathrm{U}_2\|_2^2)$ of Equation (15) with respect to $\tilde{\mathbf{b}}$ is

$$\begin{aligned}\frac{\partial \mathcal{F}_\rho(\tilde{\mathbf{b}}, \boldsymbol{\theta}^k, Z^k, \mathrm{U}^k)}{\partial \tilde{\mathbf{b}}} = &-2\tilde{\mathrm{X}}'\mathrm{Y} + 2\tilde{\mathrm{X}}'\tilde{\mathrm{X}}\tilde{\mathbf{b}} + \alpha h'(\boldsymbol{\mathcal{B}}) + \rho' h'(\boldsymbol{\mathcal{B}})h(\boldsymbol{\mathcal{B}}) + \rho(\tilde{\mathbf{b}} + \mathrm{U}_1^k - g(\boldsymbol{\theta}^k) + \tilde{\mathrm{D}}'\tilde{\mathrm{D}}\tilde{\mathbf{b}} \\ &+ \tilde{\mathrm{D}}'(\mathrm{U}_2^k - \mathrm{Z}^k)),\end{aligned} \tag{16}$$

where $h'(\boldsymbol{\mathcal{B}})$ is the gradient of $h(\boldsymbol{\mathcal{B}})$. For computation simplicity, we replace $h(\boldsymbol{\mathcal{B}})$ by its equivalence [51], i.e., $h(\boldsymbol{\mathcal{B}}) = tr[(\mathrm{I} + \alpha(\boldsymbol{\mathcal{B}} \circ \boldsymbol{\mathcal{B}})^{Ln}] - Ln$. Then $h'(\boldsymbol{\mathcal{B}}) = (\mathrm{I} + \frac{\boldsymbol{\mathcal{B}} \circ \boldsymbol{\mathcal{B}}}{Ln})^{Ln-1} \circ 2\boldsymbol{\mathcal{B}}$, where $\mathrm{I}$ is identity matrix and has the same shape with $\boldsymbol{\mathcal{B}}$. Besides, to accelerate speed of L-BFGS-B, we limit the searching space in possible locations through initial bounds according to the composition of $\boldsymbol{\mathcal{B}}$. In this way, for a $Ln \times Ln$ matrix $\boldsymbol{\mathcal{B}}$, there are only $L(m+1)n^2$ parameters to estimate with $m \ll L$.

**(b)** $\boldsymbol{\theta}^{k+1} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_\rho(\tilde{\mathbf{b}}^{k+1}, \boldsymbol{\theta}, Z^k, \mathrm{U}^k)$.

The update of $\boldsymbol{\theta}$ has the closed-form solution. The gradient of Equation (14) with respect to $\boldsymbol{\theta}$ is

$$\frac{\partial \mathcal{L}_\rho(\tilde{\mathbf{b}}^{k+1}, \boldsymbol{\theta}, Z^k, \mathrm{U}^k)}{\partial \boldsymbol{\theta}} = \lambda_3(\boldsymbol{\theta}\tilde{\mathrm{L}} + \boldsymbol{\theta}\tilde{\mathrm{L}}') + \rho\boldsymbol{\theta} - \rho g^{-1}(\mathrm{U}_1^k + \tilde{\mathbf{b}}^{k+1}), \tag{17}$$

where $\boldsymbol{\theta} = g^{-1}(\tilde{\mathbf{b}})$ transforms $\tilde{\mathbf{b}}$ back to the matrix form $\boldsymbol{\theta}$. Taking this gradient function to be zero, we can get the minimum $\boldsymbol{\theta}$ as Equation (18),

$$\boldsymbol{\theta}^{k+1} = (\mathrm{U}_1^k + \tilde{\mathbf{b}}^{k+1})(\lambda_3\tilde{\mathrm{L}} + \lambda_3\tilde{\mathrm{L}}' + \rho\mathrm{I})^{-1}, \tag{18}$$

where $\mathrm{I}$ is identity matrix and has the same size as $\tilde{\mathrm{L}}$.

**(c)** $Z^{k+1} = \arg\min_Z \mathcal{L}_\rho(\tilde{\mathbf{b}}^{k+1}, \boldsymbol{\theta}^{k+1}, Z, \mathrm{U}^k)$.

$Z$ can be written as the proximal operator of the $\ell_1$-norm, which has the closed-form solution [4]:

$$\mathrm{Z}^{k+1} = \mathbf{prox}_{\frac{\lambda_2}{\rho}}(\tilde{\mathrm{D}}\tilde{\mathbf{b}}^{k+1} + \mathrm{U}_2^k), \tag{19}$$

where $\mathbf{prox}_\kappa(a)$ is the element-wise soft-threshold function defined as

$$\mathbf{prox}_\kappa(a) = \begin{cases} a - \kappa, & a > \kappa, \\ 0, & |a| \leqslant \kappa, \\ a + \kappa, & a < -\kappa. \end{cases} \tag{20}$$

**(d)** $\mathrm{U}^{k+1} = \begin{bmatrix} \mathrm{U}_1^{k+1} \\ \mathrm{U}_2^{k+1} \end{bmatrix}$.

The update of a scaled dual variable can be easily derived as

$$\mathrm{U}^{k+1} = \begin{bmatrix} \mathrm{U}_1^{k+1} \\ \mathrm{U}_2^{k+1} \end{bmatrix} = \begin{bmatrix} \mathrm{U}_1^k \\ \mathrm{U}_2^k \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{b}}^{k+1} - g(\boldsymbol{\theta}^{k+1}) \\ \tilde{\mathrm{D}}\tilde{\mathbf{b}}^{k+1} - Z^{k+1} \end{bmatrix}. \tag{21}$$

It is to be noted that we set the initial values of all the parameters to be zeros following a commonly used and simple way. Some other alternatives can also be adopted to ensure a faster convergence rate. For example, Ghadimi et al. [16] tested the number of required iterations for

different initial states. In summary, we split the problem up into a series of sub-problems (a)–(d) and use iterative optimizations to get the globally optimal solution. For parameter settings, we follow the advice of Zheng et al. [55], and set $\rho' > 0$ and Lagrange multiplier $\alpha = \alpha + \rho' h(\mathcal{B})$. Algorithm 1 shows the detailed procedure. Based on the solved $\mathcal{B}$, we can get the dynamic structure of $\mathcal{G}_t$. Hereafter we name the constructed TVDBN, which has Segment-Wise varyINg structure with Graph information as SWING.

**ALGORITHM 1:** To find optimal parameter $\tilde{\mathbf{b}}$ for problem (14) with mixed algorithm.

**Input:** Initial guess $(\tilde{\mathbf{b}}^0, \boldsymbol{\theta}^0, \mathbf{Z}^0, \mathbf{U}^0)$ (all zero matrices(vectors)); progress rate $c1 \in (0, 1)$ and $c2 > 1$, tolerance $\epsilon, \epsilon_1 > 0$, threshold $\tau \geqslant 0$, iteration times $v, v_1 > 0$, penalty $\lambda_1, \lambda_2, \lambda_3, \rho, \rho'$ and max penalty $\rho'_m$, step k = 0;

**Output:** $\tilde{\mathbf{b}}$

**for** $k = 0$ to $v$ **do**
  $\tilde{\mathbf{b}}^{k+1}$ update via step 3 − 14
  **for** $j = 0$ to $v_1$ **do**
    **while** $\rho' < \rho_{max}$ **do**
      $\mathcal{B}_{j+1} = \arg\min_{\tilde{\mathbf{b}}} \mathcal{L}_\rho(\tilde{\mathbf{b}}, \boldsymbol{\theta}^k, \mathbf{Z}^k, \mathbf{U}^k, \alpha^j)$ with L-BFGS-B approach
      **if** $h(\mathcal{B}_{j+1}) > c1 * h(\mathcal{B}_j)$ **then**
        $\rho' = \rho' \times c2$
      **else**
        break;
    dual gradient ascent $\alpha^{j+1} \leftarrow \alpha^j + \rho' h(\mathcal{B}_j)$
    **if** $h(\mathcal{B}_{j+1}) \leqslant \epsilon$ or $\rho' \geqslant \rho'_m$ **then**
      break
    **else**
      $j = j + 1$
  $\boldsymbol{\theta}^{k+1}$ update via Equation (18)
  $\mathbf{Z}^{k+1}$ update via Equation (19)
  $\mathbf{U}^{k+1}$ update via Equation (21)
  **if** $\mathcal{L}_\rho(\tilde{\mathbf{b}}^{k+1}, \boldsymbol{\theta}^{k+1}, \mathbf{Z}^{k+1}, \mathbf{U}^{k+1}) - \mathcal{L}_\rho(\tilde{\mathbf{b}}^k, \boldsymbol{\theta}^k, \mathbf{Z}^k, \mathbf{U}^k) < \epsilon_1$ **then**
    break;
**return** $\tilde{\mathbf{b}}$

### 4.3 Complexity

For updating $\tilde{\mathbf{b}}$, similar to Zheng et al. [55], the complexity is $O(L^2n^2|S| + L^3n^3 + Ln|S|T)$, where $S$ is an active set of coordinates and $T$ is the number of inner iterations of L-BFGS-B. For updating $\boldsymbol{\theta}$, the inverse of a matrix with $k$ rows takes $O(k^3)$ computation. Then the inverse process in Equation (18) takes $O(L^3n^3(m+1)^3)$ computation. Generally, $\rho$ changes little in different iterations. Then we can prepare the inverse results in advance and save them correspondingly. For updating $\mathbf{Z}$ and $\mathbf{U}$, the time complexities are both $O(L^3n^6(m+1)^3)$. As such, the total computation complexity for one iteration of Algorithm 1 is $O(L^2n^2|S| + L^3n^3 + Ln|S|T + L^3n^6(m+1)^3)$.

## 5 NUMERICAL EXPERIMENT AND CASE STUDY

To assess the efficacy and efficiency of our proposed SWING, we perform a series of synthetic data experiments and a real-world case study. We also compare SWING with several state-of-the-art methods, including:

— `notears` [55]: it is a SBN model with linear directed conditional dependence relationships;
— `DYnotears` [38]: it is a THDBN model with both linear inter-slice and intra-slice directed conditional dependence relationships;
— `EDISON` [9]: it is a TVDBN model with only linear inter-slice directed conditional dependence relationships and segment-wise time-varying dynamic network structure;
— `LTGL` [46]: it is an undirected network with segment-wise time-varying graph lasso;
— `SWIN`: it is a simple version of SWING, which does not include graph Laplacian.

We use different criteria to evaluate the performance of SWING and baselines concerning the divergence of their estimated structures from the true ones. First, define the **true/false positive(TP/FP)** as the number of outcomes where the model correctly/incorrectly predicts the existing edges and **true/false negative(TN/FN)** as the number of outcomes where the model correctly/incorrectly predicts the non-existing edges. Then, we use the following four criteria for performance evaluation.

— accuracy = (TP+TN)/(TP+TN+FP+FN);
— precision = TP/(TP+FP);
— recall = TP/(TP+FN);
— **Structural Hamming distance(SHD)**: the total number of edges required to convert from the recovered structure into the ground truth, including removing, reversing, and adding edges.

### 5.1 Synthetic Experiments

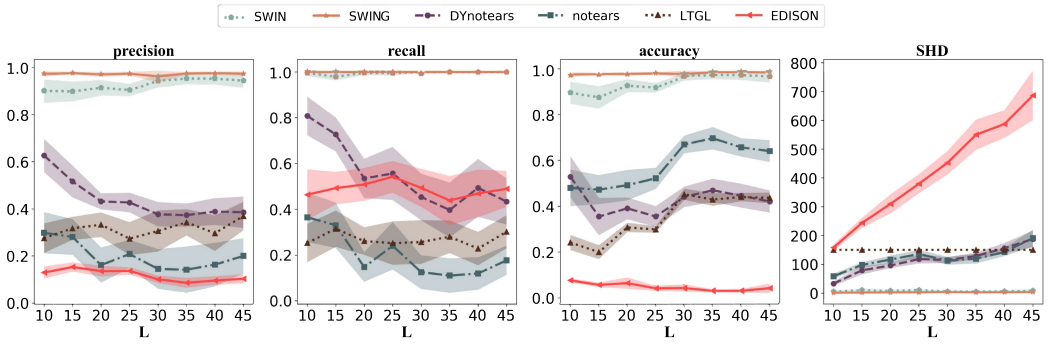
To reasonably verify the performance of SWING and the other five baselines, we generate data from the following two scenarios:

— `Scenario I`: We generate data following the model assumption of SWING:

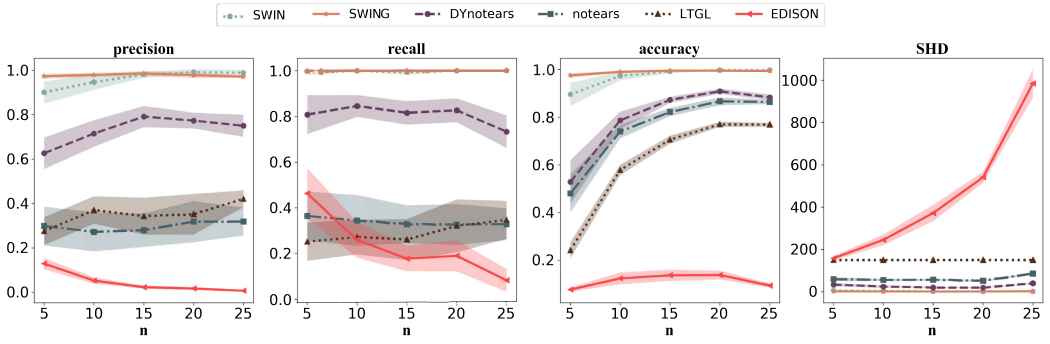
$$X_i(t) = \sum_{j \in \pi_i^0} b_{ij}^0(t) \cdot X_j(t) + \sum_{j \in \pi_i^k} \sum_{k=1}^{m} b_{ij}^k(t) \cdot X_j(t-k) + \varepsilon_i(t), i = 1, \ldots, n, t = 1, \ldots, L, \tag{22}$$

where $\boldsymbol{\varepsilon}_i(t) \sim N(0, \boldsymbol{\sigma}^2(t))$. We generate $C-1$ change-points from uniform distribution $U(1:L)$ and get $C$ segments. For each segment $c = 1, \ldots, C$, we set $b_{ij,c}^k = a_{ij,c}^k I_{ij,c}^k$, where $I_{ij,c}^k$ is drawn from Bernoulli distribution with probability $w = 0.5$ equal to 1 and $1-w$ equal to 0, and $a_{ij,c}^k$ is drawn from a uniform distribution $U((0.8,1),(-1,-0.8))$. If $I_{ij,c}^k$ formulates a cyclic network, we randomly drop certain components to be zero until it becomes acyclic. Similarity between variable $i$ and $j$ at $t$ is $W_{ij}^t = \frac{1}{|X_i(t)-X_j(t)|}$, i.e., the $i$th row and $j$th column entry of similarity matrix $\mathbf{W}^t$.
— `Scenario II`: We generate data following the model assumption of EDISON, which is a simplified version of Equation (22). It only considers lag-1 intra-slice directed conditional dependence with $b_{ij}^k(t) = 0, k \neq 1$ consistently. The data generation process is the same as Scenario I. In this case, the acyclic property of the network formulated by $I_{ij,c}^k$ can always be satisfied. The similarity matrix is generated as the scenario I.

For both scenarios, we consider segment length $v = 6$, which means some edges change every six-time point. For SWING, we manipulate the real network structure by adding noise on its edges and use the manipulated one as the corresponding similarity matrix for graph Laplacian. We generate $N = 500$ sequences of samples to calculate the average precision, recall, accuracy, and SHD, together with its 90% confidence intervals. The performance of SWING and five baselines considering different time lengths and numbers of nodes for Scenario I is shown in Figure 3. As expected,




(a) length of time changes.



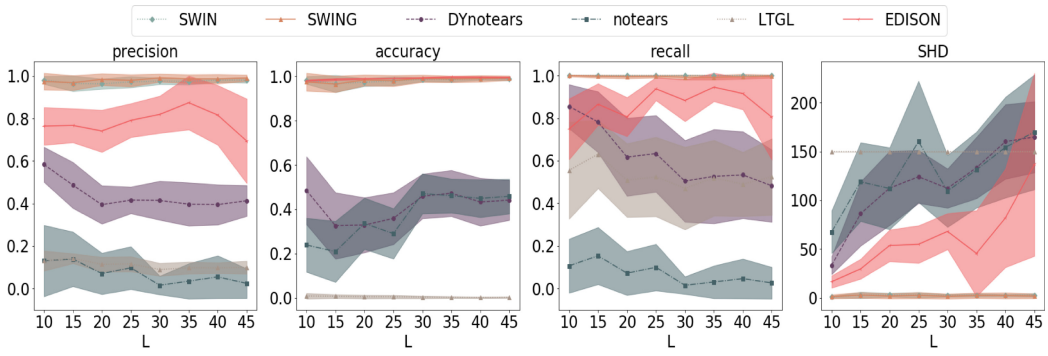

(b) number of nodes changes.

Fig. 3. Performance comparisons under Scenario I.

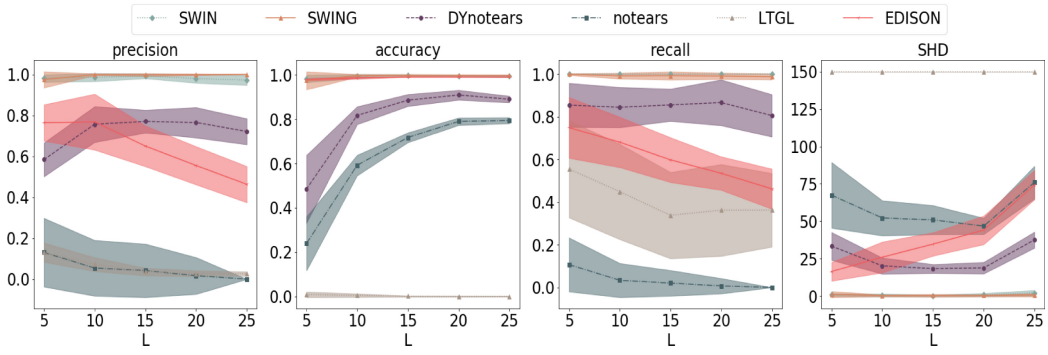
in general, SWING and SWIN perform best among all the methods. They stably achieve the best scores for all the evaluation metrics in all the settings. SWING performs slightly better than SWIN since it uses extra information which incorporates true network structure information inside. Yet their slight difference is almost insignificant. This indicates that even without external information, our method can still efficiently capture segment-wise dynamics of casual relationships. As to the other four baselines, generally, DYnotears performs the best since it still considers both intra-slice and inter-slice directed conditional dependence, followed by LTGL. As to notears, though its accuracy is not too bad, it has very unsatisfactory precision and recall. This indicates notears tend to have very high false positives due to their static structure formulation. As to EDISON, since it ignores intra-slice directed conditional dependence, its estimated network is much sparser than the true one. Consequently, it has unsatisfactory precision, though its recall is not too bad.

Furthermore, as time length increases, DYnotears performs much worse with the increasing length of time due to its time homogeneousness assumption. The other baselines also have relatively worse performance in lower precision and recall and higher SHD. This is because the number of segments increases, and the model complexity increases. Consequently, estimation with a fixed number of samples will become worse. Significantly, more false positives will occur, illustrated by the surprisingly increased accuracy. Yet SWING and SWIN are much less influenced by this. Similar conclusions also hold when the number of nodes increases, as shown in Figure 3(b), while this time, DYnotears do not deteriorate too much.

As to Scenario II shown in Figure 4, the results are similar to those of Scenario I except EDISON. Again, SWING performs the best in general, followed by SWIN with slightly worse performance.




(a) length of time changes.




(b) number of nodes changes.

Fig. 4. Performance comparisons under Scenario II.

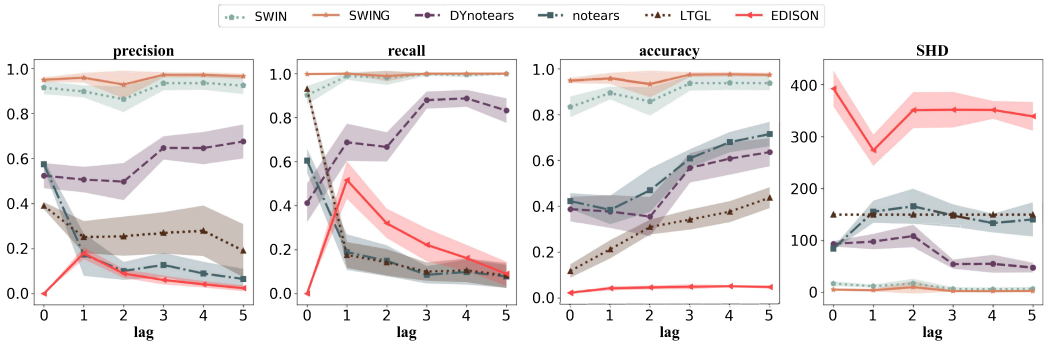
Table 1. Efficiency Comparison with $n = 5$ and $L = 21$

| Methods | SWING | SWIN | EDISON | DYnotears | notears | LTGL |
|---|---|---|---|---|---|---|
| Time(seconds) | 3.68 | 1.14 | 5.44 | 0.79 | 1.42 | 0.18 |

Though DYnotears has better precision and recall than other notears and LTGL, its SHD is as bad as others, especially for cases with larger time lengths. EDISON performs better than Scenario I since the data generation is consistent with its model assumption. EDISON also performs better than other baselines, especially under increasing time length. It is because for time-varying models, increasing time length gives more information to estimate than time-homogeneous models like DYnotears. But its performance is still worse than SWING. This is reasonable since EDISON is a special case of SWING, which further demonstrates the robustness of SWING.

In particular, we add an experiment on the efficiency comparison with node numbers $n = 5$, length of time series $L = 21$ and run in the same computer. The time costs of SWING and baselines are shown in Table 1. We could see that our method is not the most time-consuming one. Both SWING and SWIN cost less time than EDISON. DYnotears consumes less time than notears because it is more applicable for the datasets and converges faster. Besides, all these methods are offline learning methods. Thus this time cost is acceptable in practice.

To further demonstrate the superiority of SWING over DYnotears, we increase the maximum lag $m$ from zero to five. When $m = 0$, it is the case without inter-slice directed conditional dependence, which is the case most similar to the model assumption of notears. When $m$ increases, it tends to




(a) time lag changes.

Fig. 5. Sensitivity analysis when maximum lag increases for Scenario I.

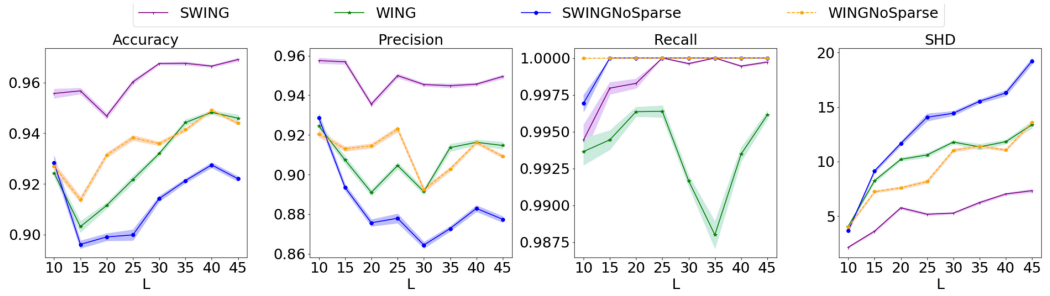



Fig. 6. Performance comparisons of ablation studies with changing length of time on sparsity and consistency.

weaken the network difference between different segments, which is the case that the most similar to the model assumption of DYnotears. When $m = 1$, it is the case most similar to the model assumption of EDISON. The results of Scenario I are shown in Figure 5. As expected, DYnotears achieves better results when time lag increases and notears achieves better results when time lag decreases. EDISON gets its best performance when $lag = 1$ and yet the worst when $lag = 0$ due to its ignoring of intra-slice directed conditional dependencies. As to LTGL, its model is designed for undirected intra-slice descriptions. Consequently, it has its best performance when $lag = 0$. However, it is unfair to mention the total number of edges required to convert its results to the ground truth because it is an undirected graph. Hence, we arbitrarily set its $SHD = 150$ fixedly without any meaning.

In particular, we do ablation studies on the sparsity regularization term and consistency regularization term. Here, we conduct three baselines: SWING without sparsity regularization as SWINGNoSparse, SWING without segment-wise regularization as WING, and SWING without both segment-wise regularization and sparsity regularization as WINGNoSparse. The performance comparisons of these three baselines and SWING are shown in Figure 6. We could know that sparsity and consistency play important roles in our model. Especially, our model is more sensitive to sparsity regularization than consistency regularization as SWINGNoSaprse performs worst in these experiments. However, compared with baselines except for SWIN illustrated in Figure 3(a), WING, SWINGNoSparse, and WINGNoSparse also outperform them in all four evaluation metrics.

We visualize the estimate $b_{ij}^k(t)$ together with the true one for one experiment with $n = 5, L = 21$, and $m = 3$. The whole network structure has two change-points with three segments in total, i.e.,

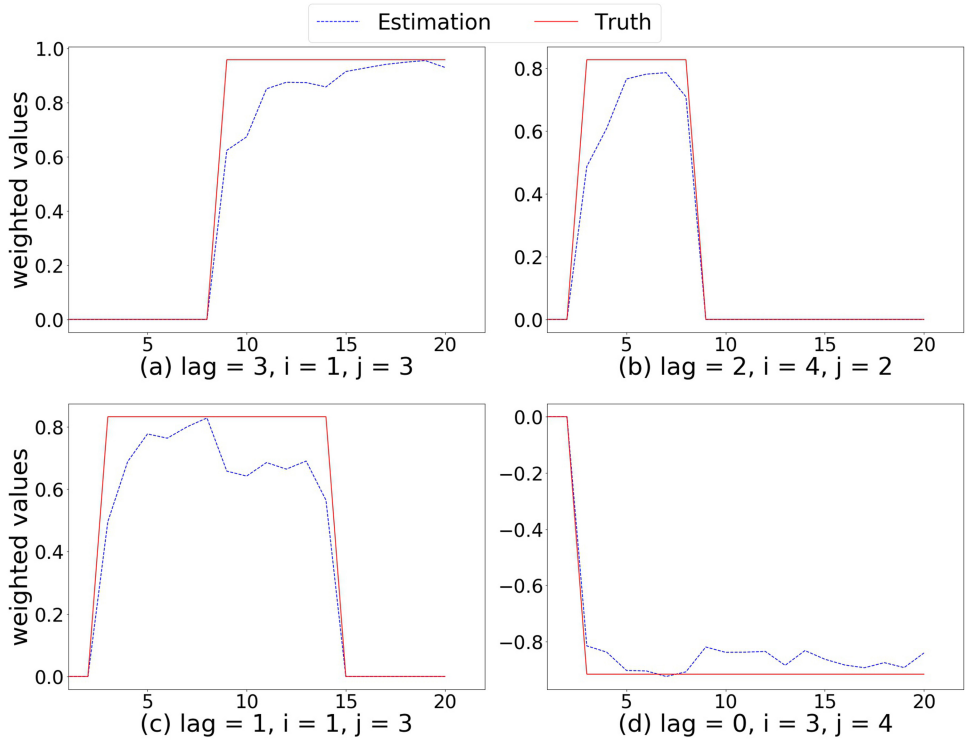



Fig. 7. Comparisons of estimated and true weighted values between different pair of variables in lags. This experiment has $SHD = 10$, $accuracy = 0.9$, $precision = 0.94$, and $recall = 1$, separately.

$[1 : 7], [8 : 14], [15 : 21]$. Yet its weight can be the same for different segments for each edge. Figure 7 demonstrates the estimated and true values of certain $b_{ij}^k(t)$ over different time steps. The estimated weights segment-wisely vary, complying with the true time-varying structure. Though for time steps without changes, the estimated weights still have small fluctuations due to random noise's influence, the fluctuation magnitude is much smaller than the signal's magnitude. In reality, we may set a threshold. If the change magnitude of edge weight is larger than the threshold, the algorithm triggers a structure change alarm. Furthermore, it shows that the estimated weights are more likely to be stable when they have a longer time duration of nonzero values. By averaging the estimated $b_{ij}^k(t)$ for each segment, Figure 8 further plots the averaged estimated $b_{ij}^k(t)$ and the true ones for different segments. For each segment, each row plots its $b_{ij}^k(t)$ for $k = 3, 2, 1, 0$, respectively. From the heatmap, we can see that although weighted values are not the same for the estimated and true ones, their difference is quite small.

### 5.2 Drosophila Muscle Development Gene Regulatory Networks

We apply SWING in a real biological dataset, i.e., Drosophila gene expression data [1], which is one of the most frequently used datasets for the performance evaluation of TVDBN models. The dataset contains expression measurements over 66 time points of 4,028 Drosophila genes throughout development and growth during the embryonic, larval, pupal, and adult stages of life. The true change-points of the four periods are located at 31, 41, and 59. We chose ten genes (eve, gfl/lmd, twi, sls, mhc, prm, actn, up, myo61f, and msp300) about Drosophila wing muscle development for analysis, following many other articles. We firstly do Granger causality hypothesis testing on

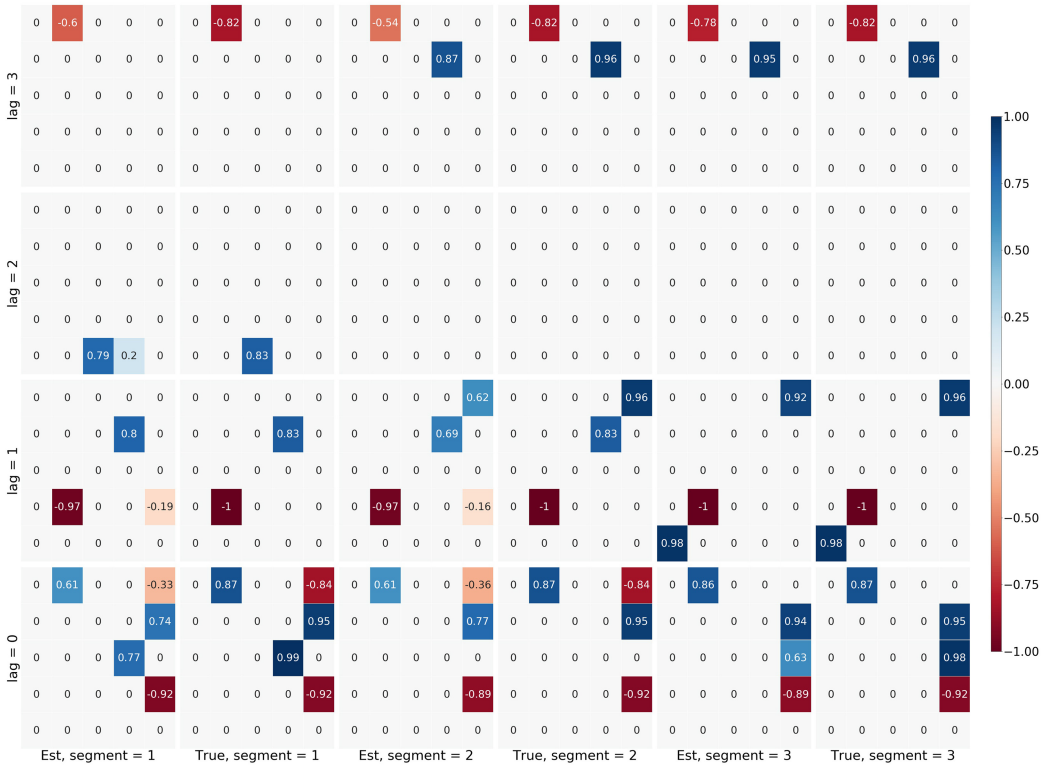
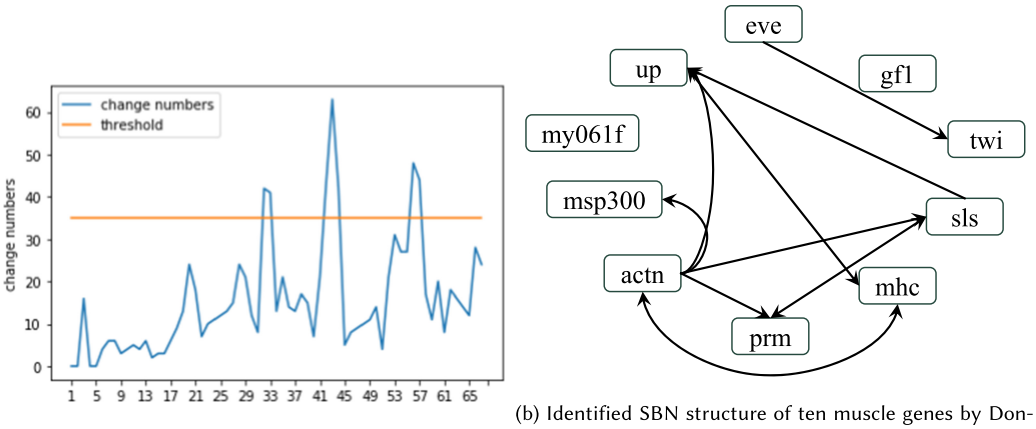



Fig. 8. Comparisons of the estimated and true network of all nodes for the three different segments. This experiment has $SHD = 10$, $accuracy = 0.9$, $precision = 0.94$, and $recall = 1$, separately.

inter-slice and intra-slice causal relationships using three methods, i.e., F test using the residual **sum of squares(SSR)** as F statistic, Chi-squared test using SSR and likelihood ratio test [18]. Take time series of twi and eve for instance, p-values in these three tests are smaller than 0.05 (0.0093, 0.0061, 0.0076 separately). Then we can conclude eve has an effect on twi with lag = 1. From a literature point of view, Dondelinger et al. [9], Li et al. [31], Robinson and Hartemink [40] have identified that there exists inter-slice causal relationships in this dataset. Besides, Monteiro et al. [34] applied a DBN model with both inter-slice and intra-slice relationships to this dataset. What's more, the biologist has identified that eve has an effect on twi with lag = 0 [39]. Therefore, there exist both inter-slice and intra-slice causal relationships in this dataset.

Dondelinger et al. [9] concluded gene relationships from a biology perspective, shown in Figure 9(b), which demonstrates static relationships between genes and also implies that there exists a similarity between genes. In our experiment, we adopt the identified SBN structure of ten muscle genes in Figure 9(b) as prior knowledge and assume the similarity matrix is static, i.e., set $W_{ij}$ be one if variable $i$ and $j$ are connected in Figure 9(b) and otherwise be zero. SWING identifies the maximum lag number as $m = 1$. After estimating $b_{ij}^k(t), k = 0, 1$, for $i, j = 1, \ldots, 10$, we calculate the number of changed edges for each time points, i.e., $\sum_{i=1}^{10}\sum_{j=1}^{10}\sum_{k=0}^{1} I(b_{ij}^k(t) - b_{ij}^k(t-1) \neq 0)$, and plot the results in Figure 9(a). The results have much more fluctuations than synthetic data analysis since the real dataset is noisier. However, we can see that there are three short time intervals with many changed edges. These three small intervals are consistent with the true stage transition time points. The only difference is the detected change-points are three small change intervals.




(a) changed numbers between nearby time points.

(b) Identified SBN structure of ten muscle genes by Dondelinger et al. [9].

Fig. 9. Number of change-points between nearby time points and identified interactions of genes.

This is reasonable since the switch of different stages cannot complete suddenly in one time point but requires a small time interval waiting for the genes' growth for this real case application.

For each identified stage, we calculate the averaged $b_{ij}^k(t)$ for each edge and plot the identified network structure in Figure 10. Though we do not have the true label of the network structure due to the absence of a gold standard in biology, our estimated network structures show similarities with the identified static network discovered by Formstecher et al. [12], Homyk Jr and Emerson Jr [22], Montana and Littleton [33], Nongthomba et al. [37], Sanchez et al. [42], as shown in Figure 9(b). For instance, we recover the interaction between two genes, eve and twi, in pupal period and adult period. This interaction is also identified and reported in Dondelinger et al. [9], while Robinson and Hartemink [41] seem to have missed it. However, SWING identifies these interactions into different stages, which could not be completed with THDBN models like Zhao et al. [54]. This shows that our dynamic time-varying structure can discover more detailed temporal information of gene interaction structures than the current works.

Furthermore, Yu et al. [50] found that maternal msp300 plays an important role in actin-dependent nuclear anchorage during cytoplasmic transport. Similar to Yu et al. [50], Figure 10 also shows that msp300 is highly connected with other genes. Msp300 can activate several genes in the embryonic period to promote cellular development. However, its activities will be inhibited in the later stages, particularly in the adult stage. Besides our consistency with other articles, we also find different results. [9, 41] only plot genes' interactions in $lag = 1$, and yet miss intra-relationships of them. We replenish the structure of $lag = 0$ and present more informative details. For example, we find that gene mhc and up are gene hubs in embryonic and larval periods. We hope our results in Figure 10 could shed light upon further drosophila gene studies in related fields. In a conclusion, our experiment on this real case study shows the efficacy and efficiency of SWING.

## 6 CONCLUSIONS AND FUTURE WORK

This article targets TVDBN modeling and structure learning. We proposed a VAR-based model to describe the intra-slice and inter-slice directed conditional dependence structure between variables while guaranteeing its acyclic property. The model allows the network structure to change segment-wisely over time. By further introducing graph Laplacian, the model can also incorporate external information helpful for structure learning. A score-based estimation algorithm based on ADMM with L-BFGS-B is constructed for structure learning. Thorough simulation studies and a

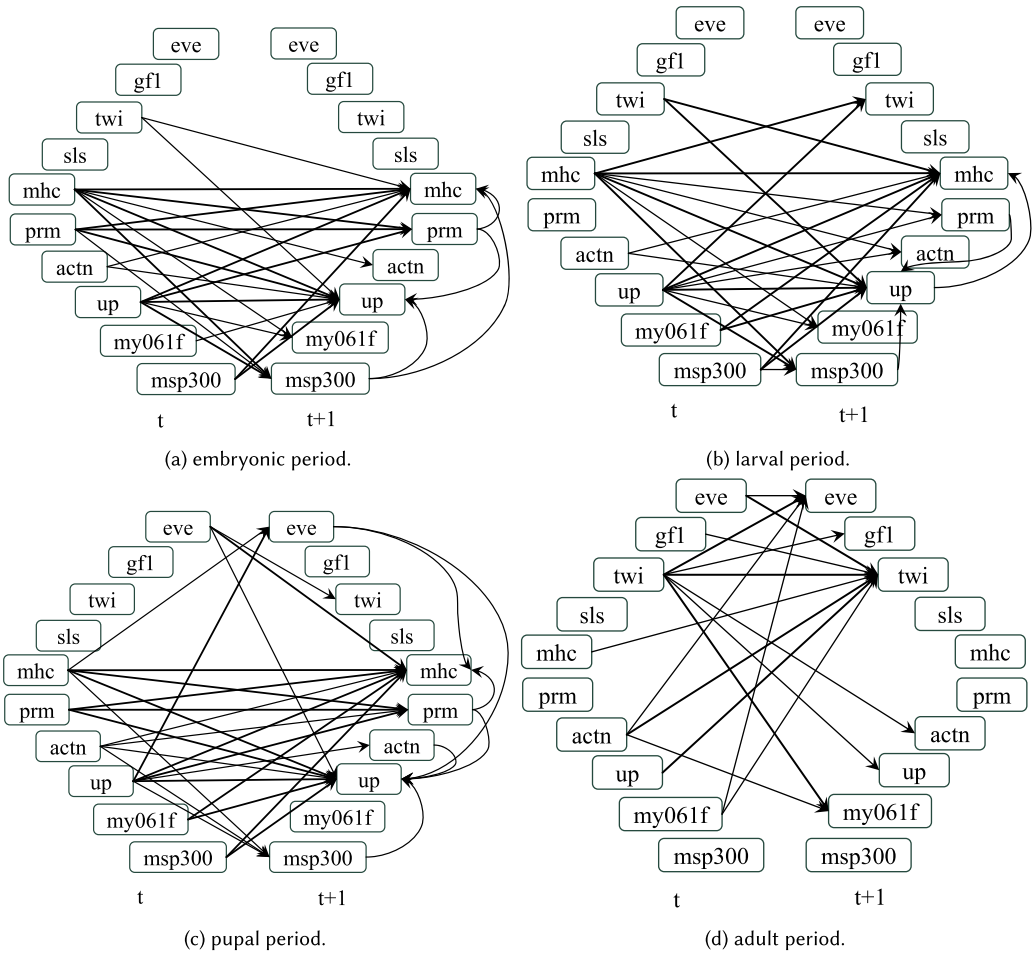



Fig. 10. Network structure learned by SWING for different stages in the muscle development.

gene regulatory case are carried out and demonstrated that this method could score satisfying results. However, this model is just a segment-wise structure and cannot enable the structure to change segmently definitely. Besides, when time-series length increases to thousands, it is hard to estimate the structure. We hope to propose a modified model to overcome these shortcomings in the future.

## REFERENCES

[1] Michelle N. Arbeitman, Eileen E. M. Furlong, Farhad Imam, Eric Johnson, Brian H. Null, Bruce S. Baker, Mark A. Krasnow, Matthew P. Scott, Ronald W. Davis, and Kevin P. White. 2002. Gene expression during the life cycle of Drosophila melanogaster. *Science* 297, 5590 (2002), 2270–2275.

[2] Mark Bartlett and James Cussens. 2017. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence* 244 (2017), 258–271.

[3] Shahab Behjati and Hamid Beigy. 2018. An order-based algorithm for learning structure of bayesian networks. In *Proceedings of the International Conference on Probabilistic Graphical Models*. 25–36.

[4] Stephen Boyd, Neal Parikh, and Eric Chu. 2011. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Publishers Inc.

[5] Victor W. Chu, Raymond K. Wong, Fang Chen, Simon Fong, and Patrick C. K. Hung. 2016. Self-regularized causal structure discovery for trajectory-based networks. *Journal of Computer and System Sciences* 82, 4 (2016), 594–609.
[6] James Cussens, David Haws, and Milan Studenỳ. 2017. Polyhedral aspects of score equivalence in Bayesian network structure learning. *Mathematical Programming* 164, 1–2 (2017), 285–324.
[7] Rónán Daly and Qiang Shen. 2009. Learning Bayesian network equivalence classes with ant colony optimization. *Journal of Artificial Intelligence Research* 35 (2009), 391–447.
[8] Shilpa Dang, Santanu Chaudhury, Brejesh Lall, and Prasun Kumar Roy. 2017. The dynamic programming high-order dynamic Bayesian networks learning for identifying effective connectivity in human brain from fMRI. *Journal of Neuroscience Methods* 285 (2017), 33–44.
[9] Frank Dondelinger, Sophie Lèbre, and Dirk Husmeier. 2013. Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Machine Learning* 90, 2 (2013), 191–230.
[10] Jonathan Eckstein and Dimitri P. Bertsekas. 1992. On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming* 55, 1 (1992), 293–318.
[11] Xiannian Fan, Brandon M. Malone, and Changhe Yuan. 2014. Finding optimal bayesian network structures with constraints learned from data. In *Proceedings of the UAI*. 200–209.
[12] Etienne Formstecher, Sandra Aresta, Vincent Collura, Alexandre Hamburger, Alain Meil, Alexandra Trehin, Céline Reverdy, Virginie Betin, Sophie Maire, Christine Brun, et al. 2005. Protein interaction mapping: A Drosophila case study. *Genome research* 15, 3 (2005), 376–384.
[13] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. 2000. Using bayesian networks to analyze expression data. *Journal of Computational Biology* 7, 3–4 (2000), 601–620.
[14] Changhe Fu, Su Deng, Guangxu Jin, Xinxin Wang, and Zu-Guo Yu. 2017. Bayesian network model for identification of pathways by integrating protein interaction with genetic interaction data. *BMC Systems Biology* 11, 4 (2017), 35–42.
[15] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3656–3663.
[16] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. 2014. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *IEEE Transactions on Automatic Control* 60, 3 (2014), 644–658.
[17] Soheila Gheisari and Mohammad Reza Meybodi. 2016. Bnc-pso: Structure learning of bayesian networks by particle swarm optimization. *Information Sciences* 348 (2016), 272–289.
[18] William H. Greene. 2000. Econometric analysis 4th edition. *International edition, New Jersey: Prentice Hall* (2000), 201–215.
[19] Marco Grzegorczyk. 2016. A non-homogeneous dynamic Bayesian network with a hidden Markov model dependency structure among the temporal data points. *Machine Learning* 102, 2 (2016), 155–207.
[20] Marco Grzegorczyk and Dirk Husmeier. 2009. Non-stationary continuous dynamic Bayesian networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 682–690.
[21] David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec. 2017. Network inference via the time-varying graphical lasso. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 205–213.
[22] T. H. Homyk Jr and C. P. Emerson Jr. 1988. Functional interactions between unlinked muscle genes within haploinsufficient regions of the Drosophila genome. *Genetics* 119, 1 (1988), 105–121.
[23] Niamat Ullah Ibne Hossain, Raed Jaradat, Seyedmohsen Hosseini, Mohammad Marufuzzaman, and Randy K. Buchanan. 2019. A framework for modeling and assessing system resilience using a Bayesian network: A case study of an interdependent electrical infrastructure system. *International Journal of Critical Infrastructure Protection* 25 (2019), 62–83.
[24] Dirk Husmeier, Frank Dondelinger, and Sophie Lebre. 2010. Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks. In *Proceedings of the NIPS*.
[25] Yi Jia and Jun Huan. 2010. Constructing non-stationary Dynamic Bayesian Networks with a flexible lag choosing mechanism. In *Proceedings of the BMC Bioinformatics*. Springer, S27.
[26] Liping Jing, Peng Wang, and Liu Yang. 2015. Sparse probabilistic matrix factorization by laplace distribution for collaborative filtering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*.
[27] Sunyong Kim, Seiya Imoto, and Satoru Miyano. 2004. Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data. *Biosystems* 75, 1–3 (2004), 57–65.
[28] Mikko Koivisto and Kismat Sood. 2004. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research* 5, (2004), 549–573.

[29] Mladen Kolar and Eric Xing. 2011. On time varying undirected graphs. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 407–415.
[30] Pedro Larrañaga, Mikel Poza, Yosu Yurramendi, Roberto H. Murga, and Cindy M. H. Kuijpers. 1996. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18, 9 (1996), 912–926.
[31] Peng Li, Chaoyang Zhang, Edward J. Perkins, Ping Gong, and Youping Deng. 2007. Comparison of probabilistic Boolean network and dynamic Bayesian network approaches for inferring gene regulatory networks. In *Proceedings of the BMC Bioinformatics*. Springer, 1–8.
[32] Brandon Malone, Changhe Yuan, and Eric Hansen. 2011. Memory-efficient dynamic programming for learning optimal Bayesian networks. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*.
[33] Enrico S. Montana and J. Troy Littleton. 2004. Characterization of a hypercontraction-induced myopathy in Drosophila caused by mutations in Mhc. *The Journal of Cell Biology* 164, 7 (2004), 1045–1054.
[34] José L. Monteiro, Susana Vinga, and Alexandra M. Carvalho. 2015. Polynomial-time algorithm for learning optimal tree-augmented dynamic Bayesian networks. In *Proceedings of the UAI*. 622–631.
[35] Chris J. Needham, James R. Bradford, Andrew J. Bulpitt, and David R. Westhead. 2007. A primer on learning in Bayesian networks for computational biology. *PLoS Computational Biology* 3, 8 (2007), e129.
[36] Martin Neil, Norman Fenton, and Manesh Tailor. 2005. Using Bayesian networks to model expected and unexpected operational losses. *Risk Analysis: An International Journal* 25, 4 (2005), 963–972.
[37] Upendra Nongthomba, Mark Cummins, Samantha Clark, Jim O. Vigoreaux, and John C. Sparrow. 2003. Suppression of muscle hypercontraction by mutations in the myosin heavy chain gene of Drosophila melanogaster. *Genetics* 164, 1 (2003), 209–222.
[38] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Paul Beaumont, Konstantinos Georgatzis, and Bryon Aragam. 2020. DYNOTEARS: Structure learning from time-series data. arXiv:2002.00498. Retrieved from https://arxiv.org/abs/2002.00498.
[39] Susan M. Parkhurst and David Ish-Horowicz. 1991. wimp, a dominant maternal-effect mutation, reduces transcription of a specific subset of segmentation genes in Drosophila. *Genes & Development* 5, 3 (1991), 341–357.
[40] Joshua Robinson and Alexander Hartemink. 2008. Non-stationary dynamic Bayesian networks. *Advances in Neural Information Processing Systems* 21 (2008), 1369–1376.
[41] Joshua W. Robinson and Alexander J. Hartemink. 2009. Non-stationary dynamic Bayesian networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 1369–1376.
[42] Catherine Sanchez, Corinne Lachaize, Florence Janody, Bernard Bellon, Laurence Röder, Jérôme Euzenat, François Rechenmann, and Bernard Jacq. 1999. Grasping at molecular interactions and genetic networks in Drosophila melanogaster using FlyNets, an Internet database. *Nucleic Acids Research* 27, 1 (1999), 89–94.
[43] Catherine Shenoy and Prakash P. Shenoy. 2000. *Bayesian Network Models of Portfolio Risk and Return*. The MIT Press.
[44] Ajit P. Singh and Andrew W. Moore. 2005. *Finding Optimal Bayesian Networks by Dynamic Programming*. Citeseer.
[45] Le Song, Mladen Kolar, and Eric P. Xing. 2009. Time-varying dynamic bayesian networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 1732–1740.
[46] Federico Tomasi, Veronica Tozzo, Saverio Salzo, and Alessandro Verri. 2018. Latent variable time-varying network inference. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2338–2346.
[47] Michail Tsagris. 2020. A new scalable bayesian network learning algorithm with applications to economics. *Computational Economics* (2020), 1–27.
[48] Nguyen Xuan Vinh, Madhu Chetty, Ross Coppel, and Pramod P. Wangikar. 2012. Gene regulatory network modeling via global optimization of high-order dynamic bayesian network. *BMC Bioinformatics* 13, 1 (2012), 1–16.
[49] Guang Wang, Tianhua Xu, Tao Tang, Tangming Yuan, and Haifeng Wang. 2017. A Bayesian network model for prediction of weather-related failures in railway turnout systems. *Expert Systems with Applications* 69 (2017), 247–256.
[50] Juehua Yu, Daniel A. Starr, Xiaohui Wu, Susan M. Parkhurst, Yuan Zhuang, Tian Xu, Rener Xu, and Min Han. 2006. The KASH domain protein MSP-300 plays an essential role in nuclear anchoring during Drosophila oogenesis. *Developmental Biology* 289, 2 (2006), 336–345.
[51] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. 2019. Dag-gnn: Dag structure learning with graph neural networks. arXiv:1904.10098. Retrieved from https://arxiv.org/abs/1904.10098.
[52] Changhe Yuan, Brandon Malone, and Xiaojian Wu. 2011. Learning optimal Bayesian networks using A* search. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*.
[53] Huaijian Zhang, Heather L. Benz, Anastasios Bezerianos, Soumyadipta Acharya, Nathan E. Crone, Anil Maybhate, Xiaoxiang Zheng, and Nitish V. Thakor. 2010. Connectivity mapping of the human ECoG during a motor task with a time-varying dynamic Bayesian network. In *Proceedings of the 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE, 130–133.

[54] Wentao Zhao, Erchin Serpedin, and Edward R. Dougherty. 2006. Inferring gene regulatory networks from time series data using the minimum description length principle. *Bioinformatics* 22, 17 (2006), 2129–2135.
[55] Xun Zheng, Bryon Aragam, Pradeep K. Ravikumar, and Eric P. Xing. 2018. DAGs with NO TEARS: Continuous optimization for structure learning. In *Proceedings of the Advances in Neural Information Processing Systems*. 9472–9483.
[56] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software* 23, 4 (1997), 550–560.

Received June 2021; revised December 2021; accepted February 2022