Many applications naturally involve time series data and the vector autoregression (VAR), and the structural VAR (SVAR) are dominant tools to investigate relations between variables in time series. In the first part of this work, we show that the SVAR method is incapable of identifying contemporaneous causal relations for Gaussian process. In addition, least squares estimators become unreliable when the scales of the problems are large and observations are limited. In the remaining part, we propose an approach to apply Bayesian network learning algorithms to identify SVARs from time series data in order to capture both temporal and contemporaneous causal relations, and avoid high-order statistical tests. The difficulty of applying Bayesian network learning algorithms to time series is that the sizes of the networks corresponding to time series tend to be large, and high-order statistical tests are required by Bayesian network learning algorithms in this case. To overcome the difficulty, we show that the search space of conditioning sets d-separating two vertices should be a subset of the Markov blankets. Based on this fact, we propose an algorithm enabling us to learn Bayesian networks locally, and make the largest order of statistical tests independent of the scales of the problems. Empirical results show that our algorithm outperforms existing methods in terms of both efficiency and accuracy.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications—Data Mining
General Terms: Algorithms, Performance
Additional Key Words and Phrases: Bayesian networks, VAR, causal modeling, graphical models

1. INTRODUCTION
The VAR model is widely used for analyzing relations between variables in time series data since the pioneering works [Sims 1980]. This method infers the relations between variables by regressing the current observation on past observations, and interprets the autoregression coefficients as causal effects. There are two main drawbacks of this method. The first one is that the VAR method provides no information on contemporaneous causal order. The second one is that least square estimators become unreliable when the dimensions of time series are large and the number of samples are limited. To capture both temporal and contemporaneous causal relations, the SVAR model is proposed that estimates contemporaneous coefficient matrices by orthogonalizing the VAR residuals [Sims 1986; Bernanke 1986; Blanchard and Watson 1987]. However, this paper shows that the SVAR model is incapable of deciding contemporaneous causal orders for Gaussian process. Recently, graphical models have been recognized useful on time series analysis, and Bayesian network learning algorithms have been used to identify the contemporaneous causal orders in SVARs [Demiralp and Hoover 2003; Chen and Chihying 2007]. The main idea of these works is using the least squares approach to estimate a VAR, and then applying Bayesian network learning algorithms to the VAR residuals to identify contemporaneous causal orders. This approach can solve the contemporaneous causal order problem but relies on the quality of the estimation of VARs. Therefore, the difficulty of unreliable least squares estimators involved by high-dimensional time series still exists in these works.

In this work, we provide a deep analysis on the reason that the SVAR model is incapable of identifying the contemporaneous causal orders for Gaussian process. We show that the incapability is because the SVAR model fails to discover the structure that is faithful to the underlying distribution. Because Bayesian network learning algorithms can be employed to identify the faithful structure to the underlying distribution, in the remaining part of the work we propose a solution of applying Bayesian network learning algorithms to mine causal relations in time series. In our approach, the Bayesian network learning algorithm is applied to time series data instead of the VAR residuals, and temporal and contemporaneous causal relations are considered in a unified way. The difficulty of doing this is determining how to find a robust Bayesian network learning algorithm to deal with large size Bayesian networks. The reason is as follows. To represent a VAR model by a Bayesian network, the number of the vertices in the Bayesian network is several times larger than that of i.i.d.
data, because the past observations need to be considered. For example, to represent a second order VAR with 10 variables, there are 30 vertices in the Bayesian network. To decide whether there exists an edge between two vertices, the Bayesian network learning algorithms need to search all subsets of the entire vertex set to find a set of vertices d-separating these two vertices, and the results of CI tests serve as indicators of d-separation. Large networks pose a difficulty on Bayesian network learning algorithms because of unreliable high-order CI tests [Ramsey et al. 2006].

The key step of using Bayesian network learning algorithm for causal mining in time series is determining how to avoid high-order CI tests even when the sizes of the underlying Bayesian networks are large. This work shows that to infer the structures of Bayesian networks, it is sufficient to test conditional independence conditioned on all subsets of Markov blankets, so that the largest order of CI tests does not depend on the size of the network, but on the sizes of Markov blankets. With this results, determining how to avoid high-order CI tests when learning Bayesian networks is equivalent to determining how to identify Markov blankets without involving high-order CI tests.

In this work, we apply the Incremental Association Markov Blanket (IAMB) algorithm [Tsamardinos et al. 2003b] to learn Markov blankets from data. The heuristic we use in this work is partial correlation, and the IAMB algorithm in this case is equivalent to the forward-backward linear regression [Draper and Smith 1966; Hocking 1976]. The IAMB algorithm enables us to identify Markov blankets of each vertex in a Bayesian network without involving high-order CI tests.

The contributions of this work are summarized as follows: (i) this work formally shows that the reason of the SVAR model being incapable of identifying contemporaneous causal order for Gaussian process is that faithfulness condition is violated under the SVAR model; (ii) this work provides a graph-theoretic method estimating SVARs and VARs, which avoids high-order regressions; (iii) this work formally proves that in Bayesian network learning if there exists a set of vertices d-separating two vertices, such a set can always be found within a subset of the Markov blanket of either of the given two vertices. The rest of the paper is organized as follows. Section 2 reviews previous works related to ours. Section 3 clarifies the problem definition and several technical preliminaries. Section 4 discusses the reason that the SVAR model is incapable of identifying contemporaneous causal order and the necessity of Bayesian network learning algorithms. Section 5 provides a theoretical analysis on using Markov blanket information to search conditioning sets, and explains how to use our heuristic Bayesian network learning algorithm to estimate SVARs and VARs. Section 6 presents some empirical comparison of our algorithm and the existing algorithms. Section 7 concludes our paper with a summary of our finds.

2. RELATED WORKS

Related works in the VAR community. The VAR model has been the focus of interest in economic time series analysis since [Sims 1980], followed by a series of varieties. One of the varieties related to causal relation mining is the SVAR model [Sims 1986; Bernanke 1986; Blanchard and Watson 1987]. In the SVAR model, contemporaneous correlations are assumed to be interpreted causally. One typical solution of this type of methods is to estimate a standard VAR model, and then orthogonalize the covariance matrix of the residuals using the Cholesky decomposition. However, this work will show that under the SVAR framework, all the contemporaneous causal orders are equivalent for Gaussian processes, and the faithfulness condition is violated. The method presented in [Swanson and Granger 1994] requires prior knowledge of the contemporaneous causal orders to identify the true SVAR models.

Related works in the Bayesian network learning community. Bayesian network learning algorithms have been widely used for causal discovery since the pioneering works [Spirtes et al. 1990; Pearl and Verma 1991]. To reduce computational complexity, the Peter and Clark (PC) algorithm [Spirtes and Glymour 1991] is a representative one that has polynomial time complexity when Bayesian networks are sparse, but not efficient when networks are relatively dense. Markov blanket information is used by Bayesian network learning algorithms to reduce computational complexity and avoid high-order CI tests. The algorithms using Markov blanket information
are the Grow-Shrink (GS) algorithm, [Margaritis and Thrun 1999], the Max-Min Hill-Climbing (MMHC) algorithm [Tsamardinos et al. 2006], and the Total Conditioning (TC) algorithm [Pellet and Elisseeff 2008]. The GS algorithm learns the Markov blankets of variables by a step-wise procedure, and uses the Markov blanket information obtained to limit the sizes of conditioning sets when inferring the structures of Bayesian networks. The Bayesian network learning algorithm in this work can be viewed as an enhanced version of the GS algorithm in the sense that we show that the search space of conditioning sets can be reduced to the subsets of the Markov blankets. The MMHC algorithm uses the Min-Max Parents and Children (MMPC) algorithm [Tsamardinos et al. 2003a] to identify the parents and the children of each variable from data. However, identifying the parents and the children of a variable is a more difficult task than identifying the Markov blanket, because identifying all the spouses of a variable requires an exponential number of CI tests. The TC algorithm uses high-order CI tests to obtain Markov blanket information. In addition, the TC algorithm does not use Markov blanket information explicitly. The Two-Phase algorithm [Wang and Chan 2010], an improved version of the Three-Phase Dependency Analysis (TPDA) algorithm [Cheng et al. 2002], is our first attempt to learn Bayesian networks from Markov random fields. In this work, we improve this basic version by using Markov blanket information.

**Related works combining VAR models and causal mining models.** Bayesian network learning algorithms are used for identifying the contemporaneous causal order of a SVAR[Demiralp and Hoover 2003; Chen and Chihying 2007]. In [Demiralp and Hoover 2003], the authors apply the PC algorithm to search the contemporaneous causal order in the residual covariance matrix. In Time Series Causal Models (TSCMs) [Chen and Chihying 2007], a score-based Bayesian network learning algorithm is used to discover the causal structures in SVARs. Besides Bayesian network learning algorithms, the Linear Non-Gaussian Acyclic Model (LiNGAM) algorithm [Shimizu et al. 2006], a causation mining algorithm based on independence component analysis, has been applied for deciding the causal orders in SVARs [Hyvarinen et al. 2008] and in that work the authors show that ignoring the contemporaneous causal effects can lead to wrong estimation of temporal causal relations. All these works need to estimate VAR models first and then apply causation mining algorithms to the residual covariance matrices. Therefore, these methods are incapable of overcoming the difficulty that least squares estimators are unreliable when the scales of the problems are large.

**Recent works in causal reference for time series.** A new regression approach for causal inference is proposed by [Mooij et al. 2009], named Hilbert-Schmidt Independence Criterion HSIC regression. This regression approach minimizes the statistical dependence between regressors and residuals, rather than minimizing squared errors as the standard linear regression does. The algorithm proposed by [Henao and Winther 2009] learns factor models and directed acyclic graphs in a unified framework. We call the algorithm FactorDAG in this paper. This algorithm is close related with VAR and Bayesian network learning, because VARs can be view as factor model with time structure, and Bayesian networks are directed acyclic graphs representing certain probability distributions. Another algorithm for identifying the causal direction of time series is proposed by [Peters et al. 2009]. This algorithm models time series as causal autoregressive moving average process, and the causal direction of the time series is determined by the dependence between regression residuals and the time series itself. However, this algorithm is incapable of dealing with contemporaneous causal orders. The algorithm proposed by [Lozano et al. 2009; Yan ] extends the concept of Granger causality [Granger 1980] to learn temporal causal relations in time series data. Because the limitation of Granger causality, this algorithm can only deal with two univariate time series at one time, and for multivariate time series, it learns the causal relations among data pairwise. Another disadvantage of this algorithm is the incapability of finding contemporaneous causal orders.
3. PRELIMINARIES

Notations. In the remainder of this paper, graphs are denoted by calligraphic letter $G$. Boldface capitals $X, Y, Z$ denote sets of random variables or vertices in a network. $V$ is the set of all variables in the analysis. Italicized capitals like $X,Y,Z$ are designated either constant matrices or random variables or the vertices in $V$, depending on the context. Vectors are set in boldface lower case, as $a, b$: scalars in italics, as the number of samples $n$ or the order of a VAR $k$. An undirected edge between vertices $X$ and $Y$ is denoted by $X \rightarrow Y$ and directed edge by $X \rightarrow Y$ or $X \leftarrow Y$, depending on the orientation of the edge.

3.1. VARs and SVARs

We denote an observed time series by $X_t(t) = x_i(t)$ for $1 \leq i \leq n$ and $1 \leq t \leq m$, where $X_t(t)$ denotes random variable, $x_i(t)$ is the corresponding observation, $i$ is the index of random variables and $t$ is the index of time. The collection of all the variables at time $t$ is denoted by $n \times 1$ vector $X(t)$ and the observations $x(t)$. The dynamic of the time series under the VAR is defined as

$$X(t) = \sum_{\tau=1}^{k} A_{\tau} X(t-\tau) + E(t)$$

where $k$ is the order of the VAR, $A_{\tau}$, $0 \leq \tau \leq k$, are $n \times n$ coefficient matrices, $E(t)$ are disturbances satisfying $E(t) \equiv E(s)$ for $1 \leq t, s \leq m, t \neq s$ and the covariance matrix of $E$ is $\text{Cov}[E] = \Sigma_E$. A VAR with order $k$ is denoted by VAR$(k)$. The SVAR assumes that the covariance matrix of disturbances can be interpreted as the contemporaneous causal structure, and the SVARs can be identified from VARs by applying Cholesky decomposition to the covariance matrix of the disturbances as follows:

$$\Sigma_E = LL'$$

where $L$ is a lower triangular matrix with strictly positive diagonal entries and $L'$ denotes the transpose of $L$. And for each $E(t)$, find $W(t)$ satisfying

$$E(t) = LW(t)$$

Plugging equation (3) into equation (1) and rearranging the formula, we get

$$X(t) = \sum_{\tau=0}^{k} B_{\tau} X(t-\tau) + W(t)$$

where $B_0 = I - L^{-1}$ and $B_1 = L^{-1} A_{\tau}$. And from equation (2) and equation (3), we know that

$$\text{Cov}[W] = L'^{-1} \Sigma_E (L^{-1})' = I$$

The VAR model identified by orthogonalizing the contemporaneous variables in equation (4) is known as a SVAR. The coefficient matrices of the VAR in equation (1) can be estimated from data by the least squares method as follows [Richard and Wichern 2002]. Denote the observations by concise matrix notations:

$$Y = [x(k+1) \cdots x(m)]$$

$$Z = \begin{bmatrix}
  x(k) & x(k+1) & \cdots & x(m-1) \\
  x(k-1) & x(k) & \cdots & x(m-2) \\
  \vdots & \vdots & \ddots & \vdots \\
  x(1) & x(2) & \cdots & x(m-k)
\end{bmatrix}$$

The least squares estimators of the coefficient matrices are given by the following formula:

$$\hat{A} = [\hat{A}_1 \ldots \hat{A}_k] = YZ'(ZZ')^{-1}$$
and the regression residuals are calculated by the formula:

$$\hat{e}(t) = x(t) - \sum_{\tau=1}^{k} \hat{A}_x(t - \tau)$$  \hspace{1cm} (9)

The drawback of using the least squares approach to estimate the structures of VARs is that when the dimensions of VARs are large and sample sizes are limited, the estimate coefficients matrices become unreliable. Even worse, when the number of samples is smaller than the production of the dimension of the VAR and the order of the VAR, namely $m < n \times k$, the matrix $ZZ^T$ becomes a singular matrix, the inverse of $ZZ^T$ does not exist, and the least squares approach fails. This is an important motivation for us to apply the Bayesian network learning algorithm directly to the time series data instead of the regression residuals.

### 3.2. Dependency analysis-based Bayesian network learning

A **Bayesian network** $G_B(V, E)$ is a directed acyclic graph used to represent a joint distribution over vertex set $V$ in which a single vertex is associated with a random variable in data set. In this work, we use vertex to denote the random variable in the domain and the corresponding vertex in the Bayesian network, and $E$ denotes the edge set of the Bayesian network. An edge from $V_i$ to $V_j$, for $V_i, V_j \in V$, indicates that $V_i$ is a direct cause of $V_j$. The joint distribution $p$ represented by a Bayesian network obeys the local Markov property that for any vertex $V \in V$, $V$ is conditionally independent of its non-descendants given its parents, denoted by $V \perp \!\!\!\!\perp \text{Nd}(V) | \text{Pa}(V)$. In the formula $\text{Nd}(V)$ is the set that consists of all non-descendants of $V$ and $\text{Pa}(V)$ is the set that consists of all parents of $V$. If we ignore the direction of all edges in $G_B(V, E)$ we get an undirected graph called the **skeleton** of $G_B(V, E)$, denoted by $\text{skeleton}(G_B(V, E))$.

Before proceeding, we provide standard definitions for adjacency path, collider, noncollider, d-separation and other important concepts related with dependency analysis Bayesian network learning.

**Definition 1** In a graph $G(V, E)$, an **adjacency path** connecting $V_0$ and $V_k$ is an ordered sequence of vertices $(V_0, V_1, ..., V_k)$ such that $V_i \rightarrow V_{i+1} \in E$ or $V_i \leftarrow V_{i+1} \in E$. An adjacency path is called a **simple adjacency path** if all the vertices contained in the path are distinct.

**Definition 2** In a Bayesian network $G_B(V, E)$, a vertex $V$ is a **collider** on a simple adjacency path $\pi$ if there are two distinct vertices adjacent to $V$ on $\pi$, and both of vertices are parents of $V$ in $G_B(V, E)$; otherwise, $V$ is called **noncollider** on $\pi$.

**Definition 3** In a Bayesian network $G_B(V, E)$, two vertices $U$ and $V$ are **d-separated** by a set of vertices $C$, denoted by $D_{sep}(U; V | C)$, if and only if every simple adjacency path connecting $U$ to $V$ is blocked by $C$. A simple adjacency path $\pi$ is **blocked** by $C$ if and only if at least one noncollider on $\pi$ is in $C$ or at least one collider and all of its descendants are not in $C$. Specifically, $U$ and $V$ can never be d-separated if there is an edge connecting $U$ and $V$. In this paper, we call the set $C$ the **conditioning set**.

Adjacency paths in a Bayesian network are different from directed paths. Take the Bayesian network in figure 1 as an example. From vertex $A$ to vertex $E$ there is a directed path $(A, C, E)$, which is also an adjacency path connecting vertex $A$ and vertex $E$. From vertex $E$ to vertex $A$, there is no directed paths but there is an adjacency path $(E, C, A)$. Note that a collider is defined by not only a vertex but also by simple adjacency path containing the vertex. In figure 1 , vertex $C$ is a collider on adjacency path $(A, C, B)$ but a noncollider on adjacency path $(A, C, E)$. Therefore, vertex $A$ and vertex $B$ are d-separated by $\emptyset$ while vertex $A$ and vertex $E$ are d-separated by $C$.  

Definition 4 A distribution $p$ is a Markov distribution with respect to a Bayesian network if $D_{sep}(U; V|C)$ implies $U \independent V|C$.

Definition 5 A distribution $p$ is faithful to a Bayesian network if $U \independent V|C$ implies $D_{sep}(U; V|C)$.

It has been proved that a distribution $p$ is a Markov distribution if and only if $p$ obeys local Markov property with respect to a Bayesian network [Lauritzen et al. 1990]. Thus, Markov property of Bayesian networks provides a connection between the structure of a Bayesian network and the independence implied. More specifically, the absence of an edge between two vertices guarantees a set of independence related with these two vertices. However, the reversed part, the existence of an edge, does not guarantees dependence between the endpoints of the edge given merely the Markov property. To guarantee the correctness, most Bayesian network learning algorithms assume that the distributions of data are not only Markov but also faithful. Faithfulness assumption is not strong in the sense that a Markov distribution on a Bayesian network satisfies the faithfulness condition almost surely [Spirtes et al. 1993]. In the remainder of the paper we assume that all distributions are both Markovian and faithful with respect to the corresponding Bayesian networks, which implies the equivalence between $D_{sep}(U; V|C)$ and $U \independent V|C$. The faithfulness assumption also implies that every edge in a Bayesian network can be identified by the absence of independence between the corresponding endpoints given any conditioning set. To illustrate this idea, we take figure 1 as an example. Suppose that the distribution over the vertex set of the Bayesian network in Figure 1 (b) is faithful. To check whether there is an edge between vertices $A$ and $B$, dependency analysis algorithms need to test independence between $A$ and $B$ conditioned on all subsets of the set $\{C, D, E\}$. If there is a conditioning set making $A$ and $B$ independent, there is no edge between these two vertices. There are $2^3 = 8$ such sets to be checked, and the largest one contains three vertices. From this case, we can see that the maximum size of conditioning sets increases as the dimension of the underlying data set grows if we use brute-force search. When data sets are high-dimensional and samples are limited, large conditioning sets cause unstable CI tests and impair the accuracy of learning algorithms. In addition, the number of CI tests grows in an exponential rate with the dimension of data, which makes brute-force search infeasible but only on small size networks.

![Diagram](image)

Fig. 1. An illustration of the dependency analysis approach inferring the structure of a Bayesian network by CI tests.

4. SVAR BEING INCAPABILITY OF IDENTIFYING CONTEMPORANEOUS CAUSAL ORDER FOR GAUSSIAN PROCESS

The information of the contemporaneous causal order of a time series is implied in the contemporaneous coefficient matrix of the true SVAR. The true SVAR is distinguished from the other SVARs by the fact that the disturbances are independent rather than uncorrelated. The true SVAR is assumed corresponding to the data-generating process [Demiralp and Hoover 2003]. We show that the procedure of identifying the true SVAR is equivalent to identifying the structure equation model (SEM) generating the disturbances of the original VAR. We use the same notations here as in section 3, and suppose that there exists a permutation matrix $P$ denoting the true contemporaneous causal order of the SVAR. The disturbances under the true contemporaneous causal order are given by

$$E_P(t) = PE(t)$$ (10)
Learning Bayesian Networks from Markov Random Fields: An Efficient Algorithm for Linear Models

Denote the diagonal matrix

\[ D = P \text{Diag}(\sigma_{11}^2, \sigma_{22}^2, ..., \sigma_{nn}^2)P' \]  

where \( \sigma_{ij}^2 \) denotes the variance of the \( i \)th entry in \( E(t) \). The covariance matrix of disturbances under the true contemporaneous causal order can be decomposed in accordance with Cholesky decomposition as follows

\[ \Sigma_{E_0} = P \Sigma_E P' = L_p L_p' = L_p D^{-\frac{1}{2}} DD^{-\frac{1}{2}} L_p' = MDM' \]  

where

\[ D^{-\frac{1}{2}} = P \text{Diag}(\frac{1}{\sigma_{11}}, \frac{1}{\sigma_{22}}, ..., \frac{1}{\sigma_{nn}})P', \quad M = L_p D^{-\frac{1}{2}} \]  

\( L_p \) is a lower triangular matrix got by applying Cholesky decomposition to \( \Sigma_{E_0} \). The main diagonal entries of \( M \) are the corresponding standard deviations of \( E_p \). We restrict the main diagonal entries of \( M \) by \( D^{-\frac{1}{2}} \) so that \( M \) is a lower triangular matrix with all the diagonal elements one. There exists a vector \( W_p(t) \) satisfying

\[ E_p(t) = L_p W_p(t) = L_p D^{-\frac{1}{2}} D\frac{1}{2} W_p(t) = MU_p(t) = (I - M^{-1})E_p(t) + U_p(t) = C_0 E_p(t) + U_p(t) \]  

where

\[ \text{Cov}[W_p(t)] = I, \quad C_0 = I - M^{-1}, \quad U_p(t) = D\frac{1}{2} W_p(t), \quad \text{Cov}[U_p(t)] = D\frac{1}{2} \text{Cov}[W_p(t)]D\frac{1}{2} = D \]  

The coefficient matrix \( C_0 \) is a strictly lower triangular matrix, because \( M^{-1} \) is a lower triangular matrix with all diagonal entries one. For the permutation matrix \( P \) corresponding to the true contemporaneous causal order, disturbances \( U_p \) satisfy \( U_{P_i} \perp U_{P_j} \) for \( 1 \leq i, j \leq n \) and \( i \neq j \); and in this case, equation (15) is the SEM generating the disturbances of the VAR in equation (1). Plugging equation (15) into equation (1) we have

\[ X_p(t) = \sum_{\tau=1}^{k} PA_{\tau}P'X_p(t-\tau) + E_p(t) = \sum_{\tau=0}^{k} C_0 X_p(t-\tau) + U_p(t) \]  

where

\[ X_p(t) = PX(t), \quad C_0 = M^{-1} PA_{P'}P', 1 \leq \tau \leq k \]  

Because the elements in \( U_p \) are mutually independent, the SVAR in equation (16) is assumed corresponding to the data-generating procedure. The strictly lower triangular coefficient matrix \( C_0 \) captures the contemporaneous causal order in this case. The non-zero elements \( c_{ij} \) in \( C_0 \) denotes that \( x_j \) is a contemporaneous cause of \( x_i \). The difficulty of analyzing contemporaneous causal order by identifying the data-generating process is that for a time series with dimension \( n \), there are \( n! \) possible orders, and under each of them we need to test independence of regression residuals. Even worse, when the given time series is a Gaussian process, the elements in residual \( U_p \) are mutually independent for arbitrary permutation \( P \), because uncorrelation is equivalent to independence for Gaussian variables. However, different \( P \) results in different contemporaneous coefficient matrix \( C_0 \) and makes contemporaneous causal order analysis fail. In this case, prior knowledge is required to resolve the confusion.

The concepts related to Bayesian network learning provide a deep insight of the reason that the SVAR model is incapable of identifying contemporaneous causal order for Gaussian process. We have shown that the SVAR model cannot identify the contemporaneous causal order for Gaussian processes, because any order will give mutually independent residuals \( U_p \). The SVAR model cannot distinguish these orders under which the faithfulness condition is satisfied from those under which the faithfulness condition is violated. The previous work shows that a distribution satisfying the Markov property of a Bayesian network is faithful to the network almost surely [Meek 1995], which means that with probability one the distribution of data is faithful to the Bayesian network.
corresponding to the true data-generating process. We illustrate the fact by an example in figure 2. In figure 2, the disturbance $E = (E_1, E_2, E_3)'$ is generated by the SEM in the left, and all $U_i$, $i = 1, 2, 3$, are drawn from standard Gaussian distribution and mutually independent. There is no causal effect between disturbances $E_1$ and $E_2$, and both $E_1$ and $E_2$ are the causes of $E_3$. The distribution of $E$ is faithful to the Bayesian network in figure 2, because $E_1 \perp\!\perp E_2$ implies that $E_1$ and $E_2$ are d-separated by empty set, which means that $E_1$ and $E_2$ are not directly connected in the network. The true permutation matrix $P$ and the corresponding parameters are

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad C_0^P = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}, \quad M^P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{Cov}[U_P] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (18)$$

Another permutation matrix $Q$ and the corresponding parameters are given as follows:

$$Q = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad C_0^Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad M^Q = \begin{bmatrix} 1 & 0 & 0 \\ -1 & -1 & 10 \\ -1 & 1 & 1 \end{bmatrix} \quad \text{Cov}[U_Q] = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (19)$$

For both permutation $P$ and $Q$, the elements in residual $U$ are mutually independent, and the SVAR model cannot tell which one corresponds the true data-generating procedure. It is obvious that the faithfulness condition is violated under permutation $Q$. The reason is as follows. There does not exist causal effect between $E_1$ and $E_2$, because these two disturbances are independent of each other. However, the regression coefficient between $E_1$ and $E_2$ under permutation $Q$, in this case $c_{032}^Q$, the entry in the third row and second column of $C_0^Q$, is not zero, which means that $E_1$ and $E_2$ are connected by an edge in the Bayesian network corresponding to permutation $Q$, and in this case $E_1$ and $E_2$ cannot be d-separated. The independence of $E_1$ and $E_2$ is the result of special regression coefficients. More specifically, for each pair of mutually independent disturbances $E_i$ and $E_j$, $i < j$, the SVAR model always finds the coefficient matrix $C_0^Q$ making

$$\text{Cov}[E_i, E_j] = \sum_{k=1} m_{ik} m_{jk} \text{Var}[U_k] = 0 \quad (20)$$

where $\text{Cov}[E_i, E_j]$ is the covariance between $E_i$ and $E_j$, $m_{ik}$ is the entry in the $i$-th row and $k$-th column of $M$. From equation (16) we know that $m_{ik}$ is a function of $C_0$. For this reason, the SVAR model cannot tell the coefficient matrices corresponding to the faithful structures from the coefficient matrices satisfying dependency relations by choosing special regression coefficients. The Bayesian network learning algorithms always find the faithful structures by CI tests. Given the fact that $E_1$ and $E_2$ independent, the corresponding entry in the coefficient matrix is restricted to be zero, so that coefficient matrix $C_0^Q$ is ruled out. In addition, Bayesian network learning algorithms identify the contemporaneous causal order by finding all the V-structures [Spirtes et al. 1993]. If the two parents of a collider are not connected by an edge, the collider and its two parents form a V-structure in a Bayesian network. In the Bayesian network in figure 2, the vertices $E_1$, $E_2$ and $E_3$ form a V-structure, and the Bayesian network learning algorithms identify it by the results that $E_1$ and $E_2$ are independent of each other conditioned on the empty set and dependent of each other conditioned on $E_3$. 

![Diagram](image-url)

Fig. 2. An illustration of a violation of faithfulness condition in VAR model
5. ESTIMATING THE SVARS BY BAYESIAN NETWORK LEARNING ALGORITHM

In this section, a novel algorithm for discovering VARs and SVARs from data is presented. The algorithm is sound under the following assumptions. (i) the coefficient matrices of SVARs are time-invariant. In other words, the VARs and SVARs in this work are stationary, which is a typical assumption for many VAR approaches and causal analysis [Dahlhaus 2000; Demiralp and Hoover 2003; Moneta and Spirtes 2006; Chen and Chihying 2007; Hyvarinen et al. 2008; Haufe et al. 2010]. (ii) the coefficient matrix $C_0$ corresponds to a directed acyclic graph. Namely, there exists a permutation matrix $P$, so that $PC_0P'$ is a strictly lower triangular matrix, which is a necessary assumption if we use Bayesian network learning algorithm in causal analysis [Demiralp and Hoover 2000; Demiralp and Hoover 2003; Shimizu et al. 2006; Moneta and Spirtes 2006; Chen and Chihying 2007; Hyvarinen et al. 2008], (iii) there exist reliable statistical tests of conditional independence. In practice, it requires sufficient samples to conduct reliable CI tests. However, reliable CI tests are commonly assumed in the theoretical correctness analysis of Bayesian network learning [Margaritis and Thrun 1999; Cheng et al. 2002; Tsamardinos et al. 2006; Pellet and Eliseeff 2007].

5.1. Represent SVARs by Bayesian networks

One of the distinction between this work and previous works using Bayesian network learning algorithms to learn SVARs is that in this work, the Bayesian network learning algorithm is applied to the time series data instead of regression residuals. To represent a SVAR($k$) by a Bayesian network, the vertex set should contain the current observation and the previous $k$ observations. Each vertex in the network corresponds a random variable $X_i(t)$ for $1 \leq i \leq n$ and $0 \leq t \leq k$, where $n$ denotes the dimension of time series and $X_i(t)$ denotes the $i$th element of $X(t)$. To represent a $n$-dimensional SVAR($k$), there are $n \times (k+1)$ vertices in the corresponding Bayesian network. There is an edge from $X_j(t-s-\tau)$ to $X_i(t-s)$ if and only if $C_{i,j} \neq 0$ for $0 \leq s \leq k-\tau$. The reason is that the SVARs considered in this work have time-invariant coefficient matrices, which implies that the causal relations between variables are also time-invariant. The non-zero entry $c_{i,j}$ indicates that $X_j(t-\tau)$ is a cause of $X_i(t)$, which also implies that $X_j(t-s-\tau)$ is a cause of $X_i(t-s)$ for all $s > 0$.

Consider a SVAR(2) with following coefficient matrices:

$$
C_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0.3 & 0 \end{bmatrix} \quad C_1 = \begin{bmatrix} 0.3 & 0 & 0.7 \\ 0 & 0 & 0 \\ 0.6 & 0.2 & 0 \end{bmatrix} \quad C_2 = \begin{bmatrix} 0 & 0.4 & 0 \\ 0.5 & 0 & 0.3 \\ 0 & 0 & 0 \end{bmatrix}
$$

The Bayesian network corresponding to this time series is given in figure 3. The dimension of the time series is 3 and the order of the SVAR is 2. Therefore, there are $3 \times (2+1) = 9$ vertices in the network. There is an edge from $X_1(t)$ to $X_2(t)$ since $c_{0,2}(2,1) = 0.5$, which also implies an edge from $X_1(t-1)$ to $X_2(t-1)$ and an edge from $X_1(t-2)$ to $X_2(t-2)$. Time series data pose a challenge on Bayesian network learning, because the sizes of the networks grow with both the dimensions and the orders of time series. The difficulty motives us to address a Bayesian network learning algorithm that has good performance on large networks.

![Figure 3](image)

Fig. 3. An illustration of representing time series by Bayesian network.
5.2. Basic Two-Phase Algorithm

To avoid high-order CI tests, dependency analysis algorithms restrict the search space of conditioning sets. More specifically, suppose that in a Bayesian network $G_B$, there is no edge between two vertices $U$ and $V$. If it can be proved that we can always find a conditioning set $d$-separating $U$ and $V$ by choosing vertices only from a subset of the entire vertex set, the search space is narrowed down from the entire vertex set to this particular subset. One of such subsets commonly used consists of all the vertices on the simple adjacency paths connecting $U$ and $V$ in $G_B$ excluding $U$ and $V$ themselves, denoted by $\text{ADJ}(U, V, G_B)$. The reason is that according to the definition of $d$-separation, only the vertices on a simple adjacency paths can block that path. Notice that the set $\text{ADJ}(U, V, G_B)$ can be an empty set, and this happens when there is no simple adjacency path connecting $U$ and $V$ or the only simple adjacency path connecting $U$ and $V$ is an edge. Take Figure 4 as an example.

In the Bayesian network in Figure 4 (b), to decide that there is no edge between vertices $A$ and $B$, it is sufficient to show that either $A \perp B | (C)$ or $A \perp B | \emptyset$ is true, because $\text{ADJ}(A, B, G_B) = \emptyset$. We also have $\text{ADJ}(A, C, G_B) = \emptyset$, because the only simple adjacent path connecting $A$ and $C$ in the network is $(A, C)$. The exact $\text{ADJ}(A, B, G_B)$ can be identified only when we know the structure of $G_B$. Therefore, we need an estimation of the underlying network, say $\tilde{G}_B$, based on which we figure out an estimated $\text{ADJ}(A, B, \tilde{G}_B)$ denoted by $\text{ADJ}(A, B, G_B)$. Correctness guarantee requires that $\text{ADJ}(A, B, G_B)$ is a subset of $\text{ADJ}(A, B, \tilde{G}_B)$, which implies that $\tilde{G}_B$ needs to contain all the true edges in $G_B$. Existing algorithms employing $\text{ADJ}$ information such as $\text{PC}$ uses estimated skeletons satisfying this condition. Phase two of the $\text{TPDA}$ algorithm adds edges into the skeleton because the maximum spanning trees obtained in phase one do not satisfy this condition.

![Fig. 4. An illustration of similarity between Markov random field and Bayesian network. The Markov random field (a) contains all the edges in the Bayesian network (b) and a redundant edge [A,B]](image)

The Two-Phase algorithm, denoted by $\text{Basic2P}$ in the remainder of the paper, is an improved version of $\text{TPDA}$. In phase one of $\text{TPDA}$, the algorithm gets the initial skeletons by constructing maximum spanning trees. Phase two of $\text{TPDA}$ adds missing edges into the initial skeletons and phase three of $\text{TPDA}$ removes redundant edges. The general idea of the $\text{Basic2P}$ algorithm comes from the observation that the structures of Bayesian networks are similar to those of Markov random fields. A Markov random field $G_M(V, E)$ is an undirected graph used to represent a joint distribution over vertex set $V$, in which a single vertex is associated with a random variable in the data set. Moreover, $E$ denotes the edge set of the Markov random field. The joint distribution $p$ represented by a Markov random field obeys the local Markov property that for any vertices $U, V \in V, U$ and $V$ are conditional independent of each other given a set of vertices $C$ if $C$ separates $U$ and $V$ in $G_M(V, E)$. With the concepts above, the $\text{Basic2P}$ algorithm works as follows:

— **Phase One** constructs a Markov random field $G_M(V, E)$ from data in accordance with the Markov property. That is, for each pair of vertices $U$ and $V$ in vertex set $V$, $\text{Basic2P}$ does not add edge $U - V$ into $G_M(V, E)$ if and only if $U \perp V | V \setminus \{U, V\}$.

— **Phase Two** removes redundant edges from $G_M(V, E)$. For each edge $U - V$ in $G_M(V, E)$, if there is a set of vertices $C \subseteq \text{ADJ}(U, V, G_M)$ making $U \perp V | C$, $\text{Basic2P}$ removes $U - V$. After all edges in $G_M(V, E)$ have been checked, $\text{Basic2P}$ orients the remaining edges.
Phase one of Basic2P obtains the initial estimations of the skeletons of Bayesian networks by learning Markov random fields from data. The advantage of using Markov random fields as initial skeletons is that Markov random fields theoretically contain all the correct edges in the corresponding Bayesian networks. Therefore, phase two of Basic2P can focus on how to remove those redundant edges by searching conditioning sets with ADJ information obtained from Markov random fields. In addition, Markov random fields contain much less redundant edges than the complete graphs used by PC as trivial estimations of underlying networks. An example is given by Figure 4. The Markov random field in (a) and the Bayesian network in (b) are defined on the same set of random variables, and the former contains all the edges in the latter. With the Markov random field obtained in phase one, phase two only needs to consider the set ADJ(A, B) to remove A − B, in this case {C}. That is, phase two only needs to test $A \perp B|\emptyset$ and $A \perp B|\{C\}$.

However, when dealing with high-dimensional data sets, Basic2P suffers from the increase of the sizes of the networks. The reason is as follows. Phase one of Basic2P learns Markov random fields according to Markov property. To be specific, for each pair of vertices, Basic2P needs to test independence conditioning on the entire vertex set, excluding the two under testing, to decide if there is an edge. Take the Markov random field in Figure 4 (a) as an example. To decide whether there is an edge between A and B, Basic2P tests if $A \perp B|\{C, D, E\}$ is true. Such tests are unstable when the sizes of the conditioning sets are large and samples are limited. Another point can be improved is that phase two of Basic2P does not make use of Markov blanket information obtained in phase one, and that information shown in this work is useful for reducing the search space of conditioning sets.

### 5.3. Markov Two-Phase Algorithm

In this section, we address an improved Two-Phase Algorithm with Markov Blanket information, named Markov2P, in order to overcome the difficulties faced by Basic2P. To avoid independence tests conditioned on large sets of variables, we use the IAMB algorithm [Tsamardinos et al. 2003b] to replace the original phase one. In phase two we use Markov blanket information to reduce the search space of conditioning sets. These two modifications are based on the concept of Markov blanket. In a Bayesian network, Markov blanket of a vertex A is defined as the set of vertices composed of its parents, children and spouses, denoted by MB(A). The heuristic Markov random field learning algorithm is based on the fact that all and only vertices in MB(A) are adjacent to A in a Markov random field. An example of Markov blanket is given in figure 5. In the Bayesian network (b), vertices from $M_0$ to $M_6$ are in MB(A), and in Markov random field (a), all these vertices are adjacent to A. Therefore, learning a Markov random field from data can be done by identifying Markov blanket for each vertex in the Markov random field by the IAMB algorithm. The heuristic we use for IAMB is partial correlation. We use $\rho_{XYZ}$ to denote partial correlation between random variables X and Y after eliminating the effect of the conditioning set Z. The intuition justification of the heuristic is that zero partial correlation indicates conditional independence in linear models [Spirtes et al. 1998; Baba et al. 2004]. The idea of IAMB is as follows. The vertices in MB(X) can never be independent of X conditioned on the other vertices in MB(X), and tend to have larger partial correlation coefficients with X than those vertices not in MB(X). Following this idea, the searching procedure initializes Candidate Markov Blanket of vertex X, abbreviated as CMB(X), according to the current edge set and keeps on adding vertex with largest partial correlation coefficient into CMB(X). The procedure IAMB is phase one of the Markov2P that learns Markov random fields from data. The IAMB with partial correlation heuristic is equivalent to the stepwise regression [Draper and Smith 1966; Hocking 1976].

After IAMB, phase two of Markov2P removes redundant edges from Markov random fields to recover the skeletons of Bayesian networks. For most dependency analysis approaches, removing redundant edges is the most difficult part. The reason is that to remove an edge from a network we need to find a set of vertices conditioned on which the two endpoints of the edge are independent and the search space is composed of all subsets of the entire vertex set. In Basic2P, the search space is restricted to the set of vertices on the simple adjacency paths connecting the endpoints. In
Fig. 5. An illustration of Markov blanket: \( \text{MB}(A) \) contains vertices \( M_0 \) to \( M_6 \) in Bayesian network (b); In Markov random field (a), vertices in \( \text{MB}(A) \) are all neighbors of \( A \).

ALGORITHM 1: IAMB

Procedure: IAMB \((\mathcal{D}, k)\)

Data: Time series data set \( \mathcal{D} \)

Result: Markov random field \( \mathcal{G}_M(\mathcal{V}, \tilde{\mathcal{E}}) \)

begin

initialize a Markov random field \( \mathcal{G}_M(\mathcal{V}, \tilde{\mathcal{E}}) \) by mapping each \( X \in \mathcal{V} \) to a random variable in \( \mathcal{D} \) or one of its \( k \) lags and setting \( \tilde{\mathcal{E}} = \emptyset \);

foreach \( X \in \mathcal{V} \) do

\( \text{MB}(X) = \{ V \mid V \in \mathcal{V} \land \{ X, V \} \in \tilde{\mathcal{E}} \} \);

repeat

\( Y = \arg\max_{Y \in \mathcal{V} \setminus \text{MB}(X)} \rho_{XY} \cdot \text{MB}(X) \);

if \( \rho_{XY} \cdot \text{MB}(X) \neq 0 \) then

\( \text{MB}(X) = \text{MB}(X) \cup \{ Y \} \);

end

to until \( \text{MB}(X) \) has not changed;

foreach \( Z \in \text{MB}(X) \) do

if \( \rho_{XZ} \cdot \text{MB}(X) \) is 0 then

\( \text{MB}(X) = \text{MB}(X) \setminus \{ Z \} \);

end

eend

add \( \{ X, Z \} \) into \( \tilde{\mathcal{E}} \) for all \( Z \in \text{MB}(X) \);

eend

end

Markov2P. we show that the search space of conditioning sets can be further reduced. To be more precise, we prove that for any two vertices \( X \) and \( Y \), if there exists a set of vertices conditioned on which \( X \) and \( Y \) are independent, we are always capable to make the two vertices independent by choosing vertices only from either \( \text{MB}(X) \) or \( \text{MB}(Y) \) instead of the entire vertex set. In addition, Markov blanket information can be combined with \( ADJ \) information to further reduce the search space because, to \( d \)-separate two vertices, it never needs vertices off the simple adjacency paths connecting them. We illustrate this idea by figure 5, and the formal proof is given by theorem 1. Consider the redundant edge \( A \rightarrow M_0 \) of the Markov random field in (a), which should be removed in phase two. Phase two of Basic2P searches conditioning sets from vertices in \( ADJ(\{ A, M_0, G_M \}) \), which is composed of the three vertices under the colored rectangular: \( M_4, M_6 \) and a grey vertex. Phase two of Markov2P reduces the search space to \( ADJ(\{ A, M_0, G_M \}) \cap \text{MB}(A) \), which contains only \( M_4 \) and \( M_6 \). In this case, we exclude one vertex from the search space and cut the size of the searching space by half because the algorithm has to go through all possible subsets in it. On large networks this improvement can be significant. Another advantage is that by reducing the size of the search space, Markov2P has less chance to test independence conditioned on large sets, which
ALGORITHM 2: Two-Phase algorithm with Markov Blanket information

Procedure: Markov2P (D, k)

Data: Time series data set D

Result: The complete PDAG \( \mathcal{G}_E(V, E) \) of underlying Bayesian network \( \mathcal{G}_B \)

begin

IA2MB (D, k); // This single line is Phase I
// Phase II starts from here
initialize \( E = \hat{E}; \)

foreach edge \( (X, Y) \in E \) do

MB(X) = neighbors of \( X \) in \( \mathcal{G}_M(V, \hat{E}) \);
MB(Y) = neighbors of \( Y \) in \( \mathcal{G}_M(V, \hat{E}) \);
S = smaller set of MB(X) and MB(Y);
S = ADJ(X, Y) \( \cap \) S;
// S is the search space of conditioning sets
if \( \exists C \subseteq S.p_{XY|C} = 0 \) then

remove \( (X, Y) \) from \( \hat{E}; \)

foreach \( V \in S \setminus C \) do

if \( V \in MB(X) \setminus MB(Y) \) then

orient \( V \) as a child of both \( X \) and \( Y; \)
end

end

end

// Edge orientation starts from here
orient edges form lags to the variables at the current time;
perform constraint propagation rules;

end


makes estimation more stable. Through CI tests with Markov blanket information, phase two of Markov2P checks every edges in the Markov random fields and removes redundant ones. Finally, Markov2P orients edges in the skeleton by identifying all colliders in the network. For two vertices \( X \) and \( Y \), if there exists an edge between \( X \) and \( Y \) in the Markov random field, and the edge is excluded from Bayesian network, \( X \) and \( Y \) must be spouses in the underlying Bayesian network and the children of both \( X \) and \( Y \) must be included in \( S = ADJ(X, Y, \mathcal{G}_M(V, \hat{E})) \cap MB(X) \cap MB(Y) \). In this case, if there exists a set \( C \subseteq S.p_{XY|C} = 0 \), the children of both \( X \) and \( Y \) cannot be in \( C \) and are therefore in \( S \setminus C \). After orienting edges in the skeleton by identifying all the V-structures in the network, Markov2P poses the directions of the undirected edges from lags to the current variables. For the remaining undirected edges, constraint propagation rules used in the MMHC algorithm are applied, which orients edges by preserving acyclicity in a Bayesian network. Without other information, an edge orienting algorithm cannot distinguish Bayesian networks in a Markov equivalent class, which contains Bayesian networks with same skeletons and V-structures. Therefore, Markov2P outputs the complete partial directed acyclic graph (PDAG) which represents the Markov equivalent class of the true Bayesian network as PC and MMHC did. The correctness of Markov2P is guaranteed by the following theorem.

**Theorem 1** For any two vertices \( X \) and \( Y \) in a Bayesian network, if there is no edge between vertices \( X \) and \( Y \), we can always construct a conditioning set d-separating \( X \) and \( Y \) by choosing vertices only from either \( MB(X) \) or \( MB(Y) \).

**Proof:** Without loss of generality, we prove that there exists a subset of \( MB(X) \) d-separating \( X \) and \( Y \) if edge \( X \rightarrow Y \) and \( X \leftarrow Y \) do not belong to \( \mathcal{G}_B \). We first prove that the simple adjacency paths between \( X \) and \( Y \) unblocked, given initial conditioning set \( C = MB(X) \), are those containing
only a collider, which is a child of both X and Y. For any simple adjacency path connecting X and Y in $G_p$, and containing only one vertex between X and Y, this vertex is in C. If this vertex is a collider on the path, the path is unblocked given C, otherwise blocked. For any simple adjacency path connecting X and Y and containing more than one vertices between X and Y, if the vertex adjacent to X on the path is a noncollider, since the vertex is in C, C blocks the path. If the vertex adjacent to X is a collider, the successive vertex on the path must be a noncollider on that path and a spouse of X. Therefore, this spouse of X must be included in C, which makes C block the path. This proves that the only unblock paths are those containing only the children of both X and Y. Based on the result, we show that we can remove the children of both X and Y from C so that it d-separates X and Y. To this end we need to prove that we can remove all the descendants of both X and Y from C and keep original blocked paths remained blocked. If removing a descendant of both X and Y, say Z, unblocks certain simple adjacency path connecting X and Y, Z must be a noncollider on that path which implies that there must exist a minimum descendant of Z, say W, on that path. W must be a collider on that path. Otherwise there would be a cycle in the Bayesian network, which is a contradiction. In addition W is also a descendant of both X and Y, so after removing all the descendants of both X and Y, the path is still blocked by W. This completes the proof.

Corollary 2 For any two vertices X and Y in a Bayesian network, if there is no edge between vertices X and Y, we can always construct a conditioning set d-separating X and Y by choosing vertices only from either $\text{ADJ}(X, Y) \cap \text{MB}(X)$ or $\text{ADJ}(X, Y) \cap \text{MB}(Y)$.

Proof: Corollary 2 is an immediate result of theorem 1 since to block a path we never need the vertices off that path.

Remark: The Markov2P algorithm can be viewed as an enhanced version of both TPDA and GS. It improves TPDA in the sense that Markov2P provides a concise step to learn the approximate skeletons of Bayesian networks from data. It improves GS because Markov2P employs both local and global features to reduce the search space of conditioning sets. More specifically, the GS algorithm restricts the search space within the Markov blankets and the Markov2P algorithm shows that it is sufficient to consider the vertices in the intersection of the ADJ set and the Markov blankets. The Markov blanket information is local because it is related with a vertex and its neighbors, whereas the ADJ information is global because it is obtained only when the whole network is taken into account.

5.4. Getting Back SVARs and VARs from Bayesian Networks
We have demonstrated how to model SVARs by Bayesian networks and provide a efficient algorithm to deal with the large size networks corresponding to the SVARs. However, some tasks may require to estimate the coefficient matrices rather than causal relations. In this section, we show that the coefficient matrices can be estimated based on the Bayesian networks obtained. More importantly,
Learning Bayesian Networks from Markov Random Fields: An Efficient Algorithm for Linear Models

high order CI tests are avoided by this approach, so the estimators are more reliable than those obtained by applying the least squares approach to the time series directly. The idea of our approach is as follows. The structures of Bayesian networks indicate which random variables should be taken as the independent variables when we use least squares approach to estimate the coefficient matrices of SVARs. The coefficient matrices of the VARs can be calculated based on those of the SVARs by the relations in equation (17). More specifically, to regress a variable in a SVAR(k), it is not necessary to take all the variable in past k time as independent variables, but only the parents of the dependent variable in the Bayesian network. The justification is an immediate result of the Markov property of Bayesian network: a variable in a Bayesian network is independent of its nondescendants given its parents. We present this idea formally in the procedure BayesVAR. In the BayesVAR algorithm, we first apply Markov2P to time series data to learn a Bayesian network that corresponds to the true SVAR. With the Bayesian network obtained, we regress every variable at the current time on its parents, and the regression coefficient is the estimator of the corresponding element in the coefficient matrix of the SVAR. The next step is finding the permutation matrix P that implies the contemporaneous causal order and calculating the coefficient matrices of the VAR according to the relation in equation (17). From the procedure, we can see that the algorithm has essential distinction from previous works [Demiralp and Hoover 2003; Chen and Chihying 2007; Hyvarinen et al. 2008]. In these works, the VAR is the necessary first step, and the Bayesian network algorithms are applied to the residuals of the VAR to identifying the contemporaneous causal order. In our work, the VARs and SVARs are identified in a reversed order. The SVARs are identified first by applying Markov2P to the time series data, and the VARs are figured out based on the SVARs. The advantage of solving the problem in this way is that high-order CI tests required by the VAR approach can be avoided by choosing efficient Bayesian network learning algorithms. With the Bayesian network information, the number of independent variables is |Pa(X_i(t))| for each X_i(t), whereas in the VAR approach, the number is k \times n for a n dimensional VAR(k) as equation (8) shows.

ALGORITHM 3: Bayesian VAR

Procedure: BayesVAR (D, k)

Data: Time series data set D
Result: The Estimate Coefficient Matrices of the SVAR(k) and the VAR(k)

begin
Markov2P (D, k); // Learn a Bayesian network from data
foreach X_i(t) \in X(t) do
   regress X_i(t) on Pa(X_i(t)) with ordinary least squares method;
end
foreach c_{ij} do
   // c_{ij} is the element at the i-th row j-th column of the coefficient matrix C_t of the SVAR(k)
   if X_j(t - \tau) \in Pa(X_i(t)) then
      assign the corresponding regression coefficient to c_{ij};
   else
      c_{ij} = 0;
   end
end
find the permutation matrix P makes PC_0P' a lower triangular matrix;
\hat{C}_t = PC_0P' for 0 \leq \tau \leq k; // k is the order of the VAR(k)
M = (I - \hat{C}_0)^{-1};
A_{\tau} = PM\hat{C}_{\tau}P' for 1 \leq \tau \leq k; // A_{\tau} are the coefficient matrices of the VAR(k)
end

Taking the SVAR in section 5.1 as an example, we explain how the BayesVAR algorithm works and why the algorithm performs better than the VAR approach. Given the data set D generated by the SVAR in section 5.1, we use BayesVAR to learn a SVAR(2) and a VAR(2). The first step is
learning a Bayesian network from data by Markov2P, and we get a Bayesian network just as figure 3 shows. The next step is estimating coefficient matrices of SVAR(2) according to the structure of the Bayesian network. To this end, BayesVAR regresses each element in $\mathbf{X}(t)$ on its parents. For example, in the Bayesian network in figure 3, the algorithm needs to regress $X_1(t)$, $X_2(t)$, $X_3(t)$ on their parents. Without loss of generality, we take $X_2(t)$ as an example. The parent set of $X_2(t)$ is $\text{Pa}(X_2(t)) = \{X_1(t), X_1(t-2), X_3(t-2)\}$. We regress $X_2(t)$ on $\text{Pa}(X_2(t))$, and get regression coefficients 0.5, 0.5, 0.3 for the three regressors. From this result, we know that $c_{211} = 0.5$, $c_{221} = 0.5$, $c_{231} = 0.3$, and the other entries in the second row of each coefficient matrix of the SVAR(2) are all zero. After all the entries in the coefficient matrix of the SVAR(2) have been estimated, we find the $C_0$ is a strictly triangular matrix, so we have the permutation matrix $P$ an identical matrix in this case. The last step is calculating the coefficient matrices of VAR(2) from the coefficient matrices of SVAR(2) according to equation (17). In this case, we have the result

$$A_1 = \begin{bmatrix} 0.6 & 0 & 1.4 \\ -1 & -0.67 & 2.33 \\ 0.7 & 0.67 & -3.03 \end{bmatrix} \quad A_2 = \begin{bmatrix} -1 & 0.8 & -0.6 \\ 0 & 1.33 & 0 \\ 1.0 & -1.73 & 0.6 \end{bmatrix}.$$  

6. EXPERIMENTAL RESULTS

In this section, we empirically evaluate the BayesVAR algorithm. We first compare BayesVAR with existing works those combine the VAR approach and Bayesian network learning algorithms to discovery causal relations on synthetic data, and then we apply the BayesVAR algorithm to some real world data sets.

6.1. Experiment on Artificial Data

To investigate the performance of the BayesVAR algorithm, we conducted a series of simulation-s. We design the experiment to check if empirical results support our theoretical analysis. To be specific, we are going to test (i) if the BayesVAR algorithm is capable to identify contemporaneous causal relations; (ii) if the Markov2P algorithm avoids high-order CI tests when dealing with large Bayesian networks. We also compare BayesVAR with the following algorithms:

— The G5 algorithm [Margaritis and Thrun 1999]. G5 is a Bayesian network learning algorithm that infers the structures of Bayesian networks locally by identifying the Markov blanket of each vertex. In this work, we apply G5 to the Bayesian networks corresponding to time series.

— The VAR+PC algorithm [Demiralp and Hoover 2003]. This approach applies VAR to time series first, and then uses PC to discover the contemporaneous causal order on regression residuals.

— The TC algorithm [Pellet and Elisseeff 2008]. TC is a Bayesian network learning algorithm that identifies Markov blankets by linear regression. In this work, we apply TC to the Bayesian networks corresponding to time series.

— The HSIC regression [Mooij et al. 2009]. This regression approach minimizes the statistical dependence between regressors and residuals, rather than minimizing squared errors as the standard linear regression does.

— The FactorDAG algorithm [Henao and Winther 2009]. FactorDAG learns directed acyclic graphs from factor models, and uses the directed acyclic graphs to represent causal relations.

— The Basic2P algorithm [Wang and Chan 2010]. Basic2P is a dependency analysis Bayesian network learning algorithm that learns Bayesian networks from Markov random field.

— The BayesVAR algorithm. BayesVAR uses Markov2P to learn the Bayesian networks corresponding to time series and estimates VARs based on the Bayesian networks obtained.

We empirically evaluate the listed algorithms on Gaussian time series, and the data are generated according to some real world Bayesian networks. The networks used in the evaluation are listed in table I, which are obtained from real decision support systems. Data are generated by SVARs corresponding to these networks by the following way. Every coefficient matrix $C_\tau$ is constructed according the structures of the Bayesian networks. Namely, the $(i, j)$-th entry of $C_\tau$ is nonzero if and only if there is an edge from variable $j$ to variable $i$. And then, for a SVAR(k), we generate $k + 1$
learning Bayesian Networks from Markov Random Fields: An Efficient Algorithm for Linear Models

Table I. Bayesian Networks Used in Evaluation

<table>
<thead>
<tr>
<th>Networks</th>
<th>Num. Variables</th>
<th>Num. Edges</th>
<th>Max In/Out Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hallfinder [Jensen and Jensen 1996]</td>
<td>56</td>
<td>66</td>
<td>4/16</td>
</tr>
</tbody>
</table>

We first check the accuracy of the BayesVAR identifying both temporal and contemporaneous causal relations and how the performance varies with the scales of the problems and the number of available observations. In this experiment, we generate SVAR(1) and SVAR(2) based on the networks and for each SVAR, the sample size varies from 128, 256, 512, 1024, 2048, 4096 to 8192. The results are summarized in figure 7. From the results, we can see that BayesVAR outperforms the other algorithms in comparison on all SVARs. The BayesVAR outperforms the other algorithms for two reasons. The first one is that the BayesVAR uses partial correlation as heuristic when identifying Markov blankets. The second reason is that BayesVAR employs ADJ information to restrict the search space of conditioning sets to subsets of Markov blankets. More importantly, we can see that the performance of BayesVAR does not decrease as the scale of the problem increases. For example, BayesVAR has higher $F_1$ value on SVAR(1) corresponding to the Hallfinder network than that corresponding to the Insurance network. The reason is that BayesVAR searches the conditioning sets within Markov blankets, and uses a heuristic Markov blanket learning algorithm to avoid coefficient matrices and in this case, there are $(k + 1) \times n$ vertices in the Bayesian network corresponding to the SVAR if the original real world network contains $n$ vertices. Weights are sampled from uniform distribution between 0.1 and 0.9 for nonzero entries in $C_r$, so that SVARs are stable. Each variable in SVARs is evaluated according to the dynamic of the SVAR for $t$ steps, where $t$ is the sample size in the experiment. Disturbance terms are drawn from standard Gaussian distribution, so that all SVARs are Gaussian processes and zero partial correlation can be tested by Fisher’s Z transformation of partial correlation coefficients [Fisher 1915], and the significant level is set to be 0.05. The performances of the algorithms are qualified in terms of efficiency and accuracy. We use the number of CI tests and CPU time to evaluate efficiency. The advantage of using the number of CI tests as the metric of efficiency is that this metric is implement platform independent, whereas the disadvantage is that it can only be applied for dependency analysis Bayesian network learning algorithms. The CPU time can be used to evaluate efficiency for all the algorithms in comparison. For accuracy, there are two metrics as well. The first one is $F_1$-measure that is defined based on the similarity between the output Bayesian network and the target one. We use adjacent matrix to represent a Bayesian network. Let $\hat{A}$ denote the target adjacent matrix, and $A$ the output one, precision $P$ and recall $R$ are defined as follows:

$$P = \frac{|\{(i,j): \hat{A}(i,j) = \hat{A}(i,j) = 1\}|}{|\{(i,j): \hat{A}(i,j) = 1\}|}$$

$$R = \frac{|\{(i,j): A(i,j) = \hat{A}(i,j) = 1\}|}{|\{(i,j): \hat{A}(i,j) = 1\}|}$$

(21)

Given precision and recall, $F_1$ is defined as:

$$F_1 = \frac{2 \times P \times R}{P + R}$$

(22)

The other is metric for accuracy is forecast error, which is a measurement of the difference between the real value of the time series and the predicted value. More specifically, we use the Mean Squared Error (MSE) defined as follows:

$$MSE = \frac{1}{s} \sum_{i=1}^{s} e_i^T e_i$$

(23)

In equation (23), $s$ is the sample size and $e_i$ is the difference between the real and predicted value at time $t$. We first check the accuracy of the BayesVAR identifying both temporal and contemporaneous causal relations and how the performance varies with the scales of the problems and the number of available observations. In this experiment, we generate SVAR(1) and SVAR(2) based on the networks and for each SVAR, the sample size varies from 128, 256, 512, 1024, 2048, 4096 to 8192. The results are summarized in figure 7. From the results, we can see that BayesVAR outperforms the other algorithms in comparison on all SVARs. The BayesVAR outperforms the other algorithms for two reasons. The first one is that the BayesVAR uses partial correlation as heuristic when identifying Markov blankets. The second reason is that BayesVAR employs ADJ information to restrict the search space of conditioning sets to subsets of Markov blankets. More importantly, we can see that the performance of BayesVAR does not decrease as the scale of the problem increases. For example, BayesVAR has higher $F_1$ value on SVAR(1) corresponding to the Hallfinder network than that corresponding to the Insurance network. The reason is that BayesVAR searches the conditioning sets within Markov blankets, and uses a heuristic Markov blanket learning algorithm to avoid...
Fig. 7. Evaluate BayesVAR against VAR+PC, GS, TC, FactorDAG, HSIC regression, and Basic2P on first and second order SVARs.
Table II. Average sizes, maximal sizes of conditioning sets and the number of CI tests required by each dependency analysis-based algorithm when 8192 samples available

(a) The SVAR(1) based on Insurance Network

<table>
<thead>
<tr>
<th># Statistics</th>
<th>VAR+PC</th>
<th>GS</th>
<th>TC</th>
<th>Basic2P</th>
<th>BayesVAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>15.1 ± 1.7</td>
<td>11.8 ± 2.4</td>
<td>13.1 ± 2.3</td>
<td>12.7 ± 1.9</td>
<td>10.8 ± 1.6</td>
</tr>
<tr>
<td>Max</td>
<td>27</td>
<td>33 ± 3.3</td>
<td>52</td>
<td>52</td>
<td>19 ± 2.7</td>
</tr>
<tr>
<td># CI tests</td>
<td>749048</td>
<td>4108</td>
<td>3186</td>
<td>3075</td>
<td>2663</td>
</tr>
</tbody>
</table>

(b) The SVAR(2) based on Insurance Network

<table>
<thead>
<tr>
<th># Statistics</th>
<th>VAR+PC</th>
<th>GS</th>
<th>TC</th>
<th>Basic2P</th>
<th>BayesVAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>21.2 ± 2.0</td>
<td>16.5 ± 2.9</td>
<td>17.7 ± 2.3</td>
<td>16.4 ± 2.0</td>
<td>14.7 ± 1.5</td>
</tr>
<tr>
<td>Max</td>
<td>54</td>
<td>48 ± 3.9</td>
<td>69</td>
<td>69</td>
<td>26 ± 3.5</td>
</tr>
<tr>
<td># CI tests</td>
<td>792303</td>
<td>4732</td>
<td>3903</td>
<td>3583</td>
<td>2840</td>
</tr>
</tbody>
</table>

(c) The SVAR(1) based on Mildew Network

<table>
<thead>
<tr>
<th># Statistics</th>
<th>VAR+PC</th>
<th>GS</th>
<th>TC</th>
<th>Basic2P</th>
<th>BayesVAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>16.4 ± 1.6</td>
<td>12.5 ± 2.8</td>
<td>14.7 ± 1.2</td>
<td>12.1 ± 1.4</td>
<td>9.7 ± 1.2</td>
</tr>
<tr>
<td>Max</td>
<td>683278</td>
<td>3285</td>
<td>2349</td>
<td>2310</td>
<td>1987</td>
</tr>
<tr>
<td># CI tests</td>
<td>794186</td>
<td>4783</td>
<td>3903</td>
<td>3583</td>
<td>2840</td>
</tr>
</tbody>
</table>

(d) The SVAR(2) based on Mildew Network

<table>
<thead>
<tr>
<th># Statistics</th>
<th>VAR+PC</th>
<th>GS</th>
<th>TC</th>
<th>Basic2P</th>
<th>BayesVAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>30.2 ± 3.6</td>
<td>22.0 ± 2.9</td>
<td>27.3 ± 3.3</td>
<td>23.2 ± 3.1</td>
<td>17.3 ± 2.9</td>
</tr>
<tr>
<td>Max</td>
<td>70</td>
<td>52 ± 4.0</td>
<td>103</td>
<td>103</td>
<td>37 ± 3.2</td>
</tr>
<tr>
<td># CI tests</td>
<td>684064</td>
<td>4873</td>
<td>3593</td>
<td>3152</td>
<td>2317</td>
</tr>
</tbody>
</table>

(e) The SVAR(1) based on Alarm Network

<table>
<thead>
<tr>
<th># Statistics</th>
<th>VAR+PC</th>
<th>GS</th>
<th>TC</th>
<th>Basic2P</th>
<th>BayesVAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>17.5 ± 2.6</td>
<td>12.4 ± 2.0</td>
<td>13.8 ± 2.2</td>
<td>12.6 ± 1.8</td>
<td>11.3 ± 1.7</td>
</tr>
<tr>
<td>Max</td>
<td>37</td>
<td>43 ± 3.0</td>
<td>72</td>
<td>72</td>
<td>33 ± 2.9</td>
</tr>
<tr>
<td># CI tests</td>
<td>794582</td>
<td>5182</td>
<td>4036</td>
<td>3781</td>
<td>3192</td>
</tr>
</tbody>
</table>

(f) The SVAR(2) based on Alarm Network

<table>
<thead>
<tr>
<th># Statistics</th>
<th>VAR+PC</th>
<th>GS</th>
<th>TC</th>
<th>Basic2P</th>
<th>BayesVAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>33.1 ± 5.6</td>
<td>24.5 ± 3.8</td>
<td>27.3 ± 4.9</td>
<td>24.0 ± 3.6</td>
<td>17.8 ± 4.2</td>
</tr>
<tr>
<td>Max</td>
<td>74</td>
<td>61 ± 4.2</td>
<td>109</td>
<td>109</td>
<td>36 ± 5.4</td>
</tr>
<tr>
<td># CI tests</td>
<td>798301</td>
<td>6835</td>
<td>5832</td>
<td>5305</td>
<td>4786</td>
</tr>
</tbody>
</table>

(g) The SVAR(1) based on Hailfinder Network

<table>
<thead>
<tr>
<th># Statistics</th>
<th>VAR+PC</th>
<th>GS</th>
<th>TC</th>
<th>Basic2P</th>
<th>BayesVAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>29.4 ± 7.6</td>
<td>18.5 ± 6.0</td>
<td>13.8 ± 7.2</td>
<td>13.2 ± 6.0</td>
<td>12.7 ± 5.3</td>
</tr>
<tr>
<td>Max</td>
<td>56</td>
<td>58 ± 3.6</td>
<td>110</td>
<td>110</td>
<td>45 ± 4.3</td>
</tr>
<tr>
<td># CI tests</td>
<td>5403893</td>
<td>7692</td>
<td>6275</td>
<td>5373</td>
<td>4524</td>
</tr>
</tbody>
</table>

(h) The SVAR(2) based on Hailfinder Network

<table>
<thead>
<tr>
<th># Statistics</th>
<th>VAR+PC</th>
<th>GS</th>
<th>TC</th>
<th>Basic2P</th>
<th>BayesVAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>49.2 ± 8.3</td>
<td>32.6 ± 5.9</td>
<td>39.7 ± 7.2</td>
<td>31.7 ± 8.1</td>
<td>23.3 ± 8.3</td>
</tr>
<tr>
<td>Max</td>
<td>112</td>
<td>74 ± 5.2</td>
<td>166</td>
<td>166</td>
<td>43 ± 4.7</td>
</tr>
<tr>
<td># CI tests</td>
<td>5790843</td>
<td>10480</td>
<td>7831</td>
<td>7408</td>
<td>6146</td>
</tr>
</tbody>
</table>

high-order CI tests when identifying Markov blankets from data. The VAR+PC, TC, and Basic2P perform worse for the reason that these algorithms rely on high-order CI tests when the scales of the problems are large. The VAR+PC algorithm applies the VAR approach to regresses each variable in the current time on all the variables in the past \( k \) time for a SVAR(k). Therefore, the maximum size of conditioning sets is \( n \times k \) for a \( n \)-dimensional SVAR(k). The TC algorithm uses linear regression to identify the Markov blanket of each variable, and the size of the conditioning set in this case is \( n \times (k + 1) - 2 \). The Basic2P uses high-order CI tests to learn Markov random field from data. Therefore, the \( F1 \) values of VAR+PC, TC, and Basic2P drop dramatically on large networks when samples are limited. For example, on SVAR(1) based on the Hailfinder network and 128 samples available, there is a sudden drop of the \( F1 \) value of the TC algorithm. The HSIC regression performs
Fig. 8. Evaluate BayesVAR against VAR+PC, GS, TC, FactorDAG, HSIC regression, and Basic2P on first order SVARs
A summary of MSE and CPU time is presented in figure 8. All algorithms were run on Intel Core 2 Duo, 2.4 GHz, 2 GB RAM running 32-bit Windows. We normalize the MSE reported by dividing by the MSE of BayesVAR on the same learning task. A normalized MSE more than one means more errors than BayesVAR. We calculate the normalized MSE and CPU time on first order SVARs and 8192 sample available. From figure 8, we can see that BayesVAR provides the best prediction on the time series because the normalized MSEs of all the other algorithms are more than one. In addition, BayesVAR is the most efficient algorithm in the sense of CPU time consumed. The CPU time required by the dependency analysis algorithms is close related with the numbers of CI tests: the more CI tests required by an algorithm, the longer CPU time consumed. For example, the VAR+PC algorithm needs more CI tests than others, so the CPU time required by VAR+PC is much longer than the other algorithms.

The error analysis of BayesVAR on 8192 samples is summarized in table III. The errors are classified into two categories, temporal errors and contemporaneous errors. The temporal errors occur between two variables at time $t$ and $s$ respectively, where $t \neq s$ and contemporaneous errors occur between two variables both in time $t$. There are three types of errors in the contemporaneous category: (i) there should be a causal relation but missed; (ii) there should not be an causal relation but wrong added; (iii) the orientation of the causal relation is reversed. In the temporal category, there are only two types because the orientation is indicated by time. For a Bayesian network with $d$ edges, the corresponding SVAR(1) contains $d$ temporal causal relations and $2d$ contemporaneous causal orders.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total # of True Edges</th>
<th>Temporal Causal Relations</th>
<th>Contemporaneous Causal Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Miss</td>
<td>Extra</td>
</tr>
<tr>
<td>BayesVAR(1)</td>
<td>156</td>
<td>8.3</td>
<td>2.7</td>
</tr>
<tr>
<td>BayesVAR(2)</td>
<td>312</td>
<td>52.7</td>
<td>8.3</td>
</tr>
<tr>
<td>BayesVAR(1)</td>
<td>138</td>
<td>4.5</td>
<td>1.8</td>
</tr>
<tr>
<td>BayesVAR(2)</td>
<td>276</td>
<td>38.6</td>
<td>4.9</td>
</tr>
<tr>
<td>BayesVAR(1)</td>
<td>138</td>
<td>3.7</td>
<td>1.5</td>
</tr>
<tr>
<td>BayesVAR(2)</td>
<td>276</td>
<td>25.4</td>
<td>2.3</td>
</tr>
<tr>
<td>BayesVAR(1)</td>
<td>198</td>
<td>5.3</td>
<td>0.4</td>
</tr>
<tr>
<td>BayesVAR(2)</td>
<td>396</td>
<td>21.0</td>
<td>2.9</td>
</tr>
</tbody>
</table>

Table III. Number of structure errors of temporal and contemporaneous causal relations

(a) The results based on Insurance Network

(b) The results based on Mildew Network

(c) The results based on Alarm Network

(d) The results based on Hailfinder Network
ones and the corresponding SVAR(2) contains 3d temporal ones and 3d contemporaneous ones. From table III, we can see that the proportion of missing temporal causal relations to the contemporaneous one is around 0.5 on SVAR(1) and that proportion is around 1 on SVAR(2). This is also true for the extra causal relations. This fact indicates that the algorithm has almost the same change to make a mistake on temporal and contemporaneous relations. In addition, most orientation of the contemporaneous causal relations can be correctly identified even though there is no time indicator and other prior knowledge.

6.2. Application in Finance

We use the BayesVAR algorithm to find the causal relations among several world stock indices. The chosen indices are DJI in USA, FTSE in UK, N225 in Japan, HSI in Hong Kong, and the SSEC in China, and all the data are public available from Yahoo finance database. We use the daily dividend/split adjusted closing prices from 3rd January 2007 to 31st December 2010. We choose the DJI trading date as the date basis and there are 1008 observations during this period. For the prices of the other indices not available in some DJI trading dates, we simply estimate the prices by linear interpolation. The analysis is based on daily return calculated by the following formula

\[ r(t) = \frac{p(t) - p(t-1)}{p(t-1)} \quad (24) \]

where \( p(t) \) and \( p(t-1) \) denote the adjusted daily closing price of the current trading date and previous trading date respectively and \( r(t) \) denotes the daily return of the current trading date.

A summary of MSE and CPU time on the stock index data is presented in figure 9. In this experiment, we estimate the first order SVARs for each model. We can see that the BayesVAR algorithm gives the best prediction on the daily return of stock index data because the normalized MSE of all the other algorithms are larger than one. In addition, the BayesVAR algorithm is the most efficient one because it requires shortest CPU time among all the algorithms in comparison.

We give a deep insight of the SVAR(1) and VAR(1) on the stock index data set estimated by the BayesVAR algorithm. The structure of the SVAR(1) is illustrated in figure 10. The numbers on the edges are regression coefficient denoting the strength of the causal effects. The SVAR(1) in figure 10 indicates that DJI has significant temporal causal effect on all the other indices, which is consistent with our knowledge. FTSE has a temporal causal effect on DJI, which indicates the close connection between the markets of USA and UK. An interesting fact is that all the autoregression coefficients for N225, FTSE, and DJI are negative. Although there are other paths implies positive causal effects from the previous observation to the current one, the overall effects are still negative. For example, there are three paths from N225_t-1 to N225_t, and the causal effect of each path equals to the production of the weights of the edges on that path. The causal effect of the path (N225_t-1, HSI_t-1, FTSE_t-1, DJI_t-1, N225_t) is \( 0.27 \times 0.3 \times 0.72 \times 0.25 \approx 0.015 \). The summation of
Learning Bayesian Networks from Markov Random Fields: An Efficient Algorithm for Linear Models

A:23

Fig. 10. The estimate SV AR(1) on the stock index data

Fig. 11. The estimate V AR(1) on the stock index data

the weights of the three paths is $-0.23$. The contemporaneous causal orders identified by BayesVAR are consistent with the time difference among the markets in comparison, although the time difference information is not provided to BayesVAR. In addition, all the contemporaneous causal effect are positive, which indicates that the markets tend to move to the same direction in one day. The estimated VAR(1) is illustrated in figure 11. We can see that no information about the contemporaneous causal effect are explicitly illustrated by the VAR(1). In addition, the temporal causal effect in the VAR(1) are quite different from those in the SVAR(1). For example, there is no causal effect from $FTSE_{t-1}$ to $DJI_t$ in the VAR(1). A more detailed discussion on the structures of VARs deviate from the true causal relation can be found in [Hyvarinen et al. 2008].

7. CONCLUDING REMARKS

In this work, we show that the VAR and the SVAR models are incapable of identifying contemporaneous causal order from Gaussian processes and provide a solution to the problem based on Bayesian network learning. The BayesVAR algorithm proposed in this work is distinct from the existing works that find contemporaneous causal order by applying Bayesian network learning algorithms to the residuals of VARs. The BayesVAR algorithm learns Bayesian networks directly from time series data and does not depend on the results of VAR models. By doing this, the algorithm avoids the high-order CI tests involved by estimating VARs from data. To overcome the difficulty that the sizes of the Bayesian networks corresponding to time series tend to be large, we propose the Markov2P algorithm to learn the Bayesian networks locally. By restricting the search space of conditioning sets within a subset of Markov blanket, the orders of the CI tests do not grow fast with the scales of the problems.

Acknowledgments

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administration Region, China.
REFERENCES


