Causation Entropy Principle and Bayesian Inference to Causal Networks

Coufal, David
2015

Institute of Computer Science
Academy of Sciences of the Czech Republic

Causation Entropy Principle and Bayesian Inference to Causal Networks

David Coufal and Jaroslav Hlinka

Technical report No. 1219

August 2016, February 2015

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: +420 266 051 111, fax: +420 286 585 789,
e-mail:ics@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Causation Entropy Principle and Bayesian Inference to Causal Networks

David Coufal and Jaroslav Hlinka[1]

Technical report No. 1219

August 2016, February 2015

Abstract:

The paper discusses the recent paper by Sun at al. [Sun 2014]. We review the causation entropy principle for identification in causal networks and propose a Bayesian approach to inference in these networks. The main purpose of the paper is to provide software implementation of both methodologies and discuss their computational effectivity.

[1]Department of Nonlinear Dynamics and Complex Systems of the Institute of Computer Science AS CR, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic. E-mail: david.coufal@cs.cas.cz.

1 Causation Entropy Principle

The causation entropy principle is a methodology for identification in causal networks. It draws on empirical data and employs techniques from theory of information. Namely, it rests on the notion of mutual information and on a decrease of mutual information due to conditioning. The decrease of mutual information indicates an influence of the conditioning variable on the distribution of a target variable, i.e., the presence of a causal link between the variables.

Mutual information between two multivariate random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ (conditioned on $\boldsymbol{Z}$) is a measure of the deviation from independence between $\boldsymbol{X}$ and $\boldsymbol{Y}$ (conditioned on $\boldsymbol{Z}$). Formally, the definition writes

$$I(\boldsymbol{X};\boldsymbol{Y}|\boldsymbol{Z}) = h(\boldsymbol{X}|\boldsymbol{Z}) - h(\boldsymbol{X}|\boldsymbol{Y},\boldsymbol{Z}) = h(\boldsymbol{Y}|\boldsymbol{Z}) - h(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{Z})$$

where $h(\cdot|\cdot)$ are the conditional entropies specified as

$$\begin{aligned} h(\boldsymbol{X}|\boldsymbol{Z}) &= -\int p(\boldsymbol{x},\boldsymbol{z}) \log p(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{x}d\boldsymbol{z} \\ h(\boldsymbol{X}|\boldsymbol{Y},\boldsymbol{Z}) &= -\int p(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \log p(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{z})\, d\boldsymbol{x}d\boldsymbol{y}d\boldsymbol{z} \end{aligned}$$

A causal network is represented by a multivariate stochastic process $\{\boldsymbol{X}_t = (X_t^{(1)},\dots,X_t^{(d)})', t \in \mathbb{N}\}$, $\boldsymbol{X}_t \in \mathbb{R}^d$, $d \in \mathbb{N}$. The components $X_t^{(i)}$, $i = 1,\dots,d$ of $\boldsymbol{X}_t$ are called nodes of the network. The network dynamics is considered in the form

$$X_t^{(i)} = f_i(A_{i1}X_{t-1}^{(1)},\dots,A_{ij}X_{t-1}^{(j)},\dots,A_{id}X_{t-1}^{(d)}),\xi_t^{(i)}), \quad i = 1,\dots,d$$

where $A_{ij} \in \mathbb{R}$ are the elements of the adjacency matrix A of the network, i.e., $A = [A_{ij}]$, $\xi_t^{(i)}$ is a random fluctuation of the i-th node at time t; and $f_i : \mathbb{R}^{d+1} \to \mathbb{R}$ models the functional dependence of the i-th node at time t on past states of other nodes with $A_{ij} \neq 0$.

Based on the adjacency matrix A, the matrix of causal links is established as

$$A_{ij}^c = \begin{cases} 1 & \text{if } A_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

The matrix determines the sets of the causal neighbors of each node i, $N_i = \{j \in \{1,\dots,d\} | A_{ij}^c = 1\}$. Thus, $A_{ij}^c = 1$ iff $j \in N_i$.

The sets of causal neighbors N_i, $i = 1,\dots,d$ can be identified on the basis of causation entropies. The causation entropy $C_{j\to i|K}$ from the node j to the node i conditioned on the set of nodes $K \subseteq \{1,\dots,d\}$ is the non-negative number specified as

$$C_{j\to i|K} = \lim_{t\to\infty}[h(X_{t+1}^{(i)}|X_t^{(k\in K)}) - h(X_{t+1}^{(i)}|X_t^{(k\in K)},X_t^{(j)})].$$

The causation entropy is a generalization of the transfer entropy $T_{i\to j}$,

$$T_{j\to i} = C_{j\to i|i} = \lim_{t\to\infty}[h(X_{t+1}^{(i)}|X_t^{(i)}) - h(X_{t+1}^{(i)}|X_t^{(i)},X_t^{(j)})].$$

Under the causation entropy principle, the non-zero values of $C_{j\to i|K}$ are used to identify the N_i sets. Let the stochastic process $\{\boldsymbol{X}_t, t \in \mathbb{N}\}$ be temporally and spatially Markov, i.e.,

$$\begin{aligned} p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{X}_{t-2},\dots) &= p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}), \\ p(X_t^{(i)}|\boldsymbol{X}_{t-1}) &= p(X_t^{(i)}|X_{t-1}^{(N_i)}), \end{aligned}$$

then for each i the set K that contains N_i is identified by the algorithm presented in Table 1 ([Sun 2014], Algorithm 2.1 adapted for $I = \{i\}$, $V = \{1,\dots,d\}$).

01. $K \leftarrow \emptyset$; $x \leftarrow \infty$; $p \leftarrow \emptyset$;
02. **while** $x > 0$,
03. $K \leftarrow K \cup \{p\}$;
04. **for** $j \in (V \setminus K)$,
05. $x_j \leftarrow C_{j \to i|K}$;
06. **end**;
07. $x \leftarrow \max_{j \in (V \setminus K)} \{x_j\}$;
08. $p \leftarrow \arg\max_{j \in (V \setminus K)} \{x_j\}$;
09. **end**;

Table 1.1: Identification of causal neighbors.

The MATLAB implementation of the algorithm reads as follows:

```
1   function [K] = sun21q(I,X,F0,F1)
2    d=size(F0,1);
3    K=[];x=Inf;p=[];Cq=1;
4    V=[1:d];
5    c=0;
6    while x>Cq
7     c=c+1;
8     K=union(K,p);
9     sdVK=setdiff(V,K);
10    xj=zeros(1,d);
11    xq=zeros(1,d);
12    for j=sdVK,
13     [xjC]=sunC([j],I,K,F0,F1);
14     xj(j)=xjC;
15     xq(j)=permC([j],I,K,X,100);
16    end;
17    [xj; xq];
18    [x,p] = max(xj);
19    Cq=xq(p);
20   end
21  end
```

In the script, at line 13, there is called the function `sunC.m`. This function computes the causation entropy $C_{j \to i|K}$. On line 15, the function `permC.m` realizes the permutation test proposed by Sun on pp. 19-20. The test is used to decide if the computed causation entropy devises significantly from zero. The code of the function is presented in the next section.

2 Gaussian process

As a benchmark, the Gaussian process is investigated because it allows to establish analytical solutions. The model writes

$$\boldsymbol{X}_t = A\boldsymbol{X}_{t-1} + \boldsymbol{\xi}_t, \quad t \in \mathbb{N}, \boldsymbol{X}_0 = 0_d. \tag{2.1}$$

In the formula, A is a matrix and the task is to identify non-zero elements of A provided that $\boldsymbol{\xi}_t$ form an i.i.d. normally distributed random variables, i.e., $\xi_t^{(i)} \sim N(0, \sigma_i)$, $\sigma_i > 0$ and therefore $\boldsymbol{\xi}_t \sim N(0, S)$ where S is the covariance matrix with entries $S_{ij} = \delta_{ij}\sigma_i^2$.

Let $\phi(\tau,t)$, $\tau\in\mathbb{N}_0$, $t\in\mathbb{N}$ be the matrices of shifted covariances with entries

$$\phi(\tau,t)_{ij}=cov(X^{(i)}_{t+\tau},X^{(j)}_t).$$

It follows from (2.1), that $\boldsymbol{X}_t\sim N(0,\Phi(0,t))$ where

$$\Phi(0,t)=\sum_{k=0}^{t}A^kS(A^k)'.$$

A sufficient condition for existence of $\lim_{t\to\infty}\phi(0,t)=\Phi(0)$ is stability of the matrix A. The matrix A is stable if its spectral norm ρ_A is lower than 1. ρ_A is defined as the maximum from the absolute values of the eigenvalues of A.

If $\Phi(0)=\sum_{k=0}^{\infty}A^kS(A^k)'$ exists, it satisfies the equation

$$A\Phi(0)A'-\Phi(0)+S=0$$

which is equivalent to

$$(I_{d^2}-A\otimes A)vec(S)=vec(S)$$

where $\otimes$ is the Kronecker product and $vec(\cdot)$ transforms a matrix to the column vector. In MATLAB the above computes as

```
reshape((eye(d^2)-kron(A,A))\S(:),[d,d])
```

The matrix $\Phi(\tau)=\Phi(\tau,t)$ for each t satisfies the relation

$$\Phi(\tau)=A\Phi(\tau-1)=A^2\Phi(\tau-2)=\cdots=A^\tau\Phi(0),$$

i.e., namely $\Phi(1)=A\Phi(0)$.

Analytical computation of matrices $\Phi(0)$ and $\Phi(1)$ is implemented as follows:

```
function [F0,F1] = sunF(A,S)
 d=size(A,1);
 F0=reshape((eye(d^2)-kron(A,A))\S(:),[d d]);
 F1=A*F0;
 %---by series---
 %F0s=zeros(n,n);
 %for k=0:1000, F0s=F0s+A^k*S*(A^k)'; end;
 %F0s;
end
```

For the Gaussian process (2.1), the following analytical expression for the causation entropy is available:

$$C_{j\to i|K} \;=\; \frac{1}{2}\log\left[\frac{\phi(0)_{i,i}-\phi(1)_{i,K}\phi(0)^{-1}_{K,K}\phi(1)'_{i,K}}{\phi(0)_{i,i}-\phi(1)_{i,\{Kj\}}\phi(0)^{-1}_{\{Kj\},\{Kj\}}\phi(1)'_{i,\{Kj\}}}\right].$$

In the formula, $\phi(0)_{ii}$ denotes the i-th row and i-th column element of the $\phi(0)$ matrix. Similarly, $\phi(1)_{i,\{Kj\}}$ denotes the submatrix of the $\phi(1)$ matrix, that is determined by the i-th row and the $k\in K$-th and j-th columns of $\phi(1)$, etc.

If $K=\{i\}$, then one gets the transfer entropy

$$T_{j\to i}=C_{j\to i|i}=\frac{1}{2}\log\left[\frac{\phi(0)_{ii}-\phi(1)_{ii}\phi(0)^{-1}_{ii}\phi(1)'_{ii}}{\phi(0)_{ii}-\phi(1)_{i,\{ij\}}\phi(0)^{-1}_{\{ij\},\{ij\}}\phi(1)'_{i,\{ij\}}}\right].$$

The MATLAB implementation reads as

```
function [C] = sunC(J,I,K,F0,F1)
 nom=F0(I,I)-F1(I,K)*inv(F0(K,K))*F1(I,K)';
 nom=det(nom);
 den=F0(I,I)-F1(I,union(K,J))*inv(F0(union(K,J),union(K,J)))*F1(I,union(K,J))';
 den=det(den);
 C=nom/den;
 C=0.5*log(nom/den);
end
```

2.1 Directed Linear Chain Example

In this case the A matrix is specifed as $A_{ij} = \delta_{i,j+1}$, i.e., for $n = 3$ one has

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

We are interested in the matrices $\Phi(0)$ and $\Phi(1)$. The analytic formulae write as

$$\Phi(0)_{ij} = \delta_{ij} \sum_{k=1}^{j} \sigma_k^2, \quad \Phi(1)_{ij} = \delta_{i,j+1} \sum_{k=1}^{j} \sigma_k^2.$$

The explicit expressions are

$$\phi(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad \phi(1) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix}.$$

Finally, the causation entropy formula has the form

$$C_{j\to i|i} = T_{j\to i} = \frac{1}{2}\delta_{i,j+1} \log\left(1 + \frac{\sum_{k=1}^{j} \sigma_k^2}{\sigma_i^2}\right),$$

which gives the matrix (j is the column index and i is the row index)

$$C_{j\to i|i} = T_{j\to i} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{3}{2} & 0 & 0 \\ 0 & \frac{6}{3} & 0 \end{bmatrix}.$$

This example is implemented as follows:

```
n=3;clc;
T=1000;
A=[0 0 0;1 0 0;0 1 0]
S=[1 0 0;0 2 0;0 0 3];
X=zeros(3,T);
Y=randn(3,T);
Y(2,:)=Y(2,:)*sqrt(2);
Y(3,:)=Y(3,:)*sqrt(3);
for i=2:T, X(:,i)=A*X(:,i-1)+Y(:,i); end;
%---
[F0,F1] = sunF(A,S);
sF0=cov(X');
sF1=zeros(n,n);
for i=1:3, for j=1:3,
```

```
15    C=cov(X(i,2:T),X(j,1:T-1));
16    sF1(i,j)=C(1,2);
17   end; end;
18   [F0 sF0]
19   [F1 sF1]
20   F0=sF0;F1=sF1;
21   Ac=zeros(n,n);
22   for i=1:n,
23    i
24    K=sun21q(i,X,F0,F1);
25    Ac(i,K)=ones(1,length(K));
26   end;
27   Ac
```

In the above script, we have two possibilities how to compute matrices F0=$\Phi(0)$ and F0=$\Phi(1)$. Either analytically due to the Gaussian character of (1) by function sunF.m; or or we may use sample counterparts of $\Phi(0)$ and $\Phi(1)$ matrices computed at lines 12-17. The lines 18-19 provide a comparison of the results from both approaches.

To conclude the section it remains to provide the code of the permutation test. It reads

```
1    function [Cq] = permC(j,I,K,X,r)
2     th=0.95;
3     [d,T]=size(X);X0=X;
4     Cr=zeros(1,r);
5     for k=1:r,
6      X=X0;
7      X(j,:)=X(j,randperm(T));
8      rF0=cov(X');
9      rF1=zeros(d,d);
10     for ii=1:d, for jj=1:d,
11      A=X(ii,2:T);
12      B=X(jj,1:T-1);
13      rF1(ii,jj)=1/(T-2)*(A-mean(A))*(B-mean(B))';
14     end; end;
15     Cr(k)=sunC([j],I,K,rF0,rF1);
16    end;
17    Cq=quantile(Cr,th);
18   end
```

The number of permutation is stored in the r parameter and the quantile θ in the th parameter. In loops at lines 10-16, we compute the sample matrices $\Phi(0)$ and $\Phi(1)$ for each permutation created at line 7.

3 Bayesian inference

Here we review an approach to causal inference that stems from the statistical analysis of time series. We follow [Anděl 1976].

Let $A_0, A_1, \ldots, A_n$, $n \in \mathbb{N}$ be square matrices of order $d \in \mathbb{N}$ such that A_0 is regular. Set $U_k = -A_0^{-1}A_k$, $k = 1, \ldots, n$ and consider the model

$$\boldsymbol{X}_t = \sum_{k=1}^{n} U_k \boldsymbol{X}_{t-k} + A_0^{-1}\boldsymbol{Y}_t, \qquad t = n+1, \ldots T \tag{3.1}$$

where $\boldsymbol{Y}_t$ are i.i.d. d-variate normal. Clearly, setting $n = 1$ and $A_0 = -I_d$ we get the model (1).

Let $\boldsymbol{X}_1 = \boldsymbol{x}_1, \ldots \boldsymbol{X}_n = \boldsymbol{x}_n$ be fixed. Because $\boldsymbol{Y}_{n+1}, \ldots, \boldsymbol{Y}_T$ are multivariate normal and (3.1) is linear, the densities of $\boldsymbol{X}_{n+1}, \ldots, \boldsymbol{X}_T$ given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are also multivariate normal.

Let $G = A_0' A_0$ and $U = [U_1, \ldots, U_n]$. Using the improper apriori density $|G|^{-1}$ for $G > 0$ (i.e., where G is positive definite) and 0 otherwise, one gets the aposteriori density of G and U given $\boldsymbol{x} = (\boldsymbol{x}_1', \ldots, \boldsymbol{x}_T')'$ in the form

$$\begin{aligned} g(G, U|\boldsymbol{x}) &= const \cdot |G|^{(T-n-1)/2} \times \\ &\exp\left[-\frac{1}{2} \sum_{t=n+1}^{T} (\boldsymbol{x}_t - \sum_{j=1}^{n} U_j \boldsymbol{x}_{t-j})' \cdot G \cdot (\boldsymbol{x}_t - \sum_{j=1}^{T} U_j \boldsymbol{x}_{t-j}) \right]. \end{aligned}$$

Establishing the vectors

$$\boldsymbol{x}_t^0 = \begin{bmatrix} \boldsymbol{x}_{t-1} \\ \vdots \\ \boldsymbol{x}_{t-n} \end{bmatrix} \quad t = n+1, \ldots, T$$

and the matrices (recall that $\boldsymbol{x}_t$ is a d-variate column vector)

$$\begin{aligned} C &= \sum_{t=n+1}^{T} \boldsymbol{x}_t \boldsymbol{x}_{t'}^0, \quad C_0 = \sum_{t=n+1}^{T} \boldsymbol{x}_t \boldsymbol{x}_{t'}, \\ S &= \sum_{t=n+1}^{T} \boldsymbol{x}_t^0 \boldsymbol{x}_{t'}^0, \quad S^* = \begin{bmatrix} C_0 & C \\ C' & S \end{bmatrix} \end{aligned}$$

we have the following theorem:

Theorem 1. *Let $S^* > 0$, i.e., be positive definite. Then the aposteriory density $g(G, U|\boldsymbol{x})$ has modus at $\hat{U}, \hat{G}$ where $\hat{U} = CS^{-1}$ and $\hat{G} = (T-n-1)(C_0 - CS^{-1}C')^{-1}$.*

The obtained estimates $\hat{U}$ and $\hat{G}$ are then used to obtain the estimates $\hat{A}_0, \hat{A}_1, \ldots \hat{A}_n$ of the $A_0, A_1, \ldots A_n$ matrices.

3.1 Tests on autoregressive matrices

Theorem 2. *Let $S^* > 0$. If $T \geq (n+1)d$, then the aposteriori distribution with the density $g(G, U|\boldsymbol{x})$ is the Wishart distribution $W_p(T - n - nd + d, C_0 - CS^{-1}C')$. Let*

$$V_0 = (T - n + p)^{1/2} (C_0 - CS^{-1}C')^{-1/2} (U_0 - \hat{U}) S^{1/2}$$

where U_0 is a matrix of the same type as matrix $\hat{U} = [\hat{U}_1, \ldots, \hat{U}_n]$. If $T - n + d \leq 2$, then the statistics $\mathrm{Tr}(V_0 V_0')$ (trace of $V_0 V_0'$) has asymptotically χ^2 distribution with the nd^2 degrees of freedom.

Setting $U_0 = 0_{(d,nd)}$ is used to test the null hypothesis that $\boldsymbol{X}_t$, $t \in \mathbb{N}$ are independent against the alternative that they form an autoregressive sequence at most of order n. Frequently, we need to test if $U_n = U_n^0$ where U_n^0 is a given matrix. Typically, U_n^0 is the zero matrix and the test concerns if the maximal lag is n. The next theorem provides the basis for the test.

Theorem 3. *Let $S^* > 0$. Denote*

$$S^{-1} = R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$$

where R_{22} is the square submatrix of order d. Let

$$V_2^0 = (T - n - nd + 2d)^{1/2} (C_0 - CS^{-1}C')^{-1/2} (U_n^0 - \hat{U}_n) S^{1/2}.$$

Then $\mathrm{Tr}(V_2^0 V_2^0)$ has asymptotically χ^2 distribution with the d^2 degrees of freedom.

The theorem gives us the tool for performing tests on submatrices of $U = [U_1, \ldots, U_n]$. The first type of these tests is testing for if lag=n, i.e., if $U_n = 0_d$. The other typical situation is for lag=1 when we are given by some mask U_1^0 for U_1 matrix and we test if the mask is consistent with the observed data.

The MATLAB function performing the above computations writes as follow:

```
function [U] = volD(X,n)
[T,d]=size(X);
C=zeros(d,n*d);
S=zeros(n*d,n*d);
C0=zeros(d,d);
x0np1=
for t=n+1:T,
 xt=X(t,:)';
 xot=[]; for i=1:n, xot=[xot; X(t-i,:)']; end;
 %xot1=flip(X(t-1:t-n,:));
 C=C+xt*xot';
 S=S+xot*xot';
 C0=C0+xt*xt';
end;
invS=inv(S);
U=C*invS;

%S1=[C0 C;C' S];
%V2o=(T-n-n*d+2*d)^(0.5)*(C0-C*invS*C')^(-0.5);
%nS=size(iS);
%R22=iS([nS-d+1:nS],[nS-d+1:nS]);
%V2=(T-n-n*d+2*d)^(0.5)*(C0-C*iS*C')^(-0.5);
end
```

3.2 Application to causal networks

The previous sections provides the concise review of the Bayesian inference in multivariate autoregressive time series. How it may be applied in the context of causal inference?

We consider model (1), therefore $d = 1$ and $A_0 = -I_d$. Given data $\boldsymbol{X}_t = \boldsymbol{x}_t$, $t \in \mathbb{N}$, $\boldsymbol{X}_t \in \mathbb{R}^d$ we estimate the matrix $\hat{U} = \hat{A}$ (Indeed, $U_1 = -A_0^{-1}A_1 = A$). Following the Sun's idea of the permutation test we repeat this r times in each particular dimension $j = 1, \ldots, d$. That is, in each permutation we permute the j-th row of $[\boldsymbol{X}_t, \boldsymbol{X}_{t-1}, \ldots, \boldsymbol{X}_1]$.

Let j be fixed, performing r permutation and estimating A, we get r matrices that differ in the j-th row and the j-th column. We find empirical $(1-\theta)\cdot 100\%$ quantiles over the r j-th rows and columns and store them as the j-th row of matrix V_{qr} and the j-th column of matrix V_{qc}. Averaging provides us with the final quantile matrix Vq, i.e., $Vq = (V_{gr} + V_{qc})/2$.

Let $\hat{A}$ be an estimation of the matrix A of (1). We compare each element of $\hat{A}$ with V_q. If $\hat{A}_{ij} \geq Vq_{ij}$, then we let $\hat{A}_{ij}$ as it is, otherwise we set the corresponding element to zero. We obtained the adjusted matrix A_M that we call the mask matrix. Clearly, if we set $U_1^0 = A_M$ in Theorem 2, we may test the significance of A_M. If $A = \hat{A}_M$ hypothesis is not rejected, then we put forward $\hat{A}^c = sign(abs(A_M))$ as the matrix of causal links identified by the Bayesian approach.

Let j be fixed, performing r permutation and estimating A, we get r matrices that differ in the j-th row and the j-th column. We keep the j-th columns of these r matrices in `U(:,j,k)` variable, where $k = 1, \ldots, r$. Let $i = 1, \ldots, d$ be the row index in the j-th columns. For each i we identify the th quantile of the absolute values of stored j the columns. We keep the result in Vq_{ij} element of the quantile matrix Vq, i.e., `Vq(i,j)=quantile(abs(Ur(i,j,:)),th)`. The quantile matrix is the square matrix of order d, i.e., $i, j = 1, \ldots, d$. The relevant MATLAB function reads as follows

```
function [Vq,Ur] = permD(X,r,th)
 [d,T]=size(X);X0=X;
```

```
  Ur=zeros(d,d,r);
  %for i=1:d, i, for j=1:d,
  % for k=1:r,
  %  X(j,:)=X0(j,randperm(T));
  %  U=volB(X',1);
  %  Ur(i,j,k)=U(i,j);
  % end;
  %end; end;
  for j=1:d, j
   for k=1:r,
    X(j,:)=X0(j,randperm(T));
    U=volB(X',1);
    Ur(:,j,k)=U(:,j);
   end;
  end;
  Vq=zeros(d,d);
  for i=1:d, for j=1:d,
   Vq(i,j)=quantile(abs(Ur(i,j,:)),th);
  end; end;
 end
```

Let $\hat{A}$ be an estimate of matrix A of (1). We compare each element of $\hat{A}$ with V_q. If $abs(\hat{A}_{ij}) \geq Vq_{ij}$, then we let $\hat{A}_{ij}$ as it is, otherwise we set the corresponding element to zero. By this operation we obtain the adjusted matrix A_M that we call the mask matrix. Corresponding matrix of causal links is set as $\hat{A}^c = sign(abs(A_M))$. That is, non zero elements of A_M are put to ones and zero elements are retained.

Clearly, if we set $U_1^0 = A_M$ in Theorem 2, we may test the significance of A_M. If $A = \hat{A}_M$ hypothesis is not rejected, then we might put forward $\hat{A}^c = sign(abs(A_M))$ as the matrix of causal links identified by the Bayesian approach and supported by the result of the corresponding χ^2 test.

The MATLAB scripts that perform the relevant computations are presented in Appendix.

4 Experiments

In this section we present results of experiments with the Bayesian inference introduced in the previous section. We have followed the experiments reported by [Sun 2014].

The matrix A of (1) corresponds to a signed Erdös-Rényi network of size d, $d \in \mathbb{N}$. That is, A is a $d \times d$ matrix with approximately pd^2 directed links (non-zero elements). The links are selected randomly with the probability of selection $p = dp/d$ where dp is the control parameter used in the scripts below. Each link is either positive or negative with the absolute value w. It means that each non-zero element of A_{ij}, $i = 1, \ldots, d$, $j = 1, \ldots d$ writes $A_{ij} = w$ or $A_{ij} = -w$. The weight w of the link is selected in such a way that the spectral norm of A has prescribed value that is below 1, i.e., $\rho_A < 1$.

Data for experiments are then generated from (1), where the matrix A corresponds to the signed Erdös-Rényi network and the noise terms are independent standard normal, i.e., $\xi_t \sim \mathcal{N}(0, 1)$.

```
function [X,A] = ernet(d,T,w,dp)
 p=dp/d;q=(1-p);
 A=rand(d);
 A=sign(sign(A-q)+1);
 W=rand(d);
 W=sign(W-0.5);
 A=w*A.*W;
 rho=max(abs(eig(A)))
 if rho>1, disp('rho>1'); end;
 %---
 X=zeros(d,T);
```

```
12    for t=2:T,
13     X(:,t)=A*X(:,t-1)+sqrt(1)*randn(d,1);
14    end;
15   end
```

Measures that are used to assess the quality of estimation of causal links are the standard false negative and false positive rates ϵ_- and ϵ_+ that are specified as

$$\epsilon_- = \frac{\text{number of } (i,j) \text{ pairs with } \chi_0(A^c)_{ij} = 1 \text{ and } \chi_0(\hat{A}^c)_{ij} = 0}{\text{number of } (i,j) \text{ pairs with } \chi_0(\hat{A}^c)_{ij} = 1}$$

$$\epsilon_+ = \frac{\text{number of } (i,j) \text{ pairs with } \chi_0(A^c)_{ij} = 0 \text{ and } \chi_0(\hat{A}^c)_{ij} = 1}{\text{number of } (i,j) \text{ pairs with } \chi_0(\hat{A}^c)_{ij} = 0}$$

```
1   function [fnr,fpr] = fp(A,Ahat)
2    d=size(A,1);
3    em=0;ep=0;t1=0;t0=0;
4    Ac=sign(abs(A));
5    Achat=sign(abs(Ahat));
6    for i=1:d, for j=1:d,
7      if ((Ac(i,j)==1)&(Achat(i,j)==0)), em=em+1; end;
8      if ((Ac(i,j)==0)&(Achat(i,j)==1)), ep=ep+1; end;
9      if (Ac(i,j)==1), t1=t1+1; end;
10     if (Ac(i,j)==0), t0=t0+1; end;
11   end; end;
12   [em ep]
13   [t1 t0]
14   fnr=em/t1;
15   fpr=ep/t0;
16  end
```

In the first set of experiments, we were interested in the influence of the quantile θ on stabilization of both rates. As it was pointed out in the Sun's article, the false negative rate ϵ_- does not depend on θ and goes to zero as $T \to \infty$. The false positive rate ϵ_+ saturates at the level $\epsilon_+ = 1 - \theta$. In the experiments the number of permutations was set $r = 100$ and further we set $\rho_A = 0.8$ ($w = 0.25$), $dp = 10$. The obtained results are presented in Fig. 1.

In the second set of experiments, we have investigated the influence of the density of links in the network that is controlled by the dp parameter; and spectral norm ρ_A. The results are provided in Fig. 2. The fixed parameters were $r = 100$, $d = 100$, and $\theta = 0.95$.

In accordance with the Sun's results, we have found that the critical sample size where ϵ_- stabilizes near 0 does not depend on the network size, but rather depends on the density of links (dp parameter) or on the spectral radius ρ_A of the adjacency matrix A.

4.1 Time consumption analysis

This section is devoted to time consumption analysis of Bayesian inference in Erdös-Rényi networks. We investigated how computation time depends on the sample size T with respect to the size of the network d, the number of permutations r, the spectral density ρ_A and the density of causal links driven by the parameter dp. In all experiments we used $\theta = 0.95$.

In the first experiment, the size of the network was gradually set to $d = 50$, $d = 100$ and $d = 150$. The other parameters were set to $r = 100$, $\rho_A = 0.8$ and $dp = 10$. From Table 2, we see that with increasing sample size T the computation time grows approximately linearly; and as would be expected it grows also with increasing d. However, we have observed that for $d = 150$ and $T > 1000$ the variance of computation time increases. In Table 2, there are presented computation times in minutes for each particular sample size T.

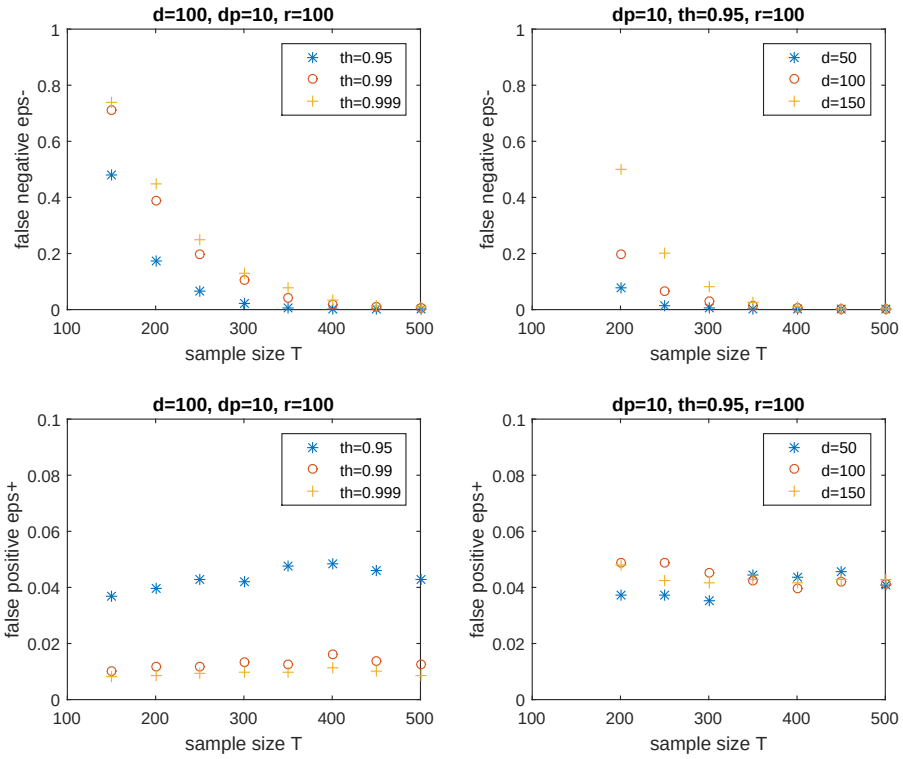

Figure 4.1: ϵ_-, ϵ_+ rates for different θs ans ds; $r = 100$, $\rho_A = 0.8$, $dp = 10$.

T	$d = 50$	$d = 100$	$d = 150$
200	0.6	3.5	9.5
400	1.0	6.7	18.8
600	1.5	9.8	31.3
800	2.0	13.7	37.3
1000	2.5	16.8	50.3
1200	3.0	22.0	56.3
1400	3.4	24.5	67.0
1600	3.9	29.0	81.0
1800	4.4	32.5	89.0
2000	5.8	34.3	111.5

Table 4.1: Computation times [mins] for different network sizes d.

In the second experiment, we were interested in the influence of the number of permutation. We had the similar setup with respect to the sample size i.e., ranging from $T = 200$ to $T = 2000$ with the step $\Delta T = 200$. The size of the network and other parameters were fixed at $d = 50$; $\rho_A = 0.8$ and $dp = 10$. The number of permutations varied as $r = 100$; $r = 500$ and $r = 1000$.

The third experiment investigated the influence of the spectral norm. We analyzed networks of size $d = 50$ with the norms set up as $\rho_A = 0.9\,(w = 0.27)$, $\rho_A = 0.6\,(w = 0.18)$ and $\rho_A = 0.3\,(w = 0.1)$. The other parameters were set to $r = 100$ and $dp = 10$. From Table 4.3, we see that the value of ρ_A does not have any substantial influence on computational times.

The last experiment concerned with the influence of the density of links in A that is governed by the value of the dp parameter. We had $d = 50$ and other parameters was set as above, i.e., $\rho_A = 0.8$ and

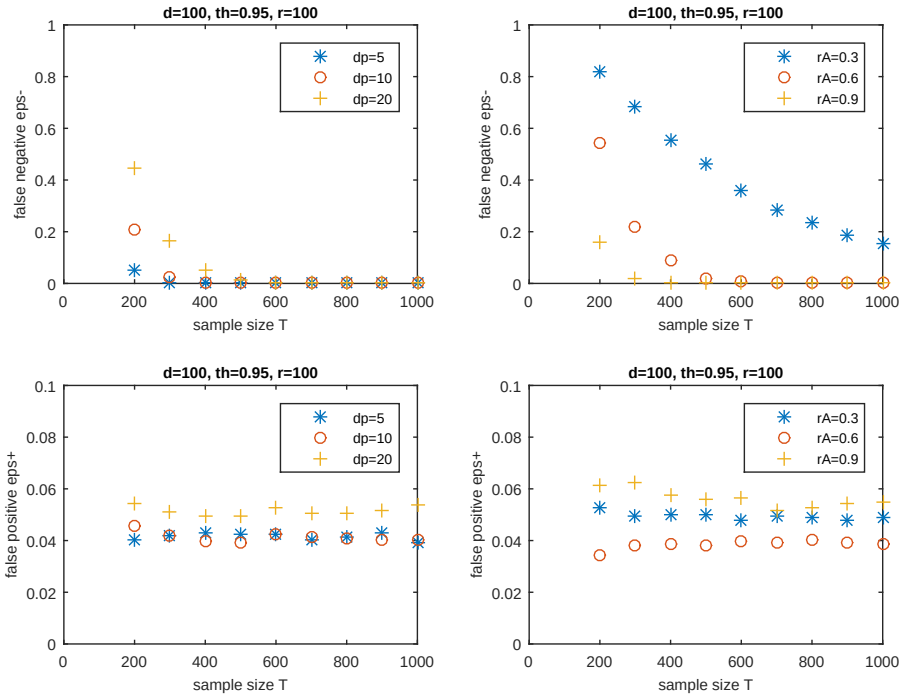

Figure 4.2: ϵ_-, ϵ_+ rates for different nd and ρ_A; $r = 100$, $d = 100$, $\theta = 0.95$.

$T, d = 50$	$r = 100$	$r = 500$	$r = 1000$
200	0.6	5.3	13.0
400	1.0	10.3	23.5
600	1.5	15.2	35.3
800	2.0	21.3	46.2
1000	2.5	25.3	58.7
1200	3.0	31.5	60.3
1400	3.4	33.5	75.3
1600	3.9	42.7	102.5
1800	4.4	44.5	105.2
2000	5.0	51.3	119.7

Table 4.2: Computation times [mins] for different numbers of permutations r.

$r = 100$. Because we kept ρ_A constant, the values of weights w varied with dp. We used $dp = 5\,(w = 0.33)$, $dp = 10\,(w = 0.25)$ and $dp = 20\,(w = 0.18)$. Results are presented in Table 4.4.

We see that the results are more or less the same as in the third experiment.

5 Parallel computing in Python

In the previous section, we presented results of causal links detection using the Bayesian approach. The computations were performed in the MATLAB computational environment on a dual-core machine. The causation entropy algorithm by Sun cannot be effectively tested on this hardware because of its computational intensity. In fact, in MATLAB we were able to run the Sun algorithm for networks having the number of nodes not exceeding $d = 10$.

$T, d = 50$	$\rho_A = 0.9$	$\rho_A = 0.6$	$\rho_A = 0.3$
200	0.7	1.2	1.1
400	1.2	2.0	2.2
600	2.0	3.0	3.2
800	2.2	4.0	4.2
1000	2.7	5.3	5.2
1200	3.2	6.0	6.2
1400	3.8	7.7	7.0
1600	4.4	7.8	8.7
1800	5.2	10.0	8.8
2000	5.3	10.7	9.8

Table 4.3: Computation times [mins] for different spectral norms ρ_A.

$T, d = 50$	$dp = 5$	$dp = 10$	$dp = 20$
200	0.7	1.1	1.1
400	1.2	2.0	2.2
600	1.8	3.0	3.3
800	2.1	4.0	4.5
1000	2.7	5.0	5.8
1200	3.5	6.0	6.5
1400	3.6	6.8	7.5
1600	4.2	7.8	9.0
1800	4.4	8.8	9.8
2000	5.2	10.0	11.0

Table 4.4: Computation times [mins] for different dp products.

To overcome the obstacle, we switched to a 24-core machine to enjoy the possibility of parallel computing. In fact, the Sun's Algorithm 2.1 reported here in Table 1, is very suitable for parallelization. The loop which cycles through the lines of the matrix A establishes the core of the algorithm; and the particular results for the individual lines of A do not interfere. Hence parallelization can be done by simply parallelizing the loop.

In MATLAB, Parallel Computing Toolbox and its `parfor` command enables parallelizing a loop straightforwardly. Unfortunately, our license of MATLAB does not support more than 6 cores. That is why we have switched to use the Python programming language. Namely, we used 2.7 version and the parallel computation interface was provided by the multiprocessing package. The code of the related Python script is provided in Appendix B.

5.1 Climate data

In our experiments, we used NCEP/NCAR Reanalysis 1 surface air temperature data. For our purposes, the original data were remapped to a sparser grid consisting of 42 places regularly spread over surface of the Earth; and we were interested in the 500hPa surface level which roughly corresponds to the altitude 10 km above sea level. More details about the remapping procedure can be found in [Hlinka et al. 2014], see Section 2.1 and the references therein.

The remapped data were used to identify a multivariate AR(1) model which is fully characterized by the matrix of its coefficients A. Clearly, A is the square matrix of order 42. Specification of the matrix was done using the standard AR process identification procedure. This gave the matrix with 88 non-zero elements of total 1764. The matrix is presented in Fig. 5.1 and its spectral norm is $\rho_A = 0.9124$.

The identified matrix was used for generating artificial data according to formula $\boldsymbol{X}_t = A\boldsymbol{X}_{t-1}$, $t \in \mathbb{N}$, $\boldsymbol{X}_0 = \boldsymbol{0}_d$. We generated data of increasing length with the step $\Delta T = 100$, i.e., we used data samples with

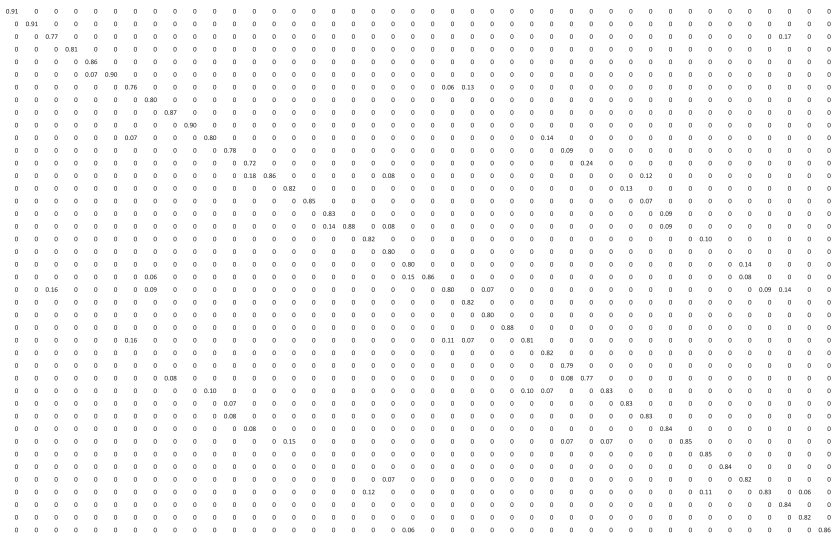

Figure 5.1: Matrix of coeficients of AR(1) model describing temperature changes.

$T = 100, 200, \ldots, 1900, 2000$. For each data sample we performed causal links detection using the Sun and Bayesian algorithm, respectively. The causation entropy algorithm was run in the Python multiprocessing setting while Bayesian one was run in MATLAB on single processor dual-core machine.

5.2 Results

The purpose of causal links identification is to detect location of the non-zero elements of AR(1) matrix presented in Fig. 5.1. This is what the Sun algorithm does. Baysian algorithm is also able to identify a strength of the link in terms of a real coefficient. However, in the presented experiment we restrict only on question if at a given position in the matrix there is or not a non-zero element. The accuracy of estimation is measured using the false negative (FN) and false positive (FP) rates introduced above. The identification was performed gradually for data of increasing length with $T = 100, 200, \ldots, 1900, 2000$.

False negative rates

In Table 5.1 below, there are presented the false negative (FN) rates for causal links identification in the generated data. The first column of the table delivers the lengths of processed data series, the second columns contains the FN rates for the Sun algorithm and the third one for the Bayesian approach. The Sun algorithm works better on shorter data series in comparison with the Bayesian algorithm. The better results, however, are obtained for price of much longer computations. The difference in error vanishes as the length of data increases. The computational burden is driven rather by the size of the network, here we have $d = 42$, than by the length of the data. Graphically, the development of the false negative rate is presented in Fig. 5.2.

False positive rates

In Table 5.2, we see the development of the false positive (FP) rates with the length of the data for both the Sun algorithm and the Bayesian approach. We can see that in both cases the rate approach zero, but at faster pace in the first case. In fact, in the Sun's case, the algorithm reaches the zero rate for lengths $T = 500$ and longer. Again, the price for better performance for short series is computational complexity. The computation for $T = 500$ took 10 hours on the multiprocessor machine when 20 cores were simultaneously used. In the case of Bayesian approach, the computation took minutes on a single processor dual core machine in MATLAB. Graphically, the development of false positive rates is presented in Fig. 5.3.

$d = 42, T$	Sun - false negative	Bayes - false negative
100	$34/88 = 0.3864$	$46/88 = 0.5227$
200	$21/88 = 0.2386$	$19/88 = 0.2159$
300	$16/88 = 0.1818$	$17/88 = 0.1932$
400	$11/88 = 0.1250$	$14/88 = 0.1591$
500	$6/88 = 0.0682$	$11/88 = 0.1250$
600	$5/88 = 0.0568$	$9/88 = 0.1023$
700	$1/88 = 0.0114$	$10/88 = 0.1136$
800	$0/88 = 0.0000$	$8/88 = 0.0909$
900	$0/88 = 0.0000$	$6/88 = 0.0682$
1000	$0/88 = 0.0000$	$4/88 = 0.0455$
1100	$0/88 = 0.0000$	$5/88 = 0.0568$
1200	$0/88 = 0.0000$	$4/88 = 0.0455$
1300	$0/88 = 0.0000$	$4/88 = 0.0455$
1400	$0/88 = 0.0000$	$4/88 = 0.0455$
1500	$0/88 = 0.0000$	$3/88 = 0.0341$
1600	$0/88 = 0.0000$	$2/88 = 0.0227$
1700	$0/88 = 0.0000$	$2/88 = 0.0227$
1800	$0/88 = 0.0000$	$1/88 = 0.0114$
1900	$0/88 = 0.0000$	$1/88 = 0.0114$
2000	$0/88 = 0.0000$	$2/88 = 0.0227$

Table 5.1: False negative rates for the Sun and Bayesian algorithms.

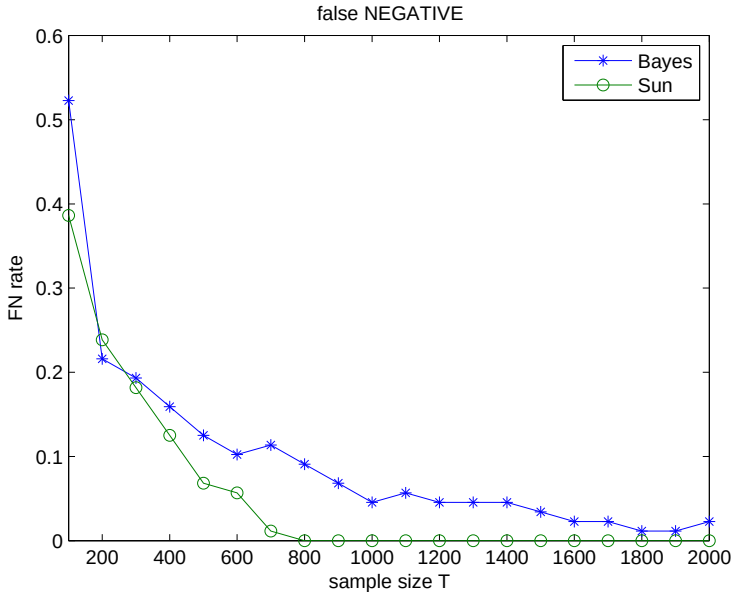

Figure 5.2: The development of the FN rates.

$d = 42, T$	Sun - false positive	Bayes - false positive
100	$119/1676 = 0.0710$	$332/1676 = 0.1981$
200	$134/1676 = 0.0800$	$303/1676 = 0.1808$
300	$115/1676 = 0.0686$	$245/1676 = 0.1462$
400	$113/1676 = 0.0674$	$184/1676 = 0.1098$
500	$89/1676 = 0.0531$	$172/1676 = 0.1026$
600	$101/1676 = 0.0603$	$147/1676 = 0.0877$
700	$103/1676 = 0.0615$	$151/1676 = 0.0901$
800	$99/1676 = 0.0591$	$131/1676 = 0.0782$
900	$89/1676 = 0.0531$	$110/1676 = 0.0656$
1000	$102/1676 = 0.0609$	$98/1676 = 0.0585$
1100	$100/1676 = 0.0597$	$94/1676 = 0.0561$
1200	$87/1676 = 0.0519$	$103/1676 = 0.0615$
1300	$77/1676 = 0.0459$	$95/1676 = 0.0567$
1400	$79/1676 = 0.0471$	$82/1676 = 0.0489$
1500	$96/1676 = 0.0573$	$87/1676 = 0.0519$
1600	$82/1676 = 0.0489$	$94/1676 = 0.0561$
1700	$81/1676 = 0.0483$	$95/1676 = 0.0567$
1800	$93/1676 = 0.0555$	$103/1676 = 0.0615$
1900	$86/1676 = 0.0513$	$98/1676 = 0.0585$
2000	$88/1676 = 0.0525$	$94/1676 = 0.0561$

Table 5.2: False positive rates for the Sun and Bayesian algorithms.

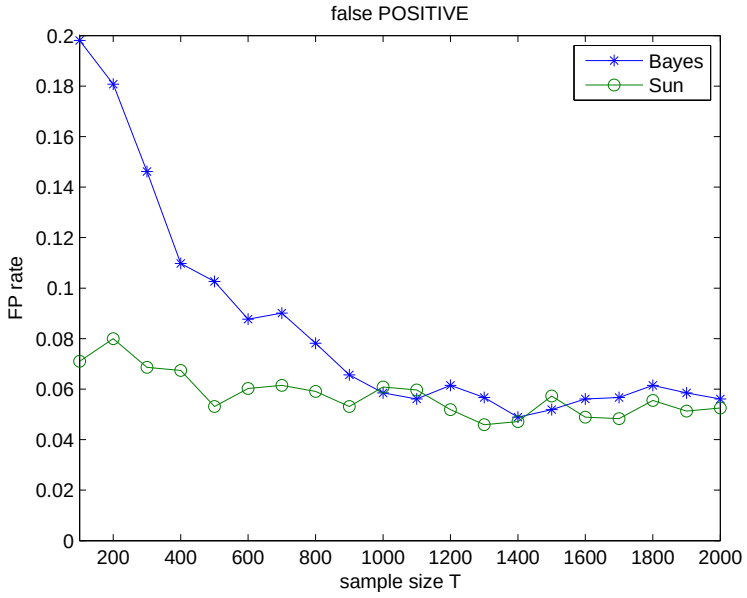

Figure 5.3: The development of the FP rates.

6 Discussion

The main goal of the presented work was the implementation of the methodology provided by [Sun 2014] and reproduction of its results. The main problem we have found with the Sun's approach is its computational complexity. We were able to recalculate his results on networks with 20 nodes and $T = 1000$ samples. However, computation for a network with $d = 50$ and $T = 2000$ took 10 hours on a single-processor computer. Hence, we were not able to undergo Sun's experiments in the whole extent ($d = 200$, $T = 2000$ and 20 runs for each experiment).

On the other hand, the Bayesian approach that we have introduced in this context is substantially faster (2 hours for $d = 150$, $T = 2000$). Using this approach we were able to discover the same patterns as reported by Sun, i.e., decrease of the fallse negative error to zero with the increasing number of samples. The speed of decrease does not depend on the size of the network; and the false positive error saturates at the selected quantile θ of the permutation test.

To overcome the problem of computational burden in the case of the Sun's approach we switch to a parallel computation in Python environment. In this setting we were able to perform effectively computation and compare results from both approaches.

The main finding is that for short lengths i.e., for $T \leq 1000$, the Sun's approach works substantially better than the Bayesian one. For $T > 1000$ both approaches are comparable in the false positive rates. In terms of the false negative rates the approaches start to be comparable for $T \geq 1800$. General observation is that the Sun's approach is superior, but much more computationally intensive, for shorter data. The difference is vanishing with increasing length of data with the benefit of much less computational intensity in the Bayesian case.

Finally, a promising idea is the introduction of the statistical test that allows testing significance of whole causal matrices (networks), instead of only individual causal links.

Acknowledgment

This study was supported by the grant COST LD13002 of the Ministry of Education, Youth and Sports of the Czech Republic and by the Czech Science Foundation project No. GA13-23940S.

Bibliography

[Sun 2014] J. Sun, D. Taylor, E. M. Bollt. *Causal Network Inference by Optimal Causation Entropy*, arXiv:1401.7574v1

[Anděl 1976] J. Anděl. Statistická analýza časových řad, SNTL 1976 *In Czech*

[Hlinka et. al. 2014] Hlinka, J., Hartman, D., Jajcay, N., Vejmelka, M., Donner, R., Marwan, N., Kurths, J., Paluš, M. *Regional and inter-regional effects in evolving climate networks*, Nonlinear Processes in Geophysics, 2(21), 2014, pp. 451–462

Appendix A

1 fbayes.m

```
function [fnr,fpr] = fbayes(X,A,r,th)
 d=size(A,1);
 [U,R22,V2]=volB(X',1);
 AM=zeros(d,d);
 Vqc=zeros(d,d);
 Vqr=zeros(d,d);
 for j=1:d, j,
  %tic,
  [Vqc(:,j),Vqr(j,:)]=permB(j,X,r,th);
  %toc,
 end;
 Vq=(Vqc+Vqr)/2;
 for i=1:d, for j=1:d;
  if abs(U(i,j))>Vq(i,j), AM(i,j)=U(i,j); end;
 end; end;
 [A AM];
 [fnr,fpr]=fp(A,AM);
end

```

2 volB.m

```
function [U,R22,V2] = volB(X,n)
[T,d]=size(X);
C=zeros(d,n*d);
S=zeros(n*d,n*d);
C0=zeros(d,d);
for t=n+1:T,
 xt=X(t,:)';
 xot=[]; for i=1:n, xot=[xot; X(t-i,:)']; end;
 C=C+xt*xot';
 S=S+xot*xot';
 C0=C0+xt*xt';
end;
S1=[C0 C;C' S];

iS=inv(S);
nS=size(iS);
R22=iS([nS-d+1:nS],[nS-d+1:nS]);

V2=(T-n-n*d+2*d)^(0.5)*(C0-C*iS*C')^(-0.5);

U=C*inv(S);
end
```

3 permB.m

```
function [Vqcj,Vqrj] = permB(j,X,r,th)
 [d,T]=size(X);X0=X;
 Urc=zeros(d,r);
 Urr=zeros(r,d);
 for k=1:r,
  X=X0;
  X(j,:)=X(j,randperm(T));
  [U]=volB(X',1);
  Urc(:,k)=U(:,j);
  Urr(k,:)=U(j,:);
 end;
 Vqcj=quantile(abs(Urc'),th)';
 Vqrj=quantile(abs(Urr ),th);
end
```