Jagiellonian University

Master Thesis

The Markov Blanket Concept in Bayesian Networks and Dynamic Bayesian Networks and Convergence Assessment in Graphical Model Selection Problems

Tomasz Kułaga

October 2006

Contents

Chapter 1

Introduction

Relations and dependencies between some entities, such as genes in genetics or any others, can be expressed in a very convenient and clear way by use of the Bayesian Network idea. It is a nice graphical representation in which edges represent direct relations between nodes. Having some experimental data we are trying to find the best graphical structure that would explain the dependencies inside the data set. For a small number of nodes, like about 5, we are able to consider all possible models, calculate for them the posterior model probabilities, and choose the best one or the best few. But where the number of nodes grows, the number of possible models explodes at exponential rate. The solution to this problem is to use the Markov chain Monte Carlo simulation to obtain an approximate sample from the posterior model distribution. In chapter 2 I introduce the concept of Bayesian Network and explain how the MCMC simulation is performed in this framework by the two most known methods: Metropolis-Hastings algorithm and Gibbs sampler.

However, the Bayesian Network concept has some limitations. The graphical structure that it represents has to be a proper Directed Acyclic Graph i.e. it cannot consist of cycles or loops. For example the edge starting and ending in the same node is forbidden in a Directed Acyclic Graph. But it is not hard to imagine such a situation where the possibility of these kind of edges is advisable. In a problem of discovering relationships between genes, sometimes it is the case that some particular gene can be self regulated, maybe together with some other genes. So the edge which starts and ends in the node which represents this gene should be present in a proper model. For this kind of applications the data set, on which basis we are trying to find the best model, is also different. The observations have to be collected over some time points. We cannot see the result of some dependency instantaneously, when nodes are influencing themselves. Formally the data is a sequence of consecutive observations. One idea of representing such a set of relationships is called a Dynamic Bayesian Network. In chapter 3 I introduce this concept in details and also explain how the MCMC simulation can be performed in this framework. For more details on this topic see Husmeier (2003).

In chapter 4 first I introduce the concept of a Markov Blanket for the BN, then I define this object for the dynamic structure. Finally I investigate a very interesting algorithm proposed by Riggelsen (2005) which is based on the idea of the Markov Blanket and which is used to perform MCMC simulation. However, although the proposition seems to be very attractive, especially in the light of speed of convergence to the limit distribution, unfortunately it turns out that this algorithm still needs some corrections and cannot be used in the present form because it gives improper results. Moreover I show, that the way to make this algorithm work properly is not so easy. I propose and test one of the possible corrections in practice. Although I justify that a new algorithm is correct, it turns out that the speed of simulation with use of this algorithm is not satisfactory. It still needs some optimization work which can be the further work in this area.

An important issue in the MCMC context is the convergence assessment. In other words when can we stop the simulation to have some confidence that the output can really be seen as taken from the limit distribution. There has been many methods proposed over the last years but not all of these can be applied in the graphical model selection framework. The model space is very specific. A model consists of many edges which can take only one of two values, presence or absence. One of the methods appropriate for this problem, the Chi-squared test, is presented in chapter 5. I also present some aspects of this test like making the output of the simulation approximately independent, which is an important assumption for this statistical test. The convergence assessment technique, and later deduction about the true structure, is performed not on the particular models sampled during the simulation, but on the labels which are assigned to each of the models. This generalization is a new idea which I propose for this kind of framework. The reason for this operation is that it allows us to significantly reduce the huge model space by giving the same label to a group of models and thus to avoid the super-exponential explosion of the size of the model space. On the other hand, it causes loss of information with comparison to the Markov chain defined on the models. However, we have no restriction on how this labelling function should look like so the reduction can be done in many different ways, depending on the problem we are interested in.

The algorithm presented in chapter 4 is implemented in Matlab (file 'dbn_mb_sampling.m' together with support file 'log_marg_prob_node_new.m') and it uses the Bayes Net Toolbox (BNT) which is needed to work properly. BNT can be downloaded from:

http://bnt.sourceforge.net/usage.html

The results from the simulation are stored in text file and next the result file is processed by perl script (file 'process.pl') which returns values of appropriate statistics. These values are again recalculated to obtain p-values with use of a C++ program (file 'p_val.cpp'). This program requires the dcdflib library which can be downloaded from:

http://www.csit.fsu.edu/ burkardt/cpp_src/dcdflib/dcdflib.html

Chapter 2

Bayesian Networks

A Bayesian Network (BN) is a very convenient and easy way of representing dependencies between some entities. In our case these entities are random variables but BNs can also be used in other applications. By such a network we mean a Directed Acyclic Graph (DAG) where nodes correspond to random variables and directed edges between nodes that represent conditional dependencies together with conditional distributions describing these dependencies. The edge from node X to Y is present when values of Y depends directly on values of X. In such a case we say that X is a parent of Y and Y is a child of X. If there is no edge between two vertices it means that they are independent given parents of both vertices. The word Acyclic means that there is no possibility that any node is its own indirect child (or parent). In other words, there is no directed path starting and ending in the same node. Let us see this on a well known example.

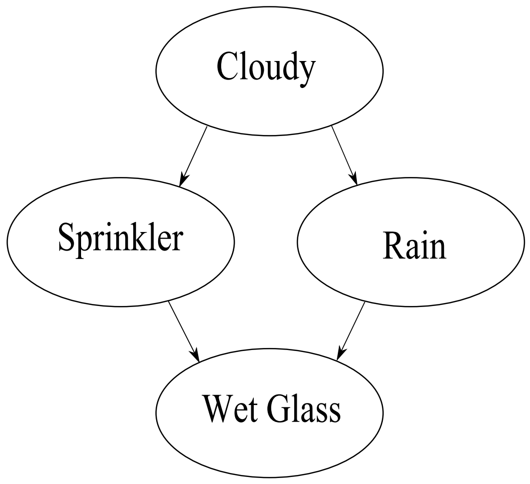

Figure 2.1: Example of a Bayesian Network

The graphical structure consist of the set of nodes {C (=Cloudy), S (=Sprinkler), R (=Rain), W (=Wet glass)} and the set of directed edges {(C, S),(C, R), (S, W), (R, W)}. Nodes in this example are discrete and take only two values $True$ or $False$. To completely define Bayesian Network we should also define conditional probabilities $P(X|parents(X))$ like $P(S = True|C = True)$ etc.

From now on we will consider Bayesian Networks only for discrete random variables. We treat nodes as these random variables.

Such a structure, besides the fact that it describes dependencies between random variables very clearly and understandable, also gives a way to rewrite joint probability distribution in a more simple way. For nodes $\{X_1, \ldots, X_n\}$ we can always do a factorization as follows

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n-1} P(X_i|X_{i+1}, \ldots, .X_n)P(X_n).$$

If nodes are ordered in such a way that for each node the indices of its parents are higher then the index of the node itself (we can do that since there are no directed cycles in the graph), then using the fact that nodes are only conditioned on their parents we can rewrite formula for the joint probability distribution as follows

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n-1} P(X_i|parents(X_i))P(X_n).$$

In our example this takes the form

$$P(C, R, S, W) = P(C)P(S|C)P(R|C)P(W|S, R).$$

2.1 Learning

Learning a Bayesian Network consists of discovering an optimal set of edges and then finding the best parameters that will describe it. We do this on the basis of the data D. The data consist of observations of the values of random variables. Because the set of nodes does never change, as a model M we mean some particular set of edges between these nodes. Finding the best parameters describing optimal set of edges is not so hard or crucial in the learning process. The first and main step focuses on finding the best model in the model space $\mathcal{M}$ that has maximum a posteriori distribution

$$\widehat{M} = argmax_{M \in \mathcal{M}} P(M|D),$$

where D denote the data. For this we take a Bayesian approach.

We consider networks with n discrete random variables (nodes) $(X_1, \ldots, X_n)$. During the process of finding the best structure we need to equip each model with the conditional distributions and their parameters with some prior distribution. Let Θ denote these parameters. For prior distribution we choose the following product Dirichlet distribution

$$P(\Theta|M) = \prod_{i=1}^{n} \prod_{\pi_i} Dir(\theta_{X_i|\pi_i}|\alpha),$$

where π_i runs through possible values of parents of node X_i and α is the vector of hyper parameters that represents the priori counts for each configuration of nodes and its parents values $\alpha(x_i, \pi_i)$. We see that $\alpha(\pi_i) = \sum_{x_i} \alpha(x_i, \pi_i)$. So we have that

$$P(\Theta|M) = \prod_{i=1}^{n} \prod_{\pi_i} Dir(\theta_{X_i|\pi_i}|\alpha) = \prod_{i=1}^{n} \prod_{\pi_i} C(i, \pi_i, \alpha) \prod_{x_i} \theta_{x_i|\pi_i}^{\alpha(x_i,\pi_i)-1},$$

with $C(i, \pi_i, \alpha)$ the normalising factor of Dirichlet distribution

$$C(i, \pi_i, \alpha) = \frac{\Gamma(\alpha(\pi_i))}{\prod_{x_i} \Gamma(\alpha(x_i, \pi_i))},$$

where $\Gamma(\cdot)$ is the gamma function.

The posterior distribution for parameters is again product Dirichlet and takes the data D into account. We denote by d the vector of counts for a particular configuration of nodes and parents values in data i.e. $d(x_i, \pi_i)$. So

$$P(\Theta|D, M) = \prod_{i=1}^{n} \prod_{\pi_i} Dir(\theta_{x_i|\pi_i}|\alpha + d) = \prod_{i=1}^{n} \prod_{\pi_i} C(i, \pi_i, \alpha + d) \prod_{x_i} \theta_{x_i|\pi_i}^{\alpha(x_i,\pi_i)+d(x_i,\pi_i)-1}.$$

We can also rewrite posteriori distribution for parameters as follows

$$P(\Theta|D, M) = \frac{P(\Theta, D, M)}{P(D, M)} = \frac{P(D|\Theta, M)P(\Theta, M)}{P(D, M)} = \frac{P(D|\Theta, M)P(\Theta|M)}{P(D|M)}.$$

And since

$$P(D|\Theta, M) = \prod_{i=1}^{n} \prod_{\pi_i} \prod_{x_i} \theta_{x_i|\pi_i}^{d(x_i,\pi_i)},$$

thus we have now close formula for marginal likelihood

$$\begin{aligned} P(D|M) &= \frac{P(D|\Theta, M)P(\Theta|M)}{P(\Theta|D, M)} \\ &= \frac{\displaystyle\prod_{i=1}^{n} \prod_{\pi_i} C(i, \pi_i, \alpha) \prod_{x_i} \theta_{x_i|\pi_i}^{\alpha(x_i,\pi_i)+d(x_i,\pi_i)-1}}{\displaystyle\prod_{i=1}^{n} \prod_{\pi_i} C(i, \pi_i, \alpha + d) \prod_{x_i} \theta_{x_i|\pi_i}^{\alpha(x_i,\pi_i)+d(x_i,\pi_i)-1}} \\ &= \prod_{i=1}^{n} \prod_{\pi_i} \frac{\Gamma(\alpha(\pi_i))}{\Gamma(\alpha(\pi_i) + d(\pi_i))} \prod_{x_i} \frac{\Gamma(\alpha(x_i, \pi_i) + d(x_i, \pi_i))}{\Gamma(\alpha(x_i, \pi_i))}. \end{aligned}$$

Applying Bayes' law allows as to compute the posterior distribution for each model as follows

$$P(M|D) = \frac{P(D|M)P(M)}{\displaystyle\sum_{m \in \mathcal{M}} P(D|m)P(m)}.$$

However here arise some problems. The main one is the computational burden concerning computing the denominator which is equivalent to computing this posterior distribution for each model. When we consider Bayesian Networks with n nodes the number of possible models is of order 2^{n^2}. For some applications like genetic regulatory interactions the number of nodes we consider can be 20, 50 or 100 and the problem becomes computationally infeasible. The second problem is, that we can discover that more than one model has approximately the maximum a posteriori distribution and that these models differ widely. What we can do and what is also very interesting in applications is to compute a posteriori value of some features like having some particular edge from node X_i to X_j (1 if yes and 0 otherwise) or any other feature that we can compute from a model. If we denote this feature by Δ then we can average over the whole model space

$$\mathbb{E}[\Delta(M)|D] = \sum_{m \in \mathcal{M}} \Delta(m)P(m|D).$$

However we still have to handle this computational issue. A possible solution for this problem is to use Markov chain Monte Carlo (MCMC) simulation which allows us to sample from this posterior distribution.

2.2 MCMC with Bayesian Networks

The main idea of Markov chain Monte Carlo simulations is to create such a Markov chain that its invariant distribution will be the distribution of interest π. A sufficient condition for that is to satisfy detailed balanced equation which has the following form

$$p(M_i, M_j)\pi(M_i) = p(M_j, M_i)\pi(M_j),$$

where $p(M_i, M_j)$ is the probability of changing from state M_i to state M_j. If this condition is satisfied, π appears to be the invariant distribution of the chain. Moreover we need to satisfy some assumptions to be sure that this Markov chain will converge to this distribution. It is sufficient to show that the probability of staying in the same state is positive (which guarantees aperiodicity of the chain) and also that the probability of reaching any other state after a finite number of transitions is positive (the chain is irreducible). Convergence to the invariant distribution is guaranteed by Ergodic Theorem. For details and proof see Jakubowski and Sztencel (2000).

In order to perform MCMC such that the distribution of interest is the stationary distribution, this distribution needs to be known up to a constant. There are two well known methods to perform MCMC: Metropolis-Hastings algorithm and Gibbs sampling. We will now go to discuss them in more details.

2.2.1 Metropolis-Hastings algorithm

In our case the distribution of interest is the posterior distribution of the model, in particular the posterior distribution of the set of edges.

When travelling the model space first we choose a new model using some arbitrary proposal distribution $q(M_{t+1}|M_t)$. This distribution is usually considered as uniform for a neighbourhood of the graph that we are visiting and the graph itself. The Neighbourhood is constructed by reversing, deleting or adding a single edge with restriction that every structure must be DAG. After generating a new model we accept this move with acceptance probability given by formula

$$a(M_t, M_{t+1}) = \min\left\{1, \frac{q(M_t|M_{t+1})P(M_{t+1}|D)}{q(M_{t+1}|M_t)P(M_t|D)}\right\}.$$

When the step is rejected we stay in the same state and a new proposal is generated. We will show that the detailed balance equation holds. Let us consider two different models $M^{'}$ and $M^{''}$. The probability of transition from $M^{'}$ to $M^{''}$ is given by formula $p(M^{'}, M^{''}) = q(M^{''}|M^{'})a(M^{'}, M^{''})$. Now we have that

$$\begin{aligned}
p(M^{'}, M^{''})P(M^{'}|D) &= q(M^{''}|M^{'})a(M^{'}, M^{''})P(M^{'}|D) \\
&= q(M^{''}|M^{'})\min\left\{1, \frac{q(M^{'}|M^{''})P(M^{''}|D)}{q(M^{''}|M^{'})P(M^{'}|D)}\right\}P(M^{'}|D) \\
&= q(M^{'}|M^{''})\min\left\{\frac{q(M^{''}|M^{'})P(M^{'}|D)}{q(M^{'}|M^{''})P(M^{''}|D)}, 1\right\}P(M^{''}|D) \\
&= p(M^{'}, M^{''})P(M^{'}|D).
\end{aligned}$$

So indeed $P(M|D)$ is the invariant distribution of the chain.

We can simplify the acceptance ratio

$$\begin{aligned}
a(M_t, M_{t+1}) &= \min\left\{1, \frac{q(M_t|M_{t+1})P(M_{t+1}|D)}{q(M_{t+1}|M_t)P(M_t|D)}\right\} \\
&= \min\left\{1, \frac{q(M_t|M_{t+1})P(D|M_{t+1})P(M_{t+1})P(D)}{q(M_{t+1}|M_t)P(D|M_t)P(M_t)P(D)}\right\} \\
&= \min\left\{1, \frac{q(M_t|M_{t+1})P(D|M_{t+1})P(M_{t+1})}{q(M_{t+1}|M_t)P(D|M_t)P(M_t)}\right\}.
\end{aligned}$$

The probabilities $P(D)$ cancel out. If we assume the uniform prior distribution for models then probabilities $P(M_t)$ and $P(M_{t+1})$ also cancels out. But sometimes when we have some knowledge on the topic it would be more advisable to put some other prior distribution for the models. We point out that in some applications, like genetics, in which the number of considered nodes is big and the data is sparse, the prior probability which we put on the models plays important role.

For this sampler we can reach any model with finite number of steps with positive probability so the chain is irreducible. In every step we can choose with positive probability to stay in the current state which ensures aperiodicity. Due to ergodic theorem we know that this chain converges to posterior model distribution.

2.2.2 Gibbs sampling

Let us consider the set $\mathcal{E}$ of all possible edges in a graph with n nodes, thus $\mathcal{E} = \{E_1, \ldots, E_c\}$, where $c = \frac{n(n-1)}{2}$. Each edge can take one direction or can be absent. We use a notion e_{-i} to denote $e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_c$. Given some visitation scheme or doing this in a random way we sample each edge of $\mathcal{E}$ using full conditionals given any other edge and data:

$$\begin{aligned} P(E_i|e_{-i}, D) &= \frac{P(E_i, e_{-i}|D)}{\sum\limits_{e_i} P(e_i, e_{-i}|D)} \\ &= \frac{P(D|E_i, e_{-i})P(E_i, e_{-i})P(D)}{\sum\limits_{e_i} P(D|e_i, e_{-i})P(e_i, e_{-i})P(D)} \\ &= \frac{P(D|E_i, e_{-i})P(E_i, e_{-i})}{\sum\limits_{e_i} P(D|e_i, e_{-i})P(e_i, e_{-i})}, \end{aligned} \tag{2.1}$$

where the terms concerning the model prior distribution cancel out if this distribution is assumed to be uniform. To show that detailed balance equations holds let us consider two models which differ only on one edge $M^{'} = \left\{e_1, \ldots, e_i^{'}, \ldots, e_c\right\}$ and $M^{''} = \left\{e_1, \ldots, e_i^{''}, \ldots, e_c\right\}$ so that

$$p(M^{'}, M^{''}) = P(e_i^{''}|e_{-i}, D).$$

Transitions are allowed only for such a models. Now we have that:

$$\begin{aligned} p(M^{'}, M^{''})P(M^{'}|D) &= \frac{P(e_i^{''}, e_{-i}|D)}{\sum\limits_{e_i} P(e_i, e_{-i}|D)} P(e_i^{'}, e_{-i}|D) \\ &= \frac{P(e_i^{'}, e_{-i}|D)}{\sum\limits_{e_i} P(e_i, e_{-i}|D)} P(e_i^{''}, e_{-i}|D) \\ &= p(M^{''}, M^{'})P(M^{''}|D), \end{aligned}$$

so $P(M|D)$ is the invariant distribution of the chain.

When sampling we consider only edges that form together a DAG. As we see the chain is irreducible because the probability of going to any other state is positive. We sample edges in some order or in a random way so every edge can be changed. For some fixed order there can be, however, a problem that we cannot add some edge to graph because the addition makes a cycle in the graph. But we still can do it in the next visitation if other edges from the potential cycle will be deleted or reversed. In every draw we can also stay in the same state with positive probability which ensures aperiodicity. Due to ergodic theorem we know this chain converges to posteriori model distribution.

When we sample an edge the probabilities in numerator and denominator are very similar and factorize in such a way that most of the factors cancel out. The only ones that stay are those concerning vertices of the drawn edge i.e. only factors for vertices that have a different parent set in the proposed model.

In fact we can also treat Gibbs sampling as a special case of Metropolis Hastings algorithm. The proposal distribution $q(M_{t+1}|M_t)$ is the full conditional from the Gibbs sampler $P(E_k|e_{-k}, D)$. For the proposed step from $M^{'} = \{e_k^{'}, e_{-k}\}$ to $M^{''} = \{e_k^{''}, e_{-k}\}$ the acceptance probability simplifies in the following manner

$$\begin{aligned}
a(M^{'}, M^{''}) &= \min\left\{1, \frac{q(M^{'}|M^{''})P(M^{''}|D)}{q(M^{''}|M^{'})P(M^{'}|D)}\right\} \\
&= \min\left\{1, \frac{\dfrac{P(e_k^{'}|e_{-k}, D)}{\sum\limits_{e_k} P(e_k|e_{-k}, D)} P(e_k^{''}, e_{-k}|D)}{\dfrac{P(e_k^{''}|e_{-k}, D)}{\sum\limits_{e_k} P(e_k|e_{-k}, D)} P(e_k^{'}, e_{-k}|D)}\right\} \\
&= \min\left\{1, \frac{P(e_k^{'}|e_{-k}, D)P(e_k^{''}|e_{-k}, D)P(e_{-k}|D)}{P(e_k^{''}|e_{-k}, D)P(e_k^{'}|e_{-k}, D)P(e_{-k}|D)}\right\} \\
&= 1.
\end{aligned}$$

Chapter 3

Dynamic Bayesian Networks

Dynamic Bayesian Networks (DBNs) can describe certain stochastic processes that change over time. Let us denote states of the process indexed by time: $\{X[1], X[2], \ldots\}$. Each time point which we call time-slice $X[t]$ is a random vector $X[t] = \{X_1[t], \ldots, X_n[t]\}$. The structure of a DBN is similar to that of a BN. We have nodes that represent random variables $X_i[t]$, $i = 1, \ldots, n$ and edges between them to represent the conditional dependencies. In the general case one consider intra-slice dependencies which are described by a BN within a time-slice as well as inter-slice dependencies which are connections between consecutive or even more distant slices. Since states of the process are indexed by time points and we cannot move back in time the inter connections always point to the future. We can see this on an example.

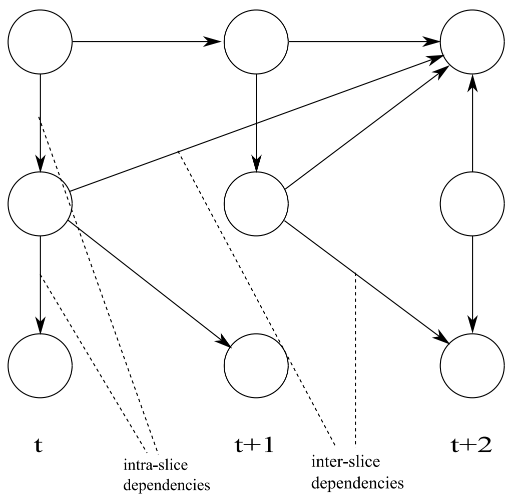

Figure 3.1: Example of a Dynamic Bayesian Network.

However adding more distant dependencies causes a great increase of the complexity of the model. To fully characterize a DBN one has to give also conditional distributions, but as in the BN case we are interested in discovering the unknown structure receding optimization of parameters for the background. On the other hand, we can only perform finding the best parameters for some specified graph and for this reason we need to find the optimal structure first. We focus on the latter issue.

We concentrate on models with no intra-slice dependencies. Moreover we consider only processes that are *Markovian* i.e. the future step does not depend on the past but only on the present state.

$$P(X[t+1]|X[t],\dots,X[1]) = P(X[t+1]|X[t]) \quad \forall t \in \mathbb{N}.$$

In other words, we assume that inter-slice dependencies are allowed only between adjoining time-slices. We will try to find the best model that describes transitions between slices made in time. The word 'dynamic' does not describe a dynamic process but is used only to emphasize the possibility of inter-slice dependencies in order to distinguish a DBN from a BN where these dependencies are not allowed. The process itself is assumed to be stationary which means

$$P(X[t+2]|X[t+1]) = P(X[t+1]|X[t]) \quad \forall t \in \mathbb{N}.$$

Dynamic Bayesian Networks do not change over time but are constant and our goal is to discover them.

So in our framework a DBN reduces to an inter graph which contains of two adjoining time-slices $X[t]$ and $X[t+1]$ and a set of edges between them to represent inter-slice dependencies. As a model M we will consider only the set of edges of inter graph because the set of nodes is fixed and does not change at all. We can also identify our model in a very clear and convenient way with the $n \times n$ binary matrix B where $B_{ij} = 1$ when there is a connection from $X_i[t]$ to $X_j[t+1]$ and $B_{ij} = 0$ otherwise. Since the process is stationary we can look only at two first time-slices $X[1]$ and $X[2]$. Let us see this on an example. We have some hypothetical inter graph

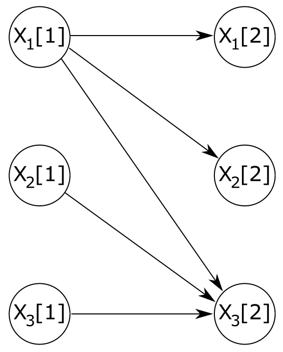

Figure 3.2: Example of the inter graph.

and the corresponding matrix B:

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

There are some advantages of using such restricted DNBs with no intra-slice dependencies and with connections only between consecutive time-slices over ordinary BNs. The first advantage is that in the BN case each graph is not always distinct from the others in terms of the marginal likelihood, and hence of posterior probability distribution given the data. It is described by equivalence class of DAGs. This equivalence class consists graphs which has the same skeleton but different *v-structure*. For details see Verma and Pearl (1990) and Chickering (2002a). This equivalence class can be represented by a completed Partially Directed Acyclic Graph that contains both directed and undirected edges. For further studies see Chickering (2002b). An undirected edge in PDAG means that the posterior probability for both graphs, one with the edge directed in one way and the second one with the edge pointed backwards, are the same. This entails that we lose substantial information about direction of the edge and therefore about causal interactions between variables. This is not the case in a DBN because edges always point to the future. There is no possibility of reversing an edge which would mean that the result precedes the cause.

A second and even more important advantage is that in some applications we are interested in graphs that allow cycles in the structure. Since feedback is an essential feature of biological systems it would be advisable to consider structures with cycles and loops. By a loop we mean the edge that starts and ends in the same node. When we model some process by a DBN with only inter-slice dependencies we assume that interactions are not instantaneous but that it takes some time for one variable to influence the other and that the result can be seen in the next time point, and the same concerns loops or cycles where feedback is delayed. So it does not pose a problem having such a network with cycles or loops to unfold it in time to create a dynamic graph with inter-slice dependencies. Let us see this on an example.

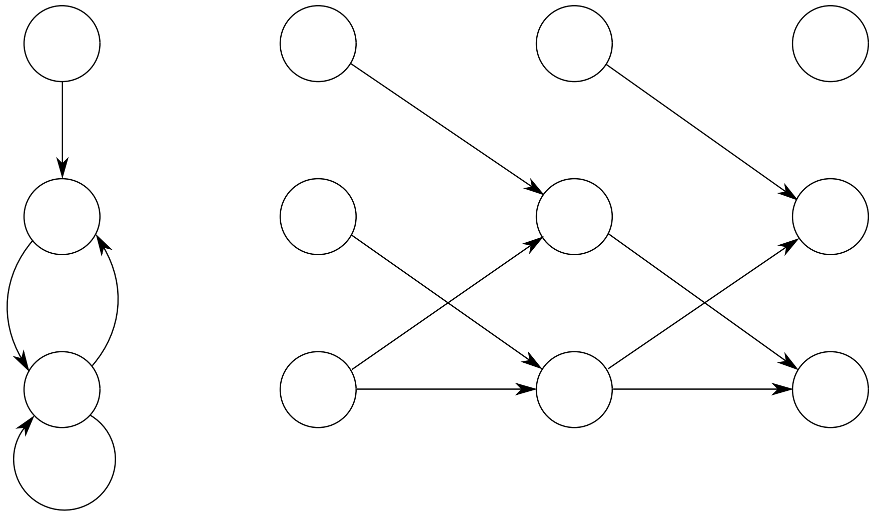

Figure 3.3: Graph with cycles and loops and corresponding unfolded dynamic structure.

A next advantage of simplified DBNs is that the model space is more simple. If we sample possible graphs using an MCMC sampler in the static case, then for every step we have to check if the graph that we consider is a DAG. Especially for the Metropolis Hastings algorithm where we need to verify this issue for all neighbouring graphs in order to compute the proposal distribution. For bigger number of nodes this can be really time consuming which makes our sampler running extremely slow. For DBNs this problem does not hold due to the fact that we have no restrictions for the graph structure. Every possible graph is allowed.

3.1 MCMC with DBN

The number of possible models for DBN reduced in such a way as described in the previous section, for a set of n random variables $\{X_1, \ldots, X_n\}$ is even bigger than in BN and now equals 2^{n^2}. So it is tempting to use Monte Carlo Markov chain simulation also to this case to avoid a lot of computations that concern finding the normalizing constant for the posterior distribution of the structures given the data. We can do this in the following manner.

In the same way as for BN we deal only with random variables that can take discrete values. Let us denote by D the data set. It consists of the observations collected over T time points:

$$D = \{d[1], \ldots, d[T]\}.$$

Each observation

$$d[t] = \{d_1[t], \ldots, d_n[t]\}$$

contains the values of each random variable X_i observed at time t denoted by $d_i[t]$. The observations are assumed to be collected at consecutive instants. It is an important issue and needs to be pointed out that the data that we deal with has to be dependent. This is significantly different from the BN case where observations were assumed to be i.i.d. In the DBN setting a particular observation at time $t+1$ needs to be related to the previous one collected at time t. An inter graph, in our DBN framework, represents the relations between two adjoining time-points so what we model is an invariable process between consecutive observations in the data.

To perform MCMC simulation we rearrange the data in the following way. We take the pairs of adjoining observations collected at times t and $t+1$ and transform them to the common vector having the form

$$d'[t] = \{d_1[t], \ldots, d_n[t], d_1[t+1], \ldots, d_n[t+1]\} \quad t = 1, \ldots, T-1.$$

The new data set D' contains $T-1$ vectors

$$D' = \{d'[1], \ldots, d'[T-1]\}.$$

Each vector $d'[t]$ is actually the observed transition between two successive time-points t and $t+1$ in the data set D.

It is also possible that we have more than just one data set and every of these sets consist of a sequence of consecutive observations where the dependencies within the data described above hold and are the same for each data set. In other words the same process has generated each of these sets. Let us assume that we have K data sets $D1, \ldots, DK$ where

$$Dk = \{dk[1], \ldots, dk[T_k]\} \quad k = 1, \ldots, K.$$

Each observation is again a vector

$$dk[t] = \{dk_1[t], \ldots, dk_n[k]\}.$$

It contains of observed values of random variables $\{X_1, \ldots, X_n\}$ at time t in data set Dk. We perform the same rearrangement for each data set Dk as for the single data set. New observations take the form

$$dk'[t] = \{dk'_1[t], \ldots, dk'_n[t], dk'_1[t+1], \ldots, dk'_n[t+1]\}$$

for $k = 1, \ldots, K$ and $t = 1, \ldots, T_k - 1$. The new data set D' is formed by gathering all new observations:

$$D' = \{d1'[1], \ldots, d1'[T_1], \ldots, dK'[1], \ldots, dK'[T_K]\}.$$

Having prepared the data we need to transform the models to fit them to the new observation set. The model is represented as the $n \times n$ matrix which is identified with the inter-graph between two first time-slices $X[1]$ and $X[2]$. We join these two slices into one set of nodes

$$\{X_1[1], \ldots, X_n[1], X_1[2], \ldots, X_n[2]\}.$$

We keep the existing edges and what we obtain is a proper DAG structure. Now we can look at these new models as at BNs and use prepared earlier data to score them as described in Chapter 2. Transformations have been made in such a way that the new model reflects transitions between two time-slices and the new data contains information about transitions made between consecutive observations in the data set even if it consists of more than just one sequence of data. So we travel the space of binary $n \times n$ matrices, which is extremely easy since we have no restrictions on such a matrix, and score them using the method presented above.

Chapter 4

Markov Blanket

When performing MCMC simulation with BNs or DBNs one has to choose how to traverse the model space. The simplest way to do this is to look at single edge components. For the Gibbs sampler this means that at every iteration we draw a single edge from the full conditional. The edge can be chosen every time in a random way or by using some visitation scheme. For the Metropolis-Hastings algorithm we move to the neighbouring structure which differs only in one single edge from the structure that we are in. However, the problem arises when we are moving around some local maximum. In other words when we travel through some likely structures that have quite good scores but the scores of the surrounding structures are very low. If we sample edges in the random way or we chose "bad" visitation scheme, we can easily get trapped in a local maximum for a very long time. This can make our sampler almost reducible i.e. the probability of going to some structure which is far away in the model space can be reduced almost to zero due to obstacles related to being trapped in the local maximum. Although theoretically it is guaranteed that we eventually will escape from such a peak in the posterior distribution, in practice this fact can be useless.

In general, to make our MCMC sampler work better, we should consider components that are as "self-contained" as possible. In other words, this should be considered as a unit that cannot be split. However, the BNs and DBNs score of the node depends directly on single edges due to the fact that it is computed on the basis of the parents set, we can also look at more than just one edge component and try to find some sets of edges that form closely relationships inside. One approach proposed by Riggelsen (2005) is based on the Markov Blanket concept and will be our main interest from now on.

We start by introducing the Markov Blanket notion together with some explanation why the set of edges based on this concept can be treated as a single component.

Let us consider a set of random variables $\{X_1, \ldots, X_n\}$. As before, by a model M we mean a set of edges between nodes that represent the random variables $\{X_1, \ldots, X_n\}$ or $\{X_1[1], \ldots, X_n[1], X_1[2], \ldots, X_n[2]\}$ for the BN or DBN case, respectively. For the DBN case we allow only connections between time-slices, leaving the intra-graph empty. We start by introducing the parents set for node X_i.

$$\mathrm{Par}(X_i) = \{X_j : (X_j, X_i) \in M\}.$$

The edges which start in nodes taken from $\mathrm{Par}(X_i)$ and end in node X_i surely belong to the common set of relationships, because they directly explain how the node X_i is influenced. The node X_i is conditionally independent of any other node conditioned on the parents set of X_i. The explanation of impact on the behaviour of the node X_i is complete. Although the dependency goes even further to parents sets of X_i's parents and so on, the impact gets less and less important with every step so we narrow our interest down to the direct dependencies set. Beside edges that explain somehow the behaviour of the node X_i we can also look in the opposite direction and ask how the node X_i influences other nodes in the structure. These dependencies can be expressed as the set of edges which start at the node X_i and end in a node taken from children set of node X_i. We define the children set of node X_i as follows.

$$\mathrm{Ch}(X_i) = \{X_j : (X_i, X_j) \in M\}.$$

As before, dependencies go deeper in the structure through children's children sets, but we only look at direct relationships so we restrict ourselves to the direct children set. But nodes from the children set of the node X_i are not only influenced by the node X_i itself but also by its parents and therefore if we consider edges that came out from node X_i we should also include to this set the edges that end up in X_i's children but start in its children's parents. These edges complete the explanation of the influence of X_i on its children.

Now we formed the set of edges that somehow make up a coherent system that describes the behaviour of the node X_i and its impact on other nodes. But when we travel the model space using MCMC sampler, what we meet is not necessarily the correct structure and the set of edges that we determine may not be true but probably some connections are mistaken and our goal is to learn the true graph. Moreover, when there is more than just one unique best structure but there are graphs equally good with the highest score, we should not get accustomed to the encountered connections. So the idea is to broaden the component that we sample to the set of all edges between nodes involved in the set of edges described above. To define it formally we can use the notion of Markov Blanket. The Markov Blanket in a BN for node X_i which we denote by $\mathrm{MB}(X_i)$ is a set of nodes composed of X_i's parents, its children and parents of its children. Formally the definition of Markov Blanket in a BN, or more general in a graph, is as follows.

$$\mathrm{MB}(X_i) = \mathrm{Par}(X_i) \cup \mathrm{Ch}(X_i) \cup \bigcup_{Y \in \mathrm{Ch}(X_i)} \mathrm{Par}(Y).$$

Let us see this on an example.

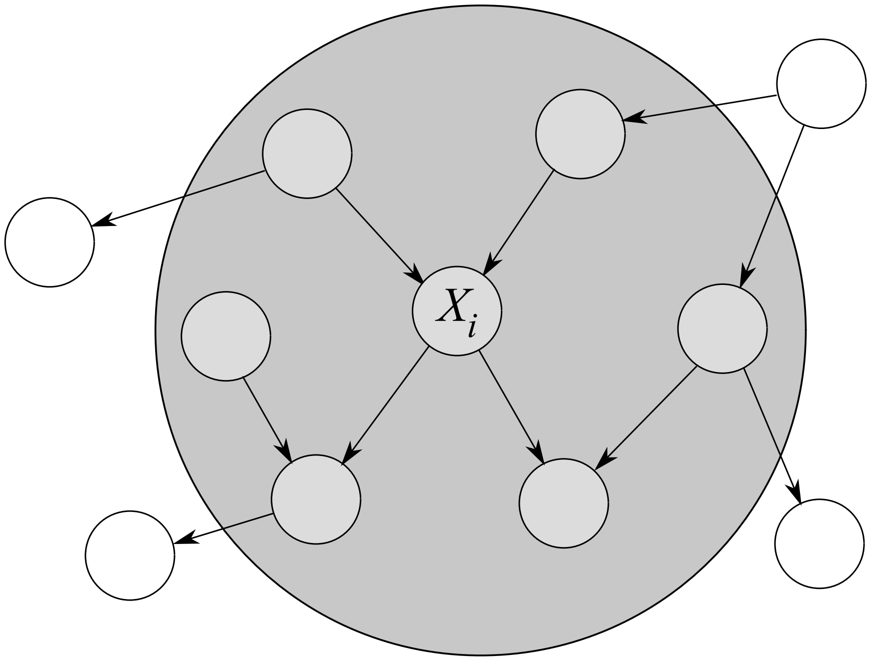

Figure 4.1: Example of Markov Blanket for node X_i.

Our interest is in DBNs, so also for this case we need to introduce the concept of a Markov Blanket. As described in chapter 3 we have two equivalent representations of our dynamic model. The first one is a graph between the first two time slices $X[1]$ and $X[2]$. However, for the Markov Blanket issue this representation has one big disadvantage. Parents that we can have in any situation can only be taken from the first time slice and children from the second one. But when sampling component based on Markov Blanket we want to have a possibility of changing the connections within the component, for example by adding backward edge and therefore making a child from a parent and vice versa. So we merge these two time-slices into one by connecting corresponding nodes from both slices. We join nodes $X_i[1]$ and $X_i[2]$ into one new node X_i but we keep the current edge structure between the nodes. Actually, this is reversing of the unfolding process described in chapter 3 so we call this merging process folding. And then we compute the Markov Blanket for node X_i within the folded graph just as described for BN. We can see that our second representation of dynamic model, binary $n \times n$ matrix B, is in fact the Adjacency Matrix for the folded graph i.e. if there is an edge from X_i to X_j then $B_{ij} = 1$ otherwise $B_{ij} = 0$. By this fact we can define Markov Blanket for the folded graph in alternative way using matrix B.

$$\mathrm{MB}(X_i) = \{X_j : B_{ji} = 1 \vee B_{ij} = 1 \vee \exists_k B_{ik} = B_{jk} = 1\}.$$

Then, if we unfold the graph again to see the dynamic structure between two time-slices, we get a set of nodes which we will call the Markov Blanket for DBN, which consist of two identically sets of nodes, where each of them correspond to the Markov Blanket within a time-slice.

$$\mathrm{MB}(X_i)[r] = \{X_j[r] : X_j \in \mathrm{MB}(X_i)\}, \quad r = 1, 2.$$

Having described what a Markov Blanket for DBN is, we can define the currently relevant edges set for node X_i as

$$\mathcal{E}_i' = \{(X_j[1], X_k[2]) : X_j[1] \in \mathrm{MB}(X_i)[1] \wedge X_k[2] \in \mathrm{MB}(X_i)[2]\}.$$

In this set we consider not only connections between current parents and children in the structure that allows us to determine the Markov Blanket but also all other possible edges within this set. The only restriction is due to the fact, that this is a dynamic structure so parents can be taken only from $X[1]$ and children from $X[2]$.

However, when we consider our new component as $\mathcal{E}_i^{'}$ we restrict ourselves only to the edges between nodes that are present in the current Markov Blanket. But if we look at folded graph, we can deduce that it would be advisable to consider also edges between node X_i and nodes which are outside $\mathrm{MB}(X_i)$ to allow our Markov Blanket to grow (or shrink). We have no confidence that current Markov Blanket contains all important nodes and we can discover true structure only by modifying connections between nodes inside $\mathrm{MB}(X_i)$. So after unfolding again the graph to dynamic structure, we introduce another set of edges which we refer to as a potentially relevant edges set for node X_i given by definition

$$\mathcal{E}_i^{''} = \{(X_j[1], X_k[2]) : X_j[1] = X_i[1] \vee X_k[2] = X_i[2]\} \setminus \mathcal{E}^{'}.$$

Now we come to the notion of the relevant edges set for node X_i which we define as the union of the two previously defined sets.

$$\mathcal{E}_i = \mathcal{E}_i^{'} \cup \mathcal{E}_i^{''}.$$

As a component that we would sample we take $\mathcal{E}_i$. It consists of the coherent set of closely related edges between vertices from Markov Blanket, which we determine in the current graph and the potentially important edges between node of interest and any other node taken from outside Makrov Blanket.

When we use bigger components than just single edge for drawing it is more likely to escape from the local maximum. We move through the model space more rapidly and it is easier to cross the low valleys in the posterior distribution. It is also more favourable from the limit distribution point of view. When we move through the whole model space in a more dispersed manner, we first get the general idea how the posterior distribution looks like and while the sampler is running we improve the shape of this density. If we move slowly we discover more or less correct shape but we still have areas that we don't know anything about.

Another important advantage of sampling with use of the concept of Markov Blanket is that the components that we would sample are not disjoint. The edges that belong to intersection of many relevant edges sets can be seen as more important for the whole structure. So if we consider these edges more often during the sampling process we can have more confidence that the values assigned to these edges are more proper.

4.1 MCMC sampling using the Markov Blanket concept

Riggelsen (2005) proposed an algorithm, that one could call 'block Gibbs sampling', in which he used the idea of Markov Blanket. As a block we mean a self-contained component i.e. the relevant edges set. Now we present this algorithm in its original form. The 'block Gibbs sampling' for every $\mathcal{E}_i$ is performed by the Metropolis-Hastings algorithm applied to single edges inside the component. We repeat Metropolis-Hastings steps for some fixed number of times. Let us call this parameter ρ. In every repetition the proposal distribution $q(M_{t+1}|M_t)$ is the same. With probability ω we switch a random edge inside $\mathcal{E}_i^{'}$ and with probability $1-\omega$ we change a random edge inside $\mathcal{E}_i^{''}$. Then the transition is accepted with probability

$$a(M_t, M_{t+1}) = \min\left\{1, \frac{q(M_{t+1}|M_t)P(M_{t+1}|D)}{q(M_t|M_{t+1})P(M_t|D)}\right\}.$$

We see that proposal probabilities cancel out because for both directions they are the same. We have to switch the same edge to go backwards and the probability of doing this change is the same for both cases. So the acceptance probability simplifies

$$a(M_t, M_{t+1}) = \min\left\{1, \frac{P(M_{t+1}|D)}{P(M_t|D)}\right\}.$$

After doing Metropolis-Hastings steps for ρ times inside one relevant edges set we move to the next one and we repeat the same procedure.

Although it seems that this way of travelling the model space should assure faster convergence as described in the previous section, the problem arises when we want to prove that the Markov chain based on this algorithm converges to the desired limit distribution at all!.

One way of looking at the algorithm presented above, is to treat it as a 'block Gibbs sampling' indeed. When the parameter ρ is big enough, then we can see all of the steps performed by the Metropolis-Hastings inside the particular relevant edges set $\mathcal{E}_i$ as a separate Markov chain simulation. However its limit distribution will be not the desired posterior distribution for the models but the full conditional conditioned on the remaining edges, denoted by $\bar{\varepsilon}_i$, and the data. The longer we sample inside the particular relevant edges set, the more confidence we have that the last sample is truly representative as a sample from the full conditional.

$$\varepsilon_i \sim P(\mathcal{E}_i|\bar{\varepsilon}_i, D).$$

The first disadvantage of this approach is that we have no clue how big the parameter ρ should be. We do not monitor the convergence of the sub-chain inside the relevant edges set. Taking care of this issue could cost us a lot of computational time and make our algorithm unattractive.

The second and more important issue concerns the fact, that in this algorithm the components that are considered to sample depend on the state that we are in during the simulation. For each model we sample only one relevant edges set and it is clear, that these sets depend on the structure of the model.

As described in Gilks, Richardson and Spiegelhalter (1996), chapter 1, it is allowed for the Metropolis-Hastings algorithm to choose a component to sample depending on the current state but in this situation the acceptance probability should be modified. Let us denote the distribution of the components to sample when being in state M as $s_M(I)$ and the proposal distribution for component I as $q_I(M_{t+1}|M_t)$. Then the acceptance probability becomes

$$a(M_t, M_{t+1}) = \min\left\{\frac{s_{M_{t+1}}(I)q_I(M_{t+1}|M_t)P(M_{t+1}|D)}{s_{M_t}(I)q_I(M_t|M_{t+1})P(M_t|D)}\right\}.$$

If we see our algorithm as a 'block Gibbs sampling' then it is clear that the proposal distribution for the component $\mathcal{E}_i$ is approximately the full conditional $\varepsilon_i \sim P(\mathcal{E}_i|\bar{\varepsilon}_i, D)$ and it is always accepted. This fact, together with the reasoning presented above, suggests that our algorithm does not converge to the posterior model distribution. To check this out, we tested two algorithms on a very simple example. The first one was the original Gibbs sampler which sample every edge in the model in some fixed order. The second one was the algorithm presented above with the parameter ρ chosen as 15 and ω as 0.9. We will call it a Markov Blanket algorithm. We tested these algorithms on a simple 3-nodes network. The data consisted of 20 observations produced on the basis of the true network. Such a small application, where the number of possible models is $2^{3^2} = 512$, gives us some advantages. The convergence to the limit distribution is really fast and can be achieved in no more than a few hundreds of iterations. Moreover, we can compute the empirical posterior distribution for all the models and compare the results between two algorithms. The empirical posterior distribution is just the number of times we visited each model divided by the length of the simulation.

$$P_e(M|D) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(M_t = M).$$

To check if these algorithms had converged, we used the Chi-square test, which is described in details in Chapter 5. For each of these algorithms we made 3 independent simulations and compared them together. It turned out that the algorithms had converged immediately. For each algorithms, the hypothesis that 3 independent runs of length 3000 give the same empirical posterior distribution for the models, was accepted with p-value equals 1 as presented below.

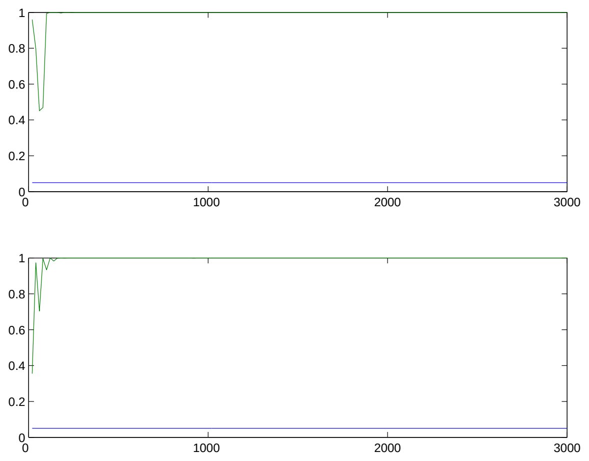

Figure 4.2: P-values for the Chi-square test for the hypothesis that 3 independent empirical posterior distributions are identically distributed. The Gibbs sampler on the top and Markov Blanket algorithm at the bottom.

Such a small amount of nodes in a network, and thus only 512 distinct models, allows us to draw the empirical posterior distribution for each of these algorithms in a details.

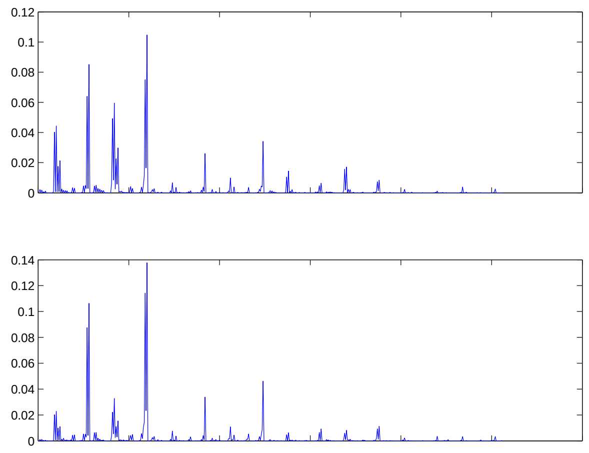

Figure 4.3: The empirical posterior models distributions. The Gibbs sampler on the top and Markov Blanket algorithm at the bottom.

We see that shapes of these distributions are similar but not the same. Again, we used the Chi-square test, described in Chapter 5, to test if all of these 6 simulations gave the same empirical distributions. It turned out that the difference was significant. P-value obtained by the test was extremely low as presented below.

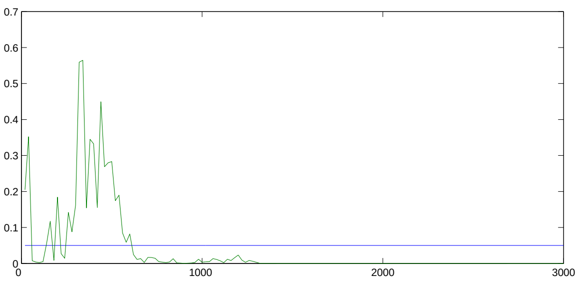

Figure 4.4: P-values for the Chi-square test for the hypothesis that all 6 empirical posterior distributions are identically distributed. Three of them are produced by Gibbs sampler and another three by Markov Blanket algorithm.

The conclusion is that the algorithm is wrong. It produces wrong results. For the 'block Gibbs sampler', if we move from the model M' to M'' by sampling edges inside $\mathcal{E}_i$ using the full conditional $P(\mathcal{E}_i|\bar{\varepsilon}_i, D)$, it can be the case that we can do the same move by sampling another relevant edges set $\mathcal{E}_j$. Probably it will be also the case that in the new model $M^{''}$ the relevant edges set $\mathcal{E}_i$ will be different than in the model $M^{'}$. This implies that the full conditional that we could use for going back is also different. All of these issues should be taken into consideration when computing the acceptance probability for the move in which should appear the probability of choosing particular relevant edges set and probability of going to next or previous state with use of appropriate full conditional. But the problem is that we do not want to compute these full conditionals because they can be really complicated if the relevant edges set is big. We obtain one sample from one of the full conditionals by the sub-simulation inside this set. We cannot repeat this procedure to obtain the probabilities for other full conditionals. So unfortunately, if we look at this algorithm as a 'block Gibbs sampler', there is no easy way to correct it.

4.1.1 New algorithm

We can use the idea of Markov Blanket in the following way. First we get rid of the parameter ρ by setting this value to 1. So we sample inside the relevant edges set only once and then we skip to the next set. Every time we choose which relevant edges set $\mathcal{E}_i$ will be considered with probability $\frac{1}{n}$. Then, as before we choose from which part of the relevant edges set, $\mathcal{E}_i^{'}$ or $\mathcal{E}_i^{''}$, the edge will be chosen with use of the parameter ω. Next we choose one edge E_k from the selected part of the relevant edges set in the uniform fashion. The new state is generated by changing the edge E_k from 1 to 0 or backwards. This is in fact the proposal distribution $q_k(M_{t+1}|M_t)$ which is fully deterministic. Finally, the move is accepted with the acceptance probability. It consists of the probability of choosing the edge for both of the models. For the model M this probability is given by the formula

$$s_M(k) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\omega}{|\mathcal{E}_i^{'}|}\mathbb{I}(E_k \in \mathcal{E}_i^{'}) + \frac{1-\omega}{|\mathcal{E}_i^{''}|}\mathbb{I}(E_k \in \mathcal{E}_i^{''})\right].$$

Then the acceptance probability takes the form

$$\begin{aligned} a(M_t, M_{t+1}) &= \min\left\{1, \frac{s_{M_{t+1}}(k)q_k(M_{t+1}|M_t)P(M_{t+1}|D)}{s_{M_t}(k)q_k(M_t|M_{t+1})P(M_t|D)}\right\} \\ &= \min\left\{1, \frac{s_{M_{t+1}}(k)P(M_{t+1}|D)}{s_{M_t}(k)P(M_t|D)}\right\}. \end{aligned}$$

We see that the proposal distributions both cancel out because they are both equal 1.

Let us present this algorithm with the help of some pseudo-code.

```
1.  for $t$ from 1 to $T$
2.      pick a random vertex $i$ from $1, \ldots, n$
3.      compute $\mathcal{E}_i^{'}$ and $\mathcal{E}_i^{''}$,
4.      sample $u \sim U[0,1]$,
5.      if $u < \omega$ then pick an edge $E_k$ from $\mathcal{E}_i^{'}$,
6.      else pick an edge $E_k$ from $\mathcal{E}_i^{''}$,
7.      end if,
8.      generate new model $M_{t+1}$ by changing the edge $E_k$ from 1 to 0 or from 0 to 1.
9.      compute the acceptance probability $a(M_t, M_{t+1})$,
10.     sample $u \sim U[0,1]$,
11.     if not $u < a(M_t, M_{t+1})$ then $M_{t+1} = M_t$,
12.     end if,
13. end for.
```

4.1.2 Evaluation

In fact this new algorithm gives us a smart and flexible visitation scheme. It changes the edges in more dense regions more often and therefore should converge faster to the desired limit distribution. The edges that are more crucial for the graph, because they influence bigger number of other edges, are sampled more often. It was also justified that the algorithm behave correctly and its limit distribution is in fact the posterior distribution for the models. However, the important issue is that the computation of the acceptance probability takes a lot of time because it requires finding the relevant edges set for each of the vertices. This fact can make the algorithm run slower and thus make it unattractive. However, the gain from using this algorithm seems to be worthy.

Unfortunately, this algorithm was not implemented in this work due to the time restrictions and there was no possibility to test it on some artificial or true data sets. However this can be the next step in the later work.

Chapter 5

Convergence diagnostics

Besides the fact that Markov chain Monte Carlo techniques are very widely used in many areas in a really efficient way, there is still a major problem in assessing the performance of the simulation i.e. gathering the evidence to support the hypothesis that the chain has reached stationarity. There are several methods to assess MCMC sampler performance. For comparative review see Cowles and Carlin (1996) and El Adlouni, Favre and Bobée (2005). However, not every diagnostic method can be used for a DBN framework. Brooks, Giudici and Philippe (2003) proposed a family of nonparametric convergence assessment techniques for general use in MCMC which are also applicable to graphical model selection problem. These methods are based on the comparison of several replications of Markov chains in terms of distance measures due to the fact that if the chains have reached stationarity the distances between these replications should be small.

We will focus on one of the techniques and use it to our DBN framework. It is based on the Chi-squared test for the homogeneity of subpopulations taken from different replications of the chain. As a subpopulation we mean a set of sampled models in each of the replications of the chain. Let us assume that we have J different runs of the MCMC algorithm and that each replication has T sampled models.

We will use the model representation as a binary matrix $n \times n$ (see Chapter 3 for more details). So the model space $\mathcal{B}$ is the set of all binary matrices with size $n \times n$.

5.1 Labelling function $\mathcal{L}$

We introduce a labelling function $\mathcal{L}$ which gives every model in a model space a label which is identified with a natural number.

$$\mathcal{L} : \mathcal{B} \mapsto \{l_1, \dots, l_L\} \subset \mathbb{N}.$$

The reason for using the labelling function is that it allows us to merge more models into a group of models having the same label. We treat this group as a new single state. It is advisable due to the size of the model space. For networks with n nodes the number of possible models is the size of 2^{n^2}. For example for a 25 node network this number grows to the size in the order of 10^{188}. However usually MCMC simulation includes 'only' about 10^5, 10^6 iterations so we see that most of the models will never be visited, but we only travel a really small part of the model space. Moreover, if we want to use the Chi-squared test for the homogeneity of subpopulations taken from few independent MCMC runs, we can be almost sure that, for such a big networks, probability that independent runs visited the same models is almost zero due to the huge number of possible models in relation to the simulation length. So by introducing the labelling function we reduce the size of the state space.

The second reason for using the labelling function will be discussed in more details in a subsequent section. It deals with the Chi-squared test and the fact that for this test to work efficiently it is advisable that the expected number of visits in the particular state is sufficiently big. We will come back to this later.

We will show now that if we look at the labels, instead of the models, a new Markov chain constructed on the basis of the labels still has desired property of convergence to the limit distribution $\pi_{\mathcal{L}}$ which is naturally defined on the label space.

$$\pi_{\mathcal{L}}(l) = \sum_{M \in \mathcal{M} : \mathcal{L}(M) = l} \pi(M).$$

Where π denotes the limit distribution for the original Markov chain. To define the transition probabilities for the new Markov chain we need to take a closer look at the following problem. Let us consider that we want to find the transition probability $p_{\mathcal{L}}(l_i, l_j)$ of going from label l_i to l_j. But being in some state l means, for the original Markov chain, that we are in one of the states M such that $\mathcal{L}(M) = l$. However, we do not know exactly in which one. So in fact moving from the label l_i to the label l_j is a step from one group of models to the other. To define $p_{\mathcal{L}}(l_i, l_j)$ with use of the transition probabilities p from the original Markov chain, we need to know what is the probability that we are in a particular state M, which has label l, given that we are in a group of models having label l. We denote this probability by $P(M|l)$. Using the definition of conditional probability we get the following formula

$$\begin{aligned} P(M|l) &= P(\text{original Markov chain is in } M|\text{new Markov chain is in } l) \\ &= \frac{P(\text{original Markov chain is in } M \text{ and new Markov chain is in } l)}{P(\text{new Markov chain is in } l)} \\ &= \frac{P(\text{original Markov chain is in } M)}{P(\text{original Markov chain is in some state having label } l)}. \end{aligned}$$

We can express the probability that the original Markov chain is in state M in terms of its limit distribution π. The same holds for the probability of being in some state having label l. So we get the following

$$P(M|l) = \frac{\pi(M)}{\sum_{M' \in \mathcal{M}: \mathcal{L}(M')=l} \pi(M')}.$$

To make a move to a state l in a new Markov chain we need to make a step to any state M in the original chain, such that $\mathcal{L}(M) = l$. So now we get to the formula for the transition probability in the new Markov chain

$$p_{\mathcal{L}}(l_i, l_j) = \sum_{M' \in \mathcal{M}: \mathcal{L}(M')=l_i} P(M'|l_i) \sum_{M'' \in \mathcal{M}: \mathcal{L}(M'')=l_j} p(M', M'').$$

Preposition 1 *Suppose we have a Markov chain defined on the finite model space $\mathcal{M}$ with transition probabilities p. Suppose that its invariant distribution is π. Then for any labelling function $\mathcal{L} : \mathcal{M} \mapsto \{l_1, \ldots, l_L\} \subset \mathbb{N}$ a new Markov chain defined on the label set $\{l_1, \ldots, l_L\}$, with transitions probabilities $p_{\mathcal{L}}$ defined as above, has its invariant distribution $\pi_{\mathcal{L}}$.*

Proof. We can treat any labelling function as a process in which we merge groups of states having the same label into a new states in the new Markov chain. By induction it is sufficient to show that the Preposition 1 holds for a specific function $\mathcal{L}$ such that it gives every model the unique label except of the two models, which get the same label. So in fact it corresponds to a merger of two states into a new one and leaving the rest the same. By induction we can do every possible merger and thus we consider every possible labelling function. So let us assume that we have a labelling function

$\mathcal{L} : \{M_1, \ldots, M_m, M_{m+1}, M_{m+2}\} \mapsto \{0, \ldots, m\}$ such that

$$\mathcal{L}(M_i) = \begin{cases} i & \text{for } i \leq m \\ 0 & \text{for } i = m+1, m+2 \end{cases} .$$

We have that

$$\pi_{\mathcal{L}}(i) = \begin{cases} \pi(M_i) & \text{for } i = 1, \ldots, m \\ \pi(M_{m+1}) + \pi(M_{m+2}) & \text{for } i = 0 \end{cases} .$$

And the transition probabilities are as follows

$$\begin{aligned}
p_{\mathcal{L}}(i,j) &= p(M_i, M_j), \\
p_{\mathcal{L}}(i,0) &= p(M_i, M_{m+1}) + p(M_i, M_{m+2}), \\
p_{\mathcal{L}}(0,j) &= P(M_{m+1}|0)p(M_{m+1}, M_j) + P(M_{m+2}|0)p(M_{m+2}, M_j), \\
p_{\mathcal{L}}(0,0) &= P(M_{m+1}|0)p(M_{m+1}, M_{m+1}) + P(M_{m+2}|0)p(M_{m+2}, M_{m+1}) \\
&+ P(M_{m+1}|0)p(M_{m+1}, M_{m+2}) + P(M_{m+2}|0)p(M_{m+2}, M_{m+2}),
\end{aligned}$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, m$. For this labelling function we have also that

$$P(M_{m+1}|0) = \frac{\pi(M_{m+1})}{\pi(M_{m+1}) + \pi(M_{m+2})},$$

$$P(M_{m+2}|0) = \frac{\pi(M_{m+2})}{\pi(M_{m+1}) + \pi(M_{m+2})}.$$

Now we have that for $i = 1, \ldots, m$

$$\begin{aligned}
\pi_{\mathcal{L}}(i) &= \pi(M_i) \\
&= \sum_{j=1}^{m+2} \pi(M_j)p(M_j, M_i) \\
&= \sum_{j=1}^{m} \pi(M_j)p(M_j, M_i) \\
&+ \pi(M_{m+1})p(M_{m+1}, M_i) + \pi(M_{m+2})p(M_{m+2}, M_i) \\
&= \sum_{j=1}^{m} \pi_{\mathcal{L}}(j)p_{\mathcal{L}}(j,i) \\
&+ [\pi(M_{m+1}) + \pi(M_{m+2})] \frac{\pi(M_{m+1})}{\pi(M_{m+1}) + \pi(M_{m+2})} p(M_{m+1}, M_i)
\end{aligned}$$

$$
\begin{aligned}
&+\quad [\pi(M_{m+1})+\pi(M_{m+2})]\,\frac{\pi(M_{m+2})}{\pi(M_{m+1})+\pi(M_{m+2})}p(M_{m+2},M_i)\\
&= \sum_{j=1}^{m}\pi_{\mathcal{L}}(j)p_{\mathcal{L}}(j,i)\\
&+\quad \pi_{\mathcal{L}}(0)\left[P(M_{m+1}|0)p(M_{m+1},M_j)+P(M_{m+2}|0)p(M_{m+2},M_j)\right]\\
&= \sum_{j=0}^{m}\pi_{\mathcal{L}}(j)p_{\mathcal{L}}(j,i).
\end{aligned}
$$

And in the same way

$$
\begin{aligned}
\pi_{\mathcal{L}}(0) &= \pi(M_{m+1})+\pi(M_{m+2})\\
&= \sum_{j=1}^{m+2}\pi(M_j)\left[p(M_j,M_{m+1})+p(M_j,M_{m+2})\right]\\
&= \sum_{j=1}^{m}\pi(M_j)\left[p(M_j,M_{m+1})+p(M_j,M_{m+2})\right]\\
&+\quad \pi(M_{m+1})\left[p(M_j,M_{m+1})+p(M_j,M_{m+2})\right]\\
&+\quad \pi(M_{m+2})\left[p(M_j,M_{m+1})+p(M_j,M_{m+2})\right]\\
&= \sum_{j=1}^{m}\pi_{\mathcal{L}}(j)p_{\mathcal{L}}(j,0)\\
&+\quad \pi_{\mathcal{L}}(0)P(M_{m+1}|0)\left[p(M_j,M_{m+1})+p(M_j,M_{m+2})\right]\\
&+\quad \pi_{\mathcal{L}}(0)P(M_{m+2}|0)\left[p(M_j,M_{m+1})+p(M_j,M_{m+2})\right]\\
&= \sum_{j=0}^{m}\pi_{\mathcal{L}}(j)p_{\mathcal{L}}(j,0).
\end{aligned}
$$

So $\pi_{\mathcal{L}}$ is the invariant distribution for the new Markov chain. $\Box$

So it is true that the invariant distribution of the new Markov chain is $\pi_{\mathcal{L}}$. It is also clear that the new Markov chain has also desired property of converging to this invariant distribution due to the convergence of the original Markov chain to π.

After running the MCMC simulation we can apply many labelling functions to the same output. It can turn out that one Markov chain based on the particular function $\mathcal{L}_1$ has converged and the other based on function $\mathcal{L}_2$ not. As an example for labelling function we can take the number of present edges in the model.

$$\mathcal{L}(B) = \sum_{i,j=1}^{n} B_{ij}.$$

This is a very popular way of monitoring convergence in graphical selection problem. It gives some information on the complexity of the model. However, usually the test for assessing convergence was based on a roughly look at the graph, without any mathematical background. Brooks, Giudici and Philippe (2003) also considered the number of edges as a feature that one can monitor during the simulation but they proposed additional statistical Chi-squared test. We will present this test in the subsequent section.

Usually after running a simulation, due to the complexity of the model, we do not look at the whole structure, but we ask about a particular feature of the model. It can be the presence of a particular edge from node X_i to X_j. The result is the empirical probability of this feature given by the formula

$$\frac{1}{T} \sum_{t=1}^{T} B_{ij}^{t}.$$

Where B^t is the binary $n \times n$ matrix for the t'th model in the simulation. But we can achieve the same result by taking the labelling function $\mathcal{L}$ such that

$$\mathcal{L}(B) = B_{ij}.$$

So we merge the states of the original Markov chain to construct a new two-state Markov chain. It can be the case that, however the original Markov chain has not converged, the new Markov chain is close to the stationarity. Using the Chi-squared test we could conclude that several independent runs of the new Markov chain give approximately the same posterior distribution and thus we cannot reject the hypothesis that the new Markov chain has converged. In this case we can have confidence that we really found the posterior probability of the presence of this particular edge.

The next example for the labelling function can be derived from genetics. Some problems in genetics concerns finding dependencies between a big number of genes that we investigate. But it can be the case that specialists have some knowledge of the genes i.e. they can expect that one or more of these genes are regulators which means that they behave independently but they influence many other genes. So in the network, that we are trying to find, there would be rather only a few or no edges starting in other nodes and ending at this particular node but we would have many edges starting in this particular node and ending somewhere else. So the natural question is which genes are influenced by our regulator, say node X_i. To answer this question we can use the labelling function $\mathcal{L}$ such that

$$\mathcal{L}(B) = \sum_{j=1}^{n} B_{ij} 2^{j-1}.$$

This function gives the same label to all the models that have the same structure on the edges which come out from the node X_i. Additionally every possible configuration of this structure has its unique label.

The similar method can be applied to find by which nodes a particular node X_i is influenced. In other words, we are only interested in edges that end up in this node. So the labelling function will be like this

$$\mathcal{L}(B) = \sum_{j=1}^{n} B_{ji} 2^{j-1}.$$

Another advantage of the labelling function is that it can be of any form. It is up to us to decide of its look. By using many different functions we can cope with many different problems. Moreover, we do not have to compute new transition probabilities for the new chain because we apply the labelling function after running the original MCMC simulation. The fact that we start with the original Markov chain assures us that we really work on the posterior model probabilities and thus the posterior feature probabilities.

5.2 Thinning parameter

To perform the Chi-squared test we need to satisfy the condition that the observations generated by the MCMC sampler are independent. The labels that we observe are generated by the Markov chain so they are dependent by definition. However, we can solve this problem by using the thinning parameter λ. We do not take adjoining simulation steps but only every λ'th ones. We will briefly explain how this can be done below. For more details see Brooks, Giudici and Philippe (2003).

The Markov chain used for the DBN framework has a finite state space. So when the chain is ergodic then it is automatically uniformly ergodic. For more details about this topic see Jones (2004) and Robert and Casella (2002). The uniform ergodicity means that the distance between t'th step transition density and the limit density of the chain is bounded above by $C\beta^t$ where C is some constant and β is a convergence rate. So if we can find index t, such that $\beta^t \simeq 0.01$ then after t iterations we are 100 times closer to the limit distribution than at the beginning of the simulation. On the other hand we can say that the dependency between two iterations which are t steps apart is 100 times smaller than for the two consecutive steps. So if we are satisfied that an error level is the size of 0.01 and we have some approximation of the convergence rate β then it would seem sensible to take the thinning parameter

$$\lambda = \frac{\log(0.01)}{\log(\beta)}.$$

However, taking the thinning parameter λ bigger reduces the dependency between iterations, we should remember that we cannot take this parameter as large as possible because this has negative consequences on the asymptotic approximation of the Chi-squared test, described below, as well as on the output of the algorithm and thus the empirical probabilities of the considered labels.

The approximation of the convergence rate can be achieved by investigation of the transition probabilities $\pi_{\mathcal{L}}(i,j)$ and a whole transition matrix. We can approximate these probabilities with help of the simulation output in the following way

$$\pi_{\mathcal{L}}(i,j) = \frac{N_{ij}}{\sum\limits_{j} N_{ij}}.$$

Where N_{ij} denotes a number of times that the transition from label i to j was observed. If we have computed the approximate transition matrix then we can find its second largest eigenvalue. The first eigenvalue corresponds to the limit distribution and equals 1. The second one describes the speed with which the other directions in the model space are vanishing. So it seems like a good approximation of the convergence rate β.

5.3 Chi-squared convergence assessment

The Chi-squared test to assess convergence is based on the comparison between J different, independent replications of the Markov chain Monte Carlo. We point out that we do not work on the particular models but, by applying the labelling function $\mathcal{L} : \mathcal{B} \mapsto \{l_1, \ldots, l_L\}$ to the sampled models, we look at their labels. For each replication we count the number of times we have sampled the particular label l

$$N_{j,l} = \sum_{t=1}^{T} \mathbb{I}(\mathcal{L}(B^{j,t}) = l).$$

Where $B^{j,t}$ is the t'th model in the j'th replication. For the convenience we assume that the length of each simulation after applying the thinning parameter λ, which means looking only at every λ'th iteration, is T. We are working on the thinned results from the simulation, so we can treat consecutive iterations as approximately independent. In fact $B^{j,t}$ denotes not t'th but λt'th model in the j'th replication.

The null hypothesis for our test is that all of the replications have reached stationarity. If this is true than we get a natural estimator for the expected counts of the label l as

$$\widehat{N}_l = \frac{\sum_{j=1}^{J} N_{j,l}}{J}.$$

The test is based upon the following statistics

$$\chi^2 = \sum_{i=1}^{L} \sum_{j=1}^{J} \frac{(N_{j,l_i} - \widehat{N}_{l_i})^2}{\widehat{N}_{l_i}}$$

Under the stationarity hypothesis χ^2 is asymptotically distributed as a Chi-squared variate with $J(L-1)$ degrees of freedom.

As described by Agresti (1990) the validity of the asymptotic approximation of the Chi-squared test depends on the expected counts $\widehat{N}_l$. It is advisable to keep these counts above a value of about 5. But another advantage of the idea of the labelling function is that we can just modify the labelling function. We merge few labels which have poor expected counts into common one which satisfy the required condition. The new Markov chain still has desired property of convergence to the proper limit distribution. It reduces the informativeness of the new chain but increases the validity of the Chi-squared test.

In the literature, see Raftery and Lewis (1992), we can find the idea of decomposing a continuous Markov chain into a two state sub-Markov chain. The convergence rate of the new chain is used as an underestimation of the convergence rate of the original chain. However, the idea of the labelling function is somewhat similar, we do not treat the convergence rate of the new Markov chain as before but use it itself. The new Markov chain is interesting in its own right. It describes the problem that we formulated with use of the labelling function. We can apply many labelling functions and each of Markov chain constructed on them has its own convergence rate and its own limit distribution. So we do not look anymore at the original Markov chain but only at the labels defined by the labelling function.

Chapter 6

Conclusions and further work

In the previous chapters we explained the concept of Bayesian Network and its equivalent for the sequenced data, Dynamic Bayesian Network. We showed how the MCMC simulations can be performed in each of these frameworks. One very interesting approach was based on the idea of Markov Blanket and our goal was to apply it for the DBN. Unfortunately, the algorithm which was originally proposed, behaved improper due to the theoretical misapprehensions. Its limit distribution was not a desired posterior model distribution. We made two suggestions on how one could correct this algorithm. However, for the first approach there is still a problem to be solved which is related to the complicated proposal distributions and necessity to compute these distributions quite frequently. The second approach is more clear, easier and looks promising. Unfortunately we could not implement the new algorithm and see how it works due to the time restrictions but this is a natural proposition for the further research.

Another aspect of this research is to give some tools to assess convergence in a graphical model selection problem. The final chapter is devoted to the Chi-squared test. The idea is to apply this test not to particular models, but, by introducing a labelling function, to deal with labels that we put on the models. Similar idea was already presented in the literature, see Brooks, Giudici and Philippe (2003), where authors investigated the number of edges to assess convergence. Similarity to the idea of labelling function follows from the fact that the number of edges is one possible labelling function. However, the labelling function is more general concept. Moreover, we can use many labelling functions at the same time. By using labelling functions we lose some details of the simulated models but for applications, where considered models are big and the complexity of the problem becomes extremely high, this is inevitable. The idea of labelling function is an attempt to find the compromise between complexity of the problem and informativeness of the output of the simulation. The further research in this area can be the validation of the idea of labelling functions in practice on some synthetic data and later on authentic applications.

List of Figures

References

A. Agresti (1990). Categorical Data Analysis. Wiley, New York.

S. El Adlouni, A. C. Favre and B. Bobée (2005). Comparison of methodologies to assess the convergence of Markov chain Monte Carlo methods. Computational Statistics & Data Analysis 50 (2006), pages 2685 - 2701.

S. P. Brooks, P. Giudici and A. Philippe (2003). Nonparametric Convergence Assessment for MCMC Model Selection. Journal of Computational and Graphical Statistics, volume 12, number 1, pages 1 - 22.

D. M. Chickering (2002a). Optimal structure identification with greedy search. The Journal of Machine Learning Research, 3, pages 507 - 554.

D. M. Chickering (2002b). Learning equivalence classes of Bayesian-network structures. The Journal of Machine Learning Research, 2(3), pages 445 - 498.

M. K. Cowles and B. P. Carlin (1996). Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. Journal of the American Statistical Association, volume 91, number 434, pages 883-904.

W. R. Gilks, S. Richardson, D. J. Spiegelhalter (1996). Markov Chain Monte Carlo in practice. Chapman & Hall.

D. Husmeier (2003). Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. Bioinformatics, volume 19, number 17, pages 2271 - 2282.

J. Jakubowski and R. Sztencel (2000). Wstęp do teorii prawdopodobieństwa. Wydawnictwo Script.

G. L. Jones (2004). On the Markov chain central limit theorem. Probability Surveys, volume 1, pages 299 - 320.

A. E. Raftery and S. M. Lewis (1992). How Many Iterations in the Gibbs Sampler? Bayesian Statistics 4, Oxford University Press, pages 763 - 774.

C. Riggelsen (2005). MCMC Learning of Bayesian Network Models by Markov Blanket Decomposition. ECML, pages 329 - 340.

C. P. Robert and G. Casella (2002). Monte Carlo statistical methods. Springer-Verlag, 3rd edition.

T. S. Verma and J. Pearl (1990). Equivalence and synthesis of causal models. In Proceedings of UAI 6, pages 220 - 227.