Published in final edited form as:
Signal Processing. 2006 April ; 86(4): 814–834.

Relationships between probabilistic Boolean networks and dynamic Bayesian networks as models of gene regulatory networks

Harri Lähdesmäki[a,b,*], **Sampsa Hautaniemi**[a], **Ilya Shmulevich**[c], and **Olli Yli-Harja**[a]
a *Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, FIN-33101 Tampere, Finland*

b *Cancer Genomics Laboratory, The University of Texas M. D. Anderson Cancer Center, Houston, TX 77030, USA*

c *Institute for Systems Biology, Seattle, WA 98103, USA*

Abstract

A significant amount of attention has recently been focused on modeling of gene regulatory networks. Two frequently used large-scale modeling frameworks are Bayesian networks (BNs) and Boolean networks, the latter one being a special case of its recent stochastic extension, probabilistic Boolean networks (PBNs). PBN is a promising model class that generalizes the standard rule-based interactions of Boolean networks into the stochastic setting. Dynamic Bayesian networks (DBNs) is a general and versatile model class that is able to represent complex temporal stochastic processes and has also been proposed as a model for gene regulatory systems. In this paper, we concentrate on these two model classes and demonstrate that PBNs and a certain subclass of DBNs can represent the same joint probability distribution over their common variables. The major benefit of introducing the relationships between the models is that it opens up the possibility of applying the standard tools of DBNs to PBNs and vice versa. Hence, the standard learning tools of DBNs can be applied in the context of PBNs, and the inference methods give a natural way of handling the missing values in PBNs which are often present in gene expression measurements. Conversely, the tools for controlling the stationary behavior of the networks, tools for projecting networks onto sub-networks, and efficient learning schemes can be used for DBNs. In other words, the introduced relationships between the models extend the collection of analysis tools for both model classes.

Keywords

Gene regulatory networks; Probabilistic Boolean networks; Dynamic Bayesian networks

1. Introduction

During recent years, it has become evident that cellular processes are executed in a highly parallel and integrated fashion and that computational modeling approaches can provide powerful methodologies for gaining deeper insight into the operation of living cells. The modeling problem that has received a considerable amount of attention is the discovery of

*Corresponding author. Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, FIN-33101 Tampere, Finland. Tel.: +358 3 3115 3859; fax: +358 3 3115 3817. *E-mail addresses:* harri.lahdesmaki@tut.fi (H. Lähdesmäki), sampsa.hautaniemi@tut.fi (S. Hautaniemi), is@ieee.org (I. Shmulevich), yliharja@cs.tut.fi (O. Yli-Harja)..

[5]For example, in Fig. 3, the PBN counterparts of $P_a(\hat{X}_j(t))$ are $\{X_i(t-1), X_{j1}(t-1), X_{j2}(t-1), X_{k1}(t-1), X_{k2}(t-1)\}$.

transcriptional level interactions. With the help of recent development in high-throughput genomic technologies, computational methods have enormous potential in the context of model inference from real measurement data and in practical use, such as drug discovery.

A number of different frameworks for gene regulatory network modeling have been proposed, ranging from differential equations too qualitative models (for an overview, see e.g. [1]). There is a clear conceptual difference between differential equation and coarse-scale models. The former can be used for a detailed representation of biochemical reactions, whereas the latter emphasize fundamental, generic principles between interacting components. In this context, models classes that are both discrete-time and discrete-state are called coarse-scale models.

Fine-scale modeling of biological interactions at the molecular level may require some type of differential equations. Although differential equations have successfully been used to simulate small (known) biochemical pathways (see e.g. [2,3]), their use in large-scale (genome-wide) modeling has considerable limitations. First of all, those models are computationally very demanding. Therefore, when modeling regulatory networks with differential equations, the model selection problem is usually ignored and the underlying biological system is assumed to be known. Because the model selection is the most important computational tool for discovering new, unknown regulatory relationships from the measured data, researchers have considered alternative modeling approaches. Also, the available analysis tools for differential equations are much more restricted than the ones for the alternative model classes (see below).

So-called graphical models can overcome the above-mentioned modeling problems, and advanced analysis tools have been developed for them. The use of holistic, coarse-scale models is also supported by the fact that the currently available data is limited both in quality and the number of samples. That is, there is no advantage using models that are much more accurate than the available data. Another constraint to be kept in mind is that the modeling framework should also be selected on the basis of the preferred goals, i.e., to what kinds of questions are we seeking answers. The two most often used large-scale modeling frameworks are Boolean and Bayesian networks (BNs). Since the Boolean network is a special case of another commonly used model class, probabilistic Boolean networks (PBNs), we will consider PBNs instead of Boolean networks.

PBNs is a model class that has been recently introduced in the context of genetic network modeling [4]. PBN is a stochastic extension of the standard Boolean network that incorporates rule-based dependencies between variables but is also stochastic in nature. The PBN model has a strong biological motivation through the standard, often used Boolean network model, originally proposed by Kauffman [5,6]. The theory of PBNs as models of genetic regulatory networks has been developed further in several papers. In particular, there has been interest in the control of stationary behavior of the network by means of gene interventions/perturbations [7], modifications of the network structure [8], and external control [9]. Another recent paper [10] introduces mappings between PBNs, including projections, node adjunctions and resolution reductions, which at the same time preserve consistency with the original probabilistic structure. Further, learning methods for PBNs have been introduced in [11,12]. More efficient learning schemes, in terms of computational complexity, but with cost of decreased accuracy, have been studied in [13]. General learning concepts have also been introduced in [14], although not in the context of PBNs, but a related setting. Kim and co-authors also show that the Markovian gene regulatory network model[1] is biologically plausible [15].

[1]The dynamics of PBNs can be studied using Markov chains.

Dynamic Bayesian networks (DBNs), also called *dynamic probabilistic networks*, are a general model class that is capable of representing complex temporal stochastic processes [16–18]. DBNs are also known to be able to capture several other often used modeling frameworks, such as hidden Markov models (and its variants) and Kalman filter models, as its special cases (see e.g. [18]). DBNs and their non-temporal versions, BNs, have successfully been used in different engineering problems, such as in speech recognition [19] and target tracking and identification [20]. Recently, BNs have also been used in modeling genetic regulation [17, 21–30].

In this study we concentrate on PBNs and DBNs, and introduce certain equivalences between them. The first part of this paper is devoted to showing that PBNs and a certain subclass of DBNs can represent the same joint probability distribution over their common variables. For that purpose, we introduce a way of conceptually expressing a PBN as a DBN and vice versa. We would like to note that because there are many PBNs that can represent the same conditional probabilities, the one-to-one connection between the two models is true only in terms of probabilistic behavior. The main motivation for introducing the relationships between the models is that it opens up the possibility of applying the advanced tools of these network models to both of them. In other words, the introduced relationships between the models extend the collection of analysis tools for both model classes. The most important consequences of the results are briefly summarized below.

From the DBN point of view, the tools for controlling the stationary behavior of PBNs, by means of interventions, structural modifications of the network, and optimal external control, become available for DBNs. To our knowledge, no such methods have been introduced in the context of DBNs thus far. The same applies to efficient learning schemes, as well as mappings between different networks, in particular, projections onto subnetworks, which at the same time preserve consistency with the original probabilistic structure.

From the PBN point of view, one can use the standard learning tools of DBNs. This is particularly useful because the learning of gene regulatory networks has turned out to be a difficult problem and, therefore, efficient and flexible tools, with a possibility to be able to combine different data sources, are needed. Furthermore, both exact and approximate inference tools give a natural way of handling the missing values in PBNs which are often present in gene expression measurements. Further discussion on the impacts of the relationships can be found in Section 6. Note that the presented results are applicable to all similar modeling approaches, not only for gene regulatory network modeling. Also note that although the relationships are presented in the binary setting, they can be generalized to finer models (more discretization levels) as well.

The paper is organized as follows. Section 2 covers the basics of both model classes and develops PBNs to the extent necessary for this paper. Sections 3 and 4 introduce the relationships between the two models while the benefits of the relationships are explained in Section 6. Section 7 is devoted to general discussion and further topics in learning gene regulatory networks.

2. Network models

In the following, we focus on distributions over a set of discrete-valued random variables. To make a distinction between random variables and their particular values we use the following notation. Upper-case letters, such as X, X_1, Y, are used to denote random variables (and corresponding nodes in the graphs). Lower-case letters, such as x, x_1, y, are used to denote the values of the corresponding random variables. Vector-valued quantities are in boldface.

2.1. Probabilistic Boolean networks

For consistency of notation, we will be using the same notation as in [4]. A PBN $G(V, F)$ is defined by a set of binary-valued nodes (genes) $V = \{X_1, \ldots, X_n\}$ and a list of function sets $F = (F_1, \ldots, F_n)$, where each function set F_i consists of $l(i)$ Boolean functions, i.e., $F_i = \{f_1^{(i)}, \ldots, f_{l(i)}^{(i)}\}$. The value of each node X_i is updated by a Boolean function taken from the corresponding set F_i. A realization of the PBN at a given time instant is defined by a vector of Boolean functions. Assuming that there are N possible realizations for the PBN, then there are N vector functions $\mathbf{f}_1, \ldots, \mathbf{f}_N$ where each $\mathbf{f}_j = (f_{j_1}^{(1)}, \ldots, f_{j_n}^{(n)})$, $1 \le j \le N$, $1 \le j_i \le l(i)$, and each $f_{j_i}^{(i)} \in F_i$. Each realization of the PBN maps (updates) the values of the nodes into their new values, i.e., $\mathbf{f}_j : \mathbb{B}^n \to \mathbb{B}^n$, where $\mathbb{B} = \{0, 1\}$. Network realizations $\mathbf{f}_1, \ldots, \mathbf{f}_N$ constitute the possible values of a random variable whose outcome is selected independently for each updating step.

In order to make the discussion more explicit, we use the notion of time with the updating step of the network. That is, $X_i(t)$ $(1 \le i \le n)$ is the discrete-time random variable (stochastic process) that denotes the attribute X_i at time t, and $\mathbf{X}(t) = (X_1(t), \ldots, X_n(t))$ is a vector of all random variables X_i, $1 \le i \le n$, at time t. So, the updating step from time $t - 1$ to t, given the current state of the nodes $\mathbf{x}(t-1)$ and a realization $\mathbf{f}_j$, is expressed as $(x_1(t), \ldots, x_n(t)) = \mathbf{f}_j(x_1(t-1), \ldots, x_n(t-1))$.

Let $F^{(i)}$ and $\mathbf{F} = (F^{(1)}, \ldots, F^{(n)})$ denote random variables taking values in $F_i = \{f_1^{(i)}, \ldots, f_{l(i)}^{(i)}\}$ and $F_1 \times \ldots \times F_n$, respectively. The probability that a certain predictor function $f_j^{(i)}$ is used to update the value of the node X_i is equal to

$$c_j^{(i)} = \Pr\{F^{(i)} = f_j^{(i)}\} = \sum_{\mathbf{f}: F^{(i)} = f_j^{(i)}} \Pr\{\mathbf{F} = \mathbf{f}\}. \quad (1)$$

A PBN is said to be independent if the elements $F^{(1)}, \ldots, F^{(n)}$ of the random variable $\mathbf{F}$ are independent, i.e., $\Pr\{\mathbf{F} = \mathbf{f}\} = \prod_{i=1}^{n} \Pr\{F^{(i)} = f^{(i)}\}$. If the PBN is independent, its predictor functions $f_j^{(i)}$, $1 \le j \le l(i)$, for all the nodes X_i $(1 \le i \le n)$, are associated with an independent selection probability $c_j^{(i)}$. A PBN is said to be dependent if the joint probability distribution of random variables $F^{(1)}, \ldots, F^{(n)}$ cannot be factored as for the independent PBN. Correspondingly, the node X_i in the PBN is said to be independent (resp. dependent) if its predictor function is (resp. is not) independent of the other predictor functions. It is also worth noting that the model $G(V, F)$ is not changing with time, i.e., it defines a homogeneous process. A basic building block of a PBN, which describes the updating mechanisms for a single node X_i, is shown in Fig. 1.

2.2. Time-series in PBNs

In the following, we are interested in the joint probability distribution of the variables expanded over a finite number of updating steps, say T steps, i.e., $\{\mathbf{X}(i) : 0 \le i \le T\}$. Given the initial state $\mathbf{x}(t-1)$, the probability of moving to some state $\mathbf{x}(t)$ after one step of the network is (see e.g. [4])

$$A(\mathbf{x}(t-1), \mathbf{x}(t)) = \sum_{j: \mathbf{f}_j(\mathbf{x}(t-1)) = \mathbf{x}(t)} \Pr\{\mathbf{F} = \mathbf{f}_j\}. \quad (2)$$

Because the network realizations are selected independently for each time instant, the joint probability distribution over all possible time-series of length $T + 1$ can be expressed as

$$\begin{aligned}&\Pr\{\boldsymbol{X}(0) = \boldsymbol{x}(0), \ \ldots, \ \boldsymbol{X}(T) = \boldsymbol{x}(T)\}\\ &= \Pr\{\boldsymbol{X}(0) = \boldsymbol{x}(0)\}\prod_{t=1}^{T} A(\boldsymbol{x}(t-1), \ \boldsymbol{x}(t)),\end{aligned} \tag{3}$$

where $\Pr\{\mathbf{X}(0) = \mathbf{x}(0)\}$ denotes the distribution of the first state. Eq. (3) shows that dynamics of PBNs can also be modeled by Markov chains [4].

Let us concentrate on independent PBNs for now and rewrite the state transition probabilities $A(\mathbf{x}(t-1), \mathbf{x}(t))$. Let $(\mathbf{x}(t))_i$, $1 \le i \le n$, denote the ith element of $\mathbf{x}(t)$, and $A(\mathbf{x}(t-1), (\mathbf{x}(t))_i)$ denote the probability that the ith element of $\mathbf{x}(t)$ will be $(\mathbf{x}(t))_i$ after one step of the network, given that the current state is $\mathbf{x}(t-1)$. (A more detailed computation of probability $A(\mathbf{x}(t-1), (\mathbf{x}(t))_i)$ is shown in Eq. (12), which will be discussed later on.) Since all nodes are assumed to be independent, each node is updated independently, and Eq. (2) can be written as [15]

$$A(\boldsymbol{x}(t-1), \ \boldsymbol{x}(t)) = \prod_{i=1}^{n} A(\boldsymbol{x}(t-1), \ (\boldsymbol{x}(t))_i). \tag{4}$$

So, given Eqs. (3) and (4), the joint probability distribution over all possible time-series of length $T + 1$ can be expressed as

$$\begin{aligned}&\Pr\{\boldsymbol{X}(0) = \boldsymbol{x}(0), \ \ldots, \ \boldsymbol{X}(T) = \boldsymbol{x}(T)\}\\ &= \Pr\{\boldsymbol{X}(0) = \boldsymbol{x}(0)\}\prod_{t=1}^{T}\prod_{i=1}^{n} A(\boldsymbol{x}(t-1), \ (\boldsymbol{x}(t))_i).\end{aligned} \tag{5}$$

Let us now concentrate on a dependent PBN. Let a dependent PBN have I independent nodes, X_i, $i = 1, \ldots, I$, and D sets of dependent but mutually exclusive nodes, $\mathbf{X}_j = (X_{j_1}, \ldots, X_{j_{d(j)}})$, $j = 1, \ldots, D$. That is, $\{X_i\} \cap \mathbf{X}_j = \varnothing$;, for all $i = 1, \ldots, I$, $j = 1, \ldots, D$, and $\mathbf{X}_j \cap \mathbf{X}_k = \varnothing$, for all $1 \le j \ne k \le D$.[2] Further, $\cup_{i=1}^{I}\{X_i\} \cup \cup_{j=1}^{D}\boldsymbol{X}_j = \{X_1, \ \ldots, \ X_n\}$. Without loss of generality, we assume that the genes are sorted and re-labeled such that the first I nodes are the independent ones and the rest are dependent. The probability distribution of the network realization can be written as

$$\begin{aligned}&\Pr\{\boldsymbol{F} = \boldsymbol{f}\}\\ &= \prod_{i=1}^{I}\Pr\{F^{(i)} = f^{(i)}\}\\ &\times \prod_{j=1}^{D}\Pr\{F^{(j_1)} = f^{(j_1)}, \ \ldots, \ F^{(j_{d(j)})} = f^{(j_{d(j)})}\}.\end{aligned} \tag{6}$$

Define $D_j = \{j_1, \ldots, j_{d(j)}\}$, and let $(\mathbf{x}(t))_{D_j}$ denote the elements $j_1, \ldots, j_{d(j)}$ of the vector $\mathbf{x}(t)$. Let $A(\mathbf{x}(t-1), (\mathbf{x}(t))_{D_j})$ denote the probability that the elements $j_1, \ldots, j_{d(j)}$ of $\mathbf{x}(t)$ are $(\mathbf{x}(t))_{D_j}$ after one step of the network, given that the current state is $\mathbf{x}(t-1)$. Without going into details, it is quite straightforward to see that the joint probability of a dependent PBN over all $T+1$ length time-series can be decomposed the same way as in Eq. (5) except that terms $A(\mathbf{x}(t-1), (\mathbf{x}(t))_i)$, which correspond to the independent nodes, must be replaced by terms $A(\mathbf{x}(t-1), (\mathbf{x}(t))_{D_j})$. That is,

[2]Throughout this paper, symbol $\mathbf{X}$ is used for both (random) vector variables and sets of (random) variables so that notations such as $\mathbf{X}_j \cap \mathbf{X}_k$ make sense.

$$\begin{aligned}&\Pr\{\boldsymbol{X}(0) = \boldsymbol{x}(0), \ \ldots, \ \boldsymbol{X}(T) = \boldsymbol{x}(T)\}\\ &= \Pr\left\{\boldsymbol{X}(0) = \boldsymbol{x}(0)\right\} \prod_{t=1}^{T} \left(\prod_{i=1}^{I} A(\boldsymbol{x}(t-1), (\boldsymbol{x}(t))_i) \right.\\ &\left. \times \prod_{j=1}^{D} A(\boldsymbol{x}(t-1), (\boldsymbol{x}(t))_{D_j}) \right).\end{aligned} \tag{7}$$

2.3. Dynamic Bayesian networks

Definitions in this section follow mainly the notation used in [17]. Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ denote the (binary-valued) random variables in the network and $\Pr\{\cdot\}$ denote the joint probability distribution of $\mathbf{X}$.[3] A BN, also called a probabilistic network, for X is a pair $B = (G, \Theta)$ that encodes the joint probability distribution over X. The first component, G, is a directed acyclic graph whose vertices correspond to the variables in X. The network structure, especially the lack of possible arcs in G, encodes a set of conditional independence properties about the variables in $\mathbf{X}$. The second component, Θ, defines a set of local conditional probability distributions (or conditional probability tables (CPT)), induced by the graph structure G, associated with each variable. Let $\mathbf{Pa}(X_i)$ denote the parents of the variable X_i in the graph G and $\mathbf{pa}(X_i)$ denote the value of the corresponding variables. Then, a BN B defines a unique joint probability distribution over $\mathbf{X}$ given by

$$\Pr\{x_1, \ \ldots, \ x_n\} = \prod_{i=1}^{n} \Pr\{x_i \mid \boldsymbol{pa}(X_i)\}. \tag{8}$$

For a detailed introduction to BNs, see, e.g., [31,32].

Temporal extension of BNs, DBNs, extend these concepts to stochastic processes. In this paper, we restrict our attention to first-order Markov processes in $\mathbf{X}$, i.e., to processes whose transition probability obeys $\Pr\{\mathbf{X}(t)|\mathbf{X}(0), \mathbf{X}(1), \ldots, \mathbf{X}(t-1)\} = \Pr\{\mathbf{X}(t)|\mathbf{X}(t-1)\}$. The transition probabilities are also assumed to be independent of t, meaning that the process is homogeneous, as is the case for PBNs.

A DBN that represents the joint probability distribution over all possible time-series of variables in $\mathbf{X}$ consists of two parts: (i) an initial BN $B_0 = (G_0, \Theta_0)$ that defines the joint distribution of the variables in $\mathbf{X}(0)$, and (ii) a transition BN $B_1 = (G_1, \Theta_1)$ that specifies the transition probabilities $\Pr\{\mathbf{X}(t)|\mathbf{X}(t-1)\}$ for all t. So, a DBN is defined by a pair (B_0, B_1). In this paper we restrict the structure of DBNs in two ways. First, the directed acyclic graph G_0 in the initial BN B_0 is assumed to have only within-slice connections, i.e., $\mathbf{Pa}(X_i(0)) \subseteq \mathbf{X}(0)$ for all $1 \leq i \leq n$. We also constrain the variables in time slice $\mathbf{X}(t)$, $t>0$, to have all their parents in slice $t-1$, i.e., $\mathbf{Pa}(X_i(t)) \subseteq \mathbf{X}(t-1)$ for all $1 \leq i \leq n$ and $t>0$. The fact that connections exist only between consecutive slices is related to our first-order Markovian assumption stated earlier. An example of the basic building blocks of DBNs, B_0 and B_1, is shown in Fig. 2.

Using Eq. (8) and the assumptions on the initial and transition BNs discussed above, the joint distribution over a finite set of random variables $\mathbf{X}(0) \cup \mathbf{X}(1) \cup \ldots \cup \mathbf{X}(T)$ can be expressed as [17]

[3] In order to follow the standard notation we write $\mathbf{Pr}\{\mathbf{x}\}$ instead of $\mathbf{Pr}\{\mathbf{X} = \mathbf{x}\}$.

$$\begin{aligned}
&\Pr\{\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(T)\} \\
&= \Pr\{\boldsymbol{x}(0)\} \prod_{t=1}^{T} \Pr\{\boldsymbol{x}(t) \mid \boldsymbol{x}(t-1)\} \\
&= \prod_{i=1}^{n} \Pr\left\{x_i(0) \mid \boldsymbol{pa}(X_i(0))\right\} \\
&\times \prod_{t=1}^{T} \prod_{j=1}^{n} \Pr\left\{x_j(t) \mid \boldsymbol{pa}(X_j(t))\right\}.
\end{aligned} \tag{9}$$

It may also be worth mentioning that we assume fully observable DBNs. That is, when used in a real application, the values of all nodes are observed. Hidden nodes are considered when extensions to PBNs are discussed in Section 5.

3. Relationships between independent PBNs and DBNs

In order to be able to establish the relationship between PBNs and DBNs, we will add an extra feature to the definition of a PBN. We will assume that the initial state **X**(0) of a PBN can have any joint probability distribution. Indeed, that definition was already used in Eqs. (3), (5) and (7). In particular, we assume **X**(0) to have the same distribution as defined by B_0 for the first slice of a DBN. For instance, in the context of genetic regulatory network modeling this initial distribution can be set equal to the stationary distribution of the corresponding Markov chain. Also note that we are not discussing the general class of DBNs in the following but only the subclass of binary-valued DBNs, as discussed in the previous section. We relax the requirement of binary-valued nodes in Sections 4.1 and 4.2.

3.1. An independent PBN as a binary-valued DBN

We first illustrate a way of expressing an independent PBN $G(V,F)$ as a DBN (B_0,B_1). Let an independent PBN $G(V,F)$ be fixed. The nodes in the graphs G_0 and G_1 must clearly correspond to the nodes in the PBN. In order to distinguish between the nodes from different models, nodes in the PBN are denoted by X_i, $1 \le i \le n$, as above, and the corresponding nodes in the DBN are denoted by $\hat{X}_i$ (similarly for the vector-valued random variables **X** and $\hat{\boldsymbol{X}}$). We refer to nodes X_i and **X** as the PBN counterparts of $\hat{X}_i$ and $\hat{\boldsymbol{X}}$, respectively, and the other way around.

Because an initial BN B_0 is capable of representing any joint probability distribution over its nodes [31,32], the distribution of the initial state of the PBN, $\Pr\{\mathbf{X}(0)\}$, can be represented by B_0. We omit the technical details since they are quite straightforward.

From Eqs. (3) and (9) it is easy to see that both PBNs and DBNs obey the first-order Markovian property. Thus, we only need to consider one-step transition probabilities, say between time instants $t-1$ and t, $\Pr\{\mathbf{X}(t)|\mathbf{X}(t-1)\}$, when expressing the PBN as a DBN. Further, from Eqs. (5) and (9) one can see that, for both models, the joint probability can also be decomposed over their nodes in the same way. Thus, when constructing a transition BN B_1, we can further concentrate only on a single node, say X_i, with other nodes being handled similarly. Let $\boldsymbol{X}_j^{(i)}(t-1) \subseteq \boldsymbol{X}(t-1)$ denote the set of essential variables (nodes) used by predictor function $f_j^{(i)}$ for gene X_i at time t. The set

$$\boldsymbol{Pa}(X_i(t)) = \bigcup_{j=1}^{l(i)} \boldsymbol{X}_j^{(i)}(t-1) \tag{10}$$

denotes the set of all variables used to predict the value of the gene X_i. Let us expand the domain of all the predictor functions in F_i by adding fictitious variables so that they are functions of variables in $\mathbf{Pa}(X_i(t))$. Thus, we can define $B_1 = (G_1,\Theta_1)$ as follows. The graph G_1 has directed

edges from $\hat{\mathbf{X}}(t-1)$ to $\hat{\mathbf{X}}(t)$ such that the parents of $\hat{X}_i(t)$ are equal to the DBN counterparts of the nodes shown in Eq. (10). As was already shown in [4], given the distribution $\mathscr{D}_i$ over the predictor variables of node X_i (denoted as $\mathbf{Pa}(X_i) \sim \mathscr{D}_i$), the probability of X_i being one is

$$\Pr\{X_i = 1\} = \sum_{j=1}^{l(i)} c_j^{(i)} \sum_{\mathbf{x}\in\{0,1\}^{|\mathbf{Pa}(X_i)|}} \mathscr{D}_i(\mathbf{x}) f_j^{(i)}, \qquad (11)$$

where the domain of predictor functions $f_j^{(i)}$ is assumed to be expanded and $f_j^{(i)}$ is treated as a real-valued function. We can interpret Eq. (11) as $\Pr\{X_i(t) = 1 | \mathbf{Pa}(X_i(t)) \sim \mathscr{D}_i\}$. Thus, by specifying $\mathscr{D}_i$ to be "deterministic" such that it corresponds to a particular predictor node configuration $\mathbf{Pa}(X_i(t)) = \mathbf{z}$, i.e., $\mathscr{D}_i(\mathbf{x}) = 1$ if $\mathbf{x} = \mathbf{z}$ and $\mathscr{D}_i(\mathbf{x}) = 0$ otherwise, we have that

$$\Pr\left\{X_i(t) = 1 \mid \mathbf{Pa}(X_i(t)) = \mathbf{z}\right\} = \sum_{j=1}^{l(i)} f_j^{(i)}(\mathbf{z}) c_j^{(i)}. \qquad (12)$$

Then, the set of local conditional probability distributions, or CPTs, Θ_1, induced by the graph structure G_1, has exactly the same entries as shown in Eq. (12) for each node. So, all terms in Eq. (5) can be represented by corresponding terms in Eq. (9). Thus, any PBN can be expressed as a binary DBN.

3.2. A binary-valued DBN as an independent PBN

To establish the converse of the above conclusion, let us see how a binary-valued DBN (B_0,B_1) can be represented as an independent PBN $G(V,F)$. Let a DBN (B_0,B_1) be given. The set of nodes V must clearly correspond to the nodes in the graphs G_0 and G_1, and the distribution of the first state $\mathbf{X}(0)$ in the PBN must be the same as for the DBN.

Following the same reasoning as in Section 3.1, we can again conclude that when constructing a PBN, one only needs to consider the predictor functions for a single node X_i between consecutive time instants $t-1$ and t. Let us assume that the local conditional probability distributions in Θ_1 are given in the form of CPTs, and the number of parents of the ith node is denoted as $q = |\mathbf{Pa}(\hat{X}_i(t))|$. Let us enumerate the entries in the CPTs that are assigned to the ith node as triplets $(z_l,\mathbf{y}_l,p_l)$, where $z_l \in \mathbb{R}$, $\mathbf{y}_l \in \mathbb{R}^q$, $p_l = \Pr\left\{\hat{X}_i(t) = z_l \mid \mathbf{Pa}(\hat{X}_i(t)) = \mathbf{y}_l\right\}$, and $1 \le l \le 2^{q+1}$. Let us also suppose that the triplets are enumerated such that the first $r = 2^q$ triplets have $z_l = 1$ and they are sorted in increasing order, i.e., $1 \le k < l \le r \Rightarrow p_k \le p_l$. Let $\mathbf{y}_l = y_{l_1} y_{l_2} \ldots y_{l_q}$ and interpret a sequence of symbols $\mathbf{x}_l = x_{l_1}^{y_{l_1}} x_{l_2}^{y_{l_2}} \cdots x_{l_q}^{y_{l_q}}$ as a conjunction, where $x_{l_i}^{y_{l_i}} = x_{l_i}$ if $y_{l_i} = 1$ and $x_{l_i}^{y_{l_i}} = \bar{x}_{l_i}$ if $y_{l_i} = 0$. The variables in the conjunction correspond to a set of specific variables, $\{X_{l_1},\ldots,X_{l_q}\}$, in the PBN, i.e., they are PBN counterparts of $\mathbf{Pa}(\hat{X}_i(t))$.

Then, for the ith node in the PBN, the set of functions F_i can be generated as follows: $F_i = \left\{f_1^{(i)}, \ldots, f_r^{(i)}, f_{r+1}^{(i)}\right\}$ where

$$f_l^{(i)} = \boldsymbol{x}_l \vee \boldsymbol{x}_{l+1} \vee \ldots \vee \boldsymbol{x}_r, \text{ for } 1 \le l \le r, \tag{13}$$

is a disjunction of conjunctions, $f_{r+1}^{(i)}$ is a zero function, i.e., $f_{r+1}^{(i)}(\boldsymbol{x}) \equiv 0$ for all $\mathbf{x} \in \mathbb{B}^q$, and $r = 2^q$.[4] The corresponding selection probabilities are set to $c_1^{(i)} = p_1$, $c_l^{(i)} = p_l - p_{l-1}$ for $2 \le l \le r$, and $c_{r+1}^{(i)} = 1 - p_r$. If some of the $c_j^{(i)}$s happen to be zero, the corresponding functions $f_j^{(i)}$ can be removed from F_i. By applying Eq. (12), we can verify that the above construction really gives an equivalent PBN to the given DBN. Let us first compute the one step prediction probabilities for the cases where $X_i(t) = 1$ and the parent nodes have value $\mathbf{y}_l$. One gets

$$\begin{aligned} &\Pr\left\{X_i(t) = 1 \mid \boldsymbol{Pa}(X_i(t)) = \boldsymbol{y}_l\right\} \\ &= \sum_{j=1}^{r+1} f_j^{(i)}(\boldsymbol{y}_l) c_j^{(i)} \\ &= \sum_{j=1}^{l} f_j^{(i)}(\boldsymbol{y}_l) c_j^{(i)} \\ &= p_1 + \sum_{j=2}^{l} (p_j - p_{j-1}) = p_l, \end{aligned} \tag{14}$$

where the second equality follows from the fact that only the first l functions $f_1^{(i)}, \ldots, f_l^{(i)}$ contain the conjunction $\mathbf{x}_l$ (see Eq. (13)), i.e., $f_j^{(i)}(\boldsymbol{y}_l) = 1$ only for $1 \le j \le l$. The probability of the corresponding complement event $\Pr\{X_i(t) = 0 | \mathbf{Pa}(X_i(t)) = \mathrm{y}_l\}$ is clearly $1 - p_l$. So, a binary-valued DBN can be represented as an independent PBN. Thus, we can state the following theorem.

Theorem 1—*Independent PBNs G(V,F) and binary-valued DBNs (B_0,B_1) whose initial and transition BNs B_0 and B_1 are assumed to have only within and between consecutive slice connections, respectively, can represent the same joint distribution over their common variables.*

The methods introduced in Sections 3.1 and 3.2 provide the actual conversions between the two modeling frameworks.

It is important to note that the mapping from a binary DBN to an independent PBN is not unique. Instead, there are many PBNs that have the same probabilistic structure. This is best illustrated by a toy example (see also the example below). Assume that node X_1 is regulated by node X_2 and all the values in the CPT of the node X_1 are equal to 0.5. Then the following two function sets have the same conditional probabilities: $F_1 = \left\{f_1^{(1)}, f_2^{(1)}\right\}$, with $c_1^{(1)} = c_2^{(1)} = 0.5$ and where the functions $f_1^{(1)} = 0$ and $f_2^{(1)} = 1$ are constant zero and unity functions, respectively, or $F_{1'} = \left\{f_{1'}^{(1)}, f_{2'}^{(1)}\right\}$, with $c_{1'}^{(1)} = c_{2'}^{(1)} = 0.5$ and where the functions $f_{1'}^{(1)} = x_2$ and $f_{2'}^{(1)} = \boldsymbol{x}_2$ are the identity and the negation functions, respectively. In other words, two fundamentally different function sets can have the same probabilistic structure. In the case of many parent variables, this issue becomes more complicated. In practice, one may want to construct the predictor functions for each node in the PBN such that the predictor

[4]Note that $f_1^{(i)}$ is essentially the unity function. Also, some of the Boolean expressions in Eq. (13) can possibly be expressed in more compact form.

functions have as few variables as possible, or such that the number of predictor functions is minimized.

Let us first consider minimizing the number of variables in the predictor functions. The above construction method produces predictor functions with the maximal number of variables. In some cases, when the CPTs are "separable," one can construct predictor functions having less variables, but at the same time being consistent with the original conditional probabilities. For example, consider a case where node X_1 is regulated by a set of nodes $X_2,\ldots,X_n$. Assume that the first parent node X_2 has a forcing (canalizing) effect on the target node X_1 such that all the values in the CPT for which $X_1 = 1$ and $X_2 = 1$ are equal to 0.9. Assume further (for simplicity) that all other values in the CPT for which $X_1 = 1$ (i.e., $X_1 = 1$ and $X_2 = 0$) are smaller than 0.9, the largest of those being, say, 0.8. Instead of blindly using the aforementioned algorithm for generating the predictor functions for X_1, it can be useful to take the special form of the CPT into account. Following the above algorithm the first functions can be constructed as explained. However, the effect of the forcing variable X_2 can be accounted for using only a single one-variable predictor function $f^{(1)} = x_2$ with $c^{(1)} = 0.9 - 0.8 = 0.1$. Alternatively, if no single-variable predictor functions can be constructed as explained above, then combinations of variables, starting with two variables, three variables, etc., can be considered. For example, if all the values in the CPT for which $X_1 = 1$, $X_2 = 1$ and $X_3 = 1$ are equal, then a single two-variable function can be defined as $f^{(1)} = x_2x_3$.

The above search for predictor variables was explained in terms of the original conditional probabilities (i.e., CPTs). Alternatively, one can try to optimize the obtained predictor functions somehow. That is, each function in F_i should be expressed in some optimal form. The first thing to be considered is the removal of fictitious variables from the functions. That can possibly result in functions which have far fewer input variables. Yet another issue is to further optimize the actual expressions of the Boolean functions using methods such as Quine–McCluskey algorithm (see e.g. [33]), which minimizes the number of terms in the disjunctive normal form. The above discussion, however, does not provide any optimal method for the predictor function construction (apart from the optimal representation of the functions). An optimal method can be described in terms of the number of predictor functions which is described next.

In the worst case, the minimum number of predictor functions is determined by the number of different values in the CPT. The above construction method automatically selects that number of predictors (plus a possible constant zero function). In some special cases, the number of predictor functions can be reduced. Consider again the same triplets as above $(z_l,\mathbf{y}_l,p_l)$ for a single node X_i and again focus on only those triplets for which $z_l = 1$. This leaves us with a set of probability values $p_1,\ldots,p_r$, $r = 2^q$. The general optimality criterion can be stated as follows. Find the smallest set of selection probabilities $\{c_1^{(i)}, \ldots, c_m^{(i)}\}$, where each $c_j^{(i)} \in [0, 1]$ and $\sum_{j=1}^{m} c_j^{(i)} = 1$, such that each p_l, $1 \le l \le r$, can be written as

$$p_l = \sum_{j \in I_l} c_j^{(i)}, \tag{15}$$

where $I_l \subseteq \{1, \ldots, m\}$. Let $J_j = \{l | j \in I_l\} \subseteq \{1, \ldots, 2^q\}$ be the set of indices of those P_l for which $c_j^{(i)}$ is used in the sum in Eq. (15). Then the function set $F_i = \{f_1^{(i)}, \ldots, f_m^{(i)}\}$ is defined as

$$f_j^{(i)} = \bigvee_{l \in J_j} \boldsymbol{x}_l,$$

where $\mathbf{x}_l$ is as above. In the same manner as in Eq. (14), we can verify the correctness of the predictor functions and the corresponding selection probabilities

$$\begin{aligned}&\Pr\left\{X_i(t)=1 \mid \boldsymbol{Pa}(X_i(t))=\boldsymbol{y}_l\right\}\\&=\sum_{j=1}^{m} f_j^{(i)}(\boldsymbol{y}_l)c_j^{(i)}=\sum_{j:\, f_j^{(i)}(\boldsymbol{y}_l)=1} c_j^{(i)}\\&=\sum_{j:\, l\in J_j} c_j^{(i)}=\sum_{j\in I_l} c_j^{(i)}=p_l,\end{aligned} \tag{16}$$

because (the third equality) $f_j^{(i)}(\boldsymbol{y}_l)=1$ only if the function $f_j^{(i)}$ contains the conjunction $\mathbf{x}_l$, i.e.,$l \in J_j$. The second to last equality follows from the fact that $l \in J_j \Leftrightarrow j \in I_l$, and the last equality from Eq. (15). Note that the optimal function set can still be non-unique.

The obtained functions in F_i can be modified further as explained above e.g. by removing the possible fictitious variables and by applying the Quine–McCluskey optimization algorithm. A computationally efficient algorithm for the search of the optimal function sets remains to be developed. Fortunately, the search problem is usually limited in the sense that each gene contains only few parent variables.

Theorem 1 says that the two model classes can represent the same probabilistic behavior. However, there are many statistically equivalent PBNs that correspond to a DBN. This means the PBN formalism is redundant from the probabilistic point of view. On the other hand, PBN formalism is richer from the functional point of view because it can explain the regulatory roles of different gene sets in more detail than the conditional probabilities in DBNs can do.

Example 2—This example illustrates the above conversions between PBNs and DBNs. Let us assume that we have the following independent PBN: $G=(V,F)$, $V=\{X_1,X_2\}$, $F=(F_1,F_2)$, $F_1=\left\{f_1^{(1)},\ f_2^{(1)}\right\}$, $f_1^{(1)}=x_1\vee x_2$, $f_2^{(1)}=x_2$, $c_1^{(1)}=0.2$, $c_2^{(1)}=0.8$, $F_2=\left\{f_1^{(2)},\ f_2^{(2)}\right\}$, $f_1^{(2)}=x_1\wedge x_2$, $f_2^{(2)}=x_1$, $c_1^{(2)}=0.6$, $c_2^{(2)}=0.4$ which we would like express as a DBN. (We omit the definition of the joint probability distribution of the first slice $\mathbf{X}(0)$.) The sets of essential variables used by predictor functions $f_1^{(1)}$ and $f_2^{(1)}$ are $\boldsymbol{X}_1^{(1)}(t-1)=\left\{X_1(t-1),\ X_2(t-1)\right\}$ and $\boldsymbol{X}_2^{(1)}(t-1)=\left\{X_2(t-1)\right\}$, respectively. Then, following Eq. (10), the parents of the node $\hat{X}_1$ are the DBN counterparts of $\boldsymbol{Pa}(X_1(t))=\cup_{j=1}^{2}\boldsymbol{X}_j^{(1)}(t-1)=\left\{X_1(t-1),\ X_2(t-1)\right\}$. In a similar manner, we obtain that $\boldsymbol{Pa}(\hat{X}_2(t))=\left\{\hat{X}_1(t-1),\ \hat{X}_2(t-1)\right\}$. For purposes of illustration, the truth tables of the functions $f_1^{(1)}$ and $f_2^{(1)}$, as well as $f_1^{(2)}$ and $f_2^{(2)}$, are shown in Table 1. Note that the domains of the functions $f_2^{(1)}$ and $f_2^{(2)}$ are expanded.

The local conditional probability distributions in the DBN can be obtained by using Eq. (12). For instance, the probability of $\hat{X}_1(t)$ being one, given that $\boldsymbol{Pa}(\hat{X}_1(t))=00$, is $\sum_{j=1}^{2} f_j^{(1)}(00)c_j^{(1)}=1\times c_1^{(1)}+0\times c_2^{(1)}=c_1^{(1)}=0.2$. The probability of the corresponding complement event is $\Pr\left\{\hat{X}_1(t)=0 \mid \boldsymbol{Pa}(\hat{X}_1(t))=00\right\}=1-0.2=0.8$. Similar calculations apply to other cases. The local conditional probability distributions are tabulated in Table 2 which concludes the conversion of the given PBN into a DBN.

Let us now assume that we are given a DBN as follows. (We again omit the definition of the initial BN B_0.) The transition BN $B_1 = \{G_1, \Theta_1\}$ is: $G_1 = \{V_1, E_1\}$, with $V_1 = \{\hat{X}_1, \hat{X}_2\}$ and $E_1 = \{\{\hat{X}_1(t-1), \hat{X}_1(t)\}, \{\hat{X}_1(t-1), \hat{X}_2(t)\}, \{\hat{X}_2(t-1), \hat{X}_1(t)\}, \{\hat{X}_2(t-1), \hat{X}_2(t)\}\}$. The CPTs for both nodes (Θ_1) are given in Table 2. Let us concentrate on the node $\hat{X}_1$ and enumerate the triplets ${}_1(y_l, \mathbf{z}_l, p_l), 1 \le l \le 8$, in increasing order as introduced above (see Table 3).

The function set F_1 contains functions $f_1^{(1)} = x_1x_2 \vee x_1x_2 \vee x_1x_2 \vee x_1x_2$, $f_2^{(1)} = x_1x_2 \vee x_1x_2 \vee x_1x_2$, $f_3^{(1)} = x_1x_2 \vee x_1x_2$, $f_4^{(1)} = x_1x_2$ and $f_5^{(1)} = 0$, where the last one denotes the zero function. The corresponding selection probabilities are $c_1^{(1)} = p_1 = 0$, $c_2^{(1)} = p_2 - p_1 = 0.2$, $c_3^{(1)} = p_3 - p_2 = 0.8$, $c_4^{(1)} = p_4 - p_3 = 0$ and $c_5^{(1)} = 1 - p_4 = 0$. Because $c_1^{(1)} = c_4^{(1)} = c_5^{(1)} = 0$, the corresponding functions can be removed from F_1. Further, the remaining functions can be manipulated as $f_2^{(1)} = x_1x_2 \vee x_1x_2 \vee x_1x_2 = x_1 \vee x_1x_2 = x_1 \vee x_2$ and $f_3^{(1)} = x_1x_2 \vee x_1x_2 = x_2$, which directly correspond to the predictor functions shown in the beginning of this Example. Note that the selection probabilities are also the same. After similar operations for the second node one gets the following list of functions: $f_2^{(2)} = x_1x_2 \vee x_1x_2 \vee x_1x_2 = x_1 \vee x_1x_2$, $f_4^{(2)} = x_1x_2$ and $f_5^{(2)} = 0$, with the corresponding selection probabilities being $c_2^{(2)} = 0.4$, $c_4^{(2)} = 0.2$ and $c_5^{(2)} = 0.4$. These functions do not directly correspond to the ones shown in the beginning of this Example. However, in light of Eq. (12), they are effectively the same because they define the same one-step element-wise prediction probabilities (see also the last column of Table 2).

Note that the number of predictor functions can be reduced for the second node. Using the optimal procedure explained above it is easy to see that, e.g., the following two functions suffice $F_2 = \{f_1^{(2)}, f_2^{(2)}\}$, with $c_1^{(2)} = 0.4$ and $c_2^{(2)} = 0.2$, where $f_1^{(2)} = x_1 \vee x_2$ and $f_2^{(2)} = x_1x_2$.

4. Relationships between dependent PBNs and DBNs

This section shows the relationships between dependent PBNs and DBNs. The methods of expressing a dependent PBN as a DBN, and vice versa, are conceptually similar to the ones for independent PBNs.

4.1. A dependent PBN as a discrete-valued DBN

Let us concentrate on a dependent PBN and illustrate a way of expressing it as a DBN. Using the same notation as in Section 2.2, let the given dependent PBN have I independent nodes, X_i, $i = 1, \ldots, I$, and D sets of dependent and mutually exclusive nodes, $\mathbf{X}_j = \{X_{j1}, \ldots, X_{jd(j)}\}$, $j = 1, \ldots, D$. Assuming that the genes are sorted and re-labeled as in Section 2.2, the probability distribution of the network realization can be written as in Eq. (6).

From now on, we relax the requirement of pure binary-valued nodes by allowing discrete-valued nodes in DBNs. In the following the discrete-valued nodes are also thought of as binary vector-valued nodes (i.e., binary vector presentation of a discrete variable). In order to define an equivalent DBN to the given PBN, we start by defining its nodes. First, a DBN is assumed to have only $I + D$ nodes. Let the first I nodes be binary-valued nodes as above, $\hat{X}_i$, $1 \le i \le I$, and the remaining D nodes be binary vector-valued, denoted as $\hat{\mathbf{X}}_j = (\hat{X}_{j_1}, \ldots, \hat{X}_{j_{d(j)}})$, $1 \le j \le D$, and $\hat{\mathbf{X}}_j \in \mathbb{B}^{jd(j)}$. The scalar- and vector-valued nodes, $\hat{X}_i$ and $\hat{\mathbf{X}}_j$, correspond to the nodes X_i and $X_{j1}, \ldots, X_{jd(j)}$ in the PBN, respectively (see Fig. 3). As before, nodes X_i and X_{j1},

…, X_{jk} are called the PBN counterparts of $\hat{X}_i$ and $\hat{\mathbf{X}}_j$, respectively. (Similarly, $\hat{X}_i$ and $\hat{\mathbf{X}}_j$ are the DBN counterparts of X_i and $\mathbf{X}_j$.) In vector form, the nodes in a DBN, when incorporating the notion of time, are $\hat{\mathbf{X}}(t) = (\hat{X}_1(t), \ldots, \hat{X}_I(t), \hat{\mathbf{X}}_1(t), \ldots, \hat{\mathbf{X}}_D(t))$.

An initial BN B_0 can be defined as in Section 3.1, such that the nodes $\hat{X}_i$ ($1 \le i \le I$) and the elements of the nodes $\hat{\mathbf{X}}_j$ ($1 \le j \le D$) in the DBN have the same initial joint distribution as the variable $\mathbf{X}(0)$ in the given PBN. Technical details are again omitted.

Let us then construct the transition BN $B_1 = (G_1, \Theta_1)$. Following the same reasoning as in Section 3.1, the dependent PBN defines a first-order Markovian process in $\mathbf{X}(t)$ (see also Eq. (7)). Even though the given PBN is dependent, nodes of a DBN are constructed in such a way that their PBN counterparts are mutually independent (see Eq. (6)). Based on Eq. (7) and the above node construction, we can again concentrate on a single node in the DBN, between consecutive time steps $t-1$ and t, when constructing the transition BN B_1.

The set $\mathbf{Pa}(X_i(t))$ denotes the set of all variables used to predict the value of the gene X_i at time t (see Eq. (10)). Similarly, let $\mathbf{Pa}(\hat{X}_i(t))$ contain the DBN counterparts of the nodes in $\mathbf{Pa}(X_i(t))$ except that variables belonging into sets of dependent nodes $\mathbf{X}_j = (X_{j1}, \ldots, X_{jd(j)})$, $1 \le j \le D$, are replaced by their corresponding DBN analogs $\hat{\mathbf{X}}_j$. For example, let us assume that some of the nodes (one or more) in $\mathbf{Pa}(X_i(t))$ belong to the set of dependent nodes, say to the jth ($1 \le j \le D$) dependent set. Then, those nodes are replaced by $\hat{\mathbf{X}}_j$ in $\mathbf{Pa}(\hat{X}_i(t))$. The above parent sets are illustrated in Fig. 3, where, e.g., $\mathbf{Pa}(X_i(t)) = \{X_i(t-1), X_{j1}(t-1)\}$ but $\mathbf{Pa}(\hat{X}_i(t)) = \{\hat{X}_i(t-1), \hat{\mathbf{X}}_j(t-1)\}$ because X_{j1} is not independent.

Construction of the graph G_1, then, goes as follows. For nodes $\hat{X}_i$, $1 \le i \le I$, the graph has directed edges from time slice $t-1$ to t such that the parents of $\hat{X}_i(t)$ are equal to $\mathbf{Pa}(\hat{X}_i(t))$. For the grouped nodes $\hat{\mathbf{X}}_j$, $1 \le j \le D$, the graph has directed edges such that the parents of $\hat{\mathbf{X}}_j(t)$ are

$$\mathbf{Pa}(\hat{\mathbf{X}}_j(t)) = \bigcup_{i=j_1}^{j_{d(j)}} \mathbf{Pa}(\hat{X}_i(t)),$$

where $\hat{X}_i$, $j_1 \le i \le j_{d(j)}$, is an element of $\hat{\mathbf{X}}_j$. This is illustrated in Fig. 3, where, e.g., $\mathbf{Pa}(\hat{\mathbf{X}}_j(t)) = \{\hat{X}_i(t-1), \hat{\mathbf{X}}_j(t-1), \hat{\mathbf{X}}_k(t-1)\}$. Let us again expand the domain of predictor functions in F_i by adding fictitious variables. However, the new domain of the functions is not assumed to be $\mathbf{Pa}(X_j(t))$ but, instead, it consists of the PBN counterparts of the nodes in $\mathbf{Pa}(\hat{X}_j(t))$. An example is again shown in Fig. 3. For example, even though $\mathbf{Pa}(X_i(t)) = \{X_i(t-1), X_{j1}(t-1)\}$, the PBN counterparts of $\mathbf{Pa}(\hat{X}_i(t))$ are $\{X_i(t-1), X_{j1}(t-1), X_{j2}(t-1)\}$. It is easy to see that the local CPTs for the first I nodes can be formed as shown in Eq. (12). Let us then define D random vectors $\mathbf{F}_j = (F^{(j1)}, \ldots, F^{(jd(j))})$, $1 \le j \le D$, taking values in $F_{j1} \times \ldots \times F_{jd(j)}$ and whose domains are expanded as discussed above. Thus, $\mathbf{F}_j : \mathbb{R}^s \to \mathbb{R}^{jd(j)}$, where s equals the number of nodes in the PBN counterparts of $\mathbf{Pa}(\hat{\mathbf{X}}_j(t))$.[5] Then, the CPT of the grouped node $\hat{\mathbf{X}}_j$, $1 \le j \le D$, can be computed as

$$\Pr\left\{\hat{\mathbf{X}}_j(t) = \mathbf{z} \,\middle|\, \mathbf{Pa}(\hat{\mathbf{X}}_j(t)) = \mathbf{y}\right\} = \sum_{\mathbf{f}_j} I(\mathbf{f}_j(\mathbf{y}) = \mathbf{z}) \Pr\left\{\mathbf{F}_j = \mathbf{f}_j\right\},$$

where $I(\cdot)$ is the indicator function and the sum is expanded over all possible realizations of $\mathbf{F}_j$. The probability distribution of $\mathbf{F}_j$, in turn, can be computed by "integrating out" the other functions (see Eq. (1)).

4.2. A discrete-valued DBN as a dependent PBN

The final step is to show a way of expressing a discrete-valued DBN (B_0, B_1) as a PBN $G(V, F)$. We again start by defining the nodes of a PBN. Discrete-valued nodes in the DBN are considered to have a binary representation. That is, let $\hat{\boldsymbol{X}}_i$ (a node in DBN) have $b(i)$ bits in its binary representation in which case we can also write $\hat{\mathbf{X}}_i = (\hat{X}_{i1},\ldots, \hat{X}_{ib(i)})$, $\hat{X}_{ij} \in \mathbb{B}$, $j = 1, \ldots, b(i)$. For notational simplicity, we will assume in the following that the number of different values for each node is a power of two. In general, that does not need to be the case. For each node $\hat{\boldsymbol{X}}_i$ in the DBN, the corresponding PBN has a set of $b(i)$ mutually dependent nodes. So, in total, the constructed PBN has $\sum_{i=1}^{n} b(i)$ nodes, where n equals the number of nodes in the given DBN.

The initial distribution of the PBN must be able to represent the same distribution as the first state of the given DBN. When the states of the DBN are considered via their binary representation it is clear, by the assumption stated in the beginning of Section 3, that the previous condition holds.

The joint probability distribution over the variables in the given discrete-valued DBN can be decomposed exactly the same way as shown in Eq. (9). Thus, the process is first order Markovian and we only need to consider the one step prediction probabilities, say from time $t-1$ to time t. We cannot, however, define the predictor functions for each node in PBN independently. Fortunately, it suffices to consider the set of $b(i)$ dependent nodes at a time.

Let us then construct the predictor functions for a PBN. Binary nodes in the DBN are not interesting since for those nodes we can use practically the same method as in Section 3.2. The only exception is that some parent nodes (in the DBN) may be non-binary. In that case, all the nodes in the PBN which correspond to a non-binary node in the DBN are used to predict the value of that gene. So, assume that we are considering a node $\hat{\boldsymbol{X}}_i$ which has $b>1$ bits in its binary representation. (The index i will be omitted from b since we concentrate on a single node $\hat{\boldsymbol{X}}_i$ at a time.) Let the corresponding nodes in the PBN be $\{X_{i_1}, \ldots, X_{i_b}\}$. The parents of the node $\hat{\mathbf{X}}_i$, $\boldsymbol{Pa}(\hat{\boldsymbol{X}}_i(t))$, are assumed to have q bits in total. Input variables of the predictor functions for the nodes $\{X_{i_1}, \ldots, X_{i_b}\}$ are required to be the PBN counterparts of $\boldsymbol{Pa}(\hat{\boldsymbol{X}}_i(t))$. Further, all those predictor functions are mutually dependent.

Let the CPTs in Θ_1 for the node $\hat{\mathbf{X}}_i$ be given: $\Pr\{\hat{\boldsymbol{X}}_i(t) = \boldsymbol{z} \mid \boldsymbol{Pa}(\hat{\boldsymbol{X}}_i(t)) = \boldsymbol{y}\}$ for all $\mathbf{z} \in \mathbb{B}^b$, $\mathbf{y} \in \mathbb{B}^q$. Assume that they are organized in 2^q lists, each containing 2^b entries (triplets): $L_l = ((\mathbf{z}_{l,1}, \mathbf{y}_l, p_{l,1}), \ldots, (\mathbf{z}_{l,2^b}, \mathbf{y}_l, p_{l,2^b}))$ where $\mathbf{z}_{l,r} \in \mathbb{B}^b$ $\mathbf{y}_l \in \mathbb{B}^q$ and $p_{l,r} = \Pr\{\hat{\boldsymbol{X}}_i(t) = \boldsymbol{z}_{l,r} \mid \boldsymbol{Pa}(\hat{\boldsymbol{X}}_i(t)) = \boldsymbol{y}_l\}$, $r = 1, \ldots, 2^b$, $l = 1, \ldots, 2^q$. So, each list shows the elements of the CPT for a single parent (input) node configuration $\mathbf{y}_l$. Thus, the probabilities in each list must sum up to unity, i.e., $\sum_{i=1}^{2^b} p_{l,i} = 1$, for all $l = 1, \ldots, 2^q$. For simplicity, let us assume that entries $(\mathbf{z}_{l,r}, \mathbf{y}_l, p_{l,r})$ for which $p_{l,r} = 0$ are removed from all lists. We now show a constructive algorithm that generates vector-valued random predictor functions $\mathbf{F}_i : \mathbb{B}^q \to \mathbb{B}^b$ for a set of dependent nodes in a PBN. Eventually, however, we will end up with a "normal" dependent PBN with standard Boolean functions for each node. In order to do so, we still

introduce an additional set of 2^q indices (pointers) $s_i^{(k)} \in \{1, \ldots, 2^b\}$, $i = 1, \ldots, 2^q$. Index $s_i^{(k)}$ is used to index (point) an element in the ith list, and k is an iteration index. A constructive algorithm, for a single node in the DBN is shown in Fig. 4.

In step 2, Eq. (17), we essentially define a function $\mathbf{f}_k : \mathbb{B}^q \rightarrow \mathbb{B}^b$ because $\mathbf{y}_l$s are different for all lists $l = 1, \ldots, 2^q$. That is, the output value of $\mathbf{f}_k$ is defined for all possible inputs. It is also important to note that in step 3, Eq. (18), we could have used $p_{l,s_l^{(k)}} := p_{l,s_l^{(k)}} - \Pr\{\boldsymbol{F}_i = \boldsymbol{f}_k\}$ as well because if $p_{l,s_l^{(k)}} = \Pr\{\boldsymbol{F}_i = \boldsymbol{f}_k\}$, then $p_{l,s_l^{(k)}} - \Pr\{\boldsymbol{F}_i = \boldsymbol{f}_k\} = \Pr\{\boldsymbol{F}_i = \boldsymbol{f}_k\} - \Pr\{\boldsymbol{F}_i = \boldsymbol{f}_k\} = 0$. During each iteration we "subtract a probability mass" $\Pr\{\mathbf{F}_i = \mathbf{f}_k\}$ from all lists (distributions). So, after each iteration $0 \le \sum_{i=1}^{2^b} p_{1,i} = \cdots = \sum_{i=1}^{2^b} p_{2^q,i} \le 1$ holds. This ensures that the condition in step 4 is meaningful and becomes valid such that all the sums $\sum_{i=1}^{2^b} p_{l,i}$, $1 \le l \le 2^q$, become equal to zero at the same time. Also, during each iteration, probability $p_{l,s_l^{(k)}}$ is set to zero and the index $s_i^{(k)}$ is incremented at least for one index $l = 1, \ldots, 2^q$. So, the criterion in step 4 is reached in at most after $2^q \times 2^b$ iterations.

In order to see that the above procedure really constructs an equivalent PBN for the given DBN, consider one entry in the CPT of the given DBN, say $\Pr\{\hat{\boldsymbol{X}}_i(t) = \boldsymbol{z}_{l,r} \mid \boldsymbol{Pa}(\hat{\boldsymbol{X}}_i(t)) = \boldsymbol{y}_l\}$. The above construction procedure generates a set of functions $\{\mathbf{f}_1, \ldots, \mathbf{f}_j\}$, with $2^b \le j \le 2^q \times 2^b$. It is easy to see that the sum of the selection probabilities of those functions for which $\mathbf{f}(\mathbf{y}_l) = \mathbf{z}_{l,r}$ is the probability $p_{l,r} = \Pr\{\hat{\boldsymbol{X}}_i(t) = \boldsymbol{z}_{l,r} \mid \boldsymbol{Pa}(\hat{\boldsymbol{X}}_i(t)) = \boldsymbol{y}_l\}$. Thus, the same conditional probabilities are preserved. So, by applying the same construction method to all nodes in the DBN, one can see that all terms of Eq. (9) can be represented by the corresponding terms in Eq. (7).

To formalize the constructed PBN in the same manner as introduced in Section 2.1, one needs to define the predictor functions for each node separately. Given the vector-valued predictor functions $\{\mathbf{f}_1, \ldots, \mathbf{f}_j\}$, $2^b \le j \le 2^q \times 2^b$, then, for each node X_{i_k}, $1 \le k \le b$, one needs to "extract" all possible Boolean functions appearing as the kth element in the vector-valued functions. The probabilities of occurrence remain the same as set by the constructive method.

Thus, since any discrete-valued DBN can be represented as a dependent PBN, we can state the following theorem.

Theorem 3—*Dependent PBNs $G(V, F)$ and discrete-valued DBNs (B_0, B_1) whose initial and transition BNs B_0 and B_1 are assumed to have only within and between consecutive slice connections, respectively, can represent the same joint distribution over their corresponding variables.*

The methods introduced in Sections 4.1 and 4.2 provide the actual conversions between the two modeling frameworks.

Example 4—We illustrate the operation of the above constructive procedure with a simple example. We apply it only to a single node, $\hat{\boldsymbol{X}}_i = (\hat{X}_{i_1}, \hat{X}_{i_2})$ whose parents are

$\hat{\boldsymbol{X}}_p = \left\{\hat{X}_{p_1}, \hat{X}_{p_2}\right\}$. Thus, both nodes are quaternary, i.e., $b = q = 2$. Further, let the CPT for that node be given as in Table 4.

So, now each column in the table corresponds to a list L_i, $1 \le i \le 4$. In each list (column), the outputs, z, are in the same order for simplicity even though that does not need to be the case. So, when step 2 is applied for the first time, we define a function $\mathbf{f}_1 : 00 \mapsto 00, 01 \mapsto 00, 10 \mapsto 00, 11 \mapsto 00$ with the selection probability $\Pr\{\mathbf{F}_i = \mathbf{f}_1\} = 0.1$. Note that, for each input (column), we pick the first non-zero element. Selected elements are shown in bold-face in Table 4 and the selection probability is the minimum of the selected elements. After step 3, the lists will be updated as shown in Table 5. During the second iteration one gets another vector-valued function (see Table 5) $\mathbf{f}_2 : 00 \mapsto 01, 01 \mapsto 00, 10 \mapsto 01, 11 \mapsto 00$, and the corresponding selection probability is $\Pr\{\mathbf{F}_i = \mathbf{f}_2\} = 0{:}1$. The lists will again be updated as shown in Table 6. In a similar manner we obtain the third function $\mathbf{f}_3 : 00 \mapsto 01, 01 \mapsto 01, 10 \mapsto 01, 11 \mapsto 01$, with the selection probability $\Pr\{\mathbf{F}_i = \mathbf{f}_1\} = 0{:}2$. The lists are updated as shown in Table 7. The remaining iterations are similar (not shown). Note that the sums of probabilities in each list (column) are the same within each iteration.

The above procedure is not guaranteed to produce optimal sets of vector-valued functions. As in the case of independent networks, one can, e.g., minimize the number of functions. To set the stage, consider again the lists L_1, $1 \le l \le 2^q$ for a single node X_i. The goal is to find the smallest set of vector-valued functions $\mathbf{F}_i = \{\mathbf{f}_1, \ldots, \mathbf{f}_m\}$ together with their selection probabilities $\Pr\{\mathbf{F}_i = \mathbf{f}_j\} \in [0, 1]$ whose sum is equal to one, such that each $p_{l,r}$ can be written as

$$p_{l,r} = \sum_{j \in \boldsymbol{I}_{l,r}} \Pr\left\{\boldsymbol{F}_i = \boldsymbol{f}_j\right\},$$

where $\boldsymbol{I}_{l,r} \subseteq \{1, \ldots, m\}$, with the constraint that $j \in \boldsymbol{I}_{l,r} \Leftrightarrow j \notin \boldsymbol{I}_{l,s}$, $s \neq r$ ($j = 1, \ldots, m$ and $r = 1, \ldots, 2^b$), for all l. The extra constraint essentially says that each individual function $\mathbf{f}_j$ is deterministic, i.e, each input $\mathbf{y}_l$ is mapped to a single output $\mathbf{z}_{l,r}$. Let $\boldsymbol{J}_j = \{(l, r) | j \in \boldsymbol{I}_{l,r}\}$, then the actual vector functions are defined as

$$\boldsymbol{f}_j : \boldsymbol{y}_l \mapsto \boldsymbol{z}_{l,r} \; \forall (l, r) \in \mathcal{J}_j,$$

and their validity can be checked as shown in Eq. (16). Efficient algorithms for the construction of the optimal vector-valued functions remain to developed.

5. Extensions of PBNs

The PBN model was further developed in [7] by introducing node perturbations. In the context of gene regulatory network modeling, the perturbations can capture various random or unknown factors, such as environmental conditions, that can possibly affect the expression value of a gene.

From a mathematical point of view, node perturbation can be defined in a variety of ways, but let us use the definition from [7]. At every time step of the network, we have a so-called *perturbation vector* $\gamma \in \mathbb{B}$. The value of the ith node in the network is flipped if the corresponding element of γ is one. We will be assuming the perturbation vector to be independent and identically distributed (i.i.d.) for simplicity, even though this is not necessary in general. Let the probability of a single node perturbation be $\Pr\{\gamma_i = 1\} = p$ for all i, where γ_i denotes the ith element of γ. Let the random perturbation process also be homogeneous. Then, given the current state $\mathbf{x}(t - 1)$ of the network and a network realization $\mathbf{f}_j$, $1 \le j \le N$, the updating step of the network is

$$\mathbf{x}(t) = \begin{cases} \mathbf{f}_j(\mathbf{x}(t-1)) & \text{with probability } (1-p)^n, \\ \mathbf{x}(t-1) \oplus \gamma(t) & \text{with probability } 1-(1-p)^n, \end{cases} \tag{19}$$

where $\oplus$ denotes addition modulo 2 and $(1-p)^n$ is the probability of no perturbation occurring. That is, if no nodes are perturbed, the standard network transition function is used as explained in Section 2.1, while in the case of random node perturbations, the next state is determined by the previous state and the perturbation vector. An alternative definition would allow the unperturbed nodes to transition according to the (random) network function and only the perturbed nodes to be flipped.

Another extension to PBNs was introduced in [34] by defining an additional binary random variable δ which controls the random network changes. If δ = 1, then the network realization is randomly selected for the next time step as discussed in Section 2. Otherwise (δ = 0) the previously used network function is used. The network change variable is assumed to homogeneous with $\Pr\{\delta = 1\} = q$. With this extension the state of a PBN consists of the actual variables **X** as well as the network function **F**.

Effectively the same model can also be defined in the DBN context. Let us concentrate on independent PBNs for simplicity. The basis of the model is the same as illustrated in Section 3.1. The effects of perturbations and random network changes can be captured by adding additional hidden nodes $\gamma(t)$, $\mathbf{F}(t)$ and $\delta(t)$, see Fig. 5.

Concerning the node perturbations, the hidden variable $\gamma(t)$ is assumed to have n-dimensional independent Bernoulli distribution with common probability p. The perturbation node $\gamma(t)$ is also added into the set of parent nodes of $\hat{X}_i(t)$, i.e., $\gamma(t) \in \mathbf{Pa}(\hat{X}_i(t))$ for all i and $t \geq 0$. The local CPTs of the nodes $\hat{X}_i(t)$ are the same as in Section 3.1, except with the following adaptation: for the parent node configurations where $\gamma(t) = (00 \ldots 0)$ the CPTs remain unchanged and only depend on the value of $\mathbf{Pa}(\hat{X}_i(t))$, and for the remaining parent node configurations (where $\gamma(t) \neq (00 \ldots 0)$, regardless of the value of $\mathbf{Pa}(\hat{X}_i(t))$) the value of the node is $x_i(t) = x_i(t-1) \oplus \gamma_i(t)$ with probability one. Also note that $\hat{X}_i(t-1)$ must belong to $\mathbf{Pa}(\hat{X}_i(t))$ for all $1 \leq i \leq n$.

The random network changes can be accounted for using the additional hidden variables $\mathbf{F}(t)$ and $\delta(t)$. The network change variable $\delta(t)$ is Bernoulli with parameter q. The value of the network function $\mathbf{F}(t)$ depends directly on the value of $\mathbf{F}(t-1)$ and $\delta(t)$, i.e., $\mathbf{Pa}(\mathbf{F}(t)) = \{\mathbf{F}(t-1), \delta(t)\}$. If $\delta(t) = 0$, then $\mathbf{f}(t) = \mathbf{f}(t-1)$. Otherwise a new network function is selected according to its selection probability. In the DBN formalism, the parents of the actual state variable, $\mathbf{Pa}(\hat{X}_i(t))$, include not only its parent variables as defined in Section 3.1 but also the hidden network function variable, i.e., $\mathbf{F}(t-1) \in \mathbf{Pa}(\hat{X}_i(t))$. Given the value of the parent variables, the CPTs of the actual state variables $\hat{X}_i(t)$ are degenerate since each network function operates deterministically. In the case of random network changes, the effects of node perturbations can be handled as explained above.

6. Benefits of relationships between PBNs and DBNs

As already discussed in Section 1, having shown the connection between PBNs and DBNs (Theorems 1 and 3), it immediately opens up the possibility of applying the advanced tools of PBNs to DBNs and vice versa. Let us give a more detailed description of the tools that become available for DBNs and PBNs in Sections 6.1 and 6.2, respectively.

6.1. Benefits for DBNs

Relationships between PBNs and DBNs make the following tools, among others, available in the context of DBNs as well. The tools originally developed for PBNs can be applied in the context DBNs, e.g., by using the detailed conversion of a DBN to a PBN. Alternatively, knowing the detailed mapping of a DBN to a PBN and that the two models can capture the same probabilistic behavior, it is possible to tailor each PBN method to be used directly in the DBN framework. The details of these tailored methods are not provided here.

In the context of genetic regulatory networks, one may want to elicit certain long-run behavior from the network. For example, a certain set of states can be deemed “undesirable” and one may wish for the network to transition into a “desirable” set of states. For example, in terms of cancer therapeutics, one may want to reach the set of states representing apoptosis in order to suppress the growth of cancer cells, which may keep proliferating.

The problem of controlling the stationary behavior of the dynamic network model has recently been addressed in several papers. Shmulevich et al. [7] considered the control by means of gene interventions. As the intervention is typically only transient, it does not affect the steady-state distribution. Thus, one may want to achieve the desired behavior as quickly as possible. For computational reasons, the control should also be achieved by intervening with as few genes as possible. A control method using the first-passage times of Markov chains was used in [7]. A related approach was taken in [8] where the control was formulated in terms of (permanently) modifying the network structure. That is, given the set of undesirable and desirable states and the original stationary distribution of the network, the goal is to find the best alteration of some preselected number of Boolean predictor functions such that a desired effect in the resulting stationary distribution is obtained. Yet another approach to control the stationary behavior, under the assumption of external variables, was introduced in [9]. Datta et al. based their approach on the theory of controlled Markov chains and used dynamic programming to find the optimal sequence of control actions that minimize a performance index over a finite time horizon. To our knowledge, no methods for controlling the stationary behavior of DBNs have been introduced so far. Therefore, the aforementioned methods can be of great value for the applications of DBNs.

Owing to the large number of variables in full genetic regulatory networks, it is often necessary to constrain one’s approach to sufficiently small sub-networks. Therefore, network projections onto sub-networks are of particular interest. Dougherty and Shmulevich considered mappings between PBNs, such as projections, node adjunctions and resolution reductions, while at the same time preserving consistency with the original probabilistic structure [10].

Learning of PBN models has been considered e.g. in [11–13,15]. A promising approach to network learning was taken in [12]. In order to limit the search space, a novel clustering-based approach to finding sets of possible predictor genes for each gene was used. Further, a reversible jump Markov chain Monte Carlo method together with the coefficient of determination were used to compute the model order and parameters. Computationally more efficient learning schemes, although with the cost of decreased accuracy, were studied in [13]. From the DBN point of view, this can be viewed as an approximate learning method.

It is also worth noting that the constructed predictor functions can provide more compact presentation of the CPTs in DBNs. For example, instead of storing the whole CPT consisting of 2^{q+1} real numbers one can represent the conditional probabilities using PBNs. Assuming only m predictor functions are required to represent a CPT, then only $m\,2^q$-length binary vectors (assuming truth table format for the Boolean functions) together with m real number (selection probabilities) are needed.

6.2. Benefits for PBNs

Having shown the relationships between PBNs and DBNs (Theorems 1 and 3), the following tools from the context of DBNs can be used for PBNs as well. As introduced in the benefits for DBNs section, the DBN tools can be applied to PBNs by using the detailed mapping of a PBN to a DBN. Alternatively, the methods originally developed for DBNs can be tailored, with the help of the detailed conversions between the two model classes, to be used directly in the PBN context.

Gene expression measurements are often corrupted by missing data values. If it so happens that the missing components are of particular relevance, one may want to compute the posterior probability of the hidden nodes, given the incomplete measurements. In order to compute that, one can use exact inference methods [35] in the context of BNs, or approximate inference methods [36] if the exact solution is intractable.

Learning of gene regulatory networks from data has turned out to be a difficult problem and, therefore, efficient and flexible tools are needed. From a theoretical point of view, optimal learning of graphical models from data is a difficult problem—the optimal learning of BN structure using a Bayesian metric, BDe score, was shown to be an NP-complete problem [37], and the temporal dimension of DBNs does not make the problem any easier. In principle, learning of graphical models can be divided into two groups, model selection and parameter estimation, the former step generally being considerably harder than the latter. As is apparent based on the relationships between PBNs and DBNs, the same division can also be used when learning PBNs. Now, the model selection step corresponds to the selection of variables for each Boolean predictor function, whereas the parameter estimation step is related to the selection of the optimal Boolean predictor functions along with their selection probabilities.

Learning of DBNs has been a topic of several studies. A good introduction to learning of (non-temporal) BNs can be found in [38,39]. Learning of DBNs has been studied, e.g., in [17,40]. More efficient inference methods based on estimating local properties (Markov blanket) of the underlying graph have also been introduced [41,42].

In the fully observable case, learning the network structure should not be overly complicated at least for small indegree, which is the maximum number of parents of each node. However, computational and theoretical problems may arise in the case of incomplete data, which is quite common in the context of gene expression measurements. Also, unknown control/regulatory factors not taken into account in the network model can be considered as hidden variables. The expectation-maximization (EM) algorithm is a standard tool for tackling the problems caused by missing data [17,43]. A more efficient structural EM (SEM) algorithm has been introduced for learning BNs (see, for instance, [17]). However, as the EM is an optimization routine, computational costs increase and there is no guarantee for the global maxima.

Data used in the gene regulatory network learning are usually both expensive and difficult to collect. Therefore, one should carefully design the experiments in order to gain maximal advantage. The use of active learning can remarkably reduce the number of observations required to learn the model by choosing data instances whose expected decrease in the uncertainty in the model learned so far is the greatest. Greedy active learning methods for BNs have recently been introduced in [44–46].

As a summary, a number of advanced methods have been developed for learning DBNs. Since PBNs and DBNs can capture the same probabilistic behavior, a straightforward way of using the DBN learning methods in the context of PBNs is as follows: learn a DBN from a given data set, e.g. by maximizing the posterior probability, and then convert it to a PBN using the

methods introduced above. However, special care must be taken in the conversion of a DBN to a PBN due to the one-to-many nature (non-uniqueness) of this transformation.

7. Discussion

The subclass of DBNs used in this study, even though not necessarily binary-valued, is by far the most commonly used DBN model class in the context of modeling dynamic gene regulatory networks. In particular, often only between-slice connections are used since that fits well in the modeling of cause-effect behavior. However, independent PBNs cannot cope with dependent effects (or nearly instantaneous interactions) between variables. The same applies to the binary-valued DBNs as well. This issue can be handled using dependent PBNs or discrete-valued (i.e., binary vector-valued) DBNs. Indeed, the discrete-valued DBNs can be viewed as binary DBNs where some individual binary variables (within slice) are clustered together, hence forming binary vector-valued variables. Since the binary vector-valued nodes can have any local conditional distributions, the individual elements of a binary vector variable can have any joint effects. The same applies to dependent PBNs due to the correspondence between the models.

In the context of learning genetic regulatory networks, the main focus so far has been on the use of gene expression data only. In the context of PBNs, several papers have been published on learning the network structure e.g. [12–15] most of which use the so-called coefficient of determination (CoD) principle [47]. Learning methods for DBNs, in the context of genetic networks, have been studied in [17,21,22,24,25,27,29]. Note, however, that some of the DBN studies have concentrated on non-temporal BNs. A problem of non-dynamic model inference is that one generally loses the causal direction of regulatory effects. This can be circumvented by using perturbations, such as gene over-expression and knock-outs [28,30].

As the DBN is a versatile model class, different information sources can be used, in a principled way, in the model inference. Recently, the use of so-called location data [48], measuring the protein–DNA interactions, was introduced in the framework of BNs [23], and with more abundant data in [49].[6] The optimal DBN model structures are the ones that maximize the a posteriori probability of the model $\mathcal{M}$ given the data $\mathcal{D}$, $\Pr\{\mathcal{M}|\mathcal{D}\} \propto \Pr\{\mathcal{D}|\mathcal{M}\}\Pr\{\mathcal{M}\}$. In [23], the location data was brought into the model inference via the model prior $\Pr\{\mathcal{M}\}$. Loosely speaking, genome-wide location data can be used to measure the degree to which a transcriptional factor, a product of a gene or several genes, binds to the promoter of a gene. So, that data provides a measure of plausibility of a certain gene being a direct parent of some other (or even the same) gene. In other words, by using location data we may be able to assess the likelihood of $X_j(t-1) \in \mathbf{Pa}(X_i(t))$.

In order to incorporate even more prior knowledge into the network learning process, we may also consider the use of sequence analysis results, possibly combined with clustering results. For instance, genes having highly homologous promoter sequences are likely to be regulated by the same transcriptional factors. An approach combining both expression data and sequence information for finding putative regulatory elements was introduced in [50] (for further reading see e.g. [51] and references therein). A purely combinatorial approach to the same problem, even though it was also validated by expression data, was introduced in [52]. Assume that two genes, say X_i and X_j, are found to have highly homologous promoter sequences and are hypothesized to be regulated by the same transcriptional factor, say a product of gene X_k. Then, one can expect that $X_k(t-1) \in \mathbf{Pa}(X_i(t)) \Leftrightarrow X_k(t-1) \in \mathbf{Pa}(X_j(t))$, where the "if and only if"

[6] Bayesian networks were not used in [49].

must be considered, of course, probabilistically, meaning that the probability of this condition can be expected to be high.

In other words, basically any additional information source can be utilized in the process of learning gene regulatory network structure (for a recent study, see [26]). For instance, recent preliminary results indicate that the genetic regulatory networks exhibit scale-free topology, as do many real-world networks [53,54]. All aforementioned information sources can be included in the model induction process through the model prior. Model inference, which utilizes several different information sources, is a topic for further studies.

Another interesting implication of the proposed relationships between the models can be seen after rewriting $\Pr\{\mathcal{M}|\mathcal{D}\}$ as $\Pr\{\mathcal{M}|\mathcal{D}\} \propto (\int \Pr\{\mathcal{D}|\mathcal{M}, \theta\} \Pr\{\theta|\mathcal{M}\}) \Pr\{\mathcal{M}\}$, where θ denotes the model parameters. Because the model parameters in DBNs correspond to the predictor functions and their selection probabilities in PBNs, one can also use the natural constraints of the predictor function classes in the learning phase in a Bayesian fashion. For instance, Harris et al. [55] recently studied more than 150 known regulatory transcriptional systems with varying number of regulating components and found that these controlling rules are strongly biased toward so-called canalizing functions. These and other natural constraints on the class of rules in genetic regulatory networks [56] can be incorporated in the learning phase.

In summary, we have shown relationships between dynamic Bayesian networks and probabilistic Boolean networks. These relationships between the models extend the collection of advanced analysis tools for both model classes. Further, as the theory of both DBNs and PBNs is under vigorous research, new advances are directly applicable to the gene regulatory networks under both models.

Acknowledgements

This study was supported by Tampere Graduate School in Information Science and Engineering (TISE) (H. L.), Academy of Finland (H. L. and O. Y.-H.) and NIH R01 GM072855-01 (I. S.).

References

1. de Jong H. Modeling and simulation of genetic regulatory systems: a literature review. J Comput Biol 2002;9(1):67–103. [PubMed: 11911796]
2. de Hoon MJL, Imoto S, Kobayashi K, Ogasawara N, Miyano S. Inferring gene regulatory networks from time-ordered gene expression data of Bacillus subtilis using differential equations. Proceedings of the Pacific Symposium on Biocomputing 2003;8:17–28.
3. Sakamoto E, Iba H. Inferring a system of differential equations for a gene regulatory network by using genetic programming. Proceedings of the Congress on Evolutionary Computation ’01 2001:720–726.
4. Shmulevich I, Dougherty ER, Seungchan K, Zhang W. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics 2002;18(2):261–274. [PubMed: 11847074]
5. Kauffman SA. Metabolic stability and epigenesis in randomly constructed genetic nets. J Theoret Biol 1969;22(3):437–467. [PubMed: 5803332]
6. The Origins of Order: Self-organization and Selection in Evolution. Oxford University Press; New York: 1993.
7. Shmulevich I, Dougherty ER, Zhang W. Gene perturbation and intervention in probabilistic Boolean networks. Bioinformatics 2002;18(10):1319–1331. [PubMed: 12376376]
8. Shmulevich I, Dougherty ER, Zhang W. Control of stationary behavior in probabilistic Boolean networks by means of structural intervention. J Biol Systems 2002;10(4):431–445.
9. Datta A, Choudhary A, Bittner ML, Dougherty ER. External control in Markovian genetic regulatory networks. Mach Learning 2003;52(1–2):169–181.

10. Dougherty ER, Shmulevich I. Mappings between probabilistic Boolean networks. Signal Processing 2003;83(4):745–761.
11. Lähdesmäki H, Shmulevich I, Yli-Harja O. On learning gene regulatory networks under the Boolean network model. Mach Learning 2003;52(1–2):147–167.
12. Zhou X, Wang X, Dougherty ER. Construction of genomic networks using mutual-information clustering and reversible-jump Markov-chain-Monte-Carlo predictor design. Signal Processing 2003;83(4):745–761.
13. Hashimoto RF, Dougherty ER, Brun M, Zhou ZZ, Bittner ML, Trent JM. Efficient selection of feature sets possessing high coefficients of determination based on incremental determinations. Signal Processing 2003;83(4):695–712.
14. Kim S, Dougherty ER, Bittner ML, Chen Y, Sivakumar K, Meltzer P, Trent JM. General nonlinear framework for the analysis of gene interaction via multivariate expression arrays. J Biomedical Optics 2000;5(4):411–424.
15. Kim S, Li H, Dougherty ER, Cao N, Chen Y, Bittner M, Suh EB. Can Markov chain models mimic biological regulation? J Biol Systems 2002;10(4):431–445.
16. Dean T, Kanazawa K. A model for reasoning about persistence and causation. Comput Intell 1989;5 (3):142–150.
17. Friedman, N.; Murphy, K.; Russell, S. , . Learning the structure of dynamic probabilistic networks, Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI); 1998. p. 139-147.
18. Dynamic Bayesian Networks: Representation, Inference and Learning. University of California, Berkeley, Computer Science Division; July 2002.
19. Madison, Wisconsin: 1998. Speech recognition with dynamic Bayesian networks, in: Proceedings of the Fifteenth National Conference on Artificial Intelligence AAAI-98; p. 173-180.
20. Chang K, Fung R. Target identification with Bayesian networks in a multiple hypothesis tracking system. Optical Eng 1994;36(3):684–691.
21. Friedman N, Linial M, Nachman I, Pe'er D. Using Bayesian networks to analyze expression data. J Comput Biol 2000;7:601–620. [PubMed: 11108481]
22. Hartemink A, Gifford D, Jaakkola T, Young R. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. Proceedings of the Pacific Symposium on Biocomputing 2001;6:422–433.
23. Hartemink A, Gifford D, Jaakkola T, Young R. Combining location and expression data for principled discovery of genetic regulatory network models. Proceedings of the Pacific Symposium on Biocomputing 2002;7:437–449.
24. Imoto S, Goto T, Miyano S. Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. Proceedings of the Pacific Symposium on Biocomputing 2002;7:175–186.
25. Imoto S, Kim S, Goto T, Aburatani S, Tashiro K, Kuhara S, Miyano S. Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network. J Bioinformatics Comput Biol 2003;1(2):231–252.
26. Imoto S, Higuchi T, Goto T, Tashiro K, Kuhara S, Miyano S. Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks. Proceedings of the Second Computational Systems Bioinformatics (CSB'03) 2003:104–113.
27. K. Murphy, S. Mian, Modelling gene expression data using dynamic Bayesian networks, Technical Report, University of California, Berkeley, 1999.
28. Pe'er D, Regev A, Elidan G, Friedman N. Inferring subnetworks from perturbed expression profiles. Bioinformatics 2001;17(Suppl 1):215S–224S.
29. Perrin B-E, Ralaivola L, Mazurie A, Bottani S, Mallet J, d'Alché-Buc F. Gene networks inference using dynamic Bayesian networks. Bioinformatics 2003;19(Suppl 2):ii138–ii148. [PubMed: 14534183]
30. Yoo C, Thorsson V, Cooper GF. Discovery of causal relationships in a gene-regulation pathway from a mixture of experimental and observational DNA microarray data. Proceedings of the Pacific Symposium on Biocomputing 2002;7:498–509.

31. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Interference. Morgan Kaufmann Publishers Inc.; Los Altos, CA: 1988.
32. Probabilistic Reasoning in Expert Systems. Wiley; New York: 1990.
33. The Complexity of Boolean Functions. Wiley; New York: 1987.
34. Zhou X, Wang X, Pal R, Ivanov I, Bittner ML, Dougherty ER. A Bayesian connectivity-based approach to constructing probabilistic gene regulatory networks. Bioinformatics 2004;20(17):2918–2927. [PubMed: 15145802]
35. Huang C, Darwiche A. Inference in belief networks: a procedural guide. Internat J Approx Reason 1996;15(3):225–263.
36. Jordan MI, Ghahramani Z, Jaakkola T, Saul LK. An introduction to variational methods for graphical models. Mach Learning 1999;37(2):183–233.
37. Learning Bayesian networks is NP-complete. In: Fisher, D.; Lenz, H-J., editors. Learning from Data: Artificial Intelligence and Statistics V. Springer; Berlin: 1996. p. 121-130.
38. D. Heckerman, A tutorial on learning with Bayesian networks, Technical Report MSR-TR-95-06, Microsoft Corporation, Redmond, USA, 1996.
39. Learning Bayesian Networks. Prentice-Hall; Englewood Cliffs, NJ: 2003.
40. Learning dynamic Bayesian networks. In: Giles, CL.; Gori, M., editors. Adaptive Processing of Sequences and Data Structures, Lecture Notes in Artificial Intelligence. Springer; Berlin: 1998. p. 168-197.
41. Lecture Notes in Artificial Intelligence. 2417. 2002. Construction of large-scale Bayesian networks by local to global search; p. 375-384.
42. Margaritis D, Thrun S. Bayesian network induction via local neighborhoods. Proceedings of the Advances in Neural Information Processing Systems 2000;12:505–511.
43. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. J Roy Stat Soc Ser B 1977;39:1–38.
44. K. Murphy, Active learning of causal Bayes net structure, Technical Report, University of California, Berkeley, March 2001.
45. Tong, S.; Koller, D. , . Active learning for parameter estimation in Bayesian networks; Proceedings of the Neural Information Processing Systems; 2000. p. 647-653.
46. Tong, S.; Koller, D. , . Active learning for structure in Bayesian networks; Proceedings of the International Joint Conference on Artificial Intelligence; 2001. p. 863-869.
47. Dougherty ER, Kim S, Chen Y. Coefficient of determination in nonlinear signal processing. Signal Processing 2000;80(10):2219–2235.
48. Ren B, Robert F, Wyrick JJ, Aparicio O, Jennings EG, Simon I, Zeitlinger J, Schreiber J, Hannett N, Kanin E, et al. Genome-wide location and function of DNA binding proteins. Science 2000;290 (5500):2306–2309. [PubMed: 11125145]
49. Lee TI, Rinaldi NJ, Robert F, Odom DT, Bar-Joseph Z, Gerber GK, Hannett NM, Harbison CT, Thompson CM, Simon I, et al. Transcriptional regulatory networks in *Saccharomyces cerevisiae*. Science 2002;298(5594):799–804. [PubMed: 12399584]
50. Vilo, J.; Brazma, A.; Jonassen, I.; Robinson, A.; Ukkonen, E. , . Mining for putative regulatory elements in the yeast genome using gene expression data; Proceedings of the International Conference on Intelligent Systems for Molecular Biology; p. 384-394.
51. DNA Microarrays and Gene Expression: From Experiments to Data Analysis and Modeling. Cambridge University Press; Cambridge, MA: 2002.
52. Pilpel Y, Sudarsanam P, Church GM. Identifying regulatory networks by combinatorial analysis of promoter elements. Nature Genetics 2001;29(2):153–159. [PubMed: 11547334]
53. Fox JJ, Hill CC. From topology to dynamics in biochemical networks. Chaos 2001;11(4):809–815. [PubMed: 12779520]
54. Oosawa C, Savageau MA. Effects of alternative connectivity on behavior of randomly constructed Boolean networks. Physica D 2002;170(2):143–161.
55. Harris SE, Sawhill BK, Wuensche A, Kauffman S. A model of transcriptional regulatory networks based on biases in the observed regulation rules. Complexity 2002;7(4):23–40.

56. Shmulevich I, Lähdesmäki H, Dougherty ER, Astola J, Zhang W. The role of certain post classes in Boolean network models of genetic networks. Proc Natl Acad Sci USA 2003;100(19):10734–10739. [PubMed: 12963822]

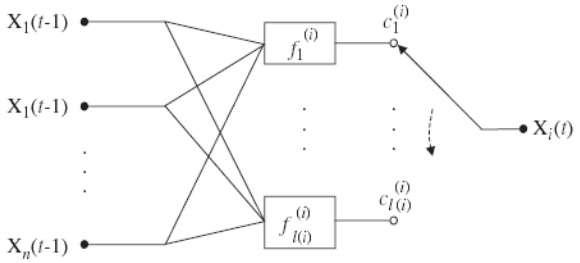

Fig. 1.
A basic building block of probabilistic Boolean networks (PBN). The graph describes the updating mechanisms for a single node X_i in the network.

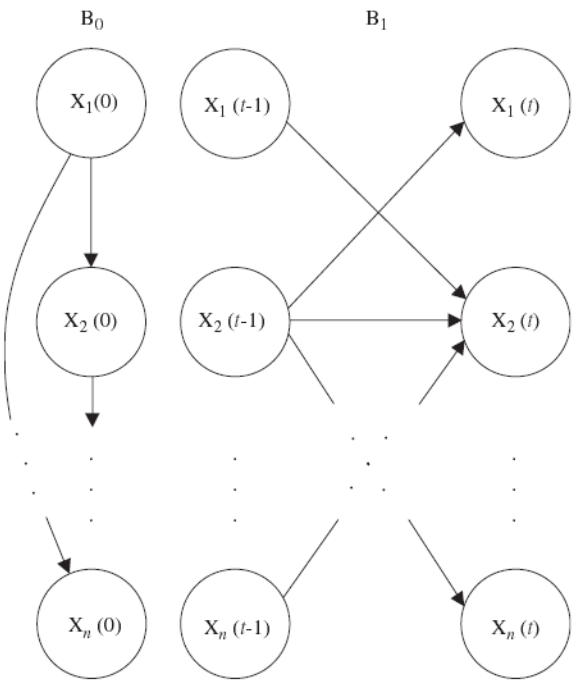

Fig. 2.
An example of the basic building blocks of dynamic Bayesian networks (DBNs) (B_0, B_1). B_0 and B_1 are the initial and transition Bayesian networks (BNs), respectively.

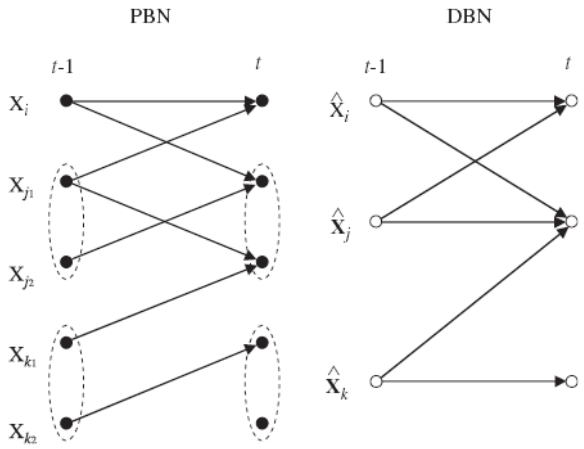

Fig. 3.
Correspondence between the nodes in a dependent PBN and a DBN (see the text for details).

(1) Initialize $k := 1$ and indices $s_i^{(k)} := 1$ for all $i = 1, \ldots, 2^q$.
(2) Define a vector-valued function

$$\mathbf{f}_k : \mathbf{y}_1 \mapsto \mathbf{z}_{1,s_1^{(k)}}, \ldots, \mathbf{y}_{2^q} \mapsto \mathbf{z}_{2^q,s_{2^q}^{(k)}} \tag{17}$$

and set its selection probability to

$$\Pr\{\mathbf{F}_i = \mathbf{f}_k\} = \min\{p_{1,s_1^{(k)}}, \ldots, p_{2^q,s_{2^q}^{(k)}}\}.$$

(3) Update the list entries. For those lists L_l for which $p_{l,s_l^{(k)}} = \Pr\{\mathbf{F}_i = \mathbf{f}_k\}$ set

$$p_{l,s_l^{(k)}} := 0 \text{ and } s_l^{(k+1)} := s_l^{(k)} + 1. \tag{18}$$

For the remaining lists set

$$p_{l,s_l^{(k)}} := p_{l,s_l^{(k)}} - \Pr\{\mathbf{F}_i = \mathbf{f}_k\} \text{ and } s_l^{(k+1)} := s_l^{(k)}$$

Increment the index $k := k + 1$
(4) Repeat steps 2 and 3 until $\sum_{j=1}^{k-1} \Pr\{\mathbf{F}_i = \mathbf{f}_j\} = 1$.

Fig. 4.
A constructive algorithm, for a single node in the DBN, for expressing a given DBN as a PBN.

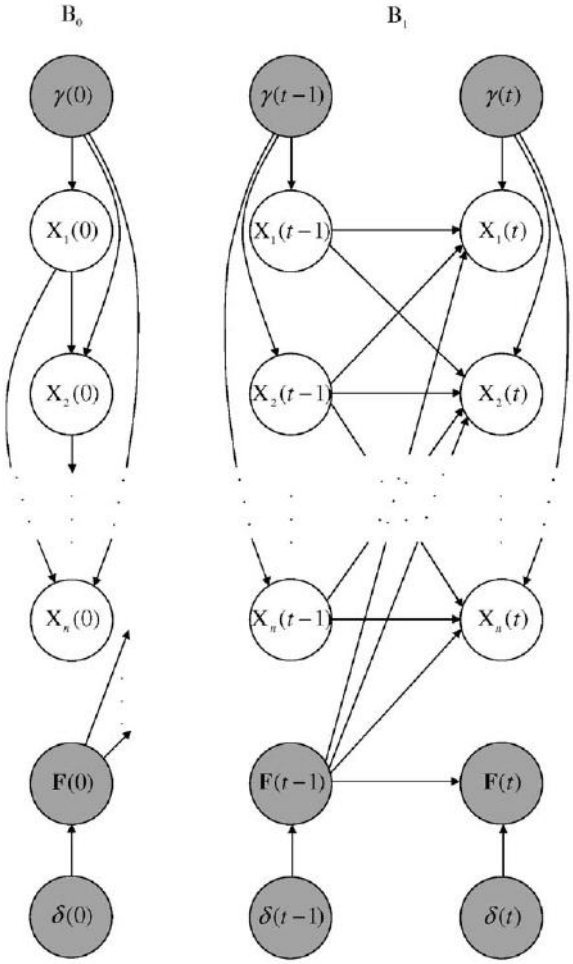

Fig. 5.

Basic building blocks of a dynamic Bayesian network (DBN) allowing node perturbations and random network changes. B_0 and B_1 are the initial and transition Bayesian networks (BNs). The hidden variables $\gamma(t)$, $\mathbf{F}(t)$ and $\delta(t)$ are shown in gray.

Table 1

The truth tables of the functions $f_1^{(1)}$, $f_2^{(1)}$, $f_1^{(2)}$ and $f_2^{(2)}$

(x_1x_2)	$f_1^{(1)}$	$f_2^{(1)}$	$f_1^{(2)}$	$f_2^{(2)}$
00	1	0	0	1
01	1	1	0	1
10	0	0	0	0
11	1	1	1	0

Note that the domains of the functions $f_2^{(1)}$ and $f_2^{(2)}$ are expanded.

Table 2

The local conditional probability distributions for the node $\hat{X}_1$ and $\hat{X}_2$

$\hat{X}_i(t)$	$Pa(\hat{X}_i(t))$	$\Pr\{\hat{X}_1(t) \mid Pa(\hat{X}_1(t))\}$	$\Pr\{\hat{X}_2(t) \mid Pa(\hat{X}_2(t))\}$
1	00	$c_1^{(1)} = 0.2$	$c_2^{(2)} = 0.4$
1	01	$c_1^{(1)} + c_2^{(1)} = 1$	$c_2^{(2)} = 0.4$
1	10	0	0
1	11	$c_1^{(1)} + c_2^{(1)} = 1$	$c_1^{(2)} = 0.6$
0	00	$1 - 0.2 = 0.8$	$1 - 0.4 = 0.6$
0	01	$1 - 1 = 0$	$1 - 0.4 = 0.6$
0	10	$1 - 0 = 1$	$1 - 0 = 1$
0	11	$1 - 1 = 0$	$1 - 0.6 = 0.4$

Table 3

The enumeration of the triplets (y_l, $\mathbf{z}_l$, p_l) for both nodes $\hat{X}_1$ and $\hat{X}_2$

$\hat{X}_1$	$\hat{X}_2$
(1, 10, 0)	(1, 10, 0)
(1, 00, 0.2)	(1, 00, 0.4)
(1, 01, 1)	(1, 01, 0.4)
(1, 11, 1)	(1, 11, 0.6)

The enumeration is shown only for those triples for which $y_l = 1$.

Table 4

The local conditional probability distributions for the node $\hat{X}_i$

z \ y	00	01	10	11
00	**0.1**	**0.2**	**0.1**	**0.2**
01	0.5	0.2	0.3	0.4
10	0.2	0.3	0.5	0.3
11	0.2	0.3	0.1	0.1

Each entry in the table defines the probability that $\Pr\{\hat{X}_I(t) = z \mid Pa(\hat{X}_I(t)) = y\}$, and, therefore, each column corresponds to a list L_i, $1 \le i \le 4$.

Table 5

The lists L_i, $1 \leq i \leq 4$, after the first iteration of the algorithm

z \ y	00	01	10	11
00	0	**0.1**	0	**0.1**
01	**0.5**	0.2	**0.3**	0.4
10	0.2	0.3	0.5	0.3
11	0.2	0.3	0.1	0.1

Table 6

The lists L_i, $1 \leq i \leq 4$, after the second iteration of the algorithm

z \ y	00	01	10	11
00	0	0	0	0
01	**0.4**	**0.2**	**0.2**	**0.4**
10	0.2	0.3	0.5	0.3
11	0.2	0.3	0.1	0.1

Table 7

The lists L_i, $1 \leq i \leq 4$, after the third iteration of the algorithm

z \ y	00	01	10	11
00	0	0	0	0
01	**0.2**	0	0	**0.2**
10	0.2	**0.3**	**0.5**	0.3
11	0.2	0.3	0.1	0.1