Dynamic Bayesian Networks: A State of the Art

V. Mihajlovic, M. Petkovic
Computer Science Department
University of Twente
The Netherlands
{vojkan, milan}@cs.utwente.nl

Abstract

One of the evolving areas that would certainly occupy computer scientists in the next decade is concerned with building software, which will be able to make conclusions based on information gathered from various sources. Interesting path to the solution of this problem would be to simulate reasoning process of humans, based on their ability to sense the environment in multiple ways and to integrate these sensed information in one global picture of the environment.

Human beings as well as the other animals integrate observations received from multiple senses to comprehend the environment and to take proper actions. Probability theory, thus, whit its inherent notions of uncertainty and confidence, had found widespread popularity in the multisensor fusion community. Various researchers had proposed many different probabilistic models for this purpose.

In this report, we will be especially interested in BNs and their extensions that try to unify temporal dimension with uncertainty. We start with the simple concepts, and then introduce static Bayesian networks, as well as basics of dynamic Bayesian networks as powerful tools for representing such uncertain events. Different levels in creating DBNs are presented. Next, we survey various dynamic models, and describe some useful computing techniques for these models, as well as various problems that arise in these computations.

1. Introduction

How confident can be one, simple information, that we receive from our senses? Can we say that leafs are green? Maybe all the leaves we saw were green, but it does not mean that this is hundred percent case? Maybe leafs look to us as they are green due to a viewing angle, or to Sun position. However, we have a lot of "maybes" here. What are we going to do with them? We have to find appropriate tool able to represent various sources of such uncertain information that describe our problem, and to join them into one inference system. This tool should be able to represent degree of uncertainty, i.e. the probability of some objects or events, and influences that exists between these objects, or events.

According to [1,2], we can deal with uncertainty in two ways: extensionally and intensionally. Extensional systems (also called rule-based systems) are generally computationally efficient but their uncertainty measures are semantically weak. On the contrary, intensional systems are generally computationally expensive and semantically strong. In probabilistic reasoning, random variables are used to represent events and/or objects in the world. By assigning various values to these random variables, we can model the current state of the world and weight the states according to the joint probabilities.

Various researchers have proposed many different probabilistic models for this purpose. According to [3], probabilistic models can be classified into four broad categories: Bayesian reasoning, evidence theory, robust statistics, and recursive operators. The question is what category is more appropriate for our domain of interest.

If we consider above mentioned probabilities and uncertainty, this will lead us to the use of Bayesian Networks (BNs), fuzzy logic, Hidden Markov Models (HMMs) or some other techniques. However, BNs bring the most appropriate representation of relative influences among real world facts. This is main reason why BNs are used for solving such problems with uncertainty. BNs has been known in the field of Artificial Intelligence (AI) and exploited in different expert systems to model complex interaction among causes and consequences as it is stated in [4]. However, Bayesian reasoning and inference procedures have only recently gained popularity in fusion of information obtained from different sources.

These problems of uncertainty are as well present in computer reasoning. One of the interpretation tasks that form the base of computer reasoning is recognition of temporal and spatial patterns of data presented in the growing multimedia computer databases. At a first glance this would lead us to the field of pattern recognition. However, pattern recognition techniques can only produce uncertain conclusions about domain. Even though some variability in data patterns exists it can be sufficiently well described in the probabilistic framework. Within the framework of statistical pattern recognition different approaches of spatio-temporal analysis and interpretation have been studied for decades. Hidden Markov Models (HMMs) have been widely used for tasks of describing time varying patterns. They are particularly successful in the area of modelling and recognition of spoken language.

As for a temporal point of view, Bayesian networks (BNs) brought a different approach in attempt to model events that include both, temporal and ambient aspect (time-series modelling). This new tool for time-series modelling is known as a Dynamic Bayesian Network (DBN). It was also shown that the successful HMMs are just a special case of dynamic Bayesian networks. Regarding to this issue, it is natural and tempting to explore and dissect BNs and DBNs for interpretation of computer's sources of information (text, video, and audio data).

This report will be organized as follows. In section 1, we will give basic issues about uncertainty and time, as well as possible ways to incorporate them into one schema. Then, we will introduce Bayesian networks, dynamic Bayesian networks, and some basic techniques that form powerful base for performing various calculations. At the end of this section, we will present different levels of creating dynamic Bayesian networks. In section 2, we will give historical review of incorporation of time into Bayesian networks, and explain more precisely important models. In section 3, we will describe basic problems that have to be solved for efficient application of dynamic Bayesian networks that depict one particular real world problem.

2. Basic Concepts

To explain the role of Bayesian networks and dynamic Bayesian networks in reasoning with uncertainty we will start from the basic issues and elements. Then, we will present more complex structures and techniques involved in these networks. Therefore, in the sequel we will give brief description of terms and concepts used for describing and explaining BNs structure and parameters. Consequently, we will present concepts of BNs and DBNs, and survey different levels of creating DBNs.

2.1. Terminology

First we will give some terms and definitions frequently used in BNs terminology. Definitions are simplified and given in a descriptive and not exact axiomatic form for the purpose of enabling less educated persons to understand the meaning and the power of this kind of network structure.

A graphical model is a tool that is used to visually illustrate and work with conditional independence among variables in a given problem [5]. A graph is composed of a set of nodes (which in graphical models represent variables) and a set of edges. They can be also termed as vertices and arcs, respectively. Each edge interlaces two nodes, and an edge can have an optional direction assigned to it, so it can be oriented or not.

A chain consists of a series of nodes where each successive node in the chain is connected to the previous one by an edge. A path is a chain where each connecting edge in the chain has a directionality going in the same direction as the chain. This property is flavour of digraphs (directed graphs). The cycle is a path that starts and ends at the same node. A simple path is a path where each node is "visited" only once. A simple cycle is a cycle where all nodes are unique, except for the start node and end node. A Directed Acyclic Graph, or DAG, is a directed graph that has no cycles.

The edge in DAG points from one node, called parent node, to the other, called child node. If X_1 is the parent of X_2 and X_2 is the parent of X_3 then X is an ancestor of X_3 and X_3 is a descendant of X. A family is the set of nodes composed of X and the parents of X. The term adjacent or a neighbour is used to describe the relationship between two nodes connected by an undirected edge.

A forest is a DAG where each node has either one parent or none at all. A tree is a forest where only one node (root node) has no parents, and every other node has exactly one parent. A moral graph is made from a DAG, by connecting the parents of each node. It means that we add an undirected edge between each parent node and after doing this we remove the directionality from all of the original edges, resulting in undirected graph.

A chord is an edge that does not appear in the path but which is between two nodes that occur in the graph. The term chordless is used to represent a simple path (or cycle) for which no chord exists. The term triangulated describes an undirected graph where any simple cycle that consists of at least four nodes also has at least one chord.

The term complete describes an undirected graph where every node is connected to all other nodes in the graph. A clique is a subset of nodes, which is complete and cannot be made any larger detaining the state of completeness. A join tree is a DAG that has been moralized and triangulated and after that had its cliques transformed to the nodes of a tree.

At the end we will note some properties that describe BNs as a type of DAG.

Node *X* in an undirected path has converging arrows if it is a child of both the previous and the following nodes in the path. If it is a child of a previous and a parent of the next node then node *X* has serial arrows. If node *X* is parent of both, the previous and the following nodes in an undirected path it has diverging arrows.

Each node in the network is conditionally independent from its non-descendants given its parents. A set of nodes A is conditionally independent of another (disjoint) set B given set C if C d-separates A and B. If *X*, *Y* and *Z* are tree disjoint subset of nodes in DAG, than *Z* is said to d-separate *X* from *Y* if there is no path between a node in *X* and a node in *Y* along which the following two conditions hold: (1) every node with converging arrows is in *Z*, or has a descendant in *Z* and (2) every other node is outside *Z* [6].

Since there exists large amount of abbreviations used in the work, we will list all of them below.

List of abbreviations	
AAP	Acute Abdominal Pain
AI	Artificial Intelligence
BBNs	Bayesian Belief Networks
BIC	Bayesian Information Criterion
BNs	Bayesian Networks
CI	Conditional Independence
CPD	Conditional Probability Distribution
C-Ph	Conditional Phase-type
CPT	Conditional Probability Table
DAG	Directed Acyclic Graph
DBBNs	Dynamic Bayesian Belief Networks
DBNs	Dynamic Bayesian Networks
DDDBN	Duration Density Dynamic Bayesian Networks
DPNs	Dynamic Probabilistic Networks
EFDBN	Error Feedback Dynamic Bayesian Networks
EM	Expectation-Maximization
GEM	Generalized Expectation-Maximization
HMMs	Hidden Markov models
LDS	Linear Dynamic Systems
MAP	Maximum A Posterior
MDL	Minimum Description Length
MI	Mutual Information
ML	Maximum Likelihood
MS-EM	Model-Selection Expectation-Maximization
pdf	Probability distribution function
Ph	Phase-type
PNs	Probabilistic networks
PTNs	Probabilistic Temporal Networks
SEM	Structural Expectation-Maximization
SFSA	Stochastic Finite-State Automata
TAP	Temporal Abduction Problem
TBN	Temporal Bayesian Networks

2.2. Bayesian Networks

Probabilistic networks (PNs), also known as Bayesian networks (BNs) or belief networks or causal networks are already well established as representations of domains involving uncertain relations among a group of random variables. In this work we will use term Bayesian networks for this type of probabilistic schema, because it is used in most of the papers in AI community.

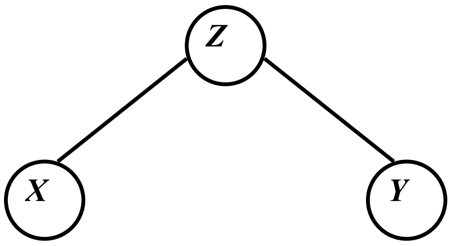


Figure 1 *A graphical model illustrating conditional independence between variables **X** and **Y** given the variable **Z***

A Bayesian Network is a specific type of graphical model that is represented as a Directed Acyclic Graph. Nodes in DAG are graphical representation of objects and events that exists in the real world, and are usually termed variables or states. Causal relations between nodes are represented drawing an arc (edge) between them. For any given edge between the variables (nodes) X and X_2, if there is a causal relationship between the variables, the edge will be directional, leading from the cause variable to the effect variable. Figure 1 illustrates how X and Y are conditionally independent, given variable Z. Here we have to notice that this is not the same as saying that X and Y are totally independent.

For each variable in the DAG there is probability distribution function (pdf), which dimensions and definition depends on the edges leading into the variable. In short, a graphical model consists of variables (nodes) $\boldsymbol{V}$={1,2,…,k} with a set $\boldsymbol{D}$ of dependencies (edges) between the variables and set $\boldsymbol{P}$ of probability distribution functions for each variable.

BNs can be defined as a special case of more general class called graphical models in which nodes represent random variables, and the lack of arc represents conditional independence assumptions between variables. Figure 2 illustrates a simple Bayesian Network.

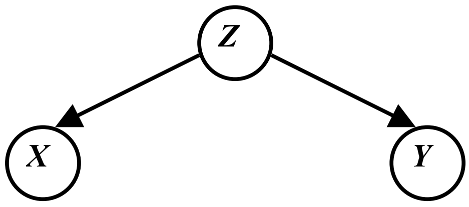


Figure 2 *Simple Bayesian Network*

Probability theory is based on three basic axioms:

1. $0 \leq P(X) \leq 1$
2. $P(X)=1$ if and only if X is certain, and
3. If X and Y are mutually exclusive, then $P(X \vee Y) = P(X) + P(Y)$,

and a fundamental rule of probability calculus:

$$P(X,Y)=P(X|Y)P(Y), \qquad (1)$$

where P(X,Y) is the probability of the joint event $X \wedge Y$.

That brings us to the Bayes' rule for computing posterior probability ($P(X \mid Y)$), given the prior one ($P(X)$), and the likelihood $P(Y \mid X)$ that Y will materialize if X is true:

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)}. \qquad (2\text{ a})$$

Here *X* represent hypothesis, *Y* represents evidence, and $P(Y)$ denotes normalizing factor, which can be determined by

$$P(Y) = P(Y \mid X)P(X) + P(Y \mid \neg X)P(\neg X), \quad (2\text{ b})$$

which can be computed by requiring that $P(X \mid Y)$ and $P(\neg X \mid Y)$ sum to unity.

If X and Y are conditionally independent of each other, like in the Figure 2, we can write the following term:

$$P(X \mid Z, Y) = P(X \mid Z). \qquad (3)$$

Bayesian Networks can be thought of as a knowledge base [3], which explicitly represents our beliefs about the elements in the system and the relationships that exist between these various elements of the system. The purpose of such knowledge base is to infer some belief or to make conclusions about some processes or events in the system.

We could say that BNs operate by propagating beliefs throughout the network, once some evidence about the existence of certain entities can be asserted. We are able to learn probabilities of all parts in the system, given our knowledge of the existence of a few of them and the conditional relationships between them. These conditional probabilities do not have to be known a priory and can be learned using statistical sampling techniques or supervised learning approaches as we would see in section 2.2.2.

In the sequel we will describe some general concepts and techniques used for various calculations that are included in BN computations, and introduce some terms and definitions that frequently pop up in the BN.

Let's resume independence. If there is an absence of a link between two variables, that indicates independence between them given that the values of their parents are known. There is also one rule associated with independence: If a node is observed, then its parents become dependent, since they are rival causes for explaining the child's value. Such rule is known as explaining away [7]. In addition to the network topology, and for purposes of preserving the computational ability, the prior probability of each state of a root node is required.

BNs have one important property that the graph can be considered as representing the joint probability distribution for all the variables. The chain rule is used to express this joint probability as the product of the conditional probabilities that need to be specified for each variable or node. The chain rule is given below:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i)), \qquad (4)$$

where $Pa(X_i)$ is the parent set of a node X_i. Taking the graph as a whole, the conditional probabilities, the structure of the BN, and joint probability distribution, can be used to determine the marginal probability or likelihood of each node holding one of its states. This procedure is called marginalisation.

The power of the belief calculations in BNs comes to light whenever we change one of these marginal probabilities. The effects of the observation are propagated throughout the network and in every propagation step the probabilities of a different neighbouring node are updated. According to [8] in simple networks the marginal probabilities or likelihood of each state can be calculated from the knowledge of the joint distribution, using the product rule, and the Bayes' theorem.

Therefore, for defining the whole structure of a BN we must specify the Conditional Probability Distribution (CPD) or pdfs of each node that has parents, and, as mentioned before, we must specify the prior probability of a root node(s). Since evidence can be assigned to any subset of the nodes (i.e. any subset of the nodes can be observed), if the real world situations do not restrict that, BNs can be used for three kinds of reasoning:

1. From known causes to unknown effects (causal reasoning), and
2. From known effects to unknown causes (diagnostic reasoning), or

3. For any combination of these two,

as it is explained in [7].

2.2.1. Inference

A significant characteristic of Bayesian networks is that we can infer conditional dependencies between variables by visually inspecting the network's graph. Therefore, we can divide set of BN nodes into non-overlapping subsets of conditional independent nodes. This decomposition is important when doing the probability inference. Inference is the task of computing the probability of each state of a node in a Bayesian network when other variables are known. To perform inference we first need to be familiar with the belief propagation. Belief propagation is the action of updating the beliefs in each variable when observations are given to some of the variables. It can be also said, like stated in [9], that inference is the task of efficiently deducing the belief distribution over a particular subset of random variables given that we know the states of some other variables in the network.

In general, variables in BNs can be divided into groups depending on their position in BNs, and taking into account the meaning of real world state that they represents, and including their observability. Consider the partition $Z_L = X_N \cup Y_M$. Let $X_N = \{x_0, x_1, ..., x_{N-1}\}$, $Y_M = \{y_0, y_1, ..., y_{M-1}\}$, and $L=N+M$ denote the two subsets as the sets of hidden and visible variables, respectively. Let U_K be an arbitrary subset of Z_L. The goal of inference is to find the conditional pdf over U given the observed variables Y, which can be written as $\Pr(U_K|Y)$.

If $U_K \subseteq Y$, we can easily find that pdf is trivially equal to $\Pr(U_K | Y) = \prod_{k=1}^{K} \delta(u_k - y_k)$, where $\delta x = 1$ for x=0 and $\delta x = 0$ otherwise. A nontrivial case arises when $U_K \subseteq X$. The desired pdf can now be obtained using the famous Bayes rule:

$$\Pr(U_K | Y) = \frac{\Pr(U_K, Y)}{\Pr(Y)} = \frac{\Pr(U_K, Y)}{\sum_{U_K} \Pr(U_K, Y)}. \quad (5)$$

Clearly, we can conclude that it is sufficient to find the joint pdf $\Pr(U_K|Y)$ and then marginalize over U_K. Moreover, the joint pdf over U_K and Y is obtained by marginalizing $\Pr(Z_L)$ over the set of hidden variables $X \setminus U_K$:

$$\Pr(U_K | Y) = \sum_{x \in X \setminus U_K} \Pr(x, U_K, Y) = \sum_{x \in X \setminus U_K} \Pr(Z_L). \quad (6)$$

Inference in arbitrary Bayesian networks is, in general NP-hard [10]. However, there are special cases of BN topologies that allow for more efficient inference algorithms. Inference can be done by:

1. Exact probability propagation in a singly connected network. To do this, we need to transform the network into singly connected structure.
2. Approximate inference (Monte Carlo inference techniques, inference by ancestral simulation, Gibbs sampling, Helmholtz machine inference, variational inference technique, etc.).

2.2.2. Learning

It is often the case that we do not know all of the conditional probabilities throughout the network. If we want to overcome this problem we must employ some learning techniques that enable us to complete the missing beliefs in the network. According to [9] the role of learning is to adjust the parameters of the Bayesian network so that the pdfs defined by the network sufficiently describes statistical behaviour of the observed data.

Let M be a parametric BN model with parameters θ of the probability distribution defined by the model. Let $\Pr(M)$ and $\Pr(\theta | M)$ be the prior distributions over the set of models and the space of parameters in these models, respectively. Given some observed data assumed to have been generated by the model, as defined by the goal of learning in Bayesian framework we have to estimate the model parameters θ, such that the posterior probability of the model-given data Z_L become maximized:

$$\Pr(M \mid Z_L) = \frac{\Pr(M)}{\Pr(Z_L)} \int_\theta \Pr(Z_L \mid \theta, M) \Pr(\theta \mid M) d\theta \,. \qquad (7)$$

To adjust this task to more appropriate form it is usually assumed that the pdf of the parameters of the model, $\Pr(\theta \mid M)$, is highly peaked around maximum likelihood (ML) estimates of those parameters. Thus we transform equitation (7) into:

$$\Pr(M \mid Z_L) \approx \frac{\Pr(M)}{\Pr(Z_L)} \Pr(Z_L \mid \theta_{ML}, M) \Pr(\theta_{ML} \mid M) \,, \qquad (8)$$

where the maximum likelihood estimate θ_{ML} for a given model *M* is obtained from next term:

$$\theta_{ML} = \arg\max_\theta \log \Pr(Z_L \mid \theta) \,. \qquad (9)$$

We can even consider a case where not all of the variables *Z* in the model of a Bayesian network *M* are observed (represented by *X*). We can describe the goal of learning in such problem as follows:

$$\hat{\theta} = \arg\max_\theta \log \sum_X P(Y, X \mid \theta) \,, \qquad (10)$$

where *P* denotes specific joint pdf defined by the network. We can, alternatively, minimize the cost function defined as

$$J(\theta) = -\log \sum_X P(Y, X \mid \theta) \,. \quad (11)$$

To minimize the cost we can use an optimisation technique that rely on the gradient of the cost function, or we can use an iterative procedure for the optimisation called Expectation-Maximization (EM) algorithm, or variance of that procedure known as Generalized EM (GEM), etc.

2.3. Dynamic Bayesian Networks

Most of the events that we meet in our everyday life are not detected based on a particular point in time, but they can be described through a multiple states of observations that yield a judgement of one complete final event. Statisticians have developed numerous methods for reasoning about temporal relationships among different entities in the world. This field is generally known as time-series analysis. According to [11] time-series is a sample realization of a stochastic process, consisting of a set of observations made sequentially over time.

Time is also an important dimension in the field of AI and reasoning. However, BNs do not provide direct mechanism for representing temporal dependencies. In attempting to add temporal dimension into the BN model various approaches has been suggested. Frequent names used to describe this new dimension in BN models are "temporal" and "dynamic". However, the difference between these models and their denomination cannot uniquely point to one typical model. Sterritt et al. [8] tried to distinguish these categories in the manner that would be described below.

Dynamic Belief Networks (DBN) should be a name of a model that describes a system that is dynamically changing or evolving over time. This model will enable users to monitor and update the system as time proceeds, and even predict further behaviour of the system. In such models word dynamic is connected with a "motive force". Changing the nature of the static BN to model "motive forces" can then be thought of as adapting it to dynamic model. Although every system that change its state involve time, authors differentiate between the two terms dynamic and temporal in that temporal models explicitly model time as continuous permanent category as opposed to other changes in the system such as the change in state of a system. Hence, temporal models would be a sub-class of dynamic. If every time slice of a temporal model corresponds to one particular state of a system, and if the movement between the slices reflects a change in state instead of time, in most cases that model is classified as a dynamic model.

According to the same authors, considering time representation, temporal approaches could be classified into two main categories, namely those models, which represent time as points (instances) or as time intervals. However time intervals can be thought of as a set of consecutive time points. Therefore, time-point representation seems to be more appropriate and more expressive.

We can distinguish two different approaches to time representation of real world states, which is explained in the sequel.

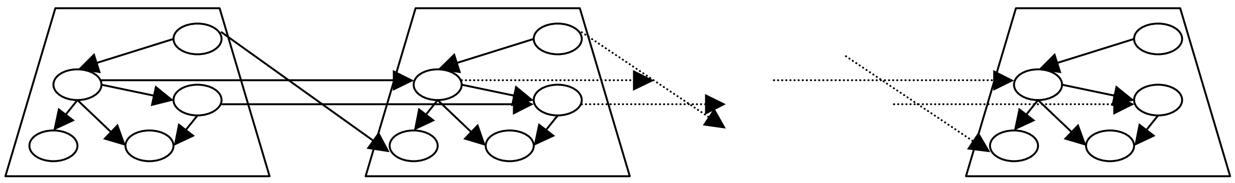


Figure 3 *Diagram showing an approach where time slice is used to represent a snapshot of the evolving temporal process*

Figure 3 represents an approach where time slice is used to represent snapshot of the evolving temporal process. We can say that the belief network consists of a sequence of sub-models each representing the system at a particular point or interval in time (time slice). These time slices are interconnected by temporal relations, which are represented by the arcs joining particular variables from two consecutive time slices.

Figure 4 represents another model where the network is composed of identical sub-models duplicated over each time slice. This means that it has the same temporal structure as previous model. However, links between state variables within a time slice are here disallowed.

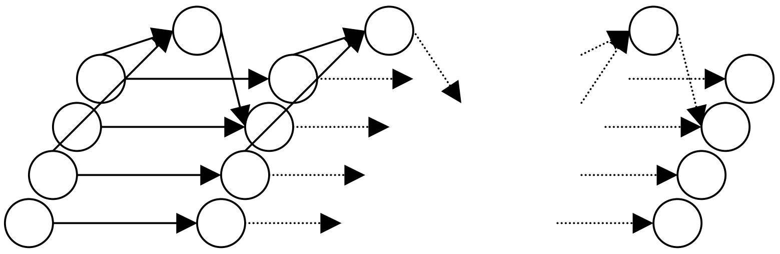


Figure 4 *Temporal model with duplicated time slices over time*

Dynamic Bayesian Networks are usually defined as special case of singly connected Bayesian Networks specifically aimed at time series modelling like stated in [9]. All the nodes, edges and probabilities that form static interpretation of a system is identical to a BN. Variables here can be denoted as the state of a DBN, because they include a temporal dimension. The states of any system described as a DBN satisfy the Markovian condition, that is defined as follows: The state of a system at time *t* depends only on its immediate past, i.e. its state at time *t*-1. Also, this property is frequently considered as a definition of First order Markov property as in [7]: the future is independent of the past given the present.

Now, as we can see, the BN model is expanded. We can allow not only connections within time slices known as intra-slice connections, but also the one between time slices. These temporal connections incorporate condition probabilities between variables from different time slices. The transition matrix that represent these time dependencies is often called a Conditional Probability Table (CPT), since it represents the CPD in tabular form. Intra-slice CPDs can also be represented by CPTs, i.e in tabular form.

The states of a dynamic model do not need to be directly observable. They may influence some other variables that we can directly measure or calculate. Also, the state of some system needs not to be a unique, simple state. It may be regarded as a complex structure of interacting states. Each state in a dynamic model at one time instance may depend on one or more states at the previous time instance or/and on some states in the same time instance. It was shown that complex structures like this could also be represented as DBNs (see [9] for example). So, generally, in DBN states of a system at time *t* may depend on systems states at time *t*-1 and possibly on current states of some other nodes in the fragment of DBN structure that represents variables at time *t*.

We can describe DBN saying that it consists of probability distribution function on the sequence of *T* hidden-state variables $X = \{x_0, ..., x_{T-1}\}$ and the sequence of *T* observable variables $Y = \{y_0, ..., y_{T-1}\}$, where *T* is the time boundary for the given event we are investigating. This can be expressed by the following term:

$$\Pr(X,Y)=\prod_{t=1}^{T-1}\Pr(x_t \mid x_{t-1})\prod_{t=0}^{T-1}\Pr(y_t \mid x_t)\Pr(x_0). \tag{12}$$

In order to completely specify a DBN we need to define three sets of parameters:

- State transition pdfs $\Pr(x_t \mid x_{t-1})$, that specifies time dependencies between the states
- Observation pdfs $\Pr(y_t \mid x_t)$, that specifies dependencies of observation nodes regarding to other nodes at time slice *t*, and
- Initial state distribution $\Pr(x_0)$, that brings initial probability distribution in the beginning of the process.

First two parameters had to be determined for all states in all time slices *t*=1,…,*T*. There is a possibility that conditional pdfs can depend of time instance, that is to be time-varying ($\Pr(x_t \mid x_{t-1})=\Pr(x_t \mid x_{t-1},t)$) or time invariant. Time invariant conditional pdfs can be parametric ($\Pr(x_t \mid x_{t-1})=\Pr(x_t \mid x_{t-1},\theta)$) or nonparametric, when they are described using probability tables (CPTs). Depending on the type of the state space of hidden and observable variables, a DBN can be discrete, continuous, or combination of these two.

Similarly, as we propose in static BNs, in DBNs we may be interested in the following tasks:

1. Inference: estimate the pdf of unknown states given some known observations, and initial probability distribution.
2. Decoding: find the best-fitting probability values for sequence of hidden states that have generated the known sequence of observations.
3. Learning: given a number of sequences of observations, estimate parameters of a DBN such that they best fit to the observed data, and make the best model for the system.
4. Pruning: distinguishing which nodes are semantically important for inference in DBN structure, and which are not, and removing them from the network.

2.3.1. Inference

Since in the DBNs only a subset of states can be observed at each time slice, we have to calculate all the unknown states in the network. This is done by procedure called inference. The problem of inference in DBNs can be represented as the problem of finding $\Pr(X_0^{T-1} \mid Y_0^{T-1})$, where Y_0^{T-1} denotes a finite set of *T* consecutive observations, $Y_0^{T-1}=\{y_0,y_1,\dots,y_{T-1}\}$ and X_0^{T-1} is the set of the corresponding hidden variables $X_0^{T-1}=\{x_1,x_2,\dots,x_{T-1}\}$. Simple graphical representation of this problem is shown in Figure 5. The shaded circle indicates the states that need to be estimated (x_t) based on observations y_t.

Different types of DBNs request different types of estimations and calculations based on their specific structure. If possible, in some circumstances it may be more appropriate (efficient), not to estimate the conditional pdfs $\Pr(X_0^{T-1} \mid Y_0^{T-1})$ for all constellations of X_0^{T-1} but instead to estimate the pdf's sufficient statistics. Therefore, we could choose significant states or a state, and estimate only their values for different time slices. This is the case where the conditional pdf is expressed as Gaussian function. In such network structure it is sufficient to estimate the mean and variance of x_t, denoted as $\left\langle x_t \mid Y_0^{T-1}\right\rangle$ and $\left\langle x_t x_t' \mid Y_0^{T-1}\right\rangle$ for every *t* as well as the covariance $\left\langle x_t x'_{t-1} \mid Y_0^{T-1}\right\rangle$. On the other hand, when x_t is discrete and the conditional pdf is given as a conditional probability table, we need to estimate $\Pr(x_t \mid Y_0^{T-1})$ (which is the same as previously described $\left\langle x_t \mid Y_0^{T-1}\right\rangle$) and $\Pr(x_t x'_{t-1} \mid Y_0^{T-1})$ [9].

If there is strong similarity between DBNs and singly connected BNs, an efficient forward-backward algorithm with some modifications can be employed. It will raise good estimations if we use it for this purpose. Naturally, we need to perform some restrictions on a regular BN algorithm in order to enable such computations.

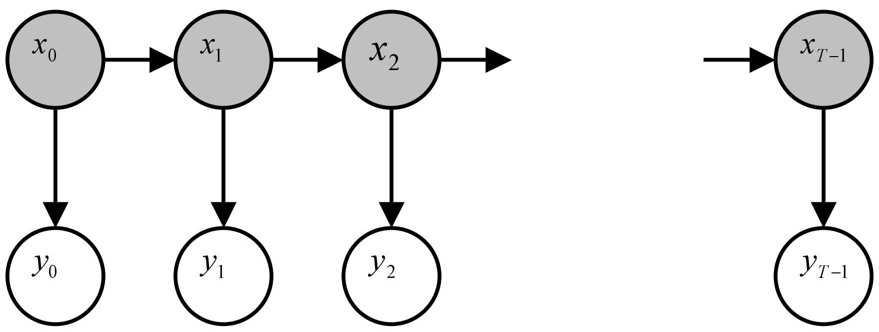


Figure 5 *Inference in DBNs. Given the values of observation nodes in every time slice* y_i *we have to estimate the values of hidden nodes* x_i*, where i receives values from 0 to T-1*

To perform this algorithm we have to execute two-step process in order to do inference task:

1. Propagation of probabilities in the time direction and then
2. The backward propagation, opposite to time direction.

Forward propagation

We will use $\alpha_t(x_t)$ to denote the forward probability distribution that describes the joint probability observations gathered upon time *t* and at time *t*:

$$\alpha_t(x_t) = \Pr(Y_0^t, x_t)\,. \tag{13}$$

If we rely on network structure, shown on Figure 5, it is not complicated to conclude that

$$\alpha_{t+1}(x_{t+1}) = \Pr(y_{t+1} \mid x_{t+1}) \sum_{x_t} \Pr(x_{t+1} \mid x_t) \alpha_t(x_t)\,, \tag{14}$$

with the initial condition $\alpha_0(x_0) = \Pr(x_0)$.

One of the interesting results of forward propagation is the term for likelihood of the observation data sequence Y_0^{T-1}. From the definition of the forward factor $\alpha_t(x_t)$ in equation (13) we can determine that

$$\Pr(Y_0^{T-1}) = \frac{\alpha_{T-1}(x_{T-1})}{\sum_{x_t} \alpha_t(x_t)}\,. \tag{15}$$

As we can see, the probability of the observation sequence is proportional to the forward factor of the last hidden state. The probability from equation (15) can also be used when we want to determine how well different DBN models corresponds to a data sequence in the framework of maximum likelihood estimation.

Backward propagation

Here we start with $\beta_t(x_t)$ as the backward probability distribution, i.e. the conditional probability of observations from time $t+1$ until the last observation at time *T*-1 conditioned on the values of the state at time *t*:

$$\beta_t(x_t) = \Pr(Y_{t+1}^{T-1} \mid x_t)\,. \tag{16}$$

We can infer the next relation from the backward factor definition:

$$\beta_{t-1}(x_{t-1}) = \sum_{x_t} \beta_t(x_t)\Pr(x_t \mid x_{t-1})\Pr(y_t \mid x_t), \tag{17}$$

with $\beta_T(x_{T-1}) = 1$ as the final value.

Smoothing

Given the expressions for forward and backward probability propagation (factors), we are in position to explore few more interesting terms. One of them is called smoothing, and is useful for inference and learning in the DBN. It is form from the equations (15) and (17) we can easily render next equation:

$$\gamma_t(x_t) = \Pr(x_t \mid Y_0^{T-1}) = \frac{\alpha_t(x_t)\beta_t(x_t)}{\sum_{x_t}\alpha_t(x_t)\beta_t(x_t)}, \tag{18}$$

where $\gamma_t(x_t)$ is the smoothing operator. We are also in position to derive higher order smoothing equations. For example a first order smoothing is defined as

$$\xi_{k,k-1}(x_t, x_{t-1}) = \Pr(x_t, x_{t-1} \mid Y_0^{T-1}) = \frac{\alpha_{t-1}(x_{t-1})\Pr(x_t, x_{t-1})\Pr(y_t \mid x_t)\beta_t(x_t)}{\sum_{x_t}\alpha_t(x_t)\beta_t(x_t)}, \tag{19}$$

These terms can be used for easier calculation of probability values of a node's state from neighbouring nodes, and for distributing this evidence to neighbouring nodes.

Prediction

Another interesting inference problem deals with predicting future observations or hidden states based on the past observation data. Namely, a prediction of a hidden states in the next time slice can be described as the following inference calculation task of $\Pr(x_{t+1} \mid Y_0^t)$ or $\Pr(y_{t+1} \mid Y_0^t)$. It is easy to show that

$$\Pr(x_{t+1} \mid Y_0^t) = \frac{\sum_{x_t}\Pr(x_{t+1} \mid x_t)\alpha_t(x_t)}{\sum_{x_t}\alpha_t(x_t)}. \tag{20}$$

In the same way we can write next:

$$\Pr(y_{t+1} \mid Y_0^t) = \frac{\sum_{x_{t+1}}\alpha_{t+1}(x_{t+1})}{\sum_{x_t}\alpha_t(x_t)}. \tag{21}$$

In some situations it may be more appropriate not to work with the pdfs themselves, but to express the prediction problem in terms of the expected or maximum likelihood estimates $\langle x_{t+1,t}\rangle = E[x_{t+1} \mid Y_0^t]$, $\langle y_{t+1,t}\rangle = E[y_{t+1} \mid Y_t]$. This is reflected in the next equations:

$$x_{t+1,t_{ML}} = \arg\max_{\mathrm{x_{t+1}}} \Pr(x_{t+1} \mid Y_0^t), \text{ and} \tag{22}$$

$$y_{t+1,t_{ML}} = \arg\max_{\mathrm{y_{t+1}}} \Pr(y_{t+1} \mid Y_0^t). \tag{23}$$

2.3.2. Decoding

Another problem that arises in DBNs is to find the most likely sequence of hidden variables given the observations. Since DBN nodes can have more than one state we need to determine the sequence of hidden states with highest probabilities. This problem is usually denoted as sequence decoding, and is described like

$$\hat{X}_0^{T-1} = \arg\max_{X_0^{T-1}} \Pr(X_0^{T-1} \mid Y_0^{T-1}) . \tag{24}$$

This task can be achieved using the dynamic programming Viterbi algorithm [12]. We will start with this equation $\delta_{t+1}(x_{t+1}) = \max_{X_0^t} \Pr(X_0^{t+1} \mid Y_0^{t+1})$. Considering previously described network topology, we can derive next term:

$$\delta_{t+1}(x_{t+1}) = \Pr(y_{t+1} \mid x_{t+1}) \max_{x_t} \left[\Pr(x_{t+1} \mid x_t) \max_{X_0^{t-1}} \Pr(X_0^t, Y_0^t) \right] = \Pr(y_{t+1} \mid x_{t+1}) \max_{x_t} \left[\Pr(x_{t+1} \mid x_t) \delta_t(x_t) \right]. \tag{25}$$

It now readily follows that

$$\max_{X_0^{T-1}} \Pr(X_0^{T-1} \mid Y_0^{T-1}) = \max_{x_{T-1}} \delta_{T-1}(x_{T-1}) . \tag{26}$$

To find $\hat{X}_0^{T-1}$ we also need to include the argument x_t that maximizes $\delta_{t+1}(x_{t+1})$, like

$$\psi_{t+1}(x_{t+1}) = \arg\max_{x_t} \left[\Pr(x_{t+1} \mid x_t) + \delta_t(x_t) \right], \tag{27}$$

and then return to the main goal:

$$\hat{x}_t = \psi_{t+1}(\hat{x}_{t+1}) . \tag{28}$$

It is significant to denote that if we intend to decode the sequence of hidden states using the Viterbi algorithm it is necessary to possess a complete set of observations $y_1, \ldots, y_T$. A less optimal solution known as the truncated Viterbi algorithm engages backtracking every time after a fixed number of observations is received ($T_{trunc} < T$).

2.3.3. Learning

A representation of real world problems by a DBN structure often requires introduction of several nodes which conditional probabilities cannot be exactly determined. Even the expert knowledge cannot offer us solution for some conditional relationships in particular domain. In such situations, we need to learn these CPDs. This process is complex and is based on the Expectation Maximisation (EM) or General Expectation Maximisation (GEM) algorithms for DBNs, shortly described in previous section, or some other similar algorithms. However, the term that describes the joint probability distribution now receives a specific form:

$$\log \Pr(X_0^{T-1}, Y_0^{T-1} \mid \theta) = \sum_{t=1}^{T-1} \log \Pr(x_t \mid x_{t-1}) + \sum_{t=0}^{T-1} \log \Pr(y_t \mid x_t) + \log \Pr(x_0) , \tag{29}$$

where θ denotes the model parameter vector. The maximization step now intends to find parameters θ that satisfy next condition:

$$\frac{\partial B(P, \hat{Q})}{\partial \theta} = \sum_{t=1}^{T-1} \left\langle \frac{\partial \log \Pr(x_t, x_{t-1})}{\partial \theta} \right\rangle + \sum_{t=0}^{T-1} \left\langle \frac{\partial \log \Pr(y_t, x_t)}{\partial \theta} \right\rangle + \left\langle \frac{\partial \log \Pr(x_0)}{\partial \theta} \right\rangle = 0 , \tag{30}$$

where the expectation has the same format as it is defined in section 2.2.2.

This learning problem can be expressed through a gradient-based learning procedure. It can be equivalently used and is implemented to perform utilization of next term $\frac{\partial B(P,\hat{Q})}{\partial \theta}$.

2.3.4. Pruning

A very important and also a difficult task in designing a DBN is the problem of pruning. It is based on DBNs possibility to change its structure and connections over time. This process will be explained in short because in most cases this procedure is omitted due to its complex nature.

Pruning the network consists of any of the following actions:

1. Deleting states from a particular node,
2. Removing the connection between two nodes, or
3. Removing a node from the network.

Pruning may be exact (without loss of information) or heuristic, when it represents just one form of approximation.

Within every time slice *t* it is possible to identify a subset of the nodes that describes current state of the world (environment), denoted as $W_1(\mathrm{t}),...,W_q(\mathrm{t})$, which represents either the entire world state, or the part we want to inspect. We call these the <u>designated world nodes</u>. These nodes are chosen in the way that if their states are known, then nodes $V(\mathrm{t_k})$, where $\mathrm{t_k} < \mathrm{t}$, are no longer relevant to the overall goal of the inference. If for every *i* in $[1,...,q]$, there is *s* such that $\Pr(W_i(\mathrm{t}) = s_i) = 1$, then the general pruning action is as follows: (1) delete all nodes $V(\mathrm{t_k})$, where $\mathrm{t_k} < \mathrm{t}$, and (2) explicitly incorporate knowledge that $W_i(\mathrm{t}) = s_i$. We explicitly incorporate the information that $W_i(\mathrm{t})$ is known to be in state s_i by deleting all states except for state s_i, and in that way, reducing the state space. Also, if a node $V(\mathrm{t})$ has no successors and if $\Pr(V(\mathrm{t}) = \mathrm{s}) = 0$, then we can delete state *s*.

In a special case, when the node $V(\mathrm{T})$ has no predecessors, its state is known to be s_i, and its other states $s_j \neq s_i$ are deleted, the conditional probability tables of its successor nodes must be updated to reflect this. Now, there may exist states that are impossible. All such states must then be deleted, and the pruning procedure performed recursively on successors.

These are just some basic pruning procedures, and they are not sufficient for controlling the inference complexity. There are also some marginal pruning procedures. For example, if there is no initialisation information, no pruning will be performed. Even with initialisation information, if a DBN models sensors failure there is always a small chance that the data is incorrect.

The basis of making a pruning decision is the trade off between the savings on execution of the inference versus the likelihood of making an error.

2.4. Different levels of creating DBNs

Table 1 Methods for creating DBNs structure and determining their parameters

Structure / Observability	**Method**
Known / Full (complete data)	Simple statistics
Known / Partial (incomplete data)	EM or gradient ascent
Unknown / Full	Search through model space
Unknown / Partial	Structural EM

Creating DBNs from data can be characterised as a very complex problem. Four cases have been distinguished in [7] for describing these problems, as shown it Table 1. Full observability (complete data) means that the values of all variables are known. Partial observability means that we do not know the values of some variable. Such case exists because in some situations variables cannot be measured, and then they are called <u>hidden variables</u>. It is possible that they can be measured in training data, but they are not, and then they are termed as missing variables. Unknown structure means that we are not in position to know the whole topology of the network.

2.4.1. Known Structure, Full Observability

We assume that the goal of learning in this case is to find the values of the parameters of each CPD which maximizes the likelihood of the training data, that contains *S* independent sequences, each of which has the observed values of all *O* observable nodes per slice for each of *T* slices. For reason of simplifying the discussion we will assume that the parameter values for all nodes are constant across time. If *S*=1, we cannot adequately estimate the parameters of the variables in the first slice, so we usually assume that these parameters are fixed a priori. This leads us to *N*=*S*(*T*-1) samples that we will use to estimate all other CPDs. In cases where *N* is small compared to the number of parameters that require fitting, we can use a numerical prior to make problem regular. In this case we call the estimates Maximum A Posterior (MAP) estimates, as opposed to Maximum Likelihood (ML) estimates.

By the chain rule of probability, that we described in section 2.2 we could find that the joint probability of all the nodes in the graph is

$$P(X_1,...,X_m) = \prod_i P(X_i | X_1,...X_{i-1}) = P(X_i | \mathrm{Pa}(X_i)), \qquad (31)$$

where m=O*(T-1) is the number of observable nodes in the unrolled network (excluding the first slice). A set of nodes denoted as a Pa(X_i) are the parents of node X_i. The normalized log likelihood of the training set is

$$L = \frac{1}{N} \log \Pr(D | G), \qquad (32)$$

where $D = \{D_1,..., D_s\}$, is a sum of terms, which corresponds to one node each:

$$L = \frac{1}{N} \sum_{i=1}^{m} \sum_{l=1}^{S} \log P(X_i | \mathrm{Pa}(X_i), D_l). \qquad (33)$$

Using this last equation we can maximize the contribution to the log likelihood of each node in the DBN independently.

2.4.2. Known Structure, Partial Observability

When some of the variables are not observable, the likelihood surface becomes multimodal, and we must use iterative methods, such as Expectation Maximization (EM) or gradient ascent, to find a local maximum of the ML/MAP function. These procedures need to use inference algorithm to compute the parameters for each node. These algorithms are sketched in section 2.3 and will not be discussed further in this section.

2.4.3. Unknown Structure, Full Observability

If we have a knowledge of a number and type of some states in the network, but we do not have the knowledge of their relation and mutual independence we should be interested in finding the way to learn the structure of DBN from observeable data and expert knowledge about the domain. Since the problems of this kind can enlarge theirs dimension in various situations we are forced to use power of modern computers. Many propositions exist that are concerned with the learning of DBN structure and they fuse different techniques and methods. They involve various modifications in structure, which can be classified as: (1) adding an edge, in any direction to the graph, (2) reversing the direction of any edge in the graph; and (3) removing an edge from the graph. The list of possible changes that we are performing in DBN structure is restricted to those that keep the graph as DAG.

As claimed in [5], to achieve this learning task we need to possess:

1. a metric for comparing potential structures against each other
2. a search algorithm for finding different potential structures

Algorithms that deal with such problems can be grouped into two categories, according to [13]. One category of algorithms uses heuristic searching methods to construct a model and then evaluates it using a scoring method. This process continues until we conclude that the structure of a new model is not significantly better than the old one. The other category of algorithms constructs Bayesian networks by analysing dependency relationships among nodes. The dependency relationships are measured using one kind of several Conditional Independence (CI) tests types.

According to Cheng et al. [13], if we compare these two types of algorithms we could conclude that the first category of algorithms is less time consuming in the worst case (when the underlying DAG is densely connected), but do not find the best solution for some real world domain, due to its heuristic nature. For the second type of algorithms certain assumptions has to be adopted, but these algorithms yield a network structure that is optimal or near the optimal solution. The restriction of these algorithms is that the CI tests with large condition sets may be unreliable unless the volume of data is enormous.

Cheng et al. [13] proposed one method that belongs to the second category of algorithms. The base of their assumption is that real world situations usually yield densely connected networks, and that they reveal very few independence relationships and thus contain little valuable information. Since CI tests with large condition sets are computationally expensive and may be unreliable, they try to avoid CI tests with large condition sets and use as few CI tests as possible.

In information theory, the mutual information of two nodes X_i and X_j is defined as

$$I(X_i, X_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}, \tag{34}$$

and the conditional mutual information as

$$I(X_i, X_j \mid Y) = \sum_{x_i, x_j, y} P(x_i, x_j, y) \log \frac{P(x_i, x_j \mid y)}{P(x_i \mid y)P(x_j \mid y)}, \tag{35}$$

where *Y* is a set of “dependent” nodes. In presented algorithm this conditional mutual information is used as CI test to measure the average information between two nodes when the statuses of some values are changed by the condition set of nodes *Y*. When $I(X\ , X_j \mid Y)$ is smaller than a certain threshold value, they d-separate this two nodes (X_i, X_j) by the condition set *Y*, and they became conditionally independent given nodes *Y*.

Construction algorithm that Cheng et al. proposed consisted of three phases: drafting, thickening, and thinning, and will not be described here. You can see details of this algorithm in [13].

Now we will resume to the algorithms that use heuristic searching in order to learn the structure of the BN or DBN. It is commonly assumed that the goal is to find the model with maximum likelihood. Given a training data set *D*, and instances of all variables in the network *X*, we will focus to find a Bayesian network $B = (G, \Theta)$ that best matches *D*. Here *G* denotes DAG whose vertices correspond to random variables of *X* ($\{X_1, X_2, ..., X_N\}$). The second component, Θ, represents the set of parameters that quantifies the network. The notion of best matches is defined using a scoring function. As we shall see, there exist several scoring functions that have been proposed in the literature.

This procedure starts by finding the model in which the sum of the Mutual Information (MI) between each node and its parents is maximal. The trouble is that MI graph will be a complete graph. Specifying that we prefer sparse models, by Bayes’ rule we gain the MAP model, which maximizes

$$\Pr(G \mid D) = \frac{\Pr(D \mid G)\Pr(G)}{\Pr(D)}. \tag{36}$$

If we take the logarithms of equation (36) we will simplify the problem of minimizing:

$$\log \Pr(G \mid D) = \log \Pr(D \mid G) + \log \Pr(G) + c. \tag{37}$$

where *c*= $\Pr(D)$ is a constant independent of G. This approach is known as Minimum Description Length (MDL) approach.

An exact Bayesian approach to model selection is usually unfeasible, since it involves computing the marginal likelihood $\Pr(D) = \sum_G \Pr(D,G)$, which is a sum over an exponential number of models. However, we can use asymptotic approximation to the posterior, called Bayesian Information Criterion (BIC) and BDe score. Both of these scores combine the likelihood of the data according to the network - $\log \Pr(D \mid B)$, with some penalty relating to the complexity of the network. When learning the structure of BNs, this complexity penalty is essential for modelling the DBNs, since the ML network is usually completely connected network [14].

Bayesian Information Criterion is defined as follows:

$$\log \Pr(G \mid D) \approx \log \Pr(D \mid G, \hat{\Theta}_G) - \frac{\log N}{2} \# G, \tag{38}$$

where N is the number of samples, $\#G$ is the dimension of the model, and $\hat{\Theta}_G$ is the ML estimate of the parameters. The BIC score has two important properties. First, since we are in log domain, the score of a DBN can be written as a sum of terms, where each term calculate the score for chosen variable and its parents. This implies that a local change of node's values that belongs to one family would affect only one of these terms. The other property is that the term that evaluates $X_i[t]$ given its parents is a function of the appropriate counts for that family. Given this, we can search for the best structure for the initial network independently of the search for the best structure for the transition network.

The most important conclusion of this section is that learning DBNs from complete data in basic uses the same algorithms as learning BNs from complete data.

2.4.4. Unknown Structure, Partial Observability

Most of the real world processes do not have structure that can be easily distinguished. Frequently, not all of the real world states can be measured or observed. This incomplete data and lack of ability to observe some crucial information demands a powerful tool for incorporating all the small peaces of information that we can gather. If we include changes that are present in almost all systems in time, the problem will further arise. Dependencies in these processes, concerning time can be between different time points and intervals. This means that even if the process is a stationary Markov process, the partial observations, we have, may not satisfy Markov property.

The most commonly used method to simplify this problem is the EM algorithm. The E-step of EM uses the currently estimated parameters to complete the data. This task is accomplished by computing the expected counts. The M-step then recalculates the maximum-likelihood parameter values considering that the expected values were true observed values. Each EM cycle is guarantied to improve the likelihood of the data given the model until it reaches a local minimum.

However, EM algorithm is developed to compute the parameters of a DBN and needs to be adjusted for learning a DBN structure from incomplete data. There are few algorithms that are in use for that purpose. Model-Selection EM (MS-EM) algorithm has been mostly used for this purpose, but the algorithm called Structural EM (SEM) algorithm yields better results. SEM algorithm has the same E-step as EM, completing the data by computing expected counts based on the current structure and parameters. The M-step has two parts. In first it is used in the same way as described above, for recalculating maximum likelihood. In the second part the M-step of SEM can use the expected counts according to the current structure to evaluate any other candidate structure. This is usually done by performing a complete search over all possible structures similar to one we have. The key step in SEM algorithm is to use an existing DBN to complete the data. All possible trajectories (i.e. joint assignments to all the variables in the training sequence) that are consistent with the partial information we have are considered. Then we average the counts in each one of these trajectories, based on the probability that our current model assigns to that trajectory [14].

The computational task for which can be said to be the most desiring in the above procedure is computing the expected counts for each family in the DBN. Thus, the key requirement is to compute the probability that a family of nodes takes on a certain set of values given the evidence. The evidence can be consisted of the past observations as well as the future ones. This can be done efficiently by converting the DBN to join tree and using algorithm, similar to the forwards-backwards algorithm used in HMMs.

3. Different Approaches for Incorporating Time in Bayesian Networks

It is extremely difficult to model time and uncertainty in a way that clearly and adequately represents the problem domains at hand. The study of probabilistic temporal formalisms is relatively new and existing approaches for integrating time and uncertainty rely on compromises in their representations of either time or uncertainty, or both.

Real-world domains that require a unified model of time and uncertainty include dealing with real-time system diagnosis, story understanding, planning and scheduling, logistics, resource management, as well as financial forecasting. Processes in these domains change over time in response to both internal and external stimuli as well as to the flow of time itself.

Extending the BNs to represent time domain is shown to be difficult problem. Therefore, different models for incorporating time into network representation were presented. Some of these techniques will be presented in following sections. Although general classification of these models has not been done yet, we will classify them into three broad categories:

1. Models that use static BNs and formal grammars to represent temporal dimension (known as Probabilistic Temporal Networks – PTNs),
2. Models that use mixture of probabilistic and non-probabilistic frameworks, and
3. Models that introduce temporal nodes into static BNs structure to represent time dependence

First we will present some earliest work that describes possible realisation of time dependent BNs, as well as some basic ideas about this concept. Then, we will describe these three models of dynamic (temporal) BNs. Since, first two categories are developed for special purposes, and has a limited usefulness, we will concentrate on third category. Another reason for this decision is that only third category retain pure BNs probabilistic framework, and just add temporal edges to this main structure.

3.1. Earliest Work on Dynamic (Temporal) Bayesian Networks

One of the first models presented for temporal reasoning in uncertain events was Berzuini's model [15]. Berzuini uses Bayesian networks for temporal reasoning about causality. His formalism considers part of the world of each period in time that is of our interest as an additional random variable in the network. This may considerably increase the number of variables in the network and the complexity of inference and in the extreme situations can make inference process intractable.

Significant research has been done exploring time nets (also called time-slice Bayesian networks) [2]. These approaches are built on the strong probabilistic semantics of BNs for expressing uncertainty. Dean and Kazanawa [16], in their earlier work, use survivor function to represent changing beliefs with time. Both, events and facts in this model are represented by random variables. These temporality representations allow inference that can be predicted and arable, but do not seem able to make inference about independent time points. Dagum et al. [17] use a dynamic belief network for forecasting. Temporal Abduction Problem (TAP) was presented by Santos [18]. In this model he uses an interval representation of time. In the TAP, each event has an associated interval during which the event occurs. Relationships between events are expressed as directed edges from cause to effect within a weighted and/or directed acyclic graph structure.

Temporal network models, no matter how they are represented have one common goal to show how random variables at time *t* are affected by static variables at time *t*, as well as by the random variables at time *t*-1.

3.1.1. Temporal Bayesian Networks

One of the earliest work that introduce temporal dimension in Bayesian networks (in a DBN manner) was Provan's these [19]. He suggested that the temporal evolution of the system could be modelled using a time-series process. Therefore, he proposed that "temporal arcs" could be used for connecting nodes in the BN over different time intervals. This means that if we have n BN models for different time instances, i.e. $BN_1, BN_2, ..., BN_n$, we could use these temporal arcs to represent a temporal sequence of Bayesian networks, called a Temporal BN or TBN in his work, but later known as DBNs.

He considers three types of approaches for TBN construction which make different tradeoffs: (1) knowledge base construction approaches, which produce the network at each time interval based on available data, (2) domain-specific time-series approaches, which model the evolution of a parsimonious subset of variables, and (3) model selection approaches, which tries to minimize some score of the predictive accuracy of the model without introducing too many parameters.

For TBN we can use heuristics to model the temporal evolution of only a subset of variables. Two different models for which variables should evolve are possible (the numbering in sequel is taken from Provan's work [19]):

1. Driving variables: This approach entails a domain-dependent identification of the system variables that are actually evolving, driving changes in other system variables. To this effect, we partition the system variables into a set of dynamic or evolving variables and a set of variables, which are either constant or whose changes are due to some dynamic variables. For completeness, we assume a set of variables that are independent of the dynamic variables. This partition should be made to trade off model accuracy for computational efficiency.
2. Observed variables: This approach seeks to model the findings that are the evidence of the internal evolution of the system. If these variables are not the ones that are driving the process under study, then one is estimating the values of the driving variables from the findings, using the model to relate the two classes of variables.

In the proposal of the approach, Acute Abdominal Pain model (AAP) was presented. AAP has three variable types: observable, intermediate (latent) and disease variables.

3.1.2. Characteristics of Temporal Networks

A probabilistic temporal representation, like most other temporal representations, must address the issue of discrete versus continuous time. Discrete time is useful for various applications and seems simpler to deal with than continuous time. However, interval based dense time seems more natural and elegant for reasoning. In temporal logics both representations are used. The aim of this section is to explain the way of incorporating both methods in DBNs, like it was described in [18]. This is possible because discrete time corresponds to a discrete probability and continuous time corresponds to continuous probability.

Probability theory can handle both this cases. In the continuous case, probability density functions are defined as functions of time. In the discrete case, probabilities themselves are functions of time. For continuous time, the evaluation of the probability of the truth of certain fluent states between two instants marking a period is the integral of the pdf between those two instants.

The second basic issue is how to associate probabilities and time. We can follow Berzuini's networks [15] and consider time as random variables, or we can parameterise probabilities with time. Representing time by a random variable complicates the network by increasing the number of nodes. It can happen that such temporal variables do not meet the definition of a random variable. A random variable may be defined roughly as a variable that takes on different values because of chance. Treating the probabilities as functions of time result in a simpler network.

3.1.3. Temporal Reasoning

From a temporal reasoning viewpoint there are at least two types of probabilistic relationships between two different time points or periods: (1) they may be completely independent of each other (unrelated), or (2) depended (related). If belief in fluent state *X* at time t_i is not affected by the knowledge of state *Y* at another time t_j than *X* at t_i is temporally independent of *Y* at time t_j. This independence means that the probability of *X* at t_i and that of *Y* in t_j are related by $\Pr(X(t_i) \mid Y(t_j)) = \Pr(X(t_i))$.

Two interesting special cases of reasoning about related time points that we need to consider in our speculation are persistence and causation. For the instantaneous standpoint, temporal reasoning is a set of atemporal Bayesian networks except for the probabilities, as they must correspond to the time point under consideration. If we are talking about dependent time instances Tafik et al. [18] proposed that we should consider probability transfer function (similar to pdf) between states of a different time instances and events. Events are changes of states of observed situations (variable), or some occurrence. A probability transfer function represents the effect of an event on other variables. A probability transfer function defines a relation between $\Pr(X|e)$ at the time *t* for all *t* where *X* is a random variable, and *e* is an event type.

Important issue about temporal BNs is about the purpose of the network. It is related to time representation and system application. It considers monitoring and occasional sampling. The distinction between monitoring and occasional sampling is a critical one. If occasional sampling is done frequently enough, it is equivalent to monitoring. The limit at which sampling can replace continuous monitoring, according to information theory, is equal to twice the maximum frequency in the signal.

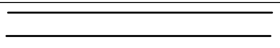
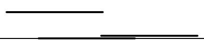
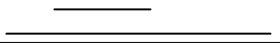
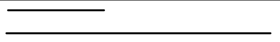
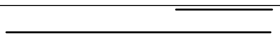
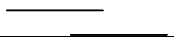
3.2. Probabilistic Temporal Networks (PTN)

This category of temporal networks tries to incorporate time into probabilistic framework introducing grammar rules. Preservation of static BNs structure will enable use of powerful BN inference technique, adjusted for this specific type of network, and grammar will introduce temporal relations between events.

Young [1] define the Probabilistic Temporal Network (PTN) as a model for representing temporal and atemporal information while remaining fully probabilistic. In this manner he proposes a technique that permits representation of time-constrained causality. The occurrence of specific event, as well as periodic and recurrent nature of a process lies in the base of these time-constraints. Bayesian networks form the foundation of the system that has to deal with probabilistic conditions, whereas, Allen's interval system and his thirteen relations provide the temporal basis.

Allen's interval algebra [20] is based on 13 relations on the intervals. Each event has an associated interval, denoted $[a,b]$ where a is the starting time point and b is the termination point. The relations between intervals are $\{=,<,>,m,mi,d,di,s,si,f,fi,o,oi\}$ (see Table 2). It is important to note that besides these thirteen relations between intervals there exist only tree relations between time points: precedes, equals, and follows.

Table 2. 13 relations of Allen's interval algebra

Symbol		Name	Relation
=	=	Equals	
<	>	Precedes	
m	mi	Meets	
d	di	During	
s	si	Starts	
f	fi	Finishes	
o	oi	Overlaps	

Of special importance for the purpose of incorporating this time relations with probabilistic framework is Allen's use of disjunctive sets to express uncertainty in the exact relationship between intervals (for example "interval A precedes or meets interval B" is written as A{<,*m*}B). As the author said "the model strictly adheres to the philosophy that intervals are primitive and that they have non-zero duration, in order to keep out some paradoxes".

BNs can be defined as probabilistic intensional systems, in which independence assumptions are used to restrict relevance. In general, if we want to investigate consequences of some real world process, we are searching for the world state with the highest likelihood. This is called belief revision in this work. Belief revision is a form of abductive reasoning.

If *W* is the set of all random variables in BN we choose for real world model, and *E* is our given evidence, any complete instantiation to all random variables that are present in *W*, which are consistent with *e* $(e \in E)$ is called explanation or interpretation of *e* [1]. Therefore, we can state the problem as finding an explanation $\hat{w}$ such that $\Pr(\hat{w} \mid e) = \max_{w} \Pr(w \mid e)$. $\hat{w}$ is termed as the most-probable explanation. The joint probability of any explanation *w*,

$$w = (X_1 = x_1) \wedge (X_2 = x_2) \wedge \ldots \wedge (X_m = x_m), \tag{39}$$

where $X_1, X_2, \ldots, X_m$ is an arbitrary ordering of random variables in W, and x is some assignment to random variable X_i, is found using chain rule:

$$\Pr(w) = \Pr(x_m \mid x_{m-1}, \ldots, x_1) \Pr(x_{m-1} \mid x_{m-2}, \ldots, x_1) \ldots \Pr(x_2 \mid x_1) \Pr(x_1). \tag{40}$$

While there is debate, in both philosophy and artificial intelligence, as to which representation, points or intervals, is most appropriate, it can be said that the intervals can be represented with beginning and end points in a point based approach. This was the leading idea in this work.

Probabilistic temporal networks presented in this section were focused on exploring recurrence and periodicity, temporal spacing between cause and effect, and modelling the time-of-reference. For further details refer to [1]. However, work presented in that paper has not fulfilled its completeness, since there is a lot of space for further improvement and research. The formulations presented in [1,2] need to be extended to perform belief updating. As one of most interesting and challenging problems, which need further investigation, is modelling continuous changes of observed facts.

PTN model yields a good inference result on repetitive, regular time events and situation. It could be implemented for slow changing and time overlapping events, but not for real time or near real time inference. For that purpose we must refer to DBNs, which are described in next sections.

3.3. DBNs as a Mixture of Probabilistic and non-Probabilistic Schemas

For representing some real world events it is not always possible to employ only probabilistic framework and simple grammar. We may need to combine several probabilistic tools or introduce some novel tools for describing a real world process.

Hidden Markov Models are usually used as a probabilistic tool for pattern varying problems. Naturally, first idea would be to integrate these models with a static structure of BNs. One of approaches based on this idea will be described in section 3.3.1. It explains how we can use BNs in conjunction with phase-type distribution, which is a specific type of HMMs.

Another work that tries to build DBNs from HMMs is presented in [21]. It introduces mixed-state DBNs for purpose of time-series classification. Building blocks of such DBN structure are HMMs and Linear Dynamic Systems (LDS). This work will be described in section 3.3.2.

3.3.1. Mixture of Phase-type Distributions and static BNs

Sterritt et al. proposed in [8] a combination of Bayesian Belief Networks - BBNs (which are the same as BNs) and Survival Analysis to create Dynamic Bayesian Belief Networks (DBBN), i.e. DBNs. They based their analysis on Markov models that are often been used to represent stochastic processes, and on structured phase-type (Ph) distributions as one type of hidden Markov models. These Ph distributions provide an intuitive and robust way of describing different static probabilistic processes. Models based on this theory are event driven. It means that when an event occurs system changes its state. Therefore, the process is presented as a sequence of latent phases as the states of a hidden Markov model.

Sterritt et al. in his approach combine this idea with the advantages of BNs (BBNs) in incorporating prior knowledge and causation into model that has a phase-type structure shown in Figure 6. The Causal network in their work is modelled as BN.

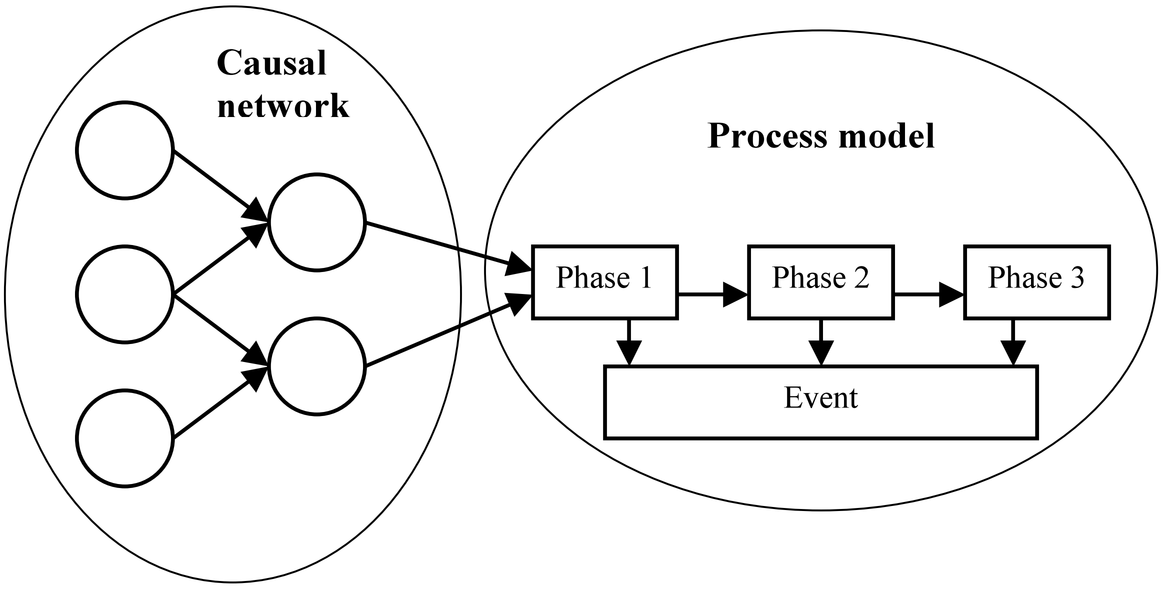


Figure 6 *DBBN consisting of interrelated causal nodes which perform temporal inference on the effect nodes that constitutes a process*

From this network we can inference the probability of particular phase in an event. The Process model can be defined with an event <E> which initiates a process <P> at *t*=0 and <P,t> indicates that the process P is active at arbitrary time *t*. Term prob<P,t> represents the probability that the process is still active at time t after initial event. This term is known as the survivor function, and is denoted by F(t). If we are working with continuous systems, the derivate of F(t), labelled f(t), represents the probability density function (pdf) of the time for which the process is active. According to author, f(t) can be defined by

$$\mathrm{f(t)}\delta\mathrm{t} = \mathrm{prob(process\ terminates\quad in\ (t, t + \delta t)\ |\ process\ is\ still\ active\ at\ t)}\,. \tag{41}$$

Variables in this model can be qualitative (the causal variables) and quantitative (the survival variables).

We can see that in this work Sterritt el al. tries to introduce the idea of Conditional Phase-type (C-Ph) distribution. Their conclusions were that this model is more appropriate for data processing in the intelligent fault management system, which is the application domain for this network.

3.3.2. Mixed-State DBNs

In [21] an integration of HMMs and Linear Dynamic Systems (LDS) is presented. It is based on estimation in LDSs and HMMs inference. These inference techniques can be considered as a special case of Bayesian inference in dynamic Bayesian networks. Nevertheless, these models are different in nature. They both use corpus of exact and approximate statistical inference and learning techniques invented for time-series modelling. HMMs are used for representing discrete states, and LDSs are used for continuous states. We can say that mixed-state DBNs are formed of HMM coupled with LDS. They are modelled in such way that outputs of a HMM form the driving input to a linear system (see Figure 7).

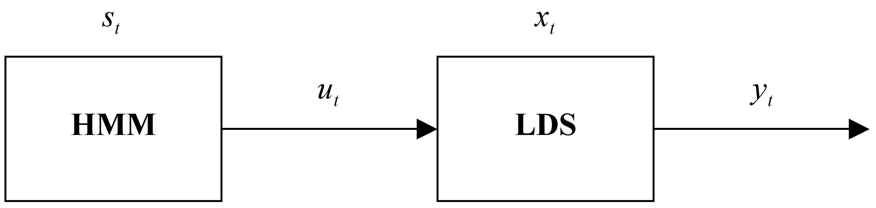


Figure 7 *Block diagram of a physical system (LDS) driven by an input generated from HMMs (equivalent to mixed-state DBNs)*

Inspiration for such a unique combination of models can be found in different applications, such as manoeuvring targets, robot control etc. A good example of a system, which is adequate for this kind of modelling is human hand/arm motion, during gestured communication, like described in [21]. Different dynamic models of simple articulated structures can be used to describe arm motion, and probabilistic concepts that influence the arm motion can be modelled as a HMM. This will enable not only to infer actions, but also to predict them.

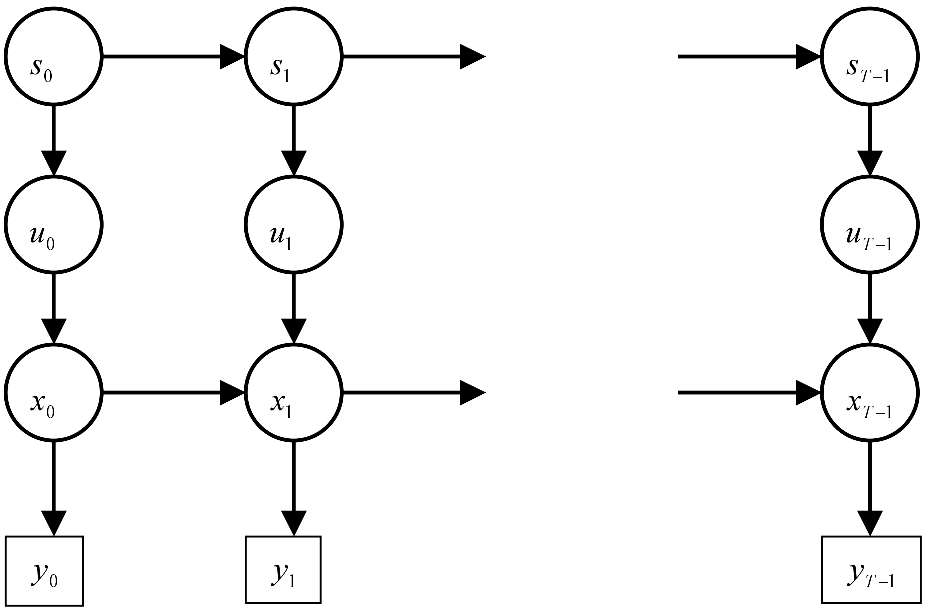


Figure 8 *Unrolled BN representation of the mixed-state DBN*

If we consider basic schema given in Figure 7 we could extend Bayesian network that represents dependencies among parameters to the mixed-state DBN. This is depicted in Figure 8, which represents unrolled DBN for T consecutive time slices. Then, we could use known parameters to compute joint distribution:

$$P(Y,X,U,S) = \Pr(s_0)\prod_{t=1}^{T-1}\Pr(s_t \mid s_{t-1})\prod_{t=0}^{T-1}\Pr(u_t \mid s_t)\Pr(x_0 \mid u_0)\prod_{t=1}^{T-1}\Pr(x_t \mid x_{t-1},u_t)\prod_{t=0}^{T-1}\Pr(y_t \mid x_t), \quad (42)$$

where *Y*, *X*, *U*, and *S* denote the sequences of observations and hidden state variables.

Parameters of mixed-state DBN can be learned using adapted ML learning algorithm for general BNs. Since this type of network has focused only on time-series models that fuses typical models of driving actions and continuous state models, we will not explained them in details. More facts and explanation of LDS mixed-state DBNs can be found in [21].

3.4. Pure Probabilistic DBNs

In this section we will describe several types of DBNs based solely on probabilistic framework. This means that we have only states, as objects, and arcs representing conditional dependence among states from same time slice, as well as ones that represent temporal dependencies among states from two consecutive time slices. The later, temporal arcs have the same nature as the arcs in static BN, except they represent probabilities that exist between states in different time instances.

3.4.1. Extensions of BNs Toward DBNs

In this type of networks we can extend static BNs for representing temporal processes in a several ways. These extensions to BNs can be classified into five categories, (similar classification can be found in [3]):

1. Addition of history node (see Figure 9), linked to the corresponding node in the Bayesian network to explicitly encode a temporal aspect into the Bayesian network.
2. Run time selection of Bayesian networks. This extension includes existence of pre-developed library consisting of fully structured DBNs that can be used at appropriate time, given the particular state of the system (entities represented in the dynamic system at the same time).
3. Dynamic structure changes in the Bayesian network to simulate reasoning strategies that are observed in human

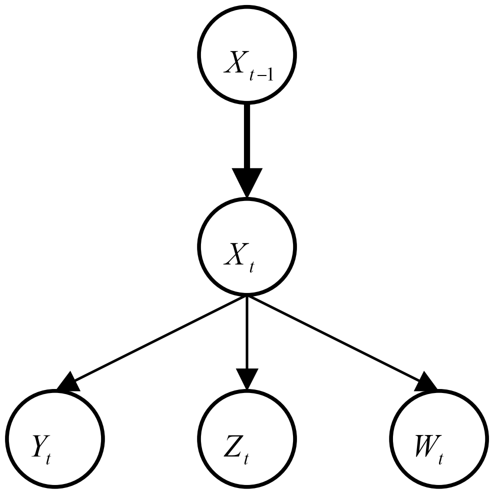


Figure 9 *Incorporating past information into BN using a history node*

   reasoning. These changes are based on a belief of the specified node, or nodes in the DBN structure. When beliefs in one node, or several of them, surpass specified threshold value, some structural changes are being triggered (see Figure 10).
4. Uniform representation of duplicated BNs for every time slice, with introduction of BNs for events representation. The nodes that belong to these event's BNs receive evidence from nodes of previous and current time slice.
5. Uniform representation of duplicated BNs for every time slice, with addition of temporal arcs between nodes that depend on some other nodes from previous time slices. Since these connections can lead to complex structure, in most cases only the connections from previous to current time slice are allowed.

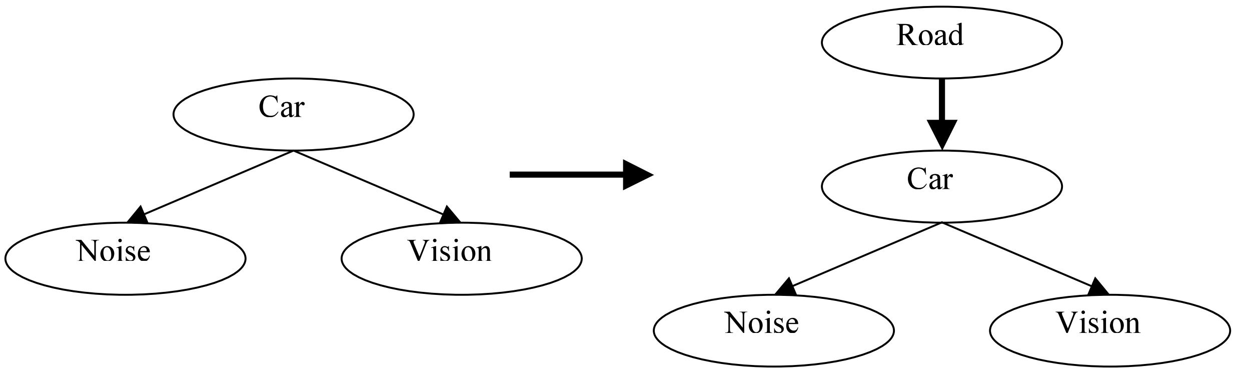


Figure 10 *A DBN for car and road detection from vision and noise sensors. The road detection node is added to the original car detection BN when the probability of the presence of a car gets more than a threshold value (0.5)*

Since networks that belong to first category can be considered as simple static BNs, with one extra node, we will not explain them further here.

Run-time selection of BNs is a method that is used in some earlier work on DBNs. We can find more about this type of networks in [3], where Singhal et al. used a locally developed Bayesian network package COBRA to work with Bayes networks and to incorporate temporality in their structure. The system selects a subset of these Bayes networks, corresponding to its current beliefs about the identities of objects in the world. The system is generally static, which means that it has a known set of DBNs, and that it cannot learn new DBNs structures. This system can be considered as dynamic system in the way that it can change its own structure, and employ a new DBN for next time instance.

These Bayes nets, presented in the library, were created by the human supervisor. Expert knowledge of these supervisors is also used to incorporate causality relationships and CPT values in the selected system environment (in this case robot's operating environment). We can draw a conclusion that the benefit of DBNs structured like this is that they can change their structure, without human interaction, depending on the current beliefs. However, they have a significant drawback, mostly due to (1) their robust system, which would naturally lead to the usage of simpler networks, and (2) that they cannot adjust itself to novel real world situations.

In the sequel we will describe other three extensions as separate sections.

3.4.2. Compositional Modelling with DBNs

Dynamic structure changes in the Bayesian network can be of two types:

1. Changes of CPT values over time, and
2. Introducing and removing arcs and/or nodes from DBNs structure over time.

However, changes that include introducing and removing parts of the network are very complex, and cannot be easily generalized. They can be described through four sub problems: (1) introducing arcs into network, (2) removing arcs from the network, (3) introducing new node, and (4) removing node from the network. All those sub problems are hard to handle, and since now, we have only few developed techniques that deal with such problems. Here we will further investigate only structural changes denoted in CPT values, and preserve above mentioned problems for section 3.

Zweig and Russell in [22] presented a model that uses decomposition techniques to represent real world situations. These decomposed situations can be divided into several temporal sequences. Such decomposition can be used in speech recognition or recognition of words in human handwriting. The basic idea that governed the researchers of stochastic temporal models in this work was that it is more convenient to create the submodels for each stage, than to create one general model. The entire process is then modelled as a composition of these stage submodels. By factoring a complex general temporal model into a combination of simpler models that represent one stage in the process, composition allows us to make reduction in the number of models. Each of these submodels needs to be learned from observations in specific time instances.

Model composition includes two important but dissimilar issues. First of them is specification of legal submodel sequences, i.e. to divide complex model into several simpler ones. For this purpose, authors used Stochastic Finite-State Automata (SFSA) algorithm, explained in [22]. Its aim is to describe a probabilistic distribution over possible submodel sequences. The second issue is submodel representation. DBNs, also called Dynamic Probabilistic Networks (DPNs) in their work, are used for this purpose.

The use of DBNs yields some advantages for submodels over HMMs that were used earlier. However, composing DBN structure is much more complicated than composing HMMs. Composite DBN has the following advantages over HMMs, as stated in [22]:

- Statistical efficiency. DBNs are factored representations of a probability distribution that reduces parameters exponentially unlike the number of parameters in representations that are not factored, such as standard HMMs. Hence these parameters can be estimated more accurately with a fixed amount of data.
- There are efficient, general-purpose algorithms for doing inference and learning in DBNs. They are upgraded versions and outbuildings of algorithms that we described in section 1.
- Different submodels can share their variables that would lead us to a natural way of describing transitional behaviour.

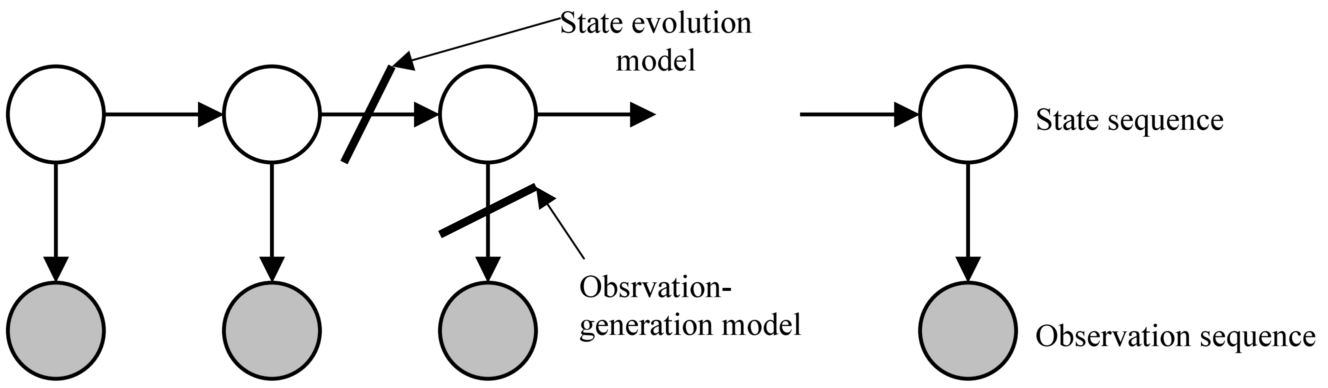


Figure 11 *Simple unrolled DPN to represent multiple time steps*

Models used in Zweig and Russell’s work are time-invariant. The topology of the network remains the same during different time instances, so they can be considered as DBNs. Therefore, CPTs that represent existing connections between temporal and atemporal nodes in the network remain the same. According to this, DBNs can be specified describing BN structure for two consecutive time slices (giving nodes, edges, and CPTs) and the links between them. When applied to a real world problem, we have a much more than two time slices. Therefore, an observation sequence is spread through all time slices, and the DBN has to be unrolled to produce a probabilistic network of the appropriate size to take into account all these observations.

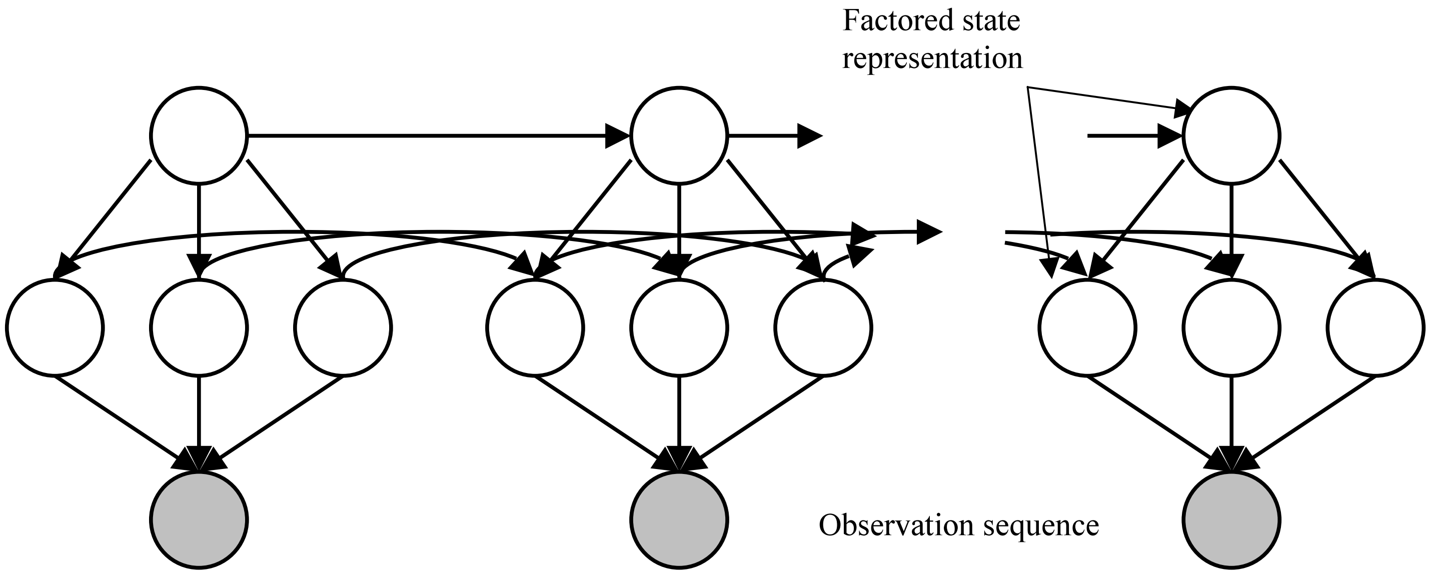


Figure 12 *A DPN with a factored state representation*

Figure 11 illustrates a generic DBN unrolled to show multiple time steps. The variables are divided into state variables, whose value is unknown, and observable variables, whose value is known. The state variables are related to the observation variables by the CPTs of the observation nodes. Figure 12 shows a more realistic DBN in which the hidden state has been factored. This means that dependencies among non-observable nodes can have different values for different time slices. These CPT values are changed in correspondence with one triggering node.

When doing inference or learning in examples such as speech recognition we must perform segmentation of observation sequences regarding to the model used at particular time instance. We need to know all partitions of an observation sequence between two consecutive models. Since the number of partitions can grow exponentially with the number of submodels presented, this is a computational demanding task that must be solved in an efficient and elegant way.

In the SFSA algorithm, the key problem is that CPTs must be uniform across time. Therefore, we can introduce an extra "index" variable in each time slice that affects the state-evolution CPTs. It is used for switching between different parameters (CPTs) that are used for describing a network for different spoken phonemes (Figure 13). This brings a uniform network structure that can generate the appropriate switching behaviours between models for inference. For further information refer to [22].

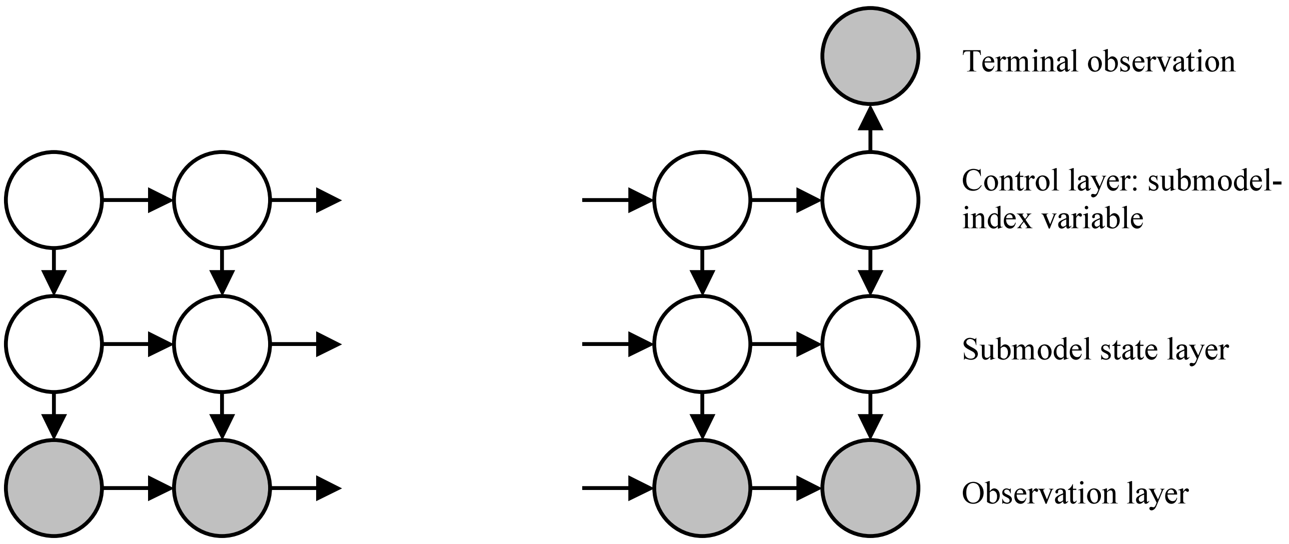


Figure 13 *A DBN structure for model composition. The submodel-index variable specifies which submodel to use at each time point.*

3.4.3. DBN for Representing Events

In this section, we will describe how we can use information obtained from different sources in order to infer events that are taking place between two consecutive time points. We can use these states from two consecutive time slices in DBN for determining the action that is happening in between. The structure of such network is shown in Figure 14. Three types of nodes can be distinguished: (1) world nodes (W), which describe the central domain variables, (2) event nodes (E), which represent a change in the state of a world node, and (3) observation nodes (O), which represent direct observations of the world nodes, or the observable effects of an event. Time intervals between different time slices are not unique and they depend on occurrence of discrete events. Each time slice within the network represents the static environment during that time interval. This is possible because the changes in state are not frequent. Therefore, we can say that the structure within time slice is often regular.

This idea is used in [23]. In this work, Nicholson proposed a generic dynamic belief network for monitoring application. Author exerts that these types of networks are usually highly connected, particularly between adjacent time slices. He supposed that the CPDs of nodes with parents in the previous time slice could change in time. This means that the CPDs are usually the function of the time interval. Information from different sensors is used for obtaining evidence, and nodes that represent this observed variables are dark-shaded in the Figure 14. When sensors detect change in state, belief updating is performed, recalculating the beliefs of current time slice (*t*) and providing prediction for the values of the world nodes at time slice *t*+1.

As we can see from Figure 14, two types of observations were presented in generic DBNs. These type of observation nodes are denoted as: (1) O(*T*) for the direct observation of a world variable, and (2) $O(T_i, T_{i+1})$ for the observation of an event.

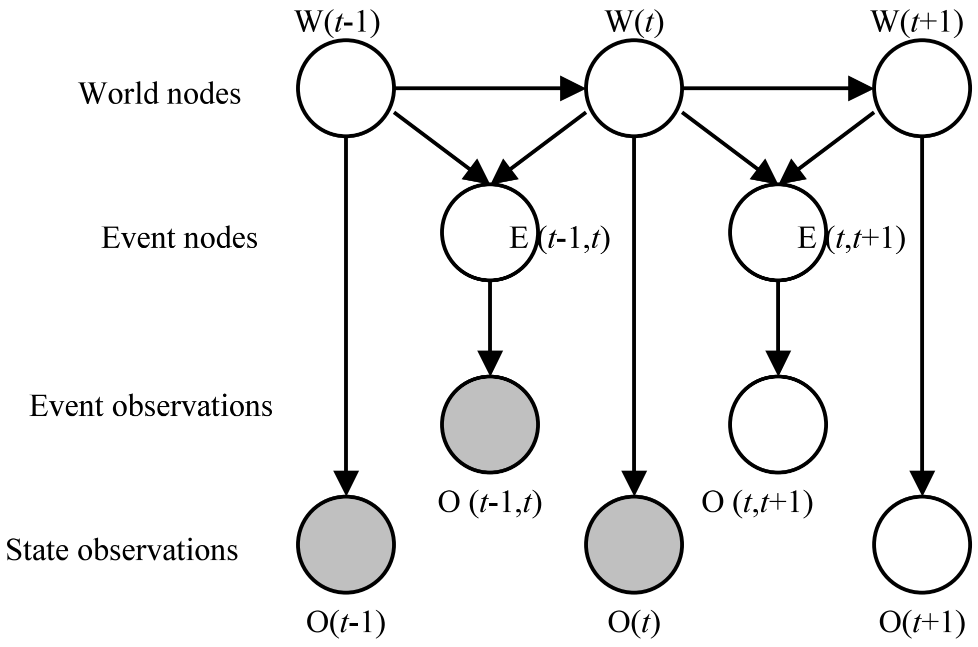


Figure 14 A *Generic structure for a dynamic belief network*

This technique was introduced for fall monitoring walking and fall diagnosis. Given evidence from sensor observations, the model outputs beliefs about the current walking status and makes predictions regarding future falls. This system yields a good performance for sparsely received observations in time, and can be used for forecasting in such situations.

3.4.4. DBNs with uniform structure

This group contains networks that have identical structure for every time slice and identical temporal conditional dependencies (arcs) between time slices. This means that conditional dependencies and states in one time slice are represented as a static BN. DBN is built over these BNs, by multiplying them for each time slice and by adding arcs between states from two consecutive time slices (nodes in BN), if they are temporally correlated. This is depicted in Figure 15.

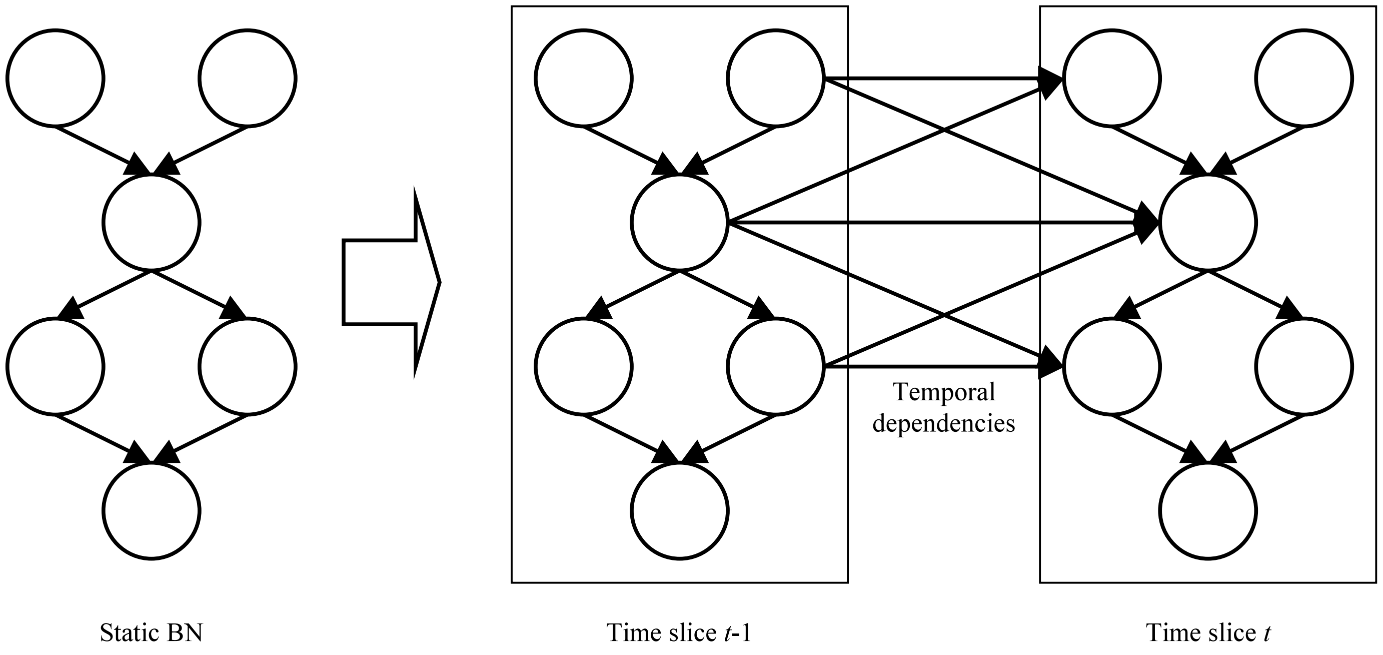


Figure 15 *Extending BNs toward DBNs*

In such structure, we have to know only probabilities of root and life nodes and CPTs that express the value of dependence between these nodes, and hidden ones, and CPTs between hidden nodes themselves. These CPTs can be obtained from experts, or learned from numerous examples, which will be explained in next section.

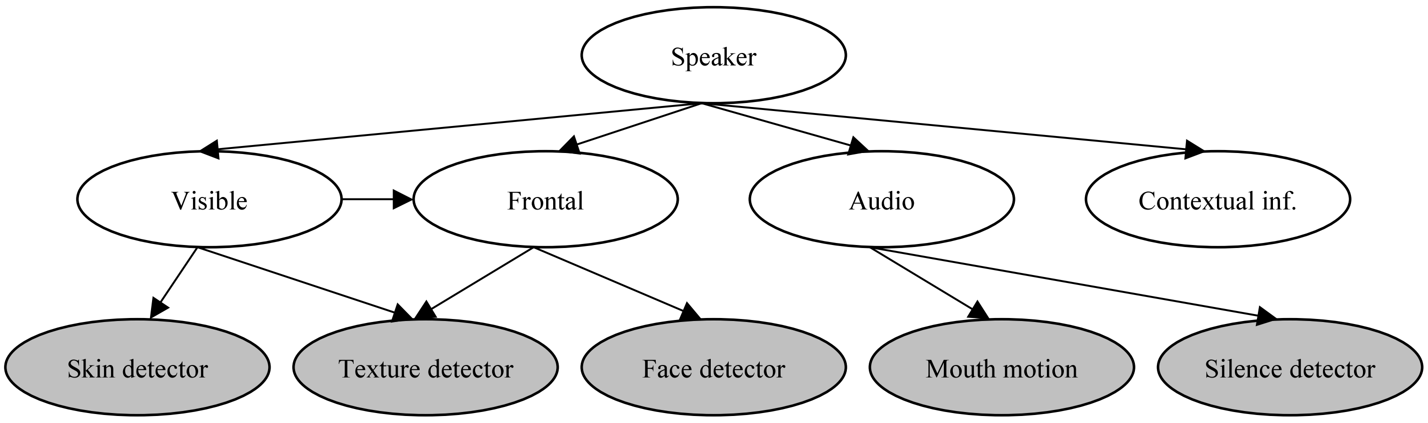


Figure 16 *Integrated audio-visual network*

For illustration, we will present two DBNs that belong to this category, given by Pavlovic et al. [24,25,26]. First they form the static network combining subnetworks that gather evidence from different sensors. These subnetworks were used to obtain visual information (visual subnetwork), audio data (audio subnetwork) and to fuse them with contextual information. This global network structure is depicted in Figure 16.

Authors' attempt to include temporal dimension into network was based on expert knowledge about the modeled problem. After introducing these temporal dependencies static BN is transformed into DBN, as shown in Figure 17. Here, shaded nodes represent observation nodes (sensor outputs).

If we want to infer if a user is speaking or not using static BN, we need to find the posterior probability for the Speaker node, i.e. the posterior probability $\Pr(S \mid SK, TX, FD, MM, CT)$.

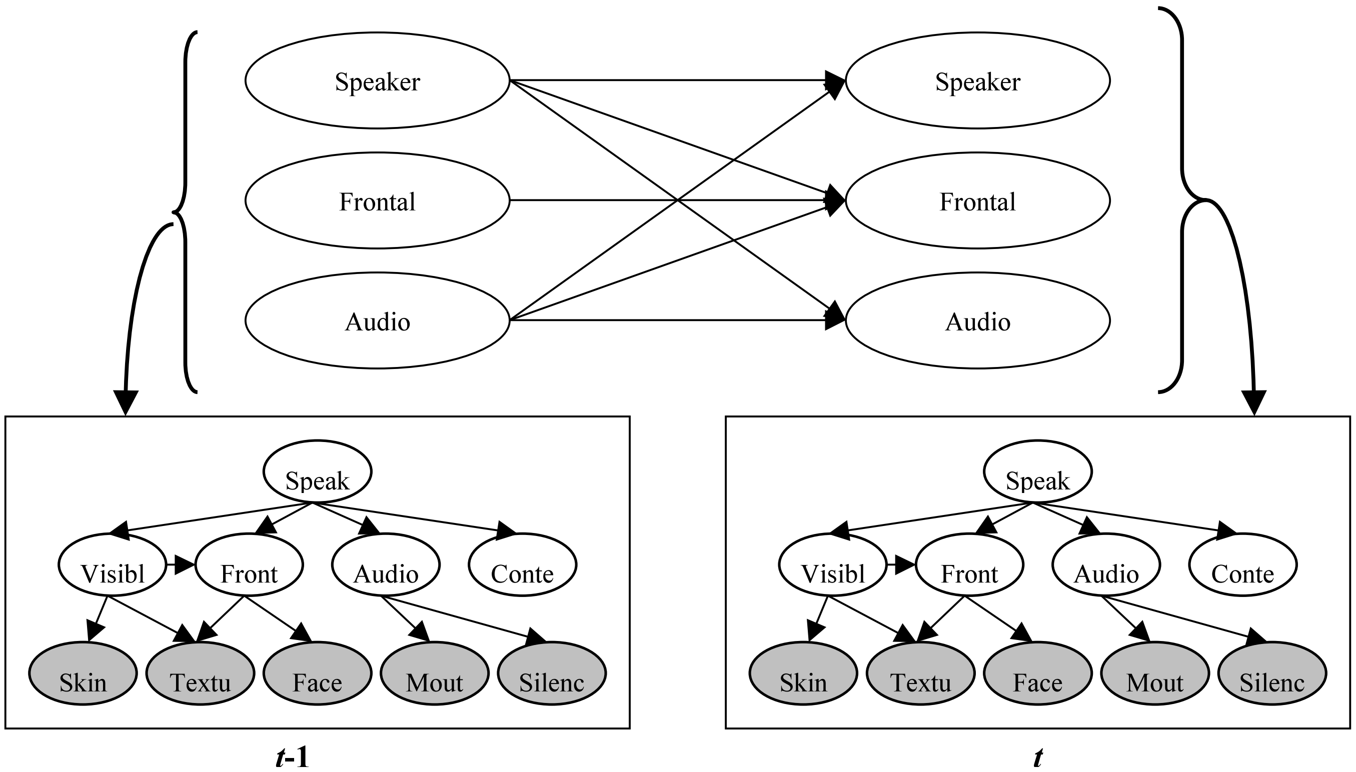


Figure 17 A *two time slices for dynamic Bayesian Network for speaker detection*

We can do this easily by marginalizing the joint probability distribution over speaker node or we can use numerous BN inference techniques. However, inference in dynamic network presented in their work is aimed to find the distribution of the speaker variable S at each time instance conditioned on a sequence of sensor observations,

$$M_1^T = \{SK_1, TX_1, FD_1, MM_1, CT_1, ..., SK_T, TX_T, FD_T, MM_T, CT_T\} . \tag{43}$$

If we want to find whether the speaker is present or not at time *t* we must compare values obtained for next two terms: $\Pr(S_t = true \mid M_1^T)$ and $\Pr(S_t = false \mid M_1^T)$. These posterior probabilities can be obtained directly from the forward-backward inference algorithm. If we want to predict the likelihood of the speaker from all the previous observations it is shown to be reachable.

Different uniform DBN models

Pavlovic et al. also explored a variant of DBNs called Duration Density DBN (DDDBN) model in [24], where state durations were explicitly modelled. This model yields better result, but according to authors, the complexity of inference in DDDBNs increases exponentially if we have to inference for long sequence of time slices. Also, learning procedure can be intractable in longer events. These problems will prevent us for using this type of DBNs if we are dealing with large number of time slices. Due to their complexity, DDDBNs can be only employed for the simpler models that have only few time slices included in the inference.

Another DBN architecture was presented recently in [26]. It is known as an Input/Output DBN (Figure 18). Current state of modelled domain (contextual information) forms an input to the network, while the observations of the sensors forms an output to the network. Together, they are used to infer the state of the speaker. Input/Output DBN structure was formed extending the input/output HMMs. The difference is that in Input/Output DBN the probabilistic dependencies between the variables have different origin, which is depicted in Figure 18. In this model, the speaker node represents the query node whose posterior probability we want to determine for every time slice. Another difference that discriminate this type of network from the others is that in Input/Output DBN continuous sensory outputs are modelled as conditional Gaussian distributions. That permits soft sensory decisions, which yields better results than discrete sensory outputs.

These DBN models are used for human-computer interaction. They are tested on audio-visual speaker detection. This project is lead by Pavlovic et al. [24,25,26] and up until now they presented several different DBN structures and learning approaches, which were tested and evaluated through numerous examples.

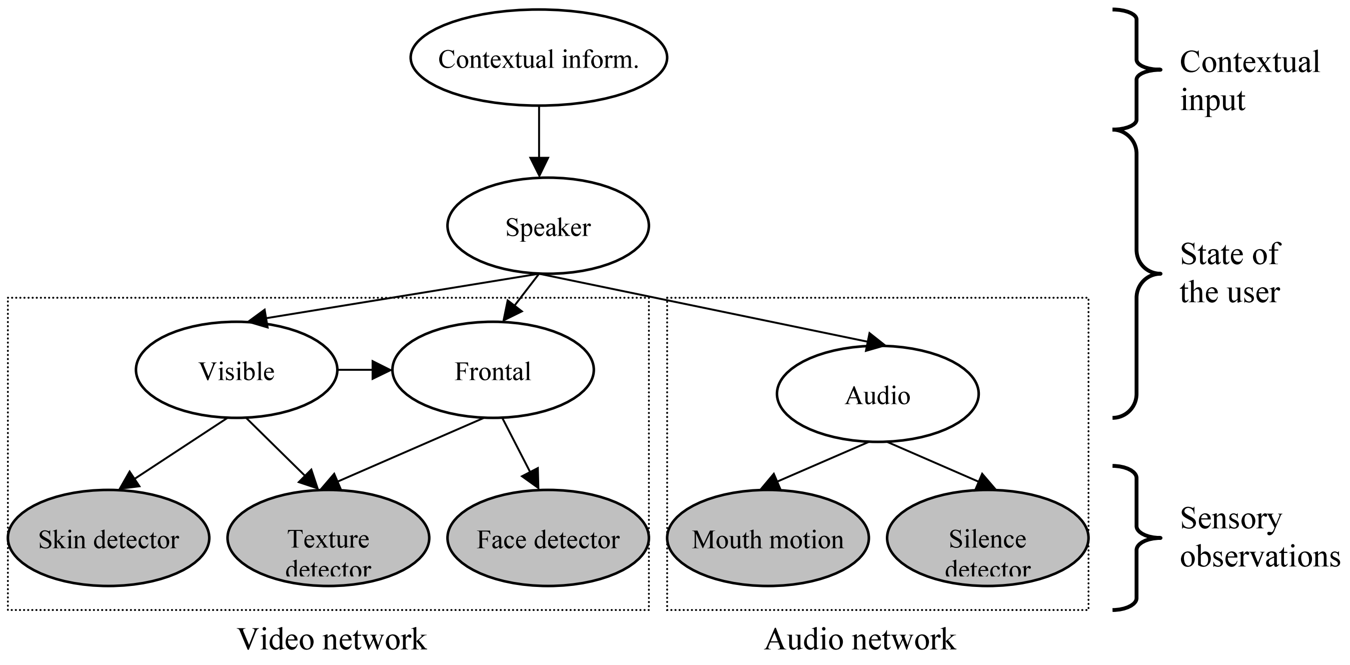


Figure 18 *Integrated audio-visual network (Input/Output structure)*

They use DBNs for human-computer interaction because it is uncertain, stochastic and because DBNs have well-constrained network structure that permits simplified inference. Another important issue that contribute to this employment is that DBNs computational power lies in the fact that the CPTs associated with the network are in most cases assumed not to vary over time. Therefore we could define network structure that is very simple and compact that would ease inference or learning.

However, these networks come with some lacks and incertitude. The problems, that arise when we transform static BNs in such uniform DBNs, are described in the next section.

4. Where the DBN Problems Came from?

Here we will describe problems that frequently arise in DBNs. Since in most cases we are dealing with exact problems, which can be easily represented by graph, we will not consider unknown structure problems. This leaves us with two major problems, one dealing with inference in DBNs, and the other is about parameter learning in DBNs.

Inference in DBNs is problematic because we have to unroll the network to represent dependencies between nodes from two consecutive time slices. In such unrolled network we must preserve conditional relationships between nodes, as well as influence that evidence nodes have on hidden nodes. Therefore, we will concentrate on unrolling network problems.

Parameter learning in DBNs can be difficult. Main reason is that usually we cannot have exact information about correctness of probabilities for some subset of hidden nodes. Expert knowledge can offer us only approximate CPTs for some node dependencies. For more exact representation we must employ efficient learning algorithms. Basic algorithms for learning are described in section 2, and here we will give only some extensions and describe novel approaches.

4.1. Unrolling Network Problems

As we described in the previous sections, DBNs allow connections between nodes from two consecutive time slices. These connections represent time dependencies, and these dependencies along with their CPTs represent state evolution model. Schafer and Weyrath [27] made general classification on different types of nodes that exist in DBNs. They distinguish three types of nodes: (1) dynamic nodes, that represent objects that evolve over time, (2) static nodes that represent objects that do not change over time, and (3) temporary nodes that receive different values over time.

When representing real world situations it is often the case that we need a structure of the network that remains the same in each time slice. That was the main constraint for incorporating powerful inference techniques that are invented for this purpose. Networks that are developed based on this constraint enable easy separation of DBNs and efficient topological speculation. This separation of DBNs is termed as unrolling or rolling up procedure. This procedure lies in the base of inference algorithm.

After performing the rollup procedure, the BN representing the time slice at time t-1 can be removed after calculating its influence on the BN representing time slice t. This pruning procedure usually makes dynamic nodes at time slice t become root nodes. Therefore, they need to receive proper prior probabilities. These priors are naturally defined throughout the belief of dynamic nodes at t before removing the part of the network. This could be generalized by forming a window of several time slices as proposed by Kjaerulff [28]. He also introduced backward smoothing slices, delaying the rollup procedure, and forecast slices for predicting the future beliefs.

Let us resume the process of inference. Inference starts by incorporating knowledge from previous time slice into current time slice. This step is often termed as rolling up or unrolling, as we saw earlier. By rolling up the network, using Markov property, evidence accumulated over time is always integrated into the current probabilistic network model. This rolling up method is based on node elimination described in [28]. However this node elimination may introduce additional links between nodes, and further complicate the network structure. According to [29], this procedure changes network structure over time, which yields many problems in calculating beliefs in the network. These problems can be divided into three major categories: (1) different node elimination sequences can result in drastically different connectivity in the resulting network, (2) modification of the network structure may force some inference algorithms (like ones based on junction trees) to perform expensive computations to modify or completely recreate internal data structures, and (3) modification requires more complex and computation-intensive code to manipulate the network.

To address these issues, four methods were developed for efficient rollup:

1. Connecting prior and transition network,
2. Temporary invariant networks,
3. Stochastic simulation, and
4. Exact representation of node dependencies.

In [14] we can find one way of representing DBNs throughout the BNs for the first time slice (prior network), and dependencies between two consecutive time slices, i.e. two BNs (transition network). They are fused to build unrolled DBN.

Temporary invariant networks and stochastic simulation will be briefly discussed as they have been described in original work [29]. These methods have intention to show how can we simulate human reasoning when driving a vehicle. Authors construct state evolution model through arcs, and their CPTs, by connecting the nodes from the BN of the time *t* to the BN at time *t*+1, and sensor model employing the CPTs for the arcs proceeding into node whose values can be observed at each time slice. In DBN structure learning approach they considered that sensor outputs are usually conditionally independent of each other given the variables that they measure. Therefore, they assume that Bayesian inference within a network will efficiently fuse together observations from several sensors, obtaining at once updated posterior probability distribution for the part of the network that gathers evidence from the sensors. For performing node elimination in inference procedure based on this sensor outputs, they considered two different techniques. One was to build temporary invariant network, which gathers evidence from previous time slice into one node, and the other was based on stochastic simulation of inference procedure.

Finally, we can decide to consider separately node dependences in the network. This would lead us to exact representation of node dependencies. As we saw nodes in DBNs can be dynamic, static, and temporary. For all temporal dependences of these nodes we can develop different unrolled representation. These various dependencies are described in section 4.1.4.

4.1.1. DBNs Built from Prior Network and Transition Network

According to Friedman et al. [14] for a dynamic Bayesian network the following needs to be defined: (1) a prior network and (2) a transition network. Figure 19 (a) indicates an example prior network. It incorporates the prior probabilities for all the

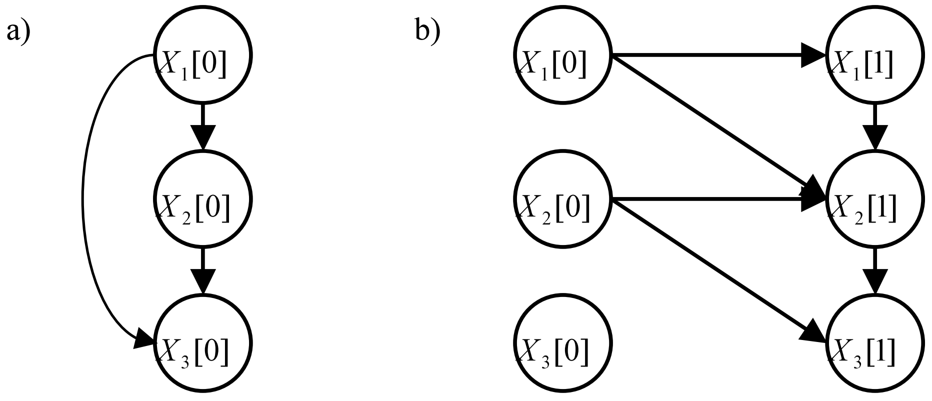


Figure 19 *A prior and a transition network defining a DBN for states X_1, X_2 and X_3*

variables in the network and the conditional probabilities at *t*=0, i.e. the initial time slice. Figure 19 (b) illustrates a transition network for the same DBN. The transition network represents conditional dependencies between consecutive time slices for all time slices *t*=1,2,…,*T*-1, as well as what the probabilities are for each state in time slice, conditioned on other states of previous and current time slice.

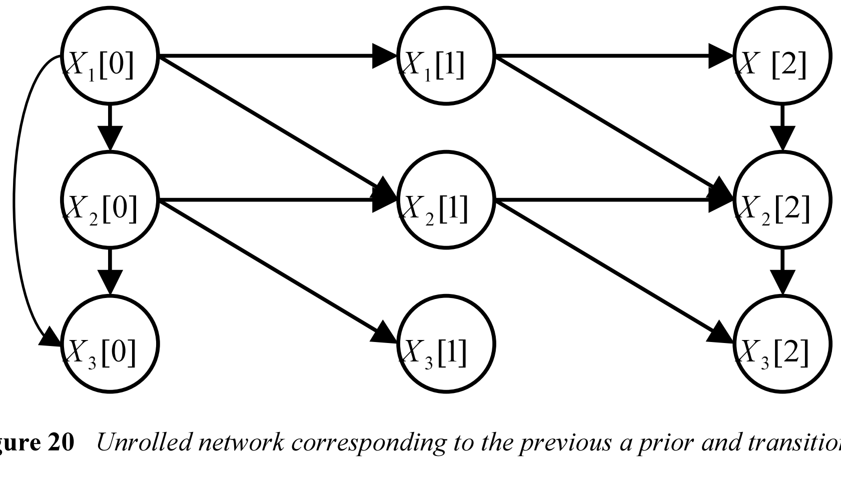


Figure 20 *Unrolled network corresponding to the previous a prior and transitional networks*

Given the prior and transition networks, we can construct a DBN of any number of states and any duration as follows. Within time slice t=0, the connections among the states and the local probability functions remains the same as in the prior network. So, we just copy the prior network. After that, for every time slice t=1,...,T-1, connections from states at time slices t-1 to time slice t, connections within time slice t, and the local probability functions (pdfs) within time slice t, have to be specified in the prior and the transition network.

This network configuration is based on assumption that this is a first order Markov network. This procedure of constructing DBN model is termed as network unrolling. Novel network structure is illustrated in the Figure 20.

4.1.2. Temporally invariant networks

Temporally invariant network can be defined as a DPN where the network structure remains the same after rollup procedure. This is only the case where only one temporal node receives all the necessary information from the previous time slice (top node in the Figure 21). Therefore roll up affects only this node and there is no need for inventing new edges in the network between nodes in the same time slice (static BN).

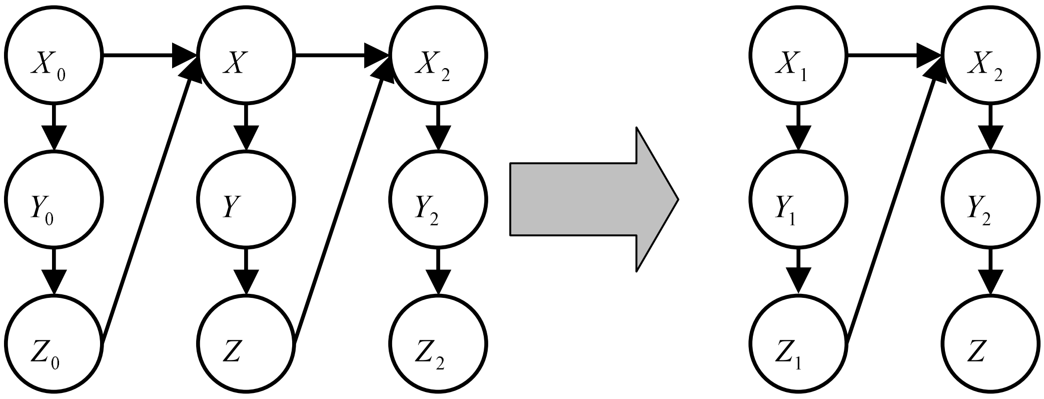


Figure 21 *A temporally invariant network whose structure is preserved after roll up*

Figure 22 shows a temporally variant network. In this case, not only one node from current time slice receives data from nodes coming from previous time slice. In the Figure 22, two nodes receives this information. This induces that these two nodes now become dependant on each other. To preserve this novel dependence relation, we must add an edge from top node to the bottom node persisting the dependence that exists between them in the previous time slice.

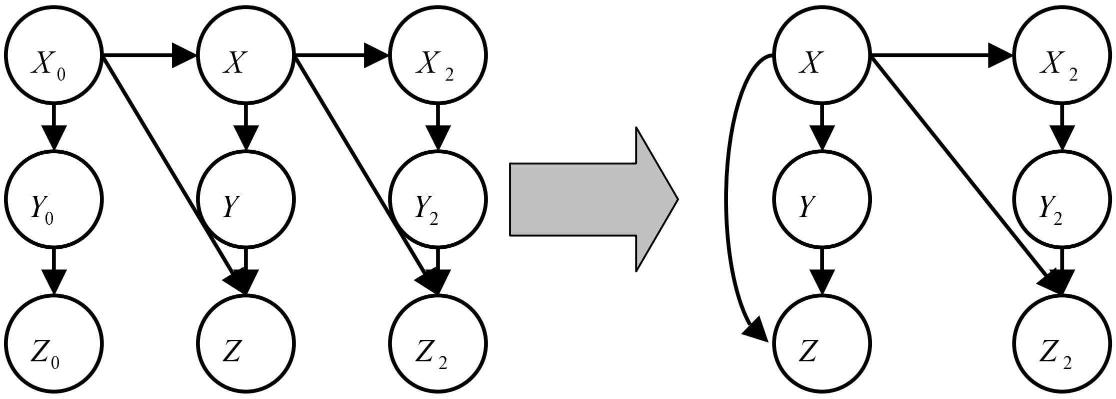


Figure 22 *A temporally variant network whose structure is altered after roll up*

Obviously temporally invariant networks are more appropriate and easier to maintain than temporary variant ones, especially for real-time applications such as driving a vehicle. In off line processing they would yield to less accurate results, and can bring many lacks and problems.

Nevertheless, in Figure 22, the edge leading from top to bottom node could be added in every time slice in the rollup, and we would create temporally invariant network. In our example this would make our graph completely connected, and in general will introduce very densely populated graph.

It is easy to show that a network with completely connected nodes in each slice is always temporally invariant, but such network is very hard for doing inference and learning. Thus, the better solution would be to transform this network into a temporally variant network, and find the more efficient algorithms for performing adequate sequence of node elimination. This could make easier our computation task and we could employ it for real time applications.

4.1.3. Stochastic simulation in DBNs

Stochastic simulation could be used to avoid the CPT manipulations involved in exact rollup like one in the temporally invariant networks. Forbes et al. [29], described stochastic simulation method as an attempt to approximate the current belief state using a collection of samples that represent real world scenario replicated to the network structure and beliefs. Each of these network's states is used to describe one possible evolution of the system involved in the consideration. The CPTs remain the same, as we have no interest to change them, because the received samples are a complete representation of our estimate of the joint distribution. Therefore the sample population and associated weighting factors are all we need to form the estimate of any marginal or joint probability in DBN.

There are several methods for stochastic simulation. In the above mentioned paper, a method known as likelihood weighting is presented. It consists of numerous trials where each trial is weighted by the probability it assigns to the observed evidence. Taking the weighted average of values on particular states obtained by these trials we can receive conclusions about probabilities of these states.

The use of likelihood weighting in DBNs reveals some problems, which are not easy to solve. Typical problem arise in situations where nodes representing observed evidence have same parent node. Two techniques exist that deal with this kind of situations. One of them is evidence reversal. This method relies on the algorithms that try to change the structure of DBN in an appropriate way and separate observed children from their parents. Another method, described in [29], is survival of the fittest sampling, which is a method that uses the likelihood weights to preferentially propagate the most likely samples. These two methods can also been employed together to bring off the best results.

4.1.4. Representing Node Dependencies in DBNs

In this section we will describe several types of node dependencies that can be found in DBNs. However, not all temporal dependencies between temporary, static, and dynamic nodes are allowed in DBN. Therefore, we can identify three types of dependencies that exist in a DBN as described in [27]: (1) between dynamic and temporary nodes, (2) between dynamic and dynamic nodes, and (3) between static and dynamic nodes. Some solutions of incorporating prior knowledge into new time slice in different consecutive network segment are described below.

Dependencies between dynamic and temporary nodes

The most frequent case in DBNs is when the child node of a dynamic node (*DN*) is a temporary node (*TN*) (also called an evidence node). The dependencies can be modelled as shown on the model A in the Figure 23. This type of dependence is common and desirable in most cases as it has been demonstrate that purpose of this dynamic node is to gather independent influences of several causes on an effect node (Heckerman and Jensen).

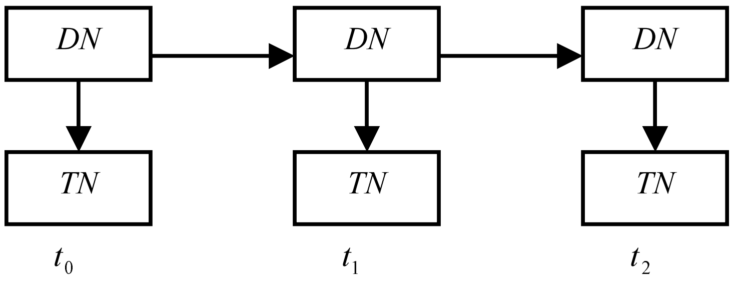


Figure 23 *Model A represents dependency between a dynamic node DN and a temporary node TN*

Dependencies between dynamic nodes

This is a situation when dynamic node *DN*2 has another dynamic node *DN*1 as a parent node. First treatment of this type of connection is similar to the previously described model A. Here, we represent *DN*2 not as a dynamic but as temporary node *TN* (model B1 in Figure 24 (a)). In each time slice we must create new instance of *TN* by a prediction. Any influence from *DN*2 in *t*-1 to *DN*2 in *t* is represented throughout common parent node. This model leads to slight loss of information on some nodes through time when we perform process of inference on a DBN, although the belief of some nodes receives the correct values.

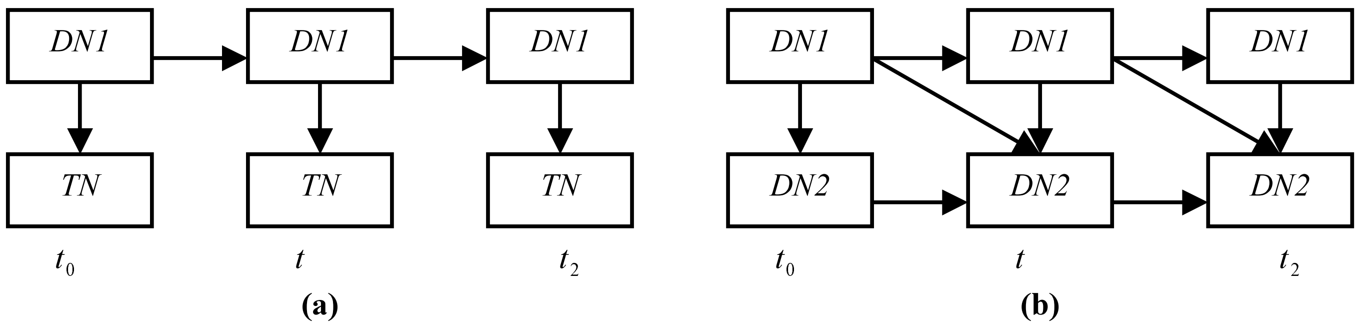


Figure 24 *(a) Model B1 represents dependency between dynamic nodes DN1 and DN2 where DN2 is simplified as a temporary node TN; (b) Model B2 represents dependency between dynamic nodes DN1 and DN2*

In another model B2 (Figure 24 (b)) there is no such loss of information, because *DN*2 is modelled, like it should be, as a dynamic node. Now, node *DN*2 collects the evidence from both, node *DN*1 in time slice *t*-1 and node *DN*1 in time slice *t*, as well as the influence of node *DN*2 from previous time slice. In this collecting evidence procedure, it is very important not to count the effects of node *DN*1 on *DN*2 two times. To overcome this problem, we introduced the edge between *DN*1 in time slice *t*-1 and *DN*1 in time slice *t*. Now, *DN*2 receives the evidence from *DN*1 in *t*-1 throughout an edge that connects them in time slice *t*-1. Although this is more appropriate and precise interpretation model, predictions and inference procedures are more costly in B2 than in B1.

Dependencies between dynamic and static nodes

A dynamic node *DN* can have a static parent node *SN*. According to [27], this dependence can be interpreted incorporating the effects of the static node *SN* in the dynamic child node *DN*. Now the dynamic node *DN* will convey all of the information

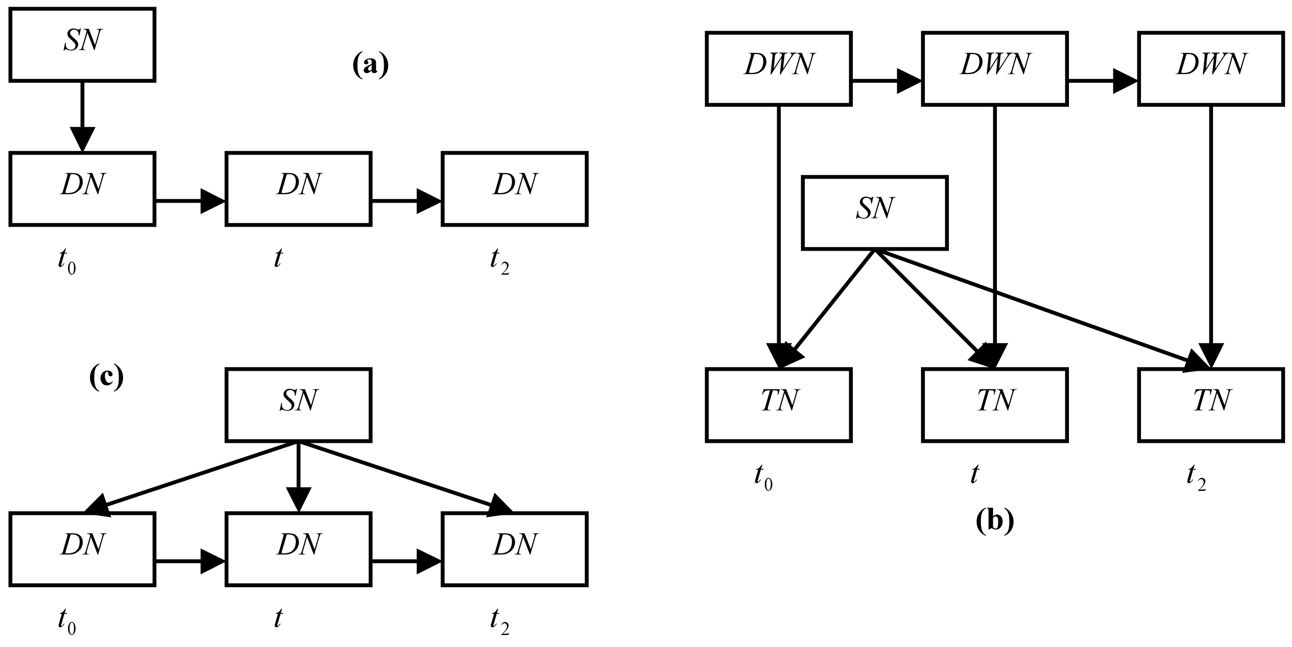


Figure 25 *Models for representing dependency between dynamic node DN and a static node SN. (a) C1 model; (b) C2 model that represents DN as a temporary model TN; (c) C3 model*

needed for inference. Therefore, we would have a simple network. This idea is represented on model C1 in Figure 25 (a). Authors claimed that the problems in this model can be differentiate in two classes: (1) node *DN* must be able to represent the effect caused by *SN*, and (2) static node *SN* is initially not known with certainty, and we can only estimate this node's value which would be changed as time progress forward. Problem (2) can be solved if we decide to trade accuracy for efficiency in a way that we delay rollup until we receive the exact probability of state *SN*.

In the model C2, *DN* will be considered as a temporary node and will be created at each time slice with corresponding belief (Figure 25 (b)). *DWN* represents the properties of *DN* without the influence of *SN*. *DN* as a temporary node receives influences of both, *SN* and *DWN*. Authors point out that in this model we ran into the same problems as in the model B1, and that we have a loss of information. Therefore, this model can be accepted only if the loss of information does not concern an important node.

Final model C3 (Figure 25 (c)) models *SN* as static node and *DN* as dynamic one. Another proposition is given in [27]. It says that we could even model *SN* as a dynamic node and simplify the model C3, which will ease the computations but leads to less precise inference.

4.2. Various Learning Techniques for DBNs Used in Audio-Visual Speaker Detection

As we stated, parameter learning in DBNs can be difficult and cumbersome. This is especially the case when we have more than one hidden node which state we cannot track. Therefore, algorithms presented in section 2 cannot be always efficient in such situations. This means that we need more powerful tools to perform learning in DBNs.

This section describes some of these techniques employed for specific DBNs, as well as problems associated with them.

4.2.1. Learning

When we use DBNs to model real world problems we rarely know all the parameters. Nevertheless, if we are in situation to know all the parameters we will be able to use DBNs directly to perform inference. However, if we do not know all the CPDs, we cannot form complete DBN with all parameters. Therefore, we will have to consult expert knowledge about specific domain of interest. Since we are dealing with problems that expert knowledge cannot support adequately, we must rely on our own knowledge of problem domain, our experiments, and observations made in this specific domain. However, experts' knowledge can give us good starting point.

The key for producing a powerful and effective DBN model would be choosing and employing adequate learning algorithm. For this purpose Maximum likelihood learning algorithm is used in many approaches similar to this one. As we mentioned, the goal of ML learning is to adjust the parameters to achieve the best fit to a set of training data. Since this is an iterative algorithm, we cannot be sure that it leads us to the best solution, or to the local minimum. Another problem that was point out in [25] is that we cannot put our confidence that the model will behave well in novel situations, i.e. would be a good classifier. Therefore, we must be cautious with employment of this learning procedure.

In addition to these problems, we could have hidden states in DBNs. This means that we have states that cannot be observed, i.e. their values cannot be uniquely determined in every time slice. If we have these hidden states in the network we cannot use ML algorithm. For such problems we must use Expectation maximization learning algorithm, which is based on ML learning algorithm. This algorithm is briefly described in the sequel.

If we denote unobserved variables with u_t, and observed with o_t, for time slice *t*, we can write probability distribution among all variables in a DBN as a:

$$\Pr(o_1,...,o_T,u_1,...,u_T) = \Pr(u_1)\Pr(o_1 \mid u_1)\prod_{t=2}^{T}\Pr(u_t \mid u_{t-1})\Pr(o_t \mid u_t). \quad (44)$$

If we denote model parameters as θ we can define this EM learning algorithm for DBN as showed below. Here θ_i^* denote optimised model parameters.

```
EM algorithm:
get initial guess of $\theta_0$ ;
do
    infer hidden variables $\hat{u}_t$ from measurements $o_1,...,o_T$ , using model $\theta_k$ ;
    $\theta^*_{k+1} = \arg\max_{\theta} \Pr(o_1,...,o_T,\hat{u}_1,...,\hat{u}_T \mid \theta)$ ;
until (convergence).
```

In order to simplify the learning procedure Pavlovic et al. [24], divided learning algorithm into two stages. In the first step they employ learning of the observation network CPTs excluding the temporal dependencies, and then used this fixed observation network to learn transition probabilities (CPTs). This learning division is obviously suboptimal but as they showed in their experiments it yields good parameter estimation. We could say that this accuracy for efficiency trade off satisfy learning criterions for this type of network.

In their papers Pavlovic et al. also refer to a new technique that is used for learning. It is called boosting and is introduced to improve accuracy of a weak classifier, as described in [26]. They adjusted this technique through error-feedback and with optimal combination of multiple classifiers. Combining it with ML learning algorithm they used it for estimating model parameters.

They termed this learning algorithm for DBN that uses boosting to improve recognition accuracy as Error Feedback DBN (EFDBN). This boosting algorithm (Adaboost) boosts the classification on a set of data points by linearly combining a number of weak models, each of which is trained to correct "mistakes" of the previous one. EFDBN framework was based on this algorithm. In short we can say that the goal of EFDBN is to change the estimate of the values of the arcs that contribute to incorrect inference in previous models. To show the power of this technique authors present their results that were obtained from numerous experiments that they conduct for evaluating this algorithm.

5. Conclusions

The question still remains: Which of previously described DBN models should we select to represent different causal dependencies over time? There is no unique answer. We can only point out that this decision should be made considering dependencies among the variables of interest in applied domain. This is because we could not trade accuracy for efficiency in every network that we create for a particular domain.

Even we decide which model to use, do we know everything we need. Can we assign proper values for every node and every edge in DBN that need assignment? Is expert knowledge sufficient enough for model characterization? Which algorithm should we use for inference, in order not to neglect important information and make computations more complex?

We can exert that DBNs with uniform structure have good theoretical base, and any real world problem can be easily transform into this networks. Most authors and most papers published describe these networks and many tools associated with them. Uniformity that goes with them is another powerful reason to employ them for purpose of inference in many real world uncertain events. Due to this uniform structure and parameter representation we can use them for problems that deal with huge number of time slices. Therefore, they are considered as main candidates for usage in recognition or modelling of uncertain events.

General conclusion can be that data and expert driven DBNs will provide a viable alternative to, often encountered, complex and ad hoc algorithms whose design is exclusively determined by the knowledge of an expert user, as stated in [24]. Therefore, we will watchfully choose structures, methods and algorithms that would yield best performance for our domain of interest.

6. References

[1] J. D. Young, "On Unifying Time and Uncertainty: The probabilistic Temporal Network", MSc Thesis, Faculty of the School of Engineering of the Air Force Institute of Technology, Air University, December 1996.

[2] J. D. Young, E. Santos Jr., "Introduction to Temporal Networks," The Seventh Midwest AI and Cognitive Science Conference, April 1996.

[3] A. Singhal, C. Brown, “Dynamic Bayes Net Approach to Multimodal Sensor Fusion,” Proceedings of the SPIE – The International Society for Optical Engineering, 3209-32020, October 1997.

[4] F. V. Jensen “An Introduction to Bayesian Networks,” University College London Press, 1996.

[5] T. A. Stephenson, “An Introduction to Bayesian Network Theory and Usage,” IDIAP Research Report 00-03, February 2000.

[6] J. Pearl, “Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference”, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[7] K. Murphy, S. Mian, “ Modelling Gene Expression Data using Dynamic Bayesian Networks,” Technical Report, Computer Science Division, University of California, Berkeley, CA, 1999.

[8] R. Sterritt, A. H. Marshal, C. M. Shapcott, S. I. McClean, “Exploring Dynamic Bayesian Belief Networks for Intelligent Fault Management System,” IEEE International Conference on Systems, Man and Cybernetics V, Nashville, Tennessee, USA, pp. 3646-3652, 2000.

[9] V. Pavlovic, “Dynamic Bayesian Networks for Information Fusion with Application to Human-Computer Interfaces,” PhD Thesis, University of Illinois at Urbana-Champaign, 1999.

[10] G. F. Cooper, “The Computational Complexity of Probabilistic Inference using Bayesian belief Networks, ” Artificial Intelligence, vol. 42, pp. 393-405, 1990.

[11] P. Dagum, A. Galper, “Forecasting Sleep Apnea with Dynamic Network Model”, Proceedings of the 9th Conference of the Uncertainty in Artificial Intelligence, 64-71, July 1993.

[12] A. J. Viterbi, “Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding algorithm, ” IEEE Trans. Information Theory, vol. 13, pp. 260-269, 1967.

[13] J. Cheng, D.A. Bell, W. Liu, “Learning Belief Networks from Data: An Information Theory Based Approach,” Proceedings of the Sixth ACM International Conference on Information and Knowledge Management.

[14] N. Friedman, K. Murphy, S. Russel, “Learning the Structure of Dynamic Probabilistic Networks,” 14th Conference on Uncertainty in Artificial Intelligence (UAI), 1998.

[15] C. Berzuini, “Representing Time in Causal Probabilistic Networks,” Uncertainty in Artificial Intelligence vol. 5, pp. 15-28, Elsevier Science Publisher B.V., 1990.

[16] T. Dean, K. Kazanawa, “A Model for reasoning about persistence and causation,” Computational Intelligence, vol. 5, pp. 142-150, 1989.

[17] P. Dagum, A. Galper, E. Horvic, “Dynamic Network Models for forecasting,” 8th Conference on Uncertainty in Artificial Intelligence, San Mateo, Morgan Kaufmann Publishers, Inc., pp. 41-48, 1992.

[18] A. Y. Tawfik, E. Neufeld, “Temporal Bayesian Networks,” Proceedings of Time94: International Workshop on Temporal Knowledge Representation and Reasoning, pp. 85-92, Pensacola, Florida, 1994.

[19] G. M.Provan, “Tradeoffs in Constructing and Evaluating Temporal Influence Diagrams,” Proceedings of the 9th Conference of the Uncertainty in Artificial Intelligence, 40-47, July 1993.

[20] J. F. Allen, “Maintaining Knowledge about Temporal Intervals,” Comm. ACM, vol.26, pp. 832-843, 1983.

[21] V. Pavlovic, B. J. Frey, T. S. Huang, “Time-Series Classification Using Mixed-State Dynamic Bayesian Networks,” CVPR’99, Ft. Collins, CO, 1999.

[22] G. Zweig, S. Russell, “Compositional Modelling With DPNs,” Report No. UCB/CSD-97-970, September 1997.

[23] A. E. Nicholson, “A Case Study in Dynamic Belief Networks: Monitoring Walking, Fall Prediction and Detection,” Technical Report 96/251, Department of Computer Science, Monash University, 1996.

[24] A. Garg, V. Pavlovic, J. M. Rehg, “Audio-Visual Speaker Detection using Dynamic Bayesian Networks,” Proceedings of 4th International Conference of Automatic Face and Gesture Recognition, Gren-Bole, France, pp. 374-471, 2000.

[25] V. Pavlovic, A. Garg, J.M. Rehg, T. S. Huang, “Multimodal Speaker Detection using Error Feedback Dynamic Bayesian Networks,” IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, SC, June, 2000

[26] V. Pavlovic, A. Garg, J. M. Rehg, “Multimodal Speaker Detection using Input/Output Dynamic Bayesian networks,” International Conference of Multimodal Interfaces, Beijing, China, October 2000.

[27] R. Schafer, T. Weyrath, “Assessing Temporally Variable User Properties With Dynamic Bayesian Networks,” User Modelling: Proceedings of the Sixth International Conference, 377-388, August 1997.

[28] U. Kjaerulff, “A Computational Scheme for Dynamic Bayesian Networks,” R-93-2018, ISSN 0106-1791, June 1993.

[29] J. Forbes, T. Huang, K. Kanazawa, S. Russell, “The BATmobile: Towards a Bayesian Automated Taxi,” Proceedings of IJCAI.