Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

Fusion of Domain Knowledge with Data for Structural Learning in Object Oriented Domains

Helge Langseth[*] HL@CS.AUC.DK

Thomas D. Nielsen TDN@CS.AUC.DK

Department of Computer Science, Aalborg University
Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark

Editor: Richard Dybowski

Abstract

When constructing a Bayesian network, it can be advantageous to employ structural learning algorithms to combine knowledge captured in databases with prior information provided by domain experts. Unfortunately, conventional learning algorithms do not easily incorporate prior information, if this information is too vague to be encoded as properties that are local to families of variables. For instance, conventional algorithms do not exploit prior information about repetitive structures, which are often found in object oriented domains such as computer networks, large pedigrees and genetic analysis.

In this paper we propose a method for doing structural learning in object oriented domains. It is demonstrated that this method is more efficient than conventional algorithms in such domains, and it is argued that the method supports a natural approach for expressing and incorporating prior information provided by domain experts.

Keywords: Bayesian networks, structural learning, object orientation, knowledge fusion

1. Introduction

The Bayesian network (BN) framework (Pearl, 1988; Jensen, 1996, 2001) has established itself as a powerful tool in many areas of artificial intelligence. However, eliciting a BN from a domain expert can be a laborious and time consuming process. Thus, methods for learning the structure of a BN from data have received much attention during the last years. For an overview see Buntine (1996) and Krause (1998). Current learning methods have been successfully applied in learning the structure of BNs based on databases. Unfortunately, though, only to a small extent do these methods incorporate prior information provided by domain experts. Prior information is typically encoded by specifying a prior BN hence, this information is restricted to the occurrence/absence of edges between specific pairs of variables.

In domains that can appropriately be described using an object oriented language (Mahoney and Laskey, 1996; Mathiasen et al., 2000) we typically find repetitive substructures or substructures that can naturally be ordered in a superclass–subclass hierarchy. For such domains, the expert is usually able to provide information about these properties. How-

∗. Current address: Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway. helgel@math.ntnu.no.

ever, this information is not easily exploited by current learning methods due to the practice mentioned above.

Recently, object oriented versions of the BN framework (termed OOBNs) have been proposed in the literature, see for example Mahoney and Laskey (1996), Laskey and Mahoney (1997), Koller and Pfeffer (1997), and Bangsø and Wuillemin (2000b). Although these object oriented frameworks relieve some of the problems when modelling large domains, it may still prove difficult to elicit the parameters and the structure of the model. Langseth and Bangsø (2001) describe a method to efficiently learn the parameters in an object oriented domain model, but the problem of specifying the structure still remains.

In this paper we propose a method for doing structural learning in an object oriented domain based on the OOBN framework. We argue that OOBNs supply a natural framework for encoding prior information about the general structure of the domain. Moreover, we show how this type of prior information can be exploited during structural learning. Empirical results demonstrate that the proposed learning algorithm is more efficient than conventional learning algorithms in object oriented domains.

2. Object Oriented Bayesian Networks

Using small and "easy-to-read" pieces as building blocks to create a complex model is an often applied technique when constructing large Bayesian networks. For instance, Pradhan et al. (1994) introduce the concept of sub-networks, which can be viewed and edited separately, and frameworks for modelling object oriented domains have been proposed by Mahoney and Laskey (1996), Laskey and Mahoney (1997), Koller and Pfeffer (1997), and Bangsø and Wuillemin (2000b).

In what follows the framework of Bangsø and Wuillemin (2000b) will be described, as it forms the formal basis for the proposed learning method. Note that we limit the description to those parts of the framework that are relevant for the learning algorithm. Further details can be found in the papers by Bangsø and Wuillemin (2000a,b).

2.1 The OOBN Framework

Consider a farm with two milk cows and two meat cows, and assume that we are interested in modelling the environment's effect on the milk and meat production of these cows.[1] Following the object oriented idea (Mathiasen et al., 2000), we construct a **Generic cow** class that describes the general properties common to all cows (see Figure 1): Specifically, as we are interested in the milk and meat production, we let *Milk* and *Meat* be *output nodes* of the class (depicted by shaded ellipses). That is to say, nodes from a class usable outside the instantiations of the class. Assuming that both the mother of a cow and the food a cow eats influence its milk and meat production, we let *Mother* and *Food* be *input nodes* (depicted by dashed ellipses) of the class; an input node is a reference to a node defined outside the scope of the instantiations of the class. Nodes that are neither input nodes nor output nodes are termed *normal nodes*. Note that the input nodes and output nodes form the interface between an instantiation and the context in which the instantiation exists. In the remainder of this paper we assume that all nodes are discrete.

1. A milk cow primarily produces milk and a meat cow primarily produces meat.

A class may be instantiated several times with different nodes having influence on the different instantiations through the input nodes hence, only the state space (the states and their ordering) of the input nodes is known at the time of specification[2] (for example, the cows might have different mothers). To avoid ambiguity when referring to a node in a specific instantiation, the name of the node will sometimes be prefixed by the name of the instantiation (that is, Instantiation-name.*Node-name*).

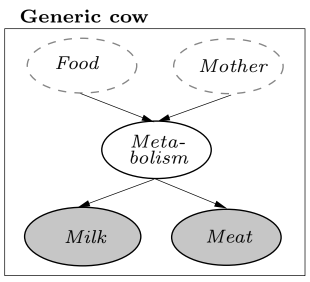


Figure 1: General properties common to all cows are described using the class **Generic cow**. The arrows are links as in normal BNs. The dashed ellipses are input nodes, and the shaded ellipses are output nodes.

In order to model the different properties of milk cows and meat cows, we introduce the two classes **Milk cow** and **Meat cow** (see Figure 2). These two cow specifications are subclasses of the **Generic cow** class (hence the "IS A **Generic cow**" in the top left corner of each of the class specifications). In a general setting, a class **S** can be a subclass of another class **C** if **S** contains at least the same set of nodes as **C**. This ensures that an instantiation of **S** can be used anywhere in the OOBN instead of an instantiation of **C** (e.g., an instantiation of **Milk cow** can be used instead of an instantiation of **Generic cow**). Each node in a subclass inherits the conditional probability table (CPT) of the corresponding node in its superclass unless the parent sets differ, or the modeler explicitly overwrites the CPT. The sub–superclass relation is transitive but not anti-symmetric, so to avoid cycles in the class hierarchy it is required that a subclass of a class cannot be a superclass of that class as well. Furthermore, multiple inheritance is not allowed, so the structure of the class hierarchy will be a tree or a collection of disjoint trees (called a *forest*).

Finally, to model the four cows in the live-stock we construct a class **Stock** that encapsulates the corresponding instantiations. In Figure 3 the boxes represent instantiations. For example, Cow1 is an instantiation of the class **Meat cow**, which is indicated by Cow1:**Meat cow** inside the Cow1 instantiation. Note that only input nodes and output nodes are visible, as they are the only part of an instantiation which directly interact with the encapsulating context (in this case the **Stock** class). This does not impose any constraints on which variables may be observed, it is merely a design technique to easier maintain large domain models. The double arrows are *reference links*. A reference link

2. This is also referred to as *strong type-checking*, see Bangsø and Wuillemin (2000a) for details.

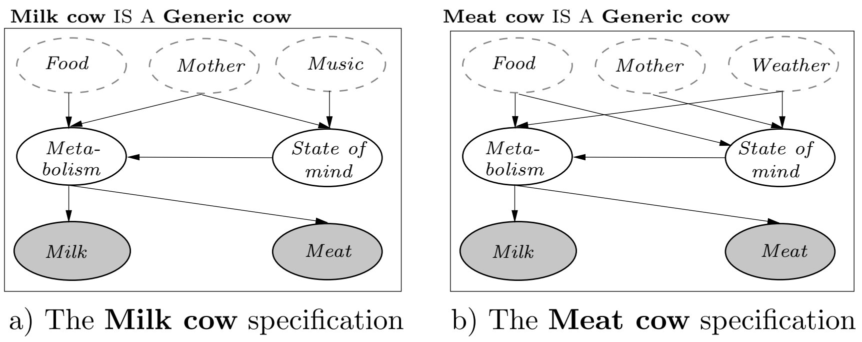

Milk cow IS A Generic cow
Food
Mother
Music
Meta-bolism
State of mind
Milk
Meat
a) The Milk cow specification
Meat cow IS A Generic cow
Food
Mother
Weather
Meta-bolism
State of mind
Milk
Meat
b) The Meat cow specification

Figure 2: a) A refined specification of a **Milk cow**. b) A refined specification of a **Meat cow**.

indicates that the leaf of the link is a reference (or *pointer*) to the root of that link.[3] For instance, the input node *Mother* of Cow1 is a reference to the node *Daisy*. This means that whenever the node *Mother* is used inside the instantiation Cow1, the node *Daisy* will be the node actually used (for instance during inference).

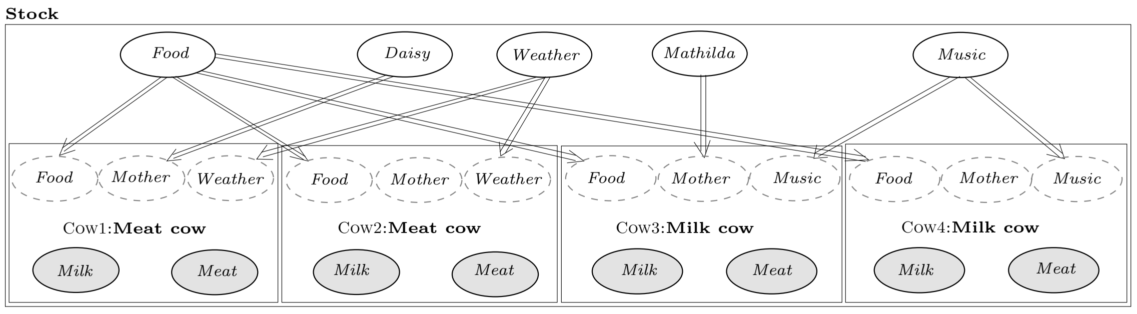


Figure 3: The **Stock** class with two instantiations of the **Milk cow** class and two instantiations of the **Meat cow** class. Note that some input nodes are not referencing any nodes.

If there is more than one instantiation of a class (for example, Cow1 and Cow2 in Figure 3), the OOBN framework gives rise to the *OO assumption* (Langseth and Bangsø, 2001). This assumption states that the CPTs of one instantiation of a class are identical to the corresponding CPTs of any other instantiation of that class (meaning that the domains of the CPTs are compatible and that the table entries are identical).

As the subclasses in a class hierarchy may have a larger set of nodes than their superclasses, the input set of a subclass **S** might be larger than the input set of its superclass **C**. Thus, if an instantiation of **S** is used instead of an instantiation of **C**, the extra input

3. To avoid confusion with the normal links in the model we do not use the terms “parent” and “child” when referring to reference links.

nodes will not be referencing any nodes. To ensure that these nodes are associated with potentials, the notion of a *default potential* is introduced: A default potential is a probability distribution over the states of an input node that is used when the input node is not referencing any node. Note that a default potential can also be used when no reference link is specified, even if this is not a consequence of subclassing. As an example we have that not all the *Mother* nodes in Figure 3 reference a node, but because of the default potential all nodes are still associated with a CPT. It is also worth noticing that the structure of references is always a tree or a forest; cycles of reference links are not possible (Bangsø and Wuillemin, 2000a).

Finally, inference can be performed by translating the OOBN into a *multiply-sectioned Bayesian network* (Xiang et al., 1993; Xiang and Jensen, 1999), see Bangsø and Wuillemin (2000a) for details on this translation. Alternatively, we can construct the *underlying BN* of the OOBN: The underlying BN of an instantiation I, BN_{I}, is the (conventional) BN that corresponds to I including all encapsulated instantiations. There is exactly one such underlying BN for a given instantiation, and it can be constructed using the following algorithm (Langseth and Bangsø, 2001):

Algorithm 1 (Underlying BN)

1. *Let* BN_{I} *be the empty graph.*
2. *Add a node to* BN_{I} *for all input nodes, output nodes and normal nodes in* I.
3. *Add a node to* BN_{I} *for each input node, output node and normal node of the instantiations encapsulated in* I, *and prefix the name of the instantiation to the node name (*Instantiation-name.*Node-name). Do the same for instantiations contained in these instantiations, and so on.*
4. *Add a link for each normal link in* I, *and repeat this for all instantiations as above.*
5. *For each reference tree, merge all the nodes into one node. This node is given all the parents and children (according to the normal links) of the nodes in the reference tree as its family. Note that only the root of the tree can have parents, as all other nodes are references to this node.*

An input node that does not reference another node will become a normal node equipped with a default potential. This can also be seen in Figure 4 which depicts the underlying BN of an instantiation of the **Stock**-class (Figure 3).

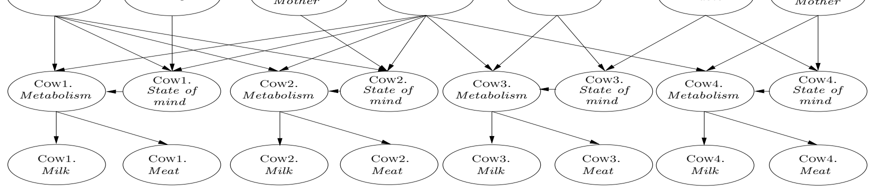


Figure 4: The underlying BN for the OOBN depicted in Figure 3.

Note that nodes associated with default potentials (Cow2.*Mother* and Cow4.*Mother*) can be marginalized out as they have no effect in the underlying BN. It is also worth emphasizing that an OOBN is just a compact representation of a (unique) BN that satisfies the OO assumption, namely the underlying BN (this can also immediately be seen from Algorithm 1).

2.2 The Insurance Network

In order to emphasize the possible use of encapsulating classes, we give an OOBN representation of the insurance network by Binder et al. (1997). The insurance network, depicted in Figure 5, is taken from *The BN repository* (Friedman et al., 1997b). The network, which consists of 27 nodes, is designed for classifying car insurance applications based on the expected claim cost. This information is captured in the nodes *PropCost* (Property cost), *ILiCost* (Liability cost) and *MedCost* (Medical cost).

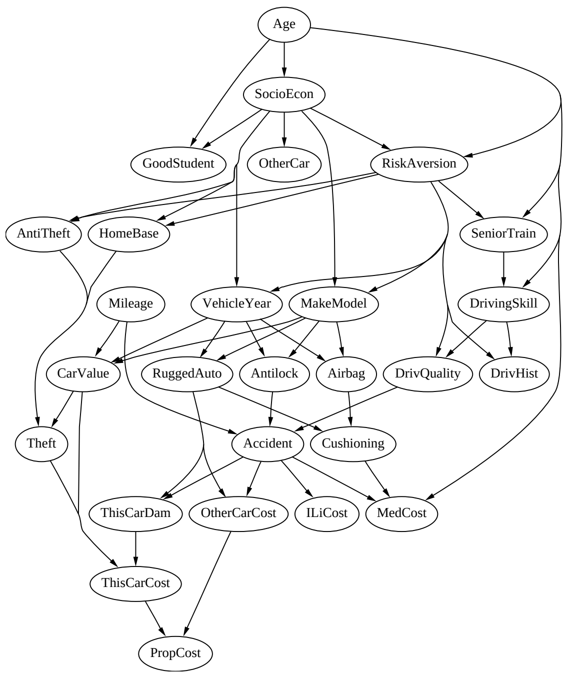


Figure 5: The insurance network, used for classifying car insurance applications.

The corresponding OOBN representation of this network is based on six classes (**Insurance**, **Theft**, **Accident**, **Car**, **Car owner** and **Driver**), which can be seen as describing different (abstract) entities in the domain. These classes are designed such that they adhere to the design principle of high internal coupling and low external coupling, see for example Mahoney and Laskey (1996) and Mathiasen et al. (2000).

For instance, the class **Car** describes the properties associated with a car (specific for this domain). The nodes *Cushioning*, *Mileage*, *CarValue*, *RuggedAuto* and *Antilock* are the only nodes "used" outside the class hence, they occur as output nodes whereas *Vehicle year* and *Make model* are input nodes and *Airbag* is a normal node (see also the encapsulated instantiation C:**Car** in Figure 6). As another example, consider the class **Driver**, which models the driving characteristics of a car owner. In the insurance context, driving characteristics are an integral part of the notion of a car owner and (by the above mentioned design principle) an instantiation of **Driver** is therefore encapsulated in the class **CarOwner**. The class **Insurance** encapsulates the corresponding instantiations of the other classes. Figure 6 depicts the final OOBN model in form of the **Insurance** class. Note that only the interfaces of the encapsulated instantiations are shown.

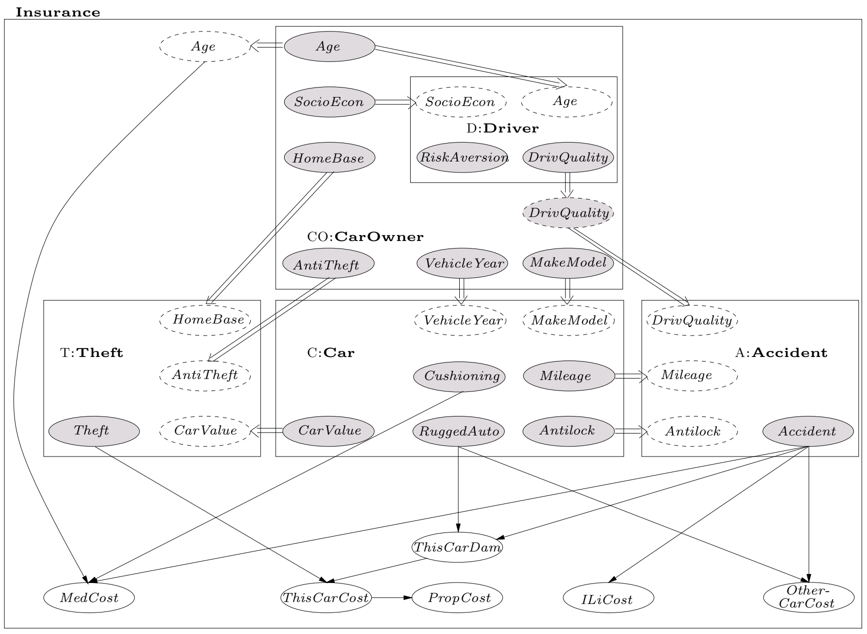


Figure 6: An OOBN representation of the insurance network. Notice that only the interfaces of the encapsulated instantiations are shown. Note also that we use a slightly non-standard graphical presentation for visualization purposes.

The **Insurance**-class is constructed so that the underlying BN of an instantiation of that class corresponds to the BN given in Figure 5. In this respect it is worth noticing the active use of reference links. For example, there are two *CarValue*-nodes in the OOBN: C.*CarValue* is defined in C:**Car**, but as C.*CarValue* is a parent of T.*Theft* (confer also the underlying BN in Figure 5), it is imported into T:**Theft** using an input node (which is named T.*CarValue*). The reference link between these two nodes shows that it is the same random variable that is used in both situations. That is, T.*CarValue* is a reference to C.*CarValue*. This is required since *CarValue* is defined outside the scope of the instantiations of the **Theft**-class.

2.3 OOBNs and Dynamic Bayesian Networks

An important set of Bayesian networks is *dynamic Bayesian networks* (DBNs), which model the stochastic evolution of a set of random variables over time, see for example Kjærulff

(1992). Traditionally, a DBN specification consists of *i)* a BN over the variables at $t = 0$, and *ii)* a transition BN over the variables at $t = 0$ and $t = 1$. These two networks can alternatively be described using OOBN classes, where the time-dependence is encoded by *self-references* between nodes; a self-reference is a reference between a node and an input node in the same class.[4] More precisely, when using the OOBN framework for modelling DBNs we construct two classes: One class representing the time-slice at $t = 0$, and another class whose instantiations correspond to the time-slices at $t > 0$. The dependence relation between a time-slice and the previous time-slice is then represented using self-references within the class specification, see also Bangsø and Wuillemin (2000b). Note that using OOBN classes for modelling time-slices also supports the introduction of encapsulated instantiations within the time slices.

3. Structural Learning

In what follows we review the basis for performing structural learning. The notation will, whenever possible, follow that of Cooper and Herskovits (1991) and Heckerman et al. (1995).

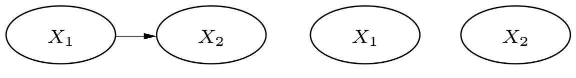
Consider a Bayesian network $BN = (B_S, \Theta_{B_S})$ over a set of discrete variables $\{X_1, X_2, \ldots, X_n\}$, where B_S is the graphical structure and Θ_{B_S} is the quantitative information. To describe B_S, the qualitative aspects of BN, we will use the following notation: r_i is the number of states for variable X_i, $q_i = \prod_{X_l \in \Pi_i} r_l$ is the number of configurations over the *parents* for X_i in B_S (denoted by Π_i), and $\Pi_i = j$ denotes the event that Π_i takes on its j'th configuration. For the quantitative properties, we use $\theta_{ijk} = P(X_i = k | \Pi_i = j, \xi)$ (we assume $\theta_{ijk} > 0$), where ξ is the prior knowledge. For ease of exposition we define

$$\Theta_{ij} = \cup_{k=1}^{r_i} \theta_{ijk}; \quad \Theta_i = \cup_{j=1}^{q_i} \Theta_{ij}; \quad \Theta_{B_S} = \cup_{i=1}^{n} \Theta_i \ .$$

Note that $\forall i, j : \sum_{k=1}^{r_i} \theta_{ijk} = 1$. Finally, we let $\mathcal{D} = \{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_N\}$ denote a *database* of N cases, where each case is a configuration $\boldsymbol{x}$ over the variables $\boldsymbol{X} = (X_1, \ldots, X_n)$.

The task is now to find a structure B_S that best describes the observed data, or in a more abstract formulation, to find the parameter space Ω_{B_S} that best restricts the parameters used to describe the family of probability distributions $\mathcal{F}_{\Omega_{B_S}} = \{f(\boldsymbol{x} \,|\, \Theta) : \Theta \in \Omega_{B_S}\}$. For example, let Ω' be the parameter space required to describe all probability distributions compatible with the complete graph for two binary variables X_1 and X_2 (see Figure 7a). With the above notation, Ω' is defined so that $(\theta_1, \theta_{21}, \theta_{22}) \in \Omega'$. For the empty graph in Figure 7b, the parameter space $\Omega'' \subset \Omega'$ corresponds to the parameter space Ω' where $\theta_{21} = \theta_{22}$ hence, Ω'' is a hyperplane in Ω'. Learning the structure B_S is therefore equivalent to finding the parameter space Ω_{B_S} that best describes the data; when learning the structure of a BN there is an injective mapping from the BN structure, B_S, to the associated parameter space Ω_{B_S}. However, as we shall see in Section 5, when we focus on learning OOBNs this is no longer true as some aspects (the OO-assumption) of an OOBN are not reflected in the underlying graphical structure. In that case it may be beneficial to think of structural learning as learning a parameter space Ω.

4. Self-references differ from reference links as the root of a self-reference is defined inside the instantiation, whereas the root of a reference link is defined outside the scope of the instantiation.



a) Complete graph b) Empty graph

Figure 7: The two BN model structures for the domain $\boldsymbol{X} = (X_1, X_2)$.

3.1 The BD Metric

A Bayesian approach for measuring the quality of a BN structure B_S, is its posterior probability given the database:

$$P(B_S|\mathcal{D}, \xi) = c \cdot P(B_S|\xi)P(\mathcal{D}|B_S, \xi),$$

where $c = 1/(\sum_B P(B|\xi)P(\mathcal{D}|B, \xi))$. The normalization constant c does not depend on B_S, thus $P(\mathcal{D}, B_S|\xi) = P(B_S|\xi)P(\mathcal{D}|B_S, \xi)$ is usually used as the network score. Note that the main computational problem is the calculation of the *marginal likelihood*:

$$P(\mathcal{D}|B_S, \xi) = \int_{\Theta_{B_S}} P(\mathcal{D}|B_S, \Theta_{B_S}, \xi)P(\Theta_{B_S}|B_S, \xi)d\Theta_{B_S}, \tag{1}$$

since the integral is over all possible parameters (conditional probabilities) Θ_{B_S} hence, over all possible BNs that encode at least the same conditional independence relations as the structure B_S.

Cooper and Herskovits (1991) showed that this probability can be computed in closed form based on the following five assumptions: 1) the database $\mathcal{D}$ is a multinomial sample from some Bayesian network B_G with parameters Θ_{B_G}, 2) the cases in the database $\mathcal{D}$ are independent given the BN model, 3) the database is complete, that is, there does not exist a case in $\mathcal{D}$ with missing values, 4) for any two configurations over the parents for a variable X_i, the parameters for the conditional probability distributions associated with X_i are marginally independent ($\Theta_{ij} \perp\!\!\!\perp \Theta_{ij'}$ for $j \neq j'$), and 5) the prior distribution of the parameters in every Bayesian network B_S has a Dirichlet distribution.[5] That is to say, there exist numbers (virtual counts) $N'_{ijk} > 0$ such that

$$P(\Theta_{ij}|B_S, \xi) = \frac{\Gamma(\sum_{k=1}^{r_i} N'_{ijk})}{\prod_{k=1}^{r_i} \Gamma(N'_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{N'_{ijk}-1}, \tag{2}$$

where Γ is the *Gamma* function satisfying $\Gamma(x+1) = x\Gamma(x)$. Note that the virtual counts can be seen as pseudo counts similar to the *sufficient statistics* derived from the database. An implicit assumption by Cooper and Herskovits (1991) is *parameter modularity*: The densities of the parameters Θ_{ij} depend only on the structure of the BN that is local to variable X_i.

Now, let N_{ijk} be the sufficient statistics given by $N_{ijk} = \sum_{l=1}^{N} \gamma(X_i = k, \Pi_i = j : \boldsymbol{D}_l)$, where $\gamma(X_i = k, \Pi_i = j : \boldsymbol{D}_l)$ takes on the value 1 if $(X_i = k, \Pi_i = j)$ occurs in case $\boldsymbol{D}_l$,

5. Cooper and Herskovits (1991) actually assume a uniform distribution, which is a special case of the Dirichlet distribution; the correctness of this generalization was proven by Cooper and Herskovits (1992).

and 0 otherwise. From assumption 1, 2 and 3 we then have

$$P(\mathcal{D}|B_S, \Theta_{B_S}, \xi) = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}}. \tag{3}$$

Substituting Equation 3 into Equation 1 gives

$$P(\mathcal{D}|B_S, \xi) = \int_{\Theta_{B_S}} \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} P(\Theta_{B_S}|B_S, \xi) d\Theta_{B_S},$$

and by assumptions 4 and 5 we get

$$\begin{aligned} P(\mathcal{D}|B_S, \xi) &= \prod_{i=1}^{n}\prod_{j=1}^{q_i} \int_{\Theta_{ij}} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} \left[\frac{\Gamma(\sum_{k=1}^{r_i} N'_{ijk})}{\prod_{k=1}^{r_i} \Gamma(N'_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{N'_{ijk}-1} \right] d\Theta_{ij} \\ &= \prod_{i=1}^{n}\prod_{j=1}^{q_i} \frac{\Gamma(\sum_{k=1}^{r_i} N'_{ijk})}{\prod_{k=1}^{r_i} \Gamma(N'_{ijk})} \int_{\Theta_{ij}} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}+N'_{ijk}-1} d\Theta_{ij}. \end{aligned}$$

The expression $\prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}+N'_{ijk}-1}$ corresponds to the last term of the Dirichlet distribution for the parameters Θ_{ij} having counts $N_{ijk} + N'_{ijk}$. Since this is a probability distribution over the parameters, the value of the integral can be read directly from Equation 2 (the integral over all parameters evaluates to 1) and we get

$$P(\mathcal{D}, B_S|\xi) = P(B_S|\xi) \prod_{i=1}^{n}\prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})}, \tag{4}$$

where $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ and $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$. This metric is known as the BD metric (Bayesian metric with Dirichlet priors), and it was first derived by Cooper and Herskovits (1992). Unfortunately it requires the specification of the virtual counts N'_{ijk} for all variable–parent configurations and for all values i, j and k.

3.2 The BDe Metric

One drawback of the BD metric is that networks, which are *likelihood equivalent*, need not be given the same score.[6] Note that data cannot be used to discriminate between such networks. Another shortcoming of the BD metric is that it does not provide an easy way of specifying prior information concerning network structure and parameters. To overcome these problems, Heckerman et al. (1995) describe the BDe metric (Bayesian metric with Dirichlet priors and equivalence) that gives the same score to likelihood equivalent networks. Hence, the metric is based on the concept of sets of likelihood equivalent network structures, where all members in a set are given the same score.

The BDe metric also provides a simple way of identifying the virtual counts N'_{ijk} (in Equation 4) by having the user specify a *prior Bayesian network* B_p for $\boldsymbol{X}$ and an *equivalent sample size* N':

$$N'_{ijk} = P(X_i = k, \Pi_i = j|B_p, \xi) \cdot N'. \tag{5}$$

6. Two networks are said to be likelihood equivalent if they encode the same assertions about conditional independence.

Note that Heckerman et al. (1995) actually condition on a complete network B_{S_c} consistent with B_p; conditioning on B_{S_c} allows Heckerman et al. (1995) to show that the Dirichlet assumption (Assumption 5) is not required. Finally, to evaluate Equation 4 we also need to define a prior probability $P(B_S|\xi)$ for the network structures. Different prior probabilities have been proposed in the literature, most of which obey the *structural modularity* assumption:

$$P(B_S|\xi) \propto \prod_{i=1}^{n} \rho(X_i, \Pi_i).$$

That is, the prior probability decomposes into a product with one term for each family in the network. From this assumption Equation 4 can be expressed as

$$P(\mathcal{D}, B_S|\xi) \propto \prod_{i=1}^{n} \rho(X_i, \Pi_i) \cdot \text{score}(X_i, \Pi_i, \mathcal{D}),$$

where

$$\text{score}(X_i, \Pi_i, \mathcal{D}) = \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})}.$$

Hence, when comparing two network structures we only need to consider the (local) scores and priors for the families for which they differ.

3.3 Learning from Incomplete Data

In real world problems we rarely have access to a complete database hence, assumption 3 of the BD metric (and the BDe metric) is likely to be violated. This implies that the parameters for a model become dependent, and known closed-form expressions cannot be used to calculate the marginal likelihood of the data. In such situations, a common approach is to apply asymptotic approximations such as the *Laplace approximation*, (see, for example, Ripley, 1996), the *Bayesian Information Criterion* (Schwarz, 1978), the *Minimum Description Length* (Rissanen, 1987) or the *Cheeseman-Stutz approximation* (Cheeseman and Stutz, 1996), see also Chichering and Heckerman (1997) for a discussion. These approximations assume that the posterior over the parameters is peaked, and the *maximum a posteriori* (MAP) parameters are used when approximating the integral in Equation 1. Thus, in order to apply these approximations we need to find the MAP parameters, for example by using the expectation-maximization (EM) algorithm (Dempster et al., 1977; Green, 1990), before we can calculate the score of a model. Thus, for each candidate model we may need to invest a considerable amount of time in order to evaluate the model.

As an alternative, Friedman (1998) describes the Structural EM (SEM) algorithm which basically “fills in” the missing values before searching the joint space of network structures and parameters (we therefore avoid the computational expensive step of calculating the MAP parameters for each candidate model). The validity of the SEM algorithm is based on the assumption that the data is *missing at random* (Little and Rubin, 1987), which is also assumed in the remainder of this paper. Informally, this means that the pattern of missingness may only depend on the values of the observed variables.[7]

The SEM algorithm maximizes $P(\mathcal{D}, B_S|\xi)$, but instead of maximizing this score directly it maximizes the expected score. Let $\boldsymbol{o}$ be the set of observations from the database $\mathcal{D}$, and let $\boldsymbol{h}$ be the set of unobserved entries in $\mathcal{D}$. The general algorithm can then be outlined as:

Algorithm 2 (SEM)

```
**Loop** *for* $n = 0, 1, \ldots$ *until convergence*
    1) *Compute the posterior* $P(\Theta_{B_S^n}|B_S^n, \boldsymbol{o})$.
    2) **E-step:** *For each* $B_S$, *compute:*
           $Q(B_S : B_S^n) = \mathbb{E}_{\boldsymbol{h}}[\log P(\boldsymbol{h}, \boldsymbol{o}, B_S)|B_S^n, \boldsymbol{o}]$
           $\qquad = \sum_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{o}, B_S^n) \log P(\boldsymbol{h}, \boldsymbol{o}, B_S)$.
    3) **M-step:** *Choose* $B_S^{n+1} \leftarrow B_S$ *that maximizes* $Q(B_S : B_S^n)$.
    4) **If** $Q(B_S^n : B_S^n) = Q(B_S^{n+1} : B_S^n)$ **then**
           **Return** $B_S^n$.
```

In the **E-step**, the algorithm completes the database by "filling-in" the unobserved entries based on the observations $\boldsymbol{o}$, the current best model B_S^n, and the posterior over the parameters for B_S^n (calculated in step 1). From the completed database the best candidate model is then selected in the **M-step**, which ensures that $Q(B_S^{l+1} : B_S^l) - Q(B_S^l : B_S^l) \geq 0$. Friedman (1998) proves that by increasing the expected score at each iteration we always obtain a better network in terms of its marginal score (this result also implies that the algorithm converges).

By exploiting linearity of expectation in the **E-step**, Friedman (1998) shows that the expected score decomposes as if the data were complete. That is, local changes to the model does not require that the entire model is reevaluated. In our context this yields (for notational convenience we assume that the structural prior, $\prod_{i=1}^n \rho(X_i, \Pi_i)$, is normalized):

$$\mathbb{E}_{\boldsymbol{h}}[\log P(\boldsymbol{h}, \boldsymbol{o}, B_S)|B_S^n, \boldsymbol{o}] = \sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{h}}[\log F_i(\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o}), B_S)|B_S^n, \boldsymbol{o}], \tag{6}$$

where $\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o})$ specifies the collection N_{ijk} according to $(\boldsymbol{h}, \boldsymbol{o})$, for all j and k, and $F_i(\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o}), B_S) = \rho(X_i, \Pi_i)\text{score}(X_i, \Pi_i, \boldsymbol{h}, \boldsymbol{o})$. Note that if $\prod_{i=1}^n \rho(X_i, \Pi_i)$ is not normalized we simply subtract $\log(c)$, where c is the normalization constant. That is to say, normalization of the prior distribution is not required. Friedman (1998) also examines an approximation for $\mathbb{E}_{\boldsymbol{h}}[\log F_i(\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o}), B_S)|B_S^n, \boldsymbol{o}]$:

$$\mathbb{E}_{\boldsymbol{h}}[\log F_i(\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o}), B_S)|B_S^n, \boldsymbol{o}] \approx \log F_i(\mathbb{E}_{\boldsymbol{h}}[\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o})|B_S^n, \boldsymbol{o}], B_S). \tag{7}$$

7. An active research area within the learning community is the discovery of *hidden variables*. These types of variables are never observed (Spirtes et al., 1993; Friedman et al., 1998; Elidan et al., 2000; Elidan and Friedman, 2001), however, hidden variables will not be considered further in this paper.

The approximation is exact if $\log F_i$ is linear in its arguments. This is, however, not the case when using the BD or BDe metric.[8] Finally, the term $\mathbb{E}_{\boldsymbol{h}}[\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h},\boldsymbol{o})|B^n_S,\boldsymbol{o}]$ can be computed as

$$\forall j,k : \mathbb{E}_{\boldsymbol{h}}[N_{ijk}(\boldsymbol{h},\boldsymbol{o})|B^n_S,\boldsymbol{o}] = \sum_{l=1}^{N} P(X_i = k, \Pi_i = j|\boldsymbol{D}_l, B^n_S).$$

3.4 Learning Dynamic Bayesian Networks

Friedman et al. (1998) describe an algorithm for learning DBNs from both complete and incomplete data. The methods proposed by Friedman et al. (1998) extend both the Bayesian Information Criterion (BIC) and the BDe score for learning DBNs from complete data. When lifting the assumption that the database is complete, Friedman et al. (1998) extend the SEM algorithm accordingly.

Friedman et al. (1998) define a DBN by partitioning the variables into time-slices s.t. the variables which occur at time t are denoted $\boldsymbol{X}[t]$. Thus, a DBN with l time-slices consists of the variables $\boldsymbol{X}[0] \cup \boldsymbol{X}[1] \cup \cdots \cup \boldsymbol{X}[l]$. It is assumed that the DBN is *Markovian* that is, $P(\boldsymbol{X}[t+1]|\boldsymbol{X}[0],\ldots,\boldsymbol{X}[t]) = P(\boldsymbol{X}[t+1]|\boldsymbol{X}[t])$. By also assuming that the DBN is *stationary* (the CPTs associated with the variables in $\boldsymbol{X}[t]$ are independent of t, for $t > 0$), a DBN can be completely described by two parts: *i*) An initial network, B^0, that specifies a distribution over $\boldsymbol{X}[0]$ and *ii*) a transition network, $B^{\rightarrow}$, over the variables $\boldsymbol{X}[0] \cup \boldsymbol{X}[1]$.

In the context of DBNs, the database is assumed to consist of N cases, where the m'th case specifies a configuration over the variables $\boldsymbol{X}[0] \cup \boldsymbol{X}[1] \cup \cdots \cup \boldsymbol{X}[l]$. Now, consider the situation where the database is complete and let $\theta^0_{ij'k}$ and $\theta^{\rightarrow}_{ijk}$ be defined as in Section 3.1 for B^0 and $B^{\rightarrow}$, respectively; we use j' and j to indicate that the parents for X_i may be different in B^0 and $B^{\rightarrow}$. Additionally, let the sufficient statistics be given by $N^0_{ij'k} = \sum_{m=1}^{N} \gamma(X_i[0] = k, \Pi_i = j' : \boldsymbol{D}_m)$ and $N^{\rightarrow}_{ijk} = \sum_{t=1}^{l} \sum_{m=1}^{N} \gamma(X_i[t] = k, \Pi_i = j : \boldsymbol{D}_m)$. By derivations similar to those of the BD metric, the following closed form expression for $P(\mathcal{D},(B^0,B^{\rightarrow})|\xi)$ is obtained:

$$\begin{aligned} P(\mathcal{D},(B^0,B^{\rightarrow})|\xi) &= P((B^0,B^{\rightarrow})|\xi) \\ &\cdot \left(\prod_{i=1}^{n} \prod_{j'=1}^{q'_i} \frac{\Gamma(N'^0_{ij'})}{\Gamma(N^0_{ij'} + N'^0_{ij'})} \prod_{k=1}^{r_i} \frac{\Gamma(N^0_{ij'k} + N'^0_{ij'k})}{\Gamma(N'^0_{ij'k})} \right) \\ &\cdot \left(\prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'^{\rightarrow}_{ij})}{\Gamma(N^{\rightarrow}_{ij} + N'^{\rightarrow}_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N^{\rightarrow}_{ijk} + N'^{\rightarrow}_{ijk})}{\Gamma(N'^{\rightarrow}_{ijk})} \right). \end{aligned}$$

Note that when maximizing this expression we can consider the terms independently assuming that $P(B^0, B^{\rightarrow}|\xi) = P(B^0|\xi) \cdot P(B^{\rightarrow}|\xi)$.

Friedman et al. (1998) overcome the problem of specifying the virtual counts for the candidate network structures by advocating the method of Heckerman et al. (1995). That is, given a prior DBN $B_p = (B^0_p, B^{\rightarrow}_p)$ and two equivalent sample sizes for B^0_p and $B^{\rightarrow}_p$, the virtual counts are found as in Equation 5.

8. Friedman (1998) shows that the error of the linear approximation vanishes as the size of the database approaches infinity.

4. Specifying Prior Information

When learning a Bayesian network, the prior information about the domain is represented by *i*) a prior distribution over the discrete space of all candidate structures, and *ii*) a prior distribution over the continuous space of probability parameters for each model. In Section 3.2 we briefly described a prior for the probability parameters, and in this section we will focus on the use of prior information regarding the structure of BNs and OOBNs.

4.1 Structural Priors in BNs

The use of structural priors when learning BNs has received only little attention in the learning community. The most obvious reason is that in most cases the effect of the prior is dominated by the likelihood term, even for relatively small databases. One exception, however, is when some of the network structures are given zero probability a priori, in which case the data cannot change that belief.

Common to most (if not all) structural priors proposed in the literature is that they obey the structural modularity assumption (see Section 3.2):

$$P(B_S \,|\, \xi) \propto \prod_{i=1}^{n} \rho(X_i, \Pi_i) \ .$$

That is, the prior decomposes into a product with one term for each family in the network structure. This assumption ensures that during structure search (given complete data – or data "completed" by the SEM algorithm) we can compare two candidate structures by only considering the local scores and priors for the families for which they differ.

Because of their relatively small influence upon the selected model, structural priors are most often used to encode ignorance, and in some cases to restrict model complexity. Examples include the uniform prior $\rho(X_i, \Pi_i) = 1$ (Cooper and Herskovits, 1991), and

$$\rho(X_i, \Pi_i) = \begin{pmatrix} n-1 \\ |\,\Pi_i\,| \end{pmatrix}^{-1}$$

used by Friedman and Koller (2003). Another prior which is frequently used is $\rho(X_i, \Pi_i) = \kappa^{\delta_i}$ (Heckerman et al., 1995), where $0 < \kappa \leq 1$ and

$$\delta_i = |\,\{\Pi_i(B_S) \cup \Pi_i(B_p)\} \setminus \{\Pi_i(B_S) \cap \Pi_i(B_p)\}\,|$$

denotes the number of parents for X_i that differs in the prior model B_p and the candidate structure B_S. Thus, each such parent is penalized by a constant κ. The flexibility of this prior can easily be extended by setting

$$\delta_i = \sum_{j \neq i} (\omega_{ij}^{+}\, \delta_{ij}^{+} + \omega_{ij}^{-}\, \delta_{ij}^{-}) \ , \tag{8}$$

where δ_{ij}^{+} is 1 if there is an edge from X_j to X_i in the candidate structure but not in the prior model, and 0 otherwise; δ_{ij}^{-} is 1 if there is an edge from X_j to X_i in the prior model, but not in B_S, and 0 otherwise. $(\omega_{ij}^{+}, \omega_{ij}^{-}) \in \mathbb{R}^{+} \times \mathbb{R}^{+}$ is a pair of weights that indicates how certain the domain expert is about the occurrence/absence of a specific edge: Complete ignorance is encoded by $\omega^{+}_{ij} = 0$, whereas certainty is encoded by $\omega^{+}_{ij} = \infty$, and similarly for ω^{-}_{ij}. When $\omega^{+}_{ij} = \omega^{-}_{ij} = 1$, $\forall\, i,\, j$, the prior reduces to that of Heckerman et al. (1995). Note that since both the prior model as well as each candidate model are restricted to be directed acyclic graphs it is not possible to give these weights a straightforward probabilistic interpretation; the occurrence of one edge is in general dependent on the occurrence of the other edges in the network structure. Finally, we note that this prior has a potential drawback since it in principle requires the elicitation of the $2n \cdot (n-1)$ weights $\omega^{(\cdot)}_{ij}$, where n is the number of variables in the domain. In practical usage, however, one can use an elicitation scheme where these weights are grouped according to the values 0, 1 or ζ (where $\zeta \gg 0$ is used to model almost certainty), see below.

4.2 Structural Priors in OOBNs

In this section we consider the additional sources of prior information available when learning in object oriented domains. We will argue that the OOBN framework is a natural language for specifying prior information. As we shall see, the underlying object oriented modelling assumptions naturally lead to zero prior probabilities for large parts of the model space.

4.2.1 The OO assumption

Langseth and Bangsø (2001) claim that for OOBN learning to be meaningful one should assume that the domain is in fact object oriented (such that the OO assumption is fulfilled). As an example, consider the special case of learning DBNs. In this situation the OO assumption states that the CPT associated with a variable $X_i[t_k]$ ($t_k > 0$) is identical to the CPT associated with any other variable $X_i[t_\ell]$ ($t_\ell > 0$). That is, the CPTs associated with the variables in $\boldsymbol{X}[t]$ are independent of t for $t > 0$. Hence, when learning DBNs, the OO assumption corresponds to the assumption that the domain is stationary, which is for instance assumed by Friedman et al. (1998). If the DBN is not stationary, one cannot define the evolving model $\boldsymbol{X}[t]$ ($t > 0$) as identical instantiations of a class, and according to Langseth and Bangsø (2001) it is not necessarily reasonable to use an object oriented domain specification in this case.

Note that the effect of making the OO assumption is that all models that violate this assumption are given zero probability a priori. Note also that the OO assumption cannot be modelled using a conventional BN as a prior model, if this model should obey structural modularity; the structural part of the OO assumption is not local to one family in the graph.

4.2.2 Relations among variables

When modelling object oriented domains, the domain expert is usually able to group the variables into substructures with high internal coupling and low external coupling. These substructures naturally correspond to instantiations in an OOBN. Moreover, analogously to the grouping of similar substructures into categories, instantiations of the same type are grouped into classes (Mahoney and Laskey, 1996; Mathiasen et al., 2000). For instance, a set of substructures may correspond to the same type of physical object or they may describe a set of entities that occur at the same instant of time.

Such types of prior information can be represented by a (partial) OOBN specification (that is, a prior model). The a priori specification of an OOBN contains a list of class specifications and a grouping of the nodes into instantiations that are classified according to the classes. This prior OOBN model can then be used as in the case of conventional prior models, and we can in principle use any of the definitions of $\rho(X_i, \Pi_i)$ outlined above.

When specifying the relations among the variables, it may be difficult for the domain expert to indicate the presence or absence of edges between *specific* nodes in the model. If, for example, two variables X and Y in an instantiation I are strongly correlated, the domain expert may be uncertain whether another node Z in the encapsulating context of I should be the parent of X or Y (even though he believes that Z should influence at least one of them). In the OOBN framework, this prior information can be encoded by specifying the interface between the instantiation I and its encapsulating context. For instance, the domain expert can indicate which instantiations are allowed (and more importantly, denied) to reference a particular node (see Figure 8). Specifically, the domain expert could be asked questions like *"Do you think it is possible that a variable Z directly influences any of the variables in instantiation* I*?"*

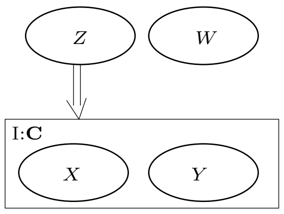


Figure 8: The figure depicts a possible way to describe knowledge about the structure of the domain. It shows an instantiation I and some of its encapsulating context (note that this is not strictly speaking an OOBN).

The use of such prior models is also supported by Equation 8, since edges that are not considered possible a priori are penalized strongly ($\omega^{+}_{ij} = \zeta \gg 0$). On the other hand, the interface of an instantiation defines edges from a single node to a group of nodes hence, missing reference links cannot be penalized (as the prior specification at the class level should obey structural modularity), and we therefore use $\omega^{-}_{ij} = 0$. As an example, see Figure 8, where we assume that the instantiation I consists of the two nodes X and Y, and that (a priori) only Z is regarded as a possible node to be referenced from I. From the discussion above, it follows that a candidate network where no node is referenced from I will not be penalized by this prior, because $\omega^{-}_{XZ} = \omega^{-}_{YZ} = 0$. If we were to use a prior that penalizes the "missing" link between Z and the instantiation I, then this prior would have to encode that the probability for a link between Z and X *depends on* the existence of a link between Z and Y (the prior only penalizes a link missing between Z and X if there is no link from Z to Y). This violates structural modularity, which says that the prior should factorize into a product of terms, where each term only depends on one family in the graph, see Section 3.2. On the other hand, if a candidate model is designed so that another node, say W, is referenced from I, it will be given a lower a priori belief (because $\omega^+_{XW} = \omega^+_{YW} = \zeta$). Note that the OOBN framework is not required to model this vague prior information; it is merely a straight forward usage of Equation 8. However, to elicit such information it turns out to be useful to have grouped the nodes into what corresponds to instantiations, and then focus on the interfaces of these (working in the framework of OOBNs).

To verify the ease of determining the interfaces a priori we conducted an experiment amongst our co-workers: The task was to identify the interfaces of the instantiations in the object oriented version of the insurance domain, see Section 2.2. The test-persons were familiar with the OOBN framework, but they had not seen the insurance network before. Initially they were given the object oriented version of the insurance network, where each node was allocated to one of the instantiations (with all edges removed). The task was then to identify the interface of all instantiations in the domain, simply by indicating which nodes inside an instantiation I_i could (possibly) be referenced from an instantiation I_j. The test-persons had no other information about the domain, except for what they were able to deduce from the names of the nodes. They where guided through the knowledge acquisition by questions of the type *"Is it possible that a person's Age can directly influence any of the nodes in the instantiation of the* **Driver***-class (RiskAversion, SeniorTrain, DrivingSkill, DrivQuality or DrivHist)?"* The result of the experiment was that out of the 702 edges that can be included in the model, only 253 were marked possible. All the 52 edges actually in the model were considered legal. The elicitation of this information took about 10 minutes; this result at least suggests that the approach is promising.

5. Learning in OOBNs

In this section we describe a method for learning in object oriented domains, casted as the problem of finding the maximum a posteriori OOBN structure given a database $\mathcal{D}$.

The basic idea of the object oriented learning method resembles that of Langseth and Bangsø (2001) who exploit the OO assumption when learning the parameters in an OOBN. Specifically, based on this assumption, Langseth and Bangsø (2001) propose to learn at the class level of the OOBN instead of in the underlying BN; cases from the instantiations of a class are considered (virtual) cases of that class.[9] Langseth and Bangsø (2001) give both theoretical as well as empirical evidence that this learning method is superior to conventional parameter learning in object oriented domains.

5.1 Structural OO learning

The goal of our learning algorithm is to find a good estimate of the unknown underlying statistical distribution function. That is, we focus on the task of *density estimation* (Silverman, 1986). Note that if focus had been on, for example, causal discovery (Heckerman, 1995a), classification (Friedman et al., 1997a), or generating a model that was able to predict well

9. Note that this approach can be seen as a generalization of the method for parameter learning in DBNs, see West and Harrison (1997).

according to a predefined query distribution (Greiner et al., 1997), the learning method would have been slightly different (the general approach, however, would still apply).

The proposed method is tightly connected to the SEM-algorithm, described in Section 3.3. The main differences concern structure search and the calculation of the expected score of a network structure. When doing structure search we restrict the space of candidate structures by employing the search operations in the class specifications instead of in the underlying BN. This has the advantages that *i*) the current best model is always guaranteed to be an OOBN, and *ii*) the learning procedure will in general require fewer steps than conventional learning because the search space is smaller.

The difference in the calculation of the expected score of an OOBN structure compared to a BN structure is a consequence of the OO assumption: Since we assume all instantiations of a given class to be identical, we treat cases from the instantiations of a given class as (virtual) cases of that class. Note that this approach can be seen as a generalization of the learning method for DBNs, described in Section 3.4, where all cases from the time-slices for $t > 0$ are used for calculating the sufficient statistics for the transition network. Before giving a formal definition of the expected score of an OOBN structure we introduce the following notation (for now we shall assume that all input sets are empty): Let $B_{\mathbf{C}_m}$ be an OOBN for class $\mathbf{C}_m$, and let $\{i : X_i \in \mathbf{C}_\ell\}$ be the set of nodes defined in class $\mathbf{C}_\ell$. Let $\mathcal{I}$ define the set of instantiations, let $\mathbf{T}(\mathrm{I})$ be the class of instantiation $\mathrm{I} \in \mathcal{I}$, and let $\{\mathrm{I} : \mathbf{T}(\mathrm{I}) = \mathbf{C}_\ell\}$ be the set of instantiations of class $\mathbf{C}_\ell$ (recall that we use $\mathrm{I}.X$ to denote node X in instantiation I).

The sufficient statistics $N_{ijk}^{\mathbf{C}_\ell}$ for a class $\mathbf{C}_\ell$, given a complete database, is then given by

$$N_{ijk}^{\mathbf{C}_\ell} = \sum_{\mathrm{I}:\mathbf{T}(\mathrm{I})=\mathbf{C}_\ell} \sum_{t=1}^{N} \gamma(\mathrm{I}.X_i = k, \mathrm{I}.\Pi_i = j : \boldsymbol{D}_t). \tag{9}$$

Based on the sufficient statistics for a class we can, under assumptions similar to those of Cooper and Herskovits (1991), derive the score for a node X_i in class $\mathbf{C}_\ell$ as

$$\text{O-score}(X_i, \Pi_i, \boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\mathcal{D}), \mathbf{C}_\ell) = \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij}^{\mathbf{C}_\ell} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk}^{\mathbf{C}_\ell} + N'_{ijk})}{\Gamma(N'_{ijk})}, \tag{10}$$

where $\boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\mathcal{D})$ specifies the collection $N_{ijk}^{\mathbf{C}_\ell}$ according to $\mathcal{D}$, and $N_{ij}^{\mathbf{C}_\ell} = \sum_{k=1}^{r_i} N_{ijk}^{\mathbf{C}_\ell}$.

Finally, we can define the BDe score for an OOBN B_S as

$$P(\mathcal{D}, B_S \mid \xi) \propto \prod_{\mathbf{C}_\ell \in \mathcal{C}} \prod_{i:X_i \in \mathbf{C}_\ell} \rho(X_i, \Pi_i, \mathbf{C}_\ell) \cdot \text{O-score}(X_i, \Pi_i, \boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\mathcal{D}), \mathbf{C}_\ell)\ , \tag{11}$$

where $\mathcal{C}$ is the set of all classes, and $\rho(X_i, \Pi_i, \mathbf{C}_\ell)$ is a function of the prior specification of $\mathbf{C}_\ell$, such that

$$P(B_S|\xi) \propto \prod_{\mathbf{C}_\ell \in \mathcal{C}} \prod_{i:X_i \in \mathbf{C}_\ell} \rho(X_i, \Pi_i, \mathbf{C}_\ell).$$

In the situation with missing data we apply a modified version of the SEM algorithm. Recall that the SEM algorithm requires the calculation of

$$Q(B_S : B_S^n) = \mathbb{E}_{\boldsymbol{h}}[\log P(\boldsymbol{o}, \boldsymbol{h}, B_S) \mid B_S^n, \boldsymbol{o}],$$

where $\boldsymbol{o}$ and $\boldsymbol{h}$ denote the observed and unobserved entries in $\mathcal{D}$, respectively, and B_S^n is the current best model. In accordance with Equation 6 and Equation 11 we have (again we assume that the prior distribution is normalized)

$$\mathbb{E}_{\boldsymbol{h}}[\log P(\boldsymbol{o},\boldsymbol{h},B_S)\,|\,B_S^n,\boldsymbol{o}] = \sum_{\mathbf{C}_\ell\in\mathcal{C}}\sum_{i:X_i\in\mathbf{C}_\ell}\mathbb{E}_{\boldsymbol{h}}[\log F_{i,\mathbf{C}_\ell}(\boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\boldsymbol{h},\boldsymbol{o}),B_S)|B_S^n,\boldsymbol{o}] \tag{12}$$

where

$$F_{i,\mathbf{C}_\ell}(\boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\boldsymbol{h},\boldsymbol{o}),B_S) = \rho(X_i,\Pi_i,\mathbf{C}_\ell)\cdot\text{O-score}(X_i,\Pi_i,\boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\boldsymbol{h},\boldsymbol{o}),\mathbf{C}_\ell).$$

Now, analogously to the SEM algorithm we advocate the approximation proposed in Equation 7 hence, for an OOBN we approximate

$$\mathbb{E}_{\boldsymbol{h}}[\log F_{i,\mathbf{C}_\ell}(\boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\boldsymbol{h},\boldsymbol{o}),B_S)|B_S^n,\boldsymbol{o}] \approx \log F_{i,\mathbf{C}_\ell}(\mathbb{E}_{\boldsymbol{h}}[\boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\boldsymbol{h},\boldsymbol{o})|B_S^n,\boldsymbol{o}],B_S).$$

Finally, the *expected counts* $\mathbb{E}_{\boldsymbol{h}}[\boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\boldsymbol{h},\boldsymbol{o})|B_S^n,\boldsymbol{o}]$ for node X_i in class $\mathbf{C}_\ell$ is given by

$$\forall j,k: \mathbb{E}_{\boldsymbol{h}}[N_{ijk}^{\mathbf{C}_\ell}(\boldsymbol{h},\boldsymbol{o})|B_S^n,\boldsymbol{o}] = \sum_{\mathrm{I}:\mathbf{T}(\mathrm{I})=\mathbf{C}_\ell}\sum_{t=1}^{N} P(\mathrm{I}.X_i=k,\,\mathrm{I}.\Pi_i=j\,|\,\boldsymbol{D}_t,B_S^n).$$

Now, both $Q(B_S:B_S^n)$ and the posterior $P(\mathcal{D},B_S\,|\,\xi)$ factorize over the variables (and therefore also over the classes). Hence, in order to compare two candidate structures which only differ w.r.t. the edge $X_i\rightarrow X_j$ we only need to re-calculate the score (Equation 10) and $\rho(X_j,\Pi_j,\mathbf{C}_\ell)$ for node X_j in the class $\mathbf{C}_\ell$ where X_j is defined. Note that this property also supports the proposed search procedure which is employed at the class level.

Unfortunately, this type of locality to a class is violated when the input sets are non-empty (this is for instance the case with the two instantiations of the class **Milk Cow** that are embedded in the **Stock** class). The problem occurs when new input nodes are added to a class interface, since the search for a "good" set of parents is not necessarily local to a class when the interface is not given. Recall that the actual nodes included through the interface of an instantiation is not defined in the class specification, but locally in each instantiation. This may result in a serious computational overhead when determining the interface since we require that the OO assumption is satisfied. As an example, assume that the node X in instantiation I_i is assigned an input node Y' as parent, and assume that Y' references the node Y. Then, due to the OO assumption, the algorithm should find a node Z that has the same influence on $\mathrm{I}_j.X$ as Y has on $\mathrm{I}_i.X$, for all instantiations I_j where $\mathbf{T}(\mathrm{I}_j)=\mathbf{T}(\mathrm{I}_i)$. The search for Z must cover all nodes in the encapsulating context of I_j. Note that Z may be non-existent in which case the default potential for the input node should be used. The complexity of finding the best candidate interface for all instantiations is exponential in the number of instantiations, and we risk using a non-negligible amount of time to evaluate network structures with low score. For example if Y' (or more precisely the node Y referenced by Y') is actually not needed as a parent for $\mathrm{I}_i.X$.

To overcome this computational difficulty we propose the following algorithm, which is inspired by the SEM algorithm (Algorithm 2). Basically, the algorithm iteratively learns *a*) the interfaces of the instantiations by keeping the structure inside the instantiations fixed according to the classes (Step *i* and *ii*), and *b*) learns the structure inside each class based on the candidate interfaces found in the previous steps (Step *iii*). Observe that Step 3 corresponds to the **E-step** in the SEM algorithm and that Step 4 corresponds to the **M-step**.

Algorithm 3 (OO–SEM)

```
a) Let $B_S^0$ be the prior OOBN model.
b) Loop for $n = 0, 1, \ldots$ until convergence
    1) Compute the posterior $P(\Theta_{B_S^n} | B_S^n, \boldsymbol{o})$, see Langseth and Bangsø (2001) and Green (1990).
    2) Set $B_S^{n,0} \leftarrow B_S^n$.
    3) For $i = 0, 1, \ldots$
        i) Let $B_S$ be the model which is obtained from $B_S^{n,i}$ by employing either none or exactly one of the operations add-edge and remove-edge, for each instantiation $\mathrm{I}$; each edge involved must have a node in both $\mathrm{I}$ and in the encapsulating context of $\mathrm{I}$ (directed into $\mathrm{I}$). The OO assumption is disregarded.[10]
        ii) For each node $X$, which is a child of an input node $Y'$ (found in step (i)) in instantiation $\mathrm{I}_j$, determine if $\mathrm{I}_k.X$ has an input node as parent with the same state space as $Y'$, for all $k \neq j$ where $\mathbf{T}(\mathrm{I}_k) = \mathbf{T}(\mathrm{I}_j)$. If this is the case, use the BDe score to determine if they should be assigned the same CPT (due to the OO assumption); otherwise introduce default potentials to ensure that they have the same CPTs.[11] Let $B'_S$ be the resulting network.
        iii) For each class $\mathbf{C}_\ell$ in $B'_S$ employ the operations add-edge or remove-edge w.r.t. the nodes in the class (excluding the input set) based on the candidate interface found in step (ii). Note that edges from instantiations encapsulated in $\mathbf{C}_\ell$ into nodes defined in $\mathbf{C}_\ell$ are also considered in this step.[12] Let $B''_S$ be the resulting OOBN.
        iv) Set $B_S^{n,i+1} \leftarrow B''_S$.
    4) Choose $B_S^{n+1} \leftarrow B_S^{n,i}$ that maximizes $Q(B_S^{n,i} : B_S^n)$ (Equation 12).
    5) If $Q(B_S^n : B_S^n) = Q(B_S^{n+1} : B_S^n)$ then
           Return $B_S^n$.
```

Note that in Step (*ii*) it may seem counterintuitive to compare CPTs using the BDe score, however, observe that this step is actually used to restrict the parameter space and the BDe score is therefore appropriate, cf. the discussion in Section 3. Moreover, it should

10. The number of operations is bounded by the product of the number of nodes in I and the number of nodes in the encapsulating context, but only the terms local to the involved families need to be re-calculated.
11. The CPTs are estimated by setting $\widehat{\theta}_{ijk}^{\mathbf{C}_\ell} = \left(N_{ijk}^{\mathbf{C}_\ell} + N'_{ijk}\right) / \left(N_{ij}^{\mathbf{C}_\ell} + N'_{ij}\right)$, where $N_{ijk}^{\mathbf{C}_\ell}$ is the expected sufficient statistics calculated according to Equation 9. Note that introducing default potentials have no effect on the underlying BN (they can just be marginalized out).
12. An example of this situation is illustrated in Figure 6, where an instantiation of **Driver** is encapsulated in the class **CarOwner**. Observe that only the terms local to the involved families need to be re-calculated.

be noticed that we use strong type-checking in Step (*ii*), which ensures that the algorithm generates a legal OOBN as defined by Bangsø and Wuillemin (2000b). If we should learn an OOBN consistent with the definition of for example Koller and Pfeffer (1997), then stronger restrictions on the input sets would apply.

In case of a complete database, the outer loop is simply evaluated once; evaluating the network structures using $Q(B_S : B_S^n)$ is identical to using the BDe score for OOBNs in this case.

Theorem 1 *Let $\mathcal{D}$ be a complete database of size N generated by an OOBN model with structure B_S^*. If $N \to \infty$, then the structure B_S returned by Algorithm 3 is likelihood equivalent to B_S^*.*

Proof Notice that the space of OOBN structures is finite, and that each OOBN structure can be visited by the inner loop of Algorithm 3. Note also that the greedy approach in step (*ii*) is asymptotically correct as the associated search space is uni-modal (as $N \to \infty$) and the operations are transitive. From these observations the proof is straightforward as the BDe score is asymptotically correct, see Heckerman (1995b) and Geiger et al. (1996). ■

Notice that the theorem above only holds when the database is complete. When the database is incomplete we have the following corollary.

Corollary 2 *Let $B_S^0, B_S^1, \ldots$ be the sequence of structures investigated by Algorithm 3, and let $\mathcal{D}$ be a database. Then $\lim_{n\to\infty} P(\boldsymbol{o}, B_S^n)$ exists, and it is a local maximum of $P(\boldsymbol{o}, B_S)$ when regarded as a function of B_S.*

Proof Follows immediately from Friedman (1998, Theorem 3.1 and Theorem 3.2) by observing that *a*) the space of OOBN structures is finite and the variables in the domain have discrete state spaces, and *b*) in Steps $(i - iii)$ we are always sure to increase the expected score of the candidate model. ■

Observe that in order to complete the operational specification of Algorithm 3, we need a search algorithm, for instance simulated annealing, for investigating the candidate structures (Step (*i*) and Step (*iii*) constitute the choice points). Note also that in order to maximize the score in Step (*ii*) we would in principle need to investigate the set of all subsets of instantiations and nodes (which have an input node as parent). To avoid this computational problem we instead consider the instantiations and nodes pairwise (randomly chosen). This still ensures that the expected score increases in each iteration hence, the algorithm will converge even though we apply hill-climbing in Step (*ii*), see also Corollary 2.

Finally it should be emphasized that the main computational problem of Algorithm 3 is in establishing the interfaces of the instantiations hence, we propose to elicit prior information based on specific enquiries about the interfaces. For instance, the domain expert can be asked to specify the nodes each instantiation is allowed to reference; as argued in Section 4.2 this is easily elicitated in an object oriented domain. Prior information of the form "*Daisy* is to Cow1 as *Mathilda* is to Cow3", related to the type-construct of Koller and Pfeffer (1997), can be exploited.

5.2 Type Uncertainty

So far we have assumed that the domain expert is able to unambiguously classify each instantiation to a specific class. Unfortunately, however, this may not be realistic in real-world applications. Not being able to classify an instantiation is an example of what is called *type uncertainty* by Pfeffer (2000); the expert is uncertain about the type (or class in our terminology) of an instantiation. However, even though we may be unable to determine whether, for instance, Cow1 is a **Milk cow** or a **Meat cow**, see Section 2, we would still like to employ the learning algorithm using all available data.

When subject to type uncertainty the main problem is as follows. Consider the situation where we have two instantiations I_i and I_j whose classes are uncertain. Assume that both I_i and I_j are a priori classified as being instantiations of $\mathbf{C}_k$, and assume that the data from I_i and I_j are somewhat different. If the data from I_i is initially used for learning in $\mathbf{C}_k$, then the class specification for $\mathbf{C}_k$ is updated and the probability of I_j being an instantiation of $\mathbf{C}_k$ may therefore change. Thus, the probability of I_j belonging to $\mathbf{C}_k$ is dependent on the classification of I_i. An immediate approach to overcome this problem is brute force, where we consider all possible combinations of allocating the uncertain instantiations to the classes. However, this method is computationally very hard, and is not suited for practical purposes if the number of combinations of instantiations and classes is large. The complexity is $O\left(|\mathcal{C}|^{|\mathcal{I}|}\right)$.

In what follows we propose an alternative algorithm for handling type uncertainty. We shall assume that the domain expert encodes his prior beliefs about the classification of the instantiations $\mathcal{I}$ as a distribution over the classes $\mathcal{C}$ (this also allows us to restrict our search in the class tree to specific subtrees, if the domain expert encodes his prior belief in that way). Recall that the main problem with type uncertainty is that learning can only be performed locally in a class specification if all instantiations are allocated to a class (with certainty). This observation forms the basis for the following algorithm, which iteratively classifies the instantiations based on the MAP distribution over the classifications of the instantiations. Note that since the learned model is dependent on the classification of the uncertain instantiations, the algorithm maximizes the joint probability $P(\mathcal{D}, B_S(\mathcal{T}), \mathcal{T})$, where $\mathcal{T} = \mathbf{T}(\mathcal{I})$; we use the notation $B_S(\mathcal{T})$ to indicate that the learned model is a function of the classifications. This probability can be computed as $P(\mathcal{D}, B_S(\mathcal{T}), \mathcal{T}) = P\left(\mathcal{D}|B_S(\mathcal{T}), \mathcal{T}\right) P\left(B_S(\mathcal{T})\,|\,\mathcal{T}\right) P\left(\mathcal{T}\right)$ where $B_S(\mathcal{T})$ is a model consistent with the classification $\mathcal{T}$. In the following we will let $\widehat{\mathcal{T}}$ denote the current estimate of the classification $\mathbf{T}(\mathcal{I})$. Furthermore, we use $\widehat{\mathcal{T}}_{\mathrm{I}} \leftarrow \mathbf{C}_\ell$ to denote that the estimate of $\mathbf{T}(\mathrm{I})$ is set to $\mathbf{C}_\ell$, and we use $\widehat{\mathcal{T}}_{-\mathrm{I}}$ to denote the estimate of $\mathbf{T}(\mathcal{I} \setminus \{\mathrm{I}\})$.

Algorithm 4 (Type Uncertainty)

```
a) Initialization: Find the classification with maximum probability according to the
   prior distribution over the classifications $P(\mathbf{T}(\mathcal{I}))$, and classify the instantiations
   accordingly. Let $\widehat{\mathcal{T}}^0$ be this initial classification.

b) Loop for $n = 0, 1, \ldots$ until convergence

      1) $\widehat{\mathcal{T}}' \leftarrow \widehat{\mathcal{T}}^n$.
```

```
2) For each uncertain instantiation I:
    i) For each classification C of I s.t. $P(\mathbf{T}(\mathrm{I}) = \mathbf{C}) > 0$:
        A) Classify I as an instantiation of class C: $\widehat{\mathcal{T}}'_{\mathrm{I}} \leftarrow \mathbf{C}$.
        B) Learn the OOBN $B'_S(\widehat{\mathcal{T}}')$ for the current classification of all instantiations (Algorithm 3).[13] Calculate the joint probability of the data, the model $B'_S(\widehat{\mathcal{T}}')$ and $\widehat{\mathcal{T}}'$:
               $f(\mathbf{C}) \leftarrow P\left(\mathcal{D}, B'_S(\widehat{\mathcal{T}}'), \widehat{\mathcal{T}}'\right)$.
    ii) Classify I to the class maximizing the joint probability
        $P(\mathcal{D}, B'_S(\widehat{\mathcal{T}}'), \widehat{\mathcal{T}}')$ by keeping the classifications $\mathbf{T}(\mathcal{I} \setminus \{\mathrm{I}\})$ fixed:
               $\widehat{\mathcal{T}}'_{\mathrm{I}} \leftarrow \arg\max_{\mathbf{C}:P(\mathbf{T}(\mathrm{I})=\mathbf{C})>0} f(\mathbf{C})$.
3) Let $\widehat{\mathcal{T}}^{n+1} \leftarrow \widehat{\mathcal{T}}'$ and let $B_S^{n+1}$ be the model found according to the classification $\widehat{\mathcal{T}}^{n+1}$.
4) If $P\left(\mathcal{D}, B_S^{n+1}(\widehat{\mathcal{T}}^{n+1}), \widehat{\mathcal{T}}^{n+1}\right) = P\left(\mathcal{D}, B_S^{n}(\widehat{\mathcal{T}}^{n}), \widehat{\mathcal{T}}^{n}\right)$ then
       Return $B_S^n(\widehat{\mathcal{T}}^n)$.
```

The algorithm attempts to maximize the joint probability $P(\mathcal{D}, B_S(\mathcal{T}), \mathcal{T})$ by iteratively maximizing 1) $P(\mathcal{D}, B_S(\widehat{\mathcal{T}}^n), \widehat{\mathcal{T}}^n)$ over the models B_S with the current classification $\widehat{\mathcal{T}}^n$ (Step B), and 2) $P(\mathcal{D}, B_S^n(\widehat{\mathcal{T}}^n_{-\mathrm{I}}, \mathbf{T}(\mathrm{I})), (\widehat{\mathcal{T}}^n_{-\mathrm{I}}, \mathbf{T}(\mathrm{I})))$ over $\mathbf{T}(\mathrm{I})$ given the classification $\widehat{\mathcal{T}}^n_{-\mathrm{I}}$ (Step *ii*). This also implies that the algorithm converges to a (local) maximum.

6. Empirical Study

In this section we describe a set of empirical tests that have been conducted to verify the proposed learning method. First, Algorithm 3 was employed to learn the OOBN model of the insurance domain. This was done to identify the effect of prior information that is not easily exploited when the domain is not regarded as object oriented. Secondly, Algorithm 3 was employed on the stock domain to consider the effect of the OO assumption, and Algorithm 4 was used to verify the method for type uncertainty calculations. Finally, Algorithm 3 was tested w.r.t. predictive accuracy in the insurance domain.

6.1 Setup of the Empirical Study

The goal of the empirical study was to evaluate whether or not the proposed learning methods generate good estimates of the unknown statistical distribution. Let $f(\boldsymbol{x}|\Theta)$ be the unknown *gold standard distribution*; $\boldsymbol{x}$ is a configuration of the domain and Θ are the parameters. $\hat{f}_N(\boldsymbol{x}|\hat{\Phi}_N)$ (or simply $\hat{f}_N$) will be used to denote the approximation of $f(\boldsymbol{x}|\Theta)$ based on N cases from the database.

Since an estimated model may have other edges than the gold standard model, the learned CPTs of $\hat{\Phi}_N$ may have other domains than the CPTs of Θ. Hence a global measure for the difference between the gold standard model and the estimated model is required. In

13. Note that only those parts of the domain that have been changed by the classification of I need to be re-learned.

the tests performed, we have measured this difference by using the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) between the gold standard model and the estimated model. The KL divergence is defined as

$$D\left(f \,||\, \hat{f}_N\right) = \sum_{\boldsymbol{x}} f(\boldsymbol{x}|\Theta) \log\left[\frac{f(\boldsymbol{x}|\Theta)}{\hat{f}_N(\boldsymbol{x}|\hat{\Phi}_N)}\right].$$

There are many arguments for using this particular measurement for calculating the quality of the approximation (see Cover and Thomas, 1991). One of them is the fact that the KL divergence bound the maximum error in the assessed probability for a particular event A, (Whittaker, 1990, Proposition 4.3.7):

$$\sup_A \left| \sum_{\boldsymbol{x}\in A} f(\boldsymbol{x} \mid \boldsymbol{\Theta}) - \sum_{\boldsymbol{x}\in A} \hat{f}_N(\boldsymbol{x}|\hat{\Phi}_N) \right| \leq \sqrt{\frac{1}{2}\cdot D\left(f \,||\, \hat{f}_N\right)}\ .$$

Similar result for the maximal error of the estimated conditional distribution is derived by van Engelen (1997). These results have made the KL divergence the "distance measure"[14] of choice in Bayesian network learning, see for example Pearl (1988), Lam and Bacchus (1994), Heckerman et al. (1995), Friedman and Yakhini (1996), Dasgupta (1997), Friedman (1998), and Cowell et al. (1999).

The learning method was tested by randomly generating a database of size N from the gold standard model, where 25% of the data was *missing completely at random*[15] (Little and Rubin, 1987; Heitjan and Basu, 1996). Note that the proposed algorithms actually only depend on the assumption that the data is missing at random. It is also worth emphasizing that all nodes in the underlying BN are observable in our tests (recall that input nodes are not part of the underlying BN as these nodes are merged with the referenced nodes, see Algorithm 1). The database was used as input to the structural learning algorithms. This was repeated a total of 50 times, with N varying from 100 to 10.000. In our tests we used Algorithm 3 with a maximum of 10 iterations (approximate convergence was typically reached in 4–5 iterations). In each iteration a simulated annealing with parameters $T_0 = 50$, $\alpha = 100$, $\beta = 100$, $\gamma = 0.75$ and $\delta = 220$ (see Heckerman et al., 1995, for notation) was performed. We refer the interested reader to Myers et al. (1999) for additional discussion on stochastic search algorithms for learning Bayesian networks.

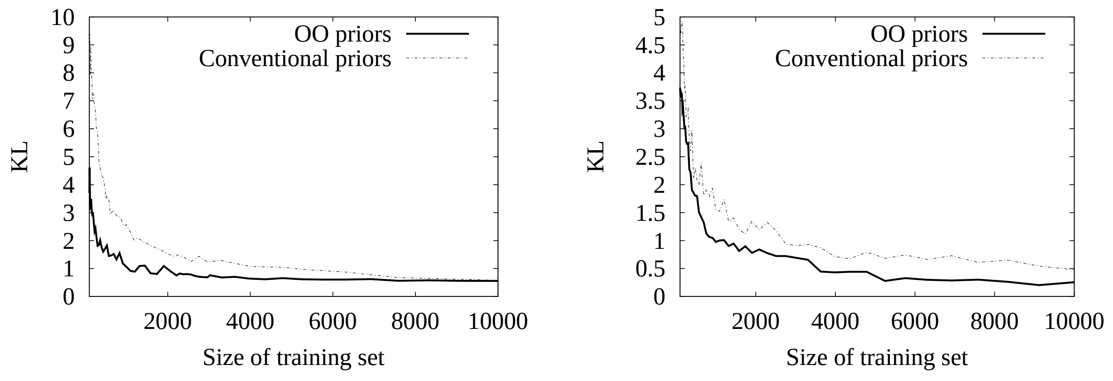
Observe that in the tests we do not consider the issue of running time. However, even though the proposed algorithms might seem more complex than the SEM algorithm (due to the nested iterations) the search space is in fact smaller. From Robinson (1973) we have that the number of DAGs over n nodes can be expressed by the recursive formula

$$G(n) = \sum_{s=1}^{n} (-1)^{s+1} \cdot \binom{n}{s} \cdot 2^{s\cdot(n-s)} \cdot G(n-s),$$

where $G(0) = 1$. Thus, the BN search space for the OMD example consists of approximately 10^{95} different DAGs, whereas the number of OOBN models for this example is not more

14. The KL divergence is not a distance measure in the mathematical sense, as $D\left(f \,||\, g\right) = D\left(g \,||\, f\right)$ does not hold in general. The term here is used in the everyday-meaning of the phrase.

15. Informally, missing completely at random means that the observability of a variable does not depend on the value of any other variable (neither missing nor observed).



a) Insurance domain b) Stock domain

Figure 9: The KL divergence of the gold standard model vs. the generated models for the two cases "Conventional priors" ($\rho(X_i, \Pi_i(B_S)) = 1/65^{|\Pi_i(B_S)|}$) and "OO priors", where parts of the search-space violating the prior information regarding the interfaces were disregarded.

than 10^{22}. This reduction in the search space should therefore (depending on the search algorithm) either produce a better result when using the same amount of time or require fewer steps as compared to the ordinary SEM algorithm.

6.2 The Empirical Results

Consider again the OOBN version of the insurance network described in Section 2.2, and recall the small experiment we performed in our research group to elicit object oriented prior information in this domain (described in Section 4.2). The goal of the experiment was to find edges in the OOBN we could disregard a priori. The result was that out of the 702 edges that can be included in the network structure, only 253 were marked possible, including all the 52 edges actually in the network structure. Based on this experiment, we employed the algorithm to examine to what extent this prior information could benefit the search algorithm.

The empirical results for the insurance domain is given in Figure 9a. The object oriented prior information regarding the interfaces was coded as absolutely certain ($\omega_{ij}^{+} = \infty$ if an edge $X_i \rightarrow X_j$ required a larger interface than given by the prior information). As expected, the KL divergence decreases with the level of information available to the learning algorithm, such that the results based on the "OO priors" is superior to the ones based on "conventional priors" (that is, the standard SEM algorithm) for smaller databases. The results seem to be at the same level for large databases, say $N > 8.000$.

The second test was conducted to analyze the effect of making the OO assumption, and was based on the stock domain. This domain consists of 2 instantiations of the **Meat cow** class and 2 instantiations of the class **Milk cow**, and it was expected that *knowing* that pairs of cows were identical would increase the learning speed; the results in Figure 9b clearly show this effect.[16] Note that learning of DBNs (see Section 3.4) is simply a special case of OOBN learning, since any DBN can be modelled by the usage of two OOBN classes (see Sections 2 and 4.2). Hence, the results by Friedman et al. (1998) can be regarded as the effect of the OO assumption in that special case.

A test was also performed to verify the type uncertainty algorithm. The test was based on the stock domain, and we assumed that the domain expert was ignorant about the classification of Cow1. We employed Algorithm 4 to this problem, and the results are shown in Figure 10, together with the results when consistently choosing the *wrong* classification (**Milk Cow**) and when consistently choosing the *correct* classification (**Meat Cow**) averaged over five runs. The results are fairly promising, as the algorithm was able to build a model which is comparable to the correct classification. Note that this problem was made fairly difficult, as can be seen from the difference in the KL divergence between the correct and the wrong classifications in Figure 10; the domain used by Langseth and Bangsø (2001) has been modified to make the differences between the classes sufficiently small for the problem to be challenging.[17]

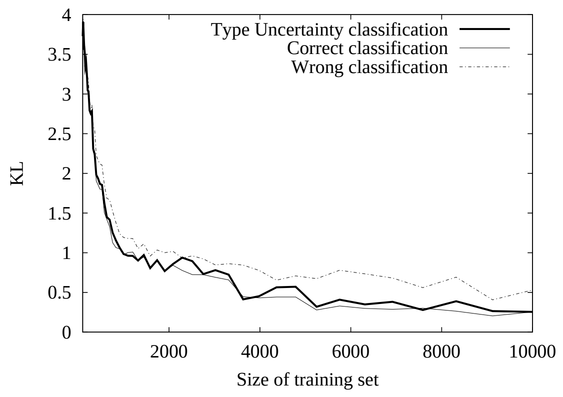


Figure 10: The KL divergence of the gold standard model vs. the generated models for the three cases "Type uncertainty classification" (Algorithm 4), the results of "Correct classification" and "Wrong classification".

Finally, a test was performed to compare the predictive performance of networks learned using Algorithm 3 and the SEM algorithm (Algorithm 2). We generated two databases from the insurance network. The databases consisted of 2000 and 8000 cases, respectively, and 25% of the data was missing completely at random. For this specific situation we tried to predict the expected cost of insurance, corresponding to the sum of the expected costs as indicated by the variables *ILiCost*, *MedCost* and *PropCost* (we assumed that the utility was linear w.r.t. the monetary values). The expected costs in the learned networks was then compared to the expected cost in the gold standard network. This was done 25.000 times in each network. The test-scenarios were sampled without missing values, but some of the

16. A related result was obtained by Friedman and Goldszmidt (1996).

17. When we used the domain as defined by Langseth and Bangsø (2001) we were able to classify the instantiation correctly for databases as small as $N = 10$ observations.

variables were subsequently removed. Specifically, we removed the variables *RiskAversion*, *Mileage*, *DrivingSkill*, *DrivQuality*, *Theft*, *Accident*, *Cushioning*, *ThisCarDam*, *OtherCarCost*, *ThisCarCost*, *ILiCost*, *MedCost* and *PropCost*. The results of the test is shown is Table 1, which specifies the relative absolute error of the predictions.

2000 cases, Algorithm 2 with uniform priors	0.49
2000 cases, Algorithm 3 with "OO priors"	0.24
8000 cases, Algorithm 2 with uniform priors	0.29
8000 cases, Algorithm 3 with "OO priors"	0.22

Table 1: The table shows the relative absolute error of the predictions for networks learned using the OO-SEM algorithm and the traditional SEM algorithm.

The results show that the predictive performance of networks learned using Algorithm 3 is superior to networks learned using the SEM algorithm for databases of 2000 cases.[18] Similar to the results using the KL divergence, we see that for 8000 cases the predictive performance of the two networks are almost the same.

7. Conclusion

In this paper we have proposed a method for doing structural learning in object oriented domains. The learning algorithm is based on the OOBN framework by Bangsø and Wuillemin (2000b), and has been implemented using a tailor-made version of the Structural EM algorithm by Friedman (1998). The proposed learning algorithm exploits an intuitive way of expressing prior information in object oriented domains, and it was shown to be more efficient than conventional learning algorithms in this setting.

Although the proposed learning algorithm is set in the framework of Bayesian model selection we conjecture that the general idea of learning in the class specifications, instead of in the underlying BN, has a broader applicability. For instance, we expect the overall approach to be applicable when learning OOBNs using constraint-based methods (Spirtes et al., 1993; Steck and Tresp, 1996).

A related area of work is the framework of *probabilistic relational models* (PRMs) (Getoor et al., 2001). A PRM specifies a probability model for classes of objects that can then be used in multiple contexts. Getoor et al. (2001) describe how these models can be learned from relational databases: as opposed to OOBNs the focus is on learning a PRM for a specific context, instead of learning subnetworks (classes) that can be applied in different contexts. Somewhat similar to the proposed algorithms, Getoor et al. (2001) also performs learning at the class level, but avoids the problem of identifying the "input sets" as the context is known, see also Taskar et al. (2001).

18. Note that due to this particular setup of the tests, it is not reasonable to argue about the general predictive performance of the learned networks.

Acknowledgments

We would like to thank our colleagues at the Decision Support Systems group, Aalborg University, for interesting discussions and helpful comments. In particular, Olav Bangsø participated in the outset of this work (Bangsø et al., 2001). We would also like to thank `Hugin Expert` (www.hugin.com) for giving us access to the *Hugin Decision Engine*, which forms the basis for our implementation. Finally, we would like to thank the anonymous reviewers for constructive comments and suggestions for improving the paper.

References

Olav Bangsø, Helge Langseth, and Thomas D. Nielsen. Structural learning in object oriented domains. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*, pages 340–344. AAAI Press, 2001.

Olav Bangsø and Pierre-Henri Wuillemin. Object oriented Bayesian networks. A framework for topdown specification of large Bayesian networks with repetitive structures. Technical report CIT-87.2-00-obphw1, Department of Computer Science, Aalborg University, 2000a.

Olav Bangsø and Pierre-Henri Wuillemin. Top-down construction and repetitive structures representation in Bayesian networks. In *Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference*, pages 282–286. AAAI Press, 2000b.

John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2–3):213–244, 1997.

Wray L. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8:195–210, 1996.

Peter Cheeseman and John Stutz. Bayesian classification (AutoClass): Theory and results. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in knowledge discovery and data mining*, pages 153–180. AAAI/MIT Press, 1996. ISBN 0-262-56097-6.

David M. Chichering and David Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2–3):181–212, 1997.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for constructing Bayesian belief networks from databases. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 86–94. Morgan Kaufmann Publishers, 1991.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley, New York, 1991. ISBN 0-471-06259-6.

Robert G. Cowell, A. Phillip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Sciences. Springer Verlag, New York, 1999. ISBN 0-387-98767-3.

Sanjoy Dasgupta. The sample complexity of learning fixed-structure Bayesian networks. *Machine Learning*, 29(2–3):165–180, 1997.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

Gal Elidan and Nir Friedman. Learning the dimensionality of hidden variables. In *Proceedings of the Seventeenth Conference on Uncertainty of Artificial Intelligence*, pages 144–151. Morgan Kaufmann Publishers, 2001.

Gal Elidan, Noam Lotner, Nir Friedman, and Daphne Koller. Discovering hidden variables: A structure-based approach. In *Advances in Neural Information Processing Systems 13*, pages 479–485. MIT Press, 2000.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2–3):131–163, 1997a.

Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1300–1309. Morgan Kaufmann Publishers, 1999.

Nir Friedman, Moises Goldszmidt, David Heckerman, and Stuart Russell. Challenge: Where is the impact of Bayesian networks in learning? In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers, 1997b. URL: http://www.cs.huji.ac.il/labs/compbio/Repository/.

Nir Friedman and Moises Goldszmidt. Learning Bayesian networks with local structure. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 252–262. Morgan Kaufmann Publishers, 1996.

Nir Friedman and Daphne Koller. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1–2):99–125, 2003.

Nir Friedman, Kevin P. Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 139–147. Morgan Kaufmann Publishers, 1998.

Nir Friedman and Zohar Yakhini. On the sample complexity of learning Bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 274–282. Morgan Kaufmann Publishers, 1996.

Nir Friedman. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138. Morgan Kaufmann Publishers, 1998.

Dan Geiger, David Heckerman, and Christopher Meek. Asymptotic model selection with hidden variables. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 283–290. Morgan Kaufmann Publishers, 1996.

Lise Getoor, Nir Friedman, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 307–338. Springer Verlag, Berlin, Germany, 2001. ISBN 3-540-42289-7. See also (Friedman et al., 1999).

Peter J. Green. On use of the EM algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society, Series B*, 52(3):443–452, 1990.

Russell Greiner, Adam J. Grove, and Dale Schuurmans. Learning Bayesian nets that perform well. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 198–207. Morgan Kaufmann Publishers, 1997.

David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

David Heckerman. A Bayesian approach to learning causal networks. Technical Report MSR-TR-95-04, Microsoft Research, 1995a.

David Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995b.

Daniel F. Heitjan and Srabashi Basu. Distinguishing "Missing At Random" and "Missing Completely At Random". *The American Statistician*, 50(3):207–213, 1996.

Finn V. Jensen. *An introduction to Bayesian networks.* UCL Press, London, UK, 1996. ISBN 1-857-28332-5.

Finn V. Jensen. *Bayesian Networks and Decision Graphs.* Springer Verlag, New York, 2001. ISBN 0-387-95259-4.

Uffe Kjærulff. A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 121–129. Morgan Kaufmann Publishers, 1992.

Daphne Koller and Avi Pfeffer. Object-oriented Bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 302–313. Morgan Kaufmann Publishers, 1997.

Paul J. Krause. Learning probabilistic networks. *The Knowledge Engineering Review*, 13 (4):321–351, 1998.

Solomon Kullback and Richard A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(4):269–293, 1994.

Helge Langseth and Olav Bangsø. Parameter learning in object oriented Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 31(1/4):221–243, 2001.

Kathryn B. Laskey and Suzanne M. Mahoney. Network fragments: Representing knowledge for constructing probabilistic models. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 334–341. Morgan Kaufmann Publishers, 1997.

Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data.* John Wiley & Sons, New York, 1987. ISBN: 0-471-80254-9.

Suzanne M. Mahoney and Kathryn B. Laskey. Network engineering for complex belief networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 389–396. Morgan Kaufmann Publishers, 1996.

Lars Mathiasen, Andreas Munk-Nielsen, Peter A. Nielsen, and Jan Stage. *Object-oriented analysis & design.* Marko Publishing ApS, Aalborg, Denmark, 2000. ISBN 8-777-51150-6.

James W. Myers, Kathryn B. Laskey, and Tod S. Levitt. Learning Bayesian networks from incomplete data with stochastic search algorithms. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 476–485. Morgan Kaufmann Publishers, 1999.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Mateo, CA., 1988. ISBN 0-934-61373-7.

Avrom J. Pfeffer. *Probabilistic Reasoning for Complex Systems.* Ph.D. thesis, Stanford University, 2000.

Malcolm Pradhan, Gregory Provan, Blackford Middleton, and Max Henrion. Knowledge engineering for large belief networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 484–490. Morgan Kaufmann Publishers, 1994.

Brian D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, UK, 1996. ISBN 0-521-46086-7.

Jorma Rissanen. Stochastic complexity (with discussion). *Journal of the Royal Statistical Society, Series B*, 49(3):223–239 and 253–265, 1987.

Robert W. Robinson. Counting labeled acyclic digraphs. In Frank Harary, editor, *New directions in the theory of graphs*, pages 239–273. Academic Press, New York, 1973.

Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis.* Monographs on statistics and applied probability. Chapman and Hall, London, UK, 1986. ISBN 0-412-24620-1.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search.* Springer Verlag, New York, 1993. ISBN 0-387-97979-4.

Harald Steck and Volker Tresp. Bayesian belief networks for data mining. In *Proceedings of the 2. Workshop on Data Mining und Data Warehousing als Grundlage moderner entscheidungsunterstützender Systeme*, pages 145–154, University of Magdeburg, Germany, 1996. ISBN 3-929-75726-5.

Benjamin Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 870–876. Morgan Kaufmann Publishers, 2001.

Robert A. van Engelen. Approximating Bayesian belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):916–920, 1997.

Mike West and Jeff Harrison. *Bayesian Forecasting and Dynamic Models.* Springer Verlag, New York, 2nd edition, 1997. ISBN 0-387-94725-6.

Joe Whittaker. *Graphical models in applied multivariate statistics.* Wiley, Chichester, 1990. ISBN 0-471-91750-8.

Yang Xiang and Finn V. Jensen. Inference in multiply sectioned Bayesian networks with extended Shafer-Shenoy and lazy propagation. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 680–687. Morgan Kaufmann Publishers, 1999.

Yang Xiang, David Poole, and Michael P. Beddoes. Multiply sectioned Bayesian networks and junction forests for large knowledge-based systems. *Computational Intelligence*, 9(2): 171–220, 1993.