Helge Langseth

Bayesian Networks with Applications in Reliability Analysis

Dr. Ing. Thesis

Department of Mathematical Sciences
Norwegian University of Science and Technology
2002

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree "Doktor Ingeniør" (Dr.Ing.) at the Norwegian University of Science and Technology (NTNU). The work is financed by a scholarship from the Norwegian Research Council of Norway.

I would like to thank my supervisors Bo Lindqvist and Agnar Aamodt for their guidance and support. I would also like to thank the members of the Decision Support Systems Group at Aalborg University for teaching me most of what I know about Bayesian networks and influence diagrams. My stay in Denmark from August 1999 to July 2001 was a wonderful period, and a special thanks to Thomas D. Nielsen, Finn Verner Jensen and Olav Bangsø for making those years so memorable. Furthermore, I would like to thank my co-authors Agnar Aamodt, Olav Bangsø, Finn Verner Jensen, Uffe Kjærulff, Brian Kristiansen, Bo Lindqvist, Thomas D. Nielsen, Claus Skaanning, Jiří Vomlel, Marta Vomlelová, and Ole Martin Winnem for inspiring cooperation. Finally, I would like to thank Mona for keeping up with me over the last couple of years. Her part in this work is larger than anybody (including myself, unfortunately) will ever know.

Trondheim, October 2002

Helge Langseth

List of papers

The thesis consists of the following 5 papers:

Paper I: Helge Langseth and Bo Henry Lindqvist: A maintenance model for components exposed to several failure modes and imperfect repair. Technical Report Statistics 10/2002, Department of Mathematical Sciences, Norwegian University of Science and Technology. Submitted as an invited paper to *Mathematical and Statistical Methods in Reliability* Kjell Doksum and Bo Henry Lindqvist (Eds.), 2002.

Paper II: Helge Langseth and Finn Verner Jensen: Decision theoretic troubleshooting of coherent systems. *Reliability Engineering and System Safety.* Forthcoming, 2002.

Paper III: Helge Langseth and Thomas D. Nielsen: Classification using hierarchical naïve Bayes models. Technical Report TR-02-004, Department of Computer Science, Aalborg University, Denmark, 2002.

Paper IV: Helge Langseth and Olav Bangsø: Parameter learning in object oriented Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32 (1/4):221–243, 2001.

Paper V: Helge Langseth and Thomas D. Nielsen: Fusion of domain knowledge with data for structural learning in object oriented domains. *Journal of Machine Learning Research.* Forthcoming, 2002.

The papers are selected to cover most of the work I have been involved in over the last years, but s.t. they all share the same core: Bayesian network technology with possible applications in reliability analysis.

All papers can be read independently of each other, although Paper IV and Paper V are closely related. Paper I is concerned with building a model for maintenance optimization; it is written for an audience of reliability data analysts. Papers II – V are related to problem solving (Paper II and Paper III) and estimation (Paper IV and Paper V) using the Bayesian network formalism. These papers are written for an audience familiar with both computer science as well as statistics, but with a terminology mostly collected from the computer scientists' vocabulary.

Background

Reliability analysis is deeply rooted in models for time to failure (survival analysis). The analysis of such time-to-event data arises in many fields, including medicine, actuarial sciences, economics, biology, public health and engineering. The Bayesian paradigm has played an important role in survival analysis because the time-to-event data can be sparse and heavily censored. The statistical models must therefore in part be based on expert judgement where *a priori* knowledge is combined with quantitative information represented by data (Martz and Waller 1982; Ibrahim et al. 2001), see also (Gelman et al. 1995). Bayesian approaches to survival analysis has lately received quite some attention due to recent advances in computational and modelling techniques (commonly referred to as computer-intensive statistical methods), and Bayesian techniques like flexible hierarchical models have for example become common in reliability analysis.

Reliability models of repairable systems often become complex, and they may be difficult to build using traditional frameworks. Additionally, reliability analyses that historically were mostly conducted for documentation purposes are now used as direct input to complex decision problems. The complexity of these decision problems can lead to a situation where the decision maker looses his overview, which in turn can lead to sub-optimal decisions. This has paved the way for formalisms that offer a transparent yet mathematically sound modelling framework; the statistical models must build on simple semantics (to interact with domain experts and the decision maker) and at the same time offer the mathematical finesse required to model the actual decision problem at hand.

The framework employed in this thesis is (discrete) Bayesian networks (BNs); BNs are described in (Pearl 1988; Jensen 1996; Lauritzen 1996; Cowell et al. 1999; Jensen 2001). A discrete BN encodes the probability mass function governing a set $\{X_1, \ldots, X_n\}$ of discrete random variables by specifying a set of conditional independence assumptions together with a set of conditional probability tables (CPTs). More specifically, a BN consists of a qualitative part; a directed acyclic graph where the nodes mirror the random variables X_i, and a quantitative part; the set of CPTs. We call the nodes with outgoing edges directed into a specific node the *parents* of that node, and say that a node X_j is a descendant of X_i if and only if there is a directed path from X_i to X_j in the graph. Now, the edges of the graph represent the assertion that a variable is conditionally independent of its non-descendants in the graph given its parents (other conditional independence statements can be read off the graph using *d-separation* rules (Pearl 1988)). Next, a CPT is specified for each variable, describing the conditional probability mass for that variable given the state of its parents. Note that a BN can represent any probability mass function, and through its factorized representation it does so in a cost-efficient manner (wrt. the number of parameters required to describe the probability mass function).

The most important task in a BN is *inference*, i.e., to calculate conditional probabilities over some target variables conditioned on the observed values of other variables (for example the probability of a system being broken given the state of some of its components). Both

exact as well as approximate inference in a BN is in general NP-hard (Cooper 1990; Dagum and Luby 1993), but fortunately both exact propagation-algorithms (Shafer and Shenoy 1990; Jensen et al. 1990; Jensen 1996) as well as MCMC simulation (Geman and Geman 1984; Gilks et al. 1994; Gilks et al. 1996) have shown useful in practice.

The Bayesian formalism offers an intuitive way to estimate models based on the combination of statistical data and expert judgement. For a given graphical structure, estimation of the conditional probability tables was considered by Spiegelhalter and Lauritzen (1990), who showed how the full posterior distribution over the parameter-space can be obtained in closed form by local computations. The *EM-algorithm* by Dempster et al. (1977) is particularly intuitive in BN models, as the sufficient statistics required for parameter learning are available in the *cliques* after propagation (Lauritzen 1995). The EM-algorithm can also be used to find *MAP-parameters* (Green 1990). Structural learning, i.e., to estimate the graphical structure of a BN (the edges of the graph), is considered in (Cooper and Herskovits 1992; Heckerman et al. 1995; Friedman 1998). A BN structure constrains the set of possible CPTs by defining their scopes, and this is utilized in (Cooper and Herskovits 1992), where it is shown how a posterior distribution over the space of directed acyclic graphs can be obtained through local computations. Heckerman et al. (1995) examine the usage of priors over the model-space, and empirically investigate the use of (stochastic) search over this space. Friedman (1998) extends these results to cope with missing data.

The fast inference algorithms and simple semantics of the BN models have lead to a continuous trend of building increasingly larger BN models. Such large models can be time consuming to build and maintain, and this problem is attacked by defining special "types" of BNs tailor-made for complex domains: Both (Koller and Pfeffer 1997) as well as (Bangsø and Wuillemin 2000) describe modelling languages where repetitive substructures play an important role during model building; these frameworks are called object oriented BNs. A language for probabilistic frame-based systems is proposed in (Koller and Pfeffer 1998), and rational models (i.e., models associated with a relational domain structure as defined for instance by a relational database) is described in (Getoor et al. 2001).

Historically, BNs have been used in two quite different settings in the safety and reliability sciences. The first body of work uses BNs solely as a tool for building complex statistical models. Analysis of lifetime data, models to extend the flexibility of classical reliability techniques (such as fault trees and reliability block diagrams), fault finding systems, and models for human errors and organizational factors all fall into this category. On the other hand, some researchers regard BNs as *causal Markov models*, and use them in for example accident investigation. The recent book by Pearl (2000), see also (Spirtes et al. 1993), gives a clear exposition of BNs as causal models, and although statisticians have traditionally been reluctant to the use of causal models (Speed (1990) wrote: *"Considerations of causality should be treated as they have always been treated in statistics: preferably not at all but, if necessary, then with great care."*) a statistical treatment of causal mechanisms and causal inference in association with Bayesian networks and influence diagrams is starting to dawn, see e.g., (Lauritzen 2001; Dawid 2002).

Summary

A common goal of the papers in this thesis is to propose, formalize and exemplify the use of Bayesian networks as a modelling tool in reliability analysis. The papers span work in which Bayesian networks are merely used as a modelling tool (Paper I), work where models are specially designed to utilize the inference algorithms of Bayesian networks (Paper II and Paper III), and work where the focus has been on extending the applicability of Bayesian networks to very large domains (Paper IV and Paper V).

Paper I is in this respect an application paper, where model building, estimation and inference in a complex time-evolving model is simplified by focusing on the conditional independence statements embedded in the model; it is written with the reliability data analyst in mind. We investigate the mathematical modelling of maintenance and repair of components that can fail due to a variety of failure mechanisms. Our motivation is to build a model, which can be used to unveil aspects of the "quality" of the maintenance performed. This "quality" is measured by two groups of model parameters: The first measures "eagerness", the maintenance crew's ability to perform maintenance at the right time to try to stop an evolving failure; the second measures "thoroughness", the crew's ability to actually stop the failure development. The model we propose is motivated by the *imperfect repair* model of Brown and Proschan (1983), but extended to model preventive maintenance as one of several *competing risks* (David and Moeschberger 1978). The competing risk model we use is based on *random signs censoring* (Cooke 1996). The explicit maintenance model helps us to avoid problems of identifiability in connection with imperfect repair models previously reported by Whitaker and Samaniego (1989). The main contribution of this paper is a simple yet flexible reliability model for components that are subject to several failure mechanisms, and which are not always given perfect repair. Reliability models that involve repairable systems with non-perfect repair, and a variety of failure mechanisms often become very complex, and they may be difficult to build using traditional reliability models. The analysis are typically performed to optimize the maintenance regime, and the complexity problems can, in the worst case, lead to sub-optimal decisions regarding maintenance strategies. Our model is represented by a Bayesian network, and we use the conditional independence relations encoded in the network structure in the calculation scheme employed to generate parameter estimates.

In **Paper II** we target the problem of *fault diagnosis*, i.e., to efficiently generate an inspection strategy to detect and repair a complex system. Troubleshooting has long traditions in reliability analysis, see e.g. (Vesely 1970; Zhang and Mei 1987; Xiaozhong and Cooke 1992; Norstrøm et al. 1999). However, traditional troubleshooting systems are built using a very restrictive representation language: One typically assumes that all attempts to inspect or repair components are successful, a repair action is related to one component only, and the user cannot supply any information to the troubleshooting system except for the outcome of repair actions and inspections. A recent trend in fault diagnosis is to use Bayesian networks to represent the troubleshooting domain (Breese and Heckerman 1996;

Jensen et al. 2001). This allows a more flexible representation, where we, e.g., can model non-perfect repair actions and questions. Questions are troubleshooting steps that do not aim at repairing the device, but merely are performed to capture information about the failed equipment, and thereby ease the identification and repair of the fault. Breese and Heckerman (1996) and Jensen et al. (2001) focus on fault finding in *serial systems*. In Paper II we relax this assumption and extend the results to any *coherent system* (Barlow and Proschan 1975). General troubleshooting is NP-hard (Sochorová and Vomlel 2000); we therefore focus on giving an approximate algorithm which generates a "good" troubleshooting strategy, and discuss how to incorporate questions into this strategy. Finally, we utilize certain properties of the domain to propose a fast calculation scheme.

Classification is the task of predicting the class of an instance from as set of attributes describing it, i.e., to apply a mapping from the attribute space to a predefined set of classes. In the context of this thesis one may for instance decide whether a component requires thorough maintenance or not based on its usage pattern and environmental conditions. *Classifier learning*, which is the theme of **Paper III**, is to automatically generate such a mapping based on a database of labelled instances. Classifier learning has a rich literature in statistics under the name of *supervised pattern recognition*, see e.g. (McLachlan 1992; Ripley 1996). Classifier learning can be seen as a model selection process, where the task is to find the model from a class of models with highest classification accuracy. With this perspective it is obvious that the model class we select the classifier from is crucial for classification accuracy. We use the class of Hierarchical Naïve Bayes (HNB) models (Zhang 2002) to generate a classifier from data. HNBs constitute a relatively new model class which extends the modelling flexibility of *Naïve Bayes* (NB) models (Duda and Hart 1973). The NB models is a class of particularly simple classifier models, which has shown to offer very good classification accuracy as measured by the 0/1-loss. However, NB models assume that all attributes are conditionally independent given the class, and this assumption is clearly violated in many real world problems. In such situations overlapping information is counted twice by the classifier. To resolve this problem, finding methods for handling the conditional dependence between the attributes has become a lively research area; these methods are typically grouped into three categories: Feature selection, feature grouping, and correlation modelling. HNB classifiers fall in the last category, as HNB models are made by introducing latent variables to relax the independence statements encoded in an NB model. The main contribution of this paper is a fast algorithm to generate HNB classifiers. We give a set of experimental results which show that the HNB classifiers can significantly improve the classification accuracy of the NB models, and also outperform other often-used classification systems.

In **Paper IV** and **Paper V** we work with a framework for modelling large domains. 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 in, e.g., (Koller and Pfeffer 1997; Bangsø and Wuillemin 2000). In domains that can appro-

priately be described using an object oriented language (Mahoney and Laskey 1996) 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. The basic building blocks available from domain experts examining such domains are information about random variables that are grouped into substructures with high internal coupling and low external coupling. These substructures naturally correspond to instantiations in an object-oriented BN (OOBN). For instance, an instantiation may correspond to a physical object or it may describe a set of entities that occur at the same instant of time (a dynamic Bayesian network (Kjærulff 1992) is a special case of an OOBN). Moreover, analogously to the grouping of similar substructures into categories, instantiations of the same type are grouped into classes. As an example, several variables describing a specific pump may be said to make up an *instantiation*. All instantiations describing the same type of pump are said to be instantiations of the same *class*. OOBNs offer an easy way of defining BNs in such object-oriented domains s.t. the object-oriented properties of the domain are taken advantage of during model building, and also explicitly encoded in the model. 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. In Paper IV and Paper V we work with learning of parameters and specifying the structure in the OOBN definition of Bangsø and Wuillemin (2000).

Paper IV describes a method for parameter learning in OOBNs. The contributions in this paper are three-fold: Firstly, we propose a method for learning parameters in OOBNs based on the *EM-algorithm* (Dempster et al. 1977), and prove that maintaining the object orientation imposed by the prior model will increase the learning speed in object oriented domains. Secondly, we propose a method to efficiently estimate the probability parameters in domains that are *not* strictly object oriented. More specifically, we show how *Bayesian model averaging* (Hoeting et al. 1999) offers well-founded tradeoff between model complexity and model fit in this setting. Finally, we attack the situation where the domain expert is unable to classify an instantiation to a given class or a set of instantiations to classes (Pfeffer (2000) calls this *type uncertainty*; a case of model uncertainty typical to object oriented domains). We show how our algorithm can be extended to work with OOBNs that are only partly specified.

In Paper V we estimate the OOBN structure. When constructing a Bayesian network, it can be advantageous to employ structural learning algorithms (Cooper and Herskovits 1992; Heckerman et al. 1995) 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 (this is for instance the case for prior information about repetitive structures). The main contribution of Paper V is a method for doing structural learning in object oriented domains. We argue that the method supports a natural approach for expressing and incorporating prior information provided by domain experts and show how this type of prior information can be exploited during structural

learning. Our method is built on the *Structural EM-algorithm* (Friedman 1998), and we prove our algorithm to be asymptotically consistent. Empirical results demonstrate that the proposed learning algorithm is more efficient than conventional learning algorithms in object oriented domains. We also consider structural learning under type uncertainty, and find through a discrete optimization technique a candidate OOBN structure that describes the data well.

References

Aamodt, A. and H. Langseth (1998). Integrating Bayesian networks into knowledge intensive CBR. In *American Association for Artificial Intelligence, Case-based reasoning integrations; Papers from the AAAI workshop – Technical Report WS-98-15*, Madison, WI., pp. 1–6. AAAI Press.

Bangsø, O., H. Langseth, and T. D. Nielsen (2001). Structural learning in object oriented domains. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*, Key West, FL., pp. 340–344. AAAI Press.

Bangsø, O. and P.-H. Wuillemin (2000). Top-down construction and repetitive structures representation in Bayesian networks. In *Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference*, Orlando, FL., pp. 282–286. AAAI Press.

Barlow, R. E. and F. Proschan (1975). *Statistical Theory of Reliability and Life Testing: Probability Models*. Silver Spring, MD.: To Begin With.

Breese, J. S. and D. Heckerman (1996). Decision-theoretic troubleshooting: A framework for repair and experiment. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 124–132. Morgan Kaufmann Publishers.

Brown, M. and F. Proschan (1983). Imperfect repair. *Journal of Applied Probability 20*, 851–859.

Cooke, R. M. (1996). The design of reliability data bases, Part I and Part II. *Reliability Engineering and System Safety 52*, 137–146 and 209–223.

Cooper, G. F. (1990). Computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence 42*(2–3), 393–405.

Cooper, G. F. and E. Herskovits (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning 9*, 309–347.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Sciences. New York: Springer Verlag.

Dagum, P. and M. Luby (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence 60*(1), 141–153.

David, H. A. and M. L. Moeschberger (1978). *Theory of Competing Risks*. London: Griffin.

Dawid, A. P. (2002). Influence diagrams for causal modelling and inference. *International Statistical Review 70*(2), 161–189.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B 39*, 1–38.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, San Fransisco, CA., pp. 129–138. Morgan Kaufmann Publishers.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (1995). *Bayesian data analysis*. London, UK: Chapman and Hall.

Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 721–741.

Getoor, L., N. Friedman, D. Koller, and A. Pfeffer (2001). Learning probabilistic relational models. In *Relational Data Mining*, pp. 307–338. Berlin, Germany: Springer Verlag.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter (1996). *Markov Chain Monte Carlo in Practice*. London, UK.: Chapman & Hall.

Gilks, W. R., A. Thomas, and D. J. Spiegelhalter (1994). A language and program for complex Bayesian modelling. *The Statistician 43*(1), 169–178.

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

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

Hoeting, J., D. Madigan, A. Raftery, and C. T. Volinsky (1999). Bayesian model averaging: A tutorial (with discussion). *Statistical Science 14*(4), 382–417.

Ibrahim, J. G., M.-H. Chen, and D. Sinha (2001). *Bayesian survival analysis*. New York: Springer.

Jensen, F. V. (1996). *An introduction to Bayesian Networks*. London, UK.: Taylor and Francis.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. New York: Springer Verlag.

Jensen, F. V., U. Kjærulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, and M. Vomlelová (2001). The SACSO methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing 15*(5), 321–333.

Jensen, F. V., S. L. Lauritzen, and K. G. Olesen (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly 4*, 269–282.

Kjærulff, U. (1992). A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, San Fransisco, CA., pp. 121–129. Morgan Kaufmann Publishers.

Koller, D. and A. Pfeffer (1997). Object-oriented Bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, San Fransisco, CA., pp. 302–313. Morgan Kaufmann Publishers.

Koller, D. and A. Pfeffer (1998). Probabilistic frame-based systems. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)*, Madison, WI., pp. 580–587. AAAI Press.

Langseth, H. (1998). Analysis of survival times using Bayesian networks. In S. Lydersen, G. K. Hansen, and H. A. Sandtorv (Eds.), *Proceedings of the ninth European Conference on Safety and Reliability - ESREL'98*, Trondheim, Norway, pp. 647 – 654. A. A. Balkema.

Langseth, H. (1999). Modelling maintenance for components under competing risk. In G. I. Schuëller and P. Kafka (Eds.), *Proceedings of the tenth European Conference on Safety and Reliability – ESREL'99*, Munich, Germany, pp. 179–184. A. A. Balkema.

Langseth, H., A. Aamodt, and O. M. Winnem (1999). Learning retrieval knowledge from data. In S. S. Anand, A. Aamodt, and D. W. Aha (Eds.), *Sixteenth International Joint Conference on Artificial Intelligence, Workshop ML-5: Automating the Construction of Case-Based Reasoners*, Stockholm, Sweden, pp. 77–82.

Langseth, H. and O. Bangsø (2001). Parameter learning in object oriented Bayesian networks. *Annals of Mathematics and Artificial Intelligence 32*(1/4), 221–243.

Langseth, H. and F. V. Jensen (2001). Heuristics for two extensions of basic troubleshooting. In H. H. Lund, B. Mayoh, and J. Perram (Eds.), *Seventh Scandinavian conference on Artificial Intelligence, SCAI'01*, Frontiers in Artificial Intelligence and Applications, Odense, Denmark, pp. 80–89. IOS Press.

Langseth, H. and F. V. Jensen (2002). Decision theoretic troubleshooting of coherent systems. *Reliability Engineering and System Safety*. Forthcoming.

Langseth, H. and B. H. Lindqvist (2002a). A maintenance model for components exposed to several failure modes and imperfect repair. Technical Report Statistics 10/2002, Department of Mathematical Sciences, Norwegian University of Science and Technology.

Langseth, H. and B. H. Lindqvist (2002b). Modelling imperfect maintenance and repair of components under competing risk. In H. Langseth and B. H. Lindqvist (Eds.), *Third International Conference on Mathematical Methods in Reliability – Methodology and Practice. Communications of the MMR'02*, Trondheim, Norway, pp. 359. Tapir Trykk.

Langseth, H. and T. D. Nielsen (2002a). Classification using Hierarchical Naïve Bayes models. Technical Report TR-02-004, Department of Computer Science, Aalborg University, Denmark.

Langseth, H. and T. D. Nielsen (2002b). Fusion of domain knowledge with data for structural learning in object oriented domains. *Journal of Machine Learning Research*. Forthcoming.

Lauritzen, S. L. (1995). The EM-algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis 19*, 191–201.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford, UK: Clarendon Press.

Lauritzen, S. L. (2001). Causal inference from graphical models. In O. E. Barndorff-Nielsen, D. R. Cox, and C. Klüppelberg (Eds.), *Complex Stochastic Systems*, pp. 63–107. London, UK: Chapman and Hall/CRC.

Mahoney, S. M. and K. B. Laskey (1996). Network engineering for complef belief networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, San Fransisco, CA., pp. 389–396. Morgan Kaufmann Publishers.

Martz, H. F. and R. A. Waller (1982). *Bayesian reliability analysis*. New York: Wiley.

McLachlan, G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Norstrøm, J., R. M. Cooke, and T. J. Bedford (1999). Value of information based inspection-strategy of a fault-tree. In *Proceedings of the tenth European Conference on Safety and Reliability*, Munich, Germany, pp. 621–626. A. A. Balkema.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA.: Morgan Kaufmann Publishers.

Pearl, J. (2000). *Causality – Models, Reasoning, and Inference*. Cambridge, UK: Cambridge University Press.

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

Pradhan, M., G. Provan, B. Middleton, and M. Henrion (1994). Knowledge engineering for large belief networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, San Fransisco, CA., pp. 484–490. Morgan Kaufmann Publishers.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge, UK: Cambridge University Press.

Shafer, G. R. and P. P. Shenoy (1990). Probability propagation. *Annals of Mathematics and Artificial Intelligence 2*, 327–352.

Sochorová, M. and J. Vomlel (2000). Troubleshooting: NP-hardness and solution methods. In *The Proceedings of the Fifth Workshop on Uncertainty Processing, WUPES'2000*, Jindřichův Hradec, Czech Republic, pp. 198–212.

Speed, T. P. (1990). Complexity, calibration and causality in influence diagrams. In R. M. Oliver and J. Q. Smith (Eds.), *Influence Diagrams, Belief Nets and Decision Analysis*, pp. 49–63. New York: Wiley.

Spiegelhalter, D. J. and S. L. Lauritzen (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks 20*, 579–605.

Spirtes, P., C. Glymour, and R. Scheines (1993). *Causation, Prediction, and Search*. New York: Springer Verlag.

Vesely, W. E. (1970). A time-dependent methodology for fault tree evaluation. *Nuclear Engineering and design 13*, 339–360.

Whitaker, L. R. and F. J. Samaniego (1989). Estimating the reliability of systems subject to imperfect repair. *Journal of American Statistical Association 84*, 301–309.

Xiaozhong, W. and R. M. Cooke (1992). Optimal inspection sequence in fault diagnosis. *Reliability Engineering and System Safety 37*, 207–210.

Zhang, N. (2002). Hierarchical latent class models for cluster analysis. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, Menlo Park, CA., pp. 230–237. AAAI Press.

Zhang, Q. and Q. Mei (1987). A sequence of diagnosis and repair for a 2-state repairable system. *IEEE Transactions on Reliability R-36*(1), 32–33.

I

A Maintenance Model for Components Exposed to Several Failure Modes and Imperfect Repair

A MAINTENANCE MODEL FOR COMPONENTS EXPOSED TO SEVERAL FAILURE MECHANISMS AND IMPERFECT REPAIR

HELGE LANGSETH

Department of Mathematical Sciences
Norwegian University of Science and Technology
N-7491 Trondheim, Norway

and

BO HENRY LINDQVIST

Department of Mathematical Sciences
Norwegian University of Science and Technology
N-7491 Trondheim, Norway

We investigate the mathematical modelling of maintenance and repair of components that can fail due to a variety of failure mechanisms. Our motivation is to build a model, which can be used to unveil aspects of the quality of the maintenance performed. The model we propose is motivated by imperfect repair models, but extended to model preventive maintenance as one of several "competing risks". This helps us to avoid problems of identifiability previously reported in connection with imperfect repair models. Parameter estimation in the model is based on maximum likelihood calculations. The model is tested using real data from the OREDA database, and the results are compared to results from standard repair models.

1. Introduction

In this paper we employ a model for components which fail due to one of a series of "competing" failure mechanisms, each acting independently on the system. The components under consideration are repaired upon failure, but are also preventively maintained. The preventive maintenance (PM) is performed periodically with some fixed period τ, but PM can also be performed out of schedule due to casual observation of an evolving failure. The maintenance need not be perfect; we use a modified version of the imperfect repair model by Brown and Proschan[1] to allow a flexible yet simple maintenance model. Our motivation for this model is to estimate quantities which describe the "goodness" of the maintenance crew; their ability to prevent failures by performing thorough maintenance at the correct time. The data required to estimate the parameters in the model we propose are the intermediate

failure times, the "winning" failure mechanism associated with each failure (i.e. the failure mechanism leading to the failure), as well as the maintenance activity. This data is found in most modern reliability data banks.

The rest of this paper is outlined as follows: We start in Section 2 with the problem definition by introducing the type of data and parameters we consider. Next, the required theoretical background is sketched in Section 3, followed by a complete description of the proposed model in Section 4. Empirical results are reported in Section 5, and we make some concluding remarks in Section 6.

2. Problem definition, typical data and model parameters

Consider a mechanical component which may fail at random times, and which after failure is immediately repaired and put back into service. In practice there can be several root causes for the failure, e.g. vibration, corrosion, etc. We call these causes failure mechanisms and denote them by $M_1, \ldots, M_k$. It is assumed that each failure can be classified as the consequence of exactly one failure mechanism.

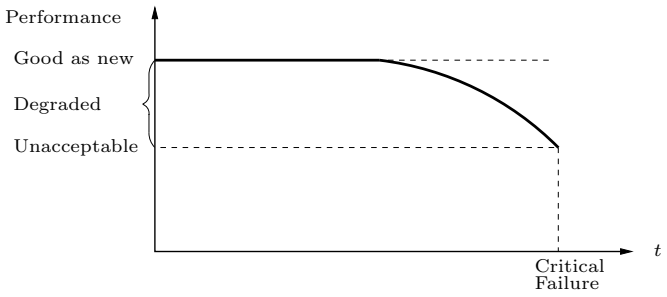


Figure 1: Component with degrading performance.

The component is assumed to undergo preventive maintenance (PM), usually at fixed time periods $\tau > 0$. In addition, the maintenance crew may perform unscheduled preventive maintenance of a component if required. The rationale for unscheduled PM is illustrated in Figure 1: We assume that the component is continuously deteriorating when used, so that the performance gradually degrades until it falls outside a preset acceptable margin. As soon as the performance is unacceptable, we say that the component experiences a critical failure. Before the component fails it may exhibit inferior but admissible performance. This is a "signal" to the maintenance crew that a critical failure is approaching, and that the inferior component may be repaired. When the maintenance crew intervenes and repairs a component before it fails critically, we call it a degraded failure, and the repair action is called (an unscheduled) preventive maintenance. On the other hand, the repair activity performed after a critical failure is called a corrective maintenance.

The history of the component may in practice be logged as shown in Table 1. The events experienced by the component can be categorized as either (*i*) Critical

Time	Event	Failure mech.	Severity
0	Put into service	—	—
314	Failure	Vibration	Critical
8.760	(Periodic) PM	External	—
17.520	(Periodic) PM	External	—
18.314	Failure	Corrosion	Degraded
20.123	Taken out of service	External	—

Table 1: Example of data describing the history of a fictitious component.

failures, (*ii*) Degraded failures, or (*iii*) External events (component taken out of service, periodic PM, or other kind of censoring).

The data for a single component can now formally be given as an ordered sequence of points

$$(Y_i, K_i, J_i);\ i = 1, 2, \ldots, n\ , \tag{1}$$

where each point represents an event (see Figure 2). Here

$$\begin{aligned} Y_i &= \text{inter-event time, i.e. time since previous event} \\ &\quad \text{(time since start of service if } i = 1\text{)} \\ K_i &= \begin{cases} m \text{ if failure mechanism } M_m \ (m = 1, \ldots, k) \\ 0 \text{ if external event} \end{cases} \\ J_i &= \begin{cases} 0 \text{ if critical failure} \\ 1 \text{ if degraded failure} \\ 2 \text{ if external event .} \end{cases} \end{aligned} \tag{2}$$

The data in Table 1 can thus be coded as (with M_1 = Vibration, M_2 = Corrosion),

$$(314, 1, 0),\ (8446, 0, 2),\ (8760, 0, 2),\ (794, 2, 1),\ (1809, 0, 2)\ .$$

A complete set of data will typically involve events from several similar components. The data can then be represented as

$$(Y_{ij}, K_{ij}, J_{ij});\ i = 1, 2, \ldots, n_j;\ j = 1, \ldots, r\ , \tag{3}$$

where j is the index which labels the component.

In practice there may also be observed covariates with such data. The models considered in this paper will, however, not include this possibility even though they could easily be modified to do so.

Our aim is to present a model for data of type (1) (or (3)). The basic ingredients in such a model are the hazard rates $\omega_m(t)$ at time t for each failure mechanism M_m, for a component which is new at time $t = 0$. We assume that $\omega_m(t)$ is a continuous and integrable function on $[0, \infty)$. In practice it will be important to estimate $\omega_m(\cdot)$ since this information may, e.g., be used to plan future maintenance strategies.

The most frequently used models for repairable systems assume either perfect repair (renewal process models) or minimal repair (nonhomogeneous Poisson-process models). Often none of these may be appropriate, and we shall here adopt the idea of the imperfect repair model presented by Brown and Proschan[1]. This will introduce two parameters per failure mechanism:

$$\begin{aligned} p_m &= \text{probability of perfect repair for a preventive maintenance of } M_m \\ \pi_m &= \text{probability of perfect repair for a corrective maintenance of } M_m. \end{aligned}$$

These quantities are of interest since they can be used as indications of the quality of maintenance. The parameters may in practice be compared between plants and companies, and thereby unveil maintenance improvement potential.

Finally, our model will take into account the relation between preventive and corrective maintenance. It is assumed that the component gives some kind of "signal", which will alert the maintenance crew to perform a preventive maintenance before a critical failure occurs. Thus it is not reasonable to model the (potential) times for preventive and corrective maintenance as stochastically independent. We shall therefore adopt the random signs censoring of Cooke[2]. This will eventually introduce a single new parameter q_m for each failure mechanism, with interpretation as the probability that a critical failure is avoided by a preceding unscheduled preventive maintenance.

In the cases where there is a single failure mechanism, we shall drop the index m on the parameters above.

3. Basic ingredients of the model

In this section we describe and discuss the two main building blocks of our final model. In Section 3.1 we consider the concept of imperfect repair, as defined by Brown and Proschan[1]. Then in Section 3.2 we introduce our basic model for the relation between preventive and corrective maintenance. Throughout the section we assume that there is a single failure mechanism ($k = 1$).

3.1. *Imperfect repair*

Our point of departure is the imperfect repair model of Brown and Proschan[1], which we shall denote BP in the following. Consider a single sequence of failures, occurring at successive times $T_1, T_2, \ldots$ As in the previous section we let the Y_i be times between events, see Figure 2. Furthermore, $N(t)$ is the number of events in $(0, t]$, and $N(t^-)$ is the number of events in $(0, t)$.

For the explanation of imperfect repair models it is convenient to use the conditional intensity

$$\lambda(t \mid \mathcal{F}_{t^-}) = \lim_{\Delta t \downarrow 0} \frac{P(\text{event in } [t, t + \Delta t) \mid \mathcal{F}_{t^-})}{\Delta t} ,$$

where $\mathcal{F}_{t^-}$ is the history of the counting process[3] up to time t. This notation enables us to review some standard repair models. Let $\omega(t)$ be the hazard rate of a com-

ponent of "age" t. Then perfect repair is modelled by $\lambda(t \mid \mathcal{F}_{t^-}) = \omega\left(t - T_{N(t^-)}\right)$ which means that the age of the component at time t equals $t - T_{N(t^-)}$, the time elapsed since the last event. Minimal repair is modelled by $\lambda(t \mid \mathcal{F}_{t^-}) = \omega(t)$, which means that the age at any time t equals the calendar time t. Imperfect repair can be modelled by $\lambda(t \mid \mathcal{F}_{t^-}) = \omega\left(\Xi_{N(t^-)} + t - T_{N(t^-)}\right)$ where $0 \leq \Xi_i \leq T_i$ is some measure of the effective age of the component immediately after the ith event, more precisely, immediately after the corresponding repair. In the BP model, Ξ_i is defined indirectly by letting a failed component be given perfect repair with probability p, and minimal repair with probability $1 - p$.

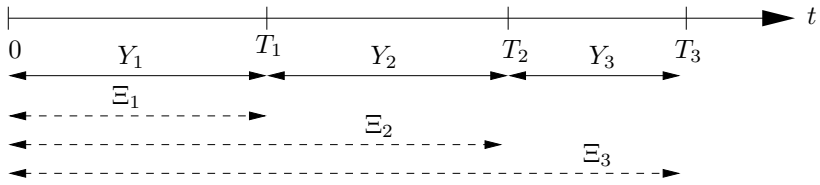


Figure 2: In imperfect repair models there are three time dimensions to measure the age of a component: Age versus calendar time T_i, age versus inter-event times Y_i, and effective age Ξ_i. The values of Ξ_i, $i > 1$, depend upon both inter-event times and maintenance history. This is indicated by dotted lines for the Ξ_i.

For simplicity of notation we follow Kijima[4] and introduce random variables D_i to denote the outcome of the repair immediately after the ith event. If we put $D_i = 0$ for a perfect repair and $D_i = 1$ for a minimal one, it follows that

$$\Xi_i = \sum_{j=1}^{i} \left(\prod_{k=j}^{i} D_k \right) Y_j \ . \tag{4}$$

The BP model with parameter p corresponds to assuming that the D_i are *i.i.d.* and independent of $Y_1, Y_2, \ldots$, with $P(D_i = 0) = p$, $P(D_i = 1) = 1 - p$, $i = 1, \ldots, n$.

BP type models have been considered by several authors, including Block *et al.*[5] who extended the model to allow the parameter p to be time varying, Kijima[4] who studied two general repair models for which BP is a special case, Hollander *et al.*[6] who studied statistical inference in the model, Dorado *et al.*[7] who proposed a more general model with BP as a special case, and most notably for the present work, Whitaker and Samaniego[8] whose results we discuss in further detail below.

Whitaker and Samaniego[8] found non-parametric maximum likelihood estimators for (p, F) in the BP model, where F is the distribution function corresponding to the hazard $\omega(\cdot)$. They noted that p is in general not identifiable if only the inter-event times Y_i are observed. The problem is related to the memoryless property of the exponential distribution, and is hardly a surprise. To ensure identifiability, Whitaker and Samaniego made strong assumptions about data availability, namely that the type of repair (minimal or perfect) is reported for each repair action (i.e.,

50	44	102	72	22	39	3	15
197	188	79	88	46	5	5	36
22	139	210	97	30	23	13	14

Table 2: Proschan's air conditioner data; inter-event times of plane 7914.

the variables D_j are actually observed). In real applications, however, exact information on the type of repair is rarely available. As we shall see in Section 4.2, identifiability of p is still possible in the model by appropriately modelling the maintenance actions.

In order to illustrate estimation in the BP model based on the Y_i alone, we consider the failure times of Plane 7914 from the air conditioner data of Proschan[9] given in Table 2. These data were also used by Whitaker and Samaniego[8]. The joint density of the observations $Y_1, \ldots, Y_n$ can be calculated as a product of conditional densities,

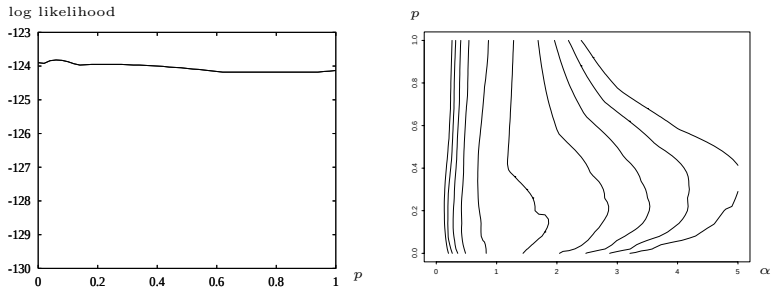
$$f(y_1, \ldots, y_n) = f(y_1) f(y_2|y_1) \cdots f(y_n|y_1, \ldots, y_{n-1}) \ .$$

For computation of the ith factor we condition on the unobserved $D_1, \ldots, D_{i-1}$, getting

$$\begin{aligned}
f(y_i \mid y_1, \ldots, y_{i-1}) &= \sum_{d_1,\ldots,d_{i-1}} f(y_i \mid y_1, \ldots, y_{i-1}, d_1, \ldots, d_{i-1}) \\
&\quad \times f(d_1, \ldots, d_{i-1} \mid y_1, \ldots, y_{i-1}) \\
&= \sum_{j=1}^{i} f(y_i \mid y_1, \ldots, y_{i-1}, d_{j-1} = 0, d_j = \cdots = d_{i-1} = 1) \\
&\quad \times P(D_{j-1} = 0, D_j = \cdots = D_{i-1} = 1) \\
&= \sum_{j=1}^{i} \omega \left(\sum_{k=j}^{i} y_k \right) e^{-\left[\Omega\left(\sum_{k=j}^{i} y_k\right) - \Omega\left(\sum_{k=j}^{i-1} y_k\right)\right]} (1-p)^{i-j} \, p^{\delta(j>1)} \ ,
\end{aligned}$$

where $\Omega(x) = \int_0^x \omega(t)dt$ is the cumulative hazard function and $\delta(j > 1)$ is 1 if $j > 1$ and 0 otherwise. The idea is to partition the set of vectors $(d_1, \ldots, d_{i-1})$ according to the number of 1s immediately preceding the ith event.

Let the cumulative hazard be given by $\Omega(x) = \mu x^\alpha$ for unknown μ and α. The profile log likelihoods of the single parameter p and the pair (α, p) are shown in Figure 3a) and Figure 3b) respectively. The maximum likelihood estimates are $\hat{\alpha} = 1.09$, $\hat{\mu} = \exp(-4.81)$, and $\hat{p} = 0.01$. However, the data contain very little information about p; this is illustrated in Figure 3a). It is seen that both $p = 0$, corresponding to an NHPP, and $p = 1$, corresponding to a Weibull renewal process are "equally" possible models here. The problem is closely connected to the problem of unidentifiability of p, noting that the maximum likelihood estimate of α is close to 1. Indeed, the exponential model with $\alpha = 1$ fixed gives the maximum log likelihood -123.86 while the maximum value in the full model (including μ, α and p) is only marginally larger, -123.78.



a) Profile log likelihood of p b) Profile log likelihood of (α, p)

Figure 3: Profile log likelihoods for the data in Table 2. Figure 3a) shows the profile likelihood of p, Figure 3b) shows the (α, p)-profile likelihood.

3.2. *Modelling preventive versus corrective maintenance*

Recall from Section 2 that PM interventions are basically periodic with some fixed period τ, but that unscheduled preventive maintenance may still be performed within a PM period, reported as degraded failures. Thus degraded failures may censor critical failures, and the two types of failure may be highly correlated.

A number of possible ways to model interaction between degraded and critical failures are discussed by Cooke[2]. We adopt one of these, called random signs censoring. In the notation introduced in Section 2 we consider here the case when we observe pairs (Y_i, J_i) where the Y_i are inter-event times whereas the J_i are indicators of failure type (critical or degraded). For a typical pair (Y, J) we let Y be the minimum of the potential critical failure time X and the potential degraded failure time Z, while $J = I(Z < X)$ is the indicator of the event $\{Z < X\}$ (assuming that $P(Z = X) = 0$ and that there are no external events). Thus we have a competing risk problem. However, while X and Z would traditionally be treated as independent, random signs censoring makes them dependent in a special way.

The basic assumption of random signs censoring is that the event of successful preventive maintenance, $\{Z < X\}$, is stochastically independent of the potential critical failure time X. In other words, the conditional probability $q(x) = P(Z < X | X = x)$ does not depend on the value of x.

Let X have hazard rate function $\omega(x)$ and cumulative hazard $\Omega(x)$. In addition to the assumption of random signs censoring, we will assume that conditionally, given $Z < X$ and $X = x$, the distribution of the intervention time Z satisfies

$$P(Z \leq z \mid X = x, Z < X) = \frac{\Omega(z)}{\Omega(x)},\ 0 \leq z \leq x\ . \tag{5}$$

To see why (5) is reasonable, consider Figure 4. When "Nature" has chosen in favour of the crew and has selected the time to critical failure, $X = x$, which the

crew will have to beat, she first draws a value u uniformly from $[0, \Omega(x)]$. Then the time for preventive maintenance is chosen as $Z = \Omega^{-1}(u)$, where $\Omega^{-1}(\cdot)$ is the inverse function of $\Omega(\cdot)$. Following this procedure makes the conditional density of Z proportional to the intensity of the underlying failure process. This seems like a coarse but somewhat reasonable description of the behaviour of a competent maintenance crew.

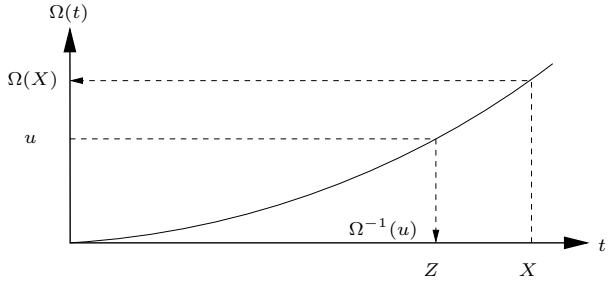


Figure 4: Time to PM conditioned on $\{Z < X, X = x\}$.

Our joint model for (X, Z) is thus defined from the following:

(*i*) X has hazard rate $\omega(\cdot)$.

(*ii*) $\{Z < X\}$ and X are stochastically independent.

(*iii*) Z given $Z < X$ and $X = x$ has distribution function (5).

These requirements determine the distribution of the observed pair (Y, J) as follows. First, by (*ii*) we get

$$\begin{aligned} P(y \le Y \le y + dy, J = 0) &= P(y \le X \le y + dy, X < Z) \\ &= (1 - q)\, \omega(y)\, \exp(-\Omega(y))\, dy \end{aligned}$$

where we introduce the parameter $q = P(Z < X)$. Next,

$$\begin{aligned} &P(y \le Y \le y + dy, J = 1) \\ &\quad = P(y \le Z \le y + dy, Z < X) \\ &\quad = \int_y^\infty P(y \le Z \le y + dy | X = x, Z < X) \\ &\qquad\qquad \times P(Z < X | X = x)\, \omega(x) \exp(-\Omega(x))\, dx \\ &\quad = q\, \omega(y)\, dy \int_y^\infty \omega(x)\, \exp(-\Omega(x)) \,/\, \Omega(x)\, dx \\ &\quad = q\, \omega(y)\, \mathrm{Ie}(\Omega(y))\, dy\ , \end{aligned}$$

where $\mathrm{Ie}(t) = \int_t^\infty \exp(-u)/u\, du$ is known as the exponential integral[10].

It is now straightforward to establish the density and distribution function of Y,

$$f_Y(y) = (1-q)\,\omega(y)\,\exp\left(-\Omega(y)\right) + q\,\omega(y)\,\mathrm{Ie}(\Omega(y)) \tag{6}$$

and

$$F_Y(y) = P(Y \le y) = 1 - \exp(-\Omega(y)) + q\,\Omega(y)\,\mathrm{Ie}(\Omega(y))\ . \tag{7}$$

Note that the proposed maintenance model introduces only one new parameter, namely q. We can interpret this parameter in terms of the alertness of the maintenance crew; a large value of q corresponds to a crew that is able to prevent a large part of the critical failures.

The distribution (6) for Y is a mixture distribution, with one component representing the failure distribution one would have without preventive maintenance, and the other mixture component being the conditional density of time for PM given that PM "beats" critical failure. It is worth noticing that the distribution with density $\omega(y)\,\mathrm{Ie}(\Omega(y))$ is stochastically smaller than the distribution with density $\omega(y)\,\exp(-\Omega(y))$; this is a general consequence of random signs censoring.

4. General model

Recall that the events in our most general setting are either critical failures, degraded failures or external events; consider Figure 2. We shall assume that corrective maintenance is always performed following a critical failure, while preventive maintenance is performed both after degraded failures and external events. Moreover, in the case of several failure mechanisms, any failure is treated as an external event for all failure mechanisms except the one failing.

4.1. *Single failure mechanism*

In this case the data for one component are $(Y_i, J_i);\ i = 1, \ldots, n$ with J_i now defined as in (2) with three possible values. Suppose for a moment that all repairs, both corrective and preventive, are perfect. Then we shall assume that the (Y_i, J_i) are *i.i.d.* observations of (Y, J) where $Y = \min(X, Z, U)$, (X, Z) is distributed as in Section 3.2, and U is the (potential) time of an external event. The U is assumed to be stochastically independent of (X, Z) and to have a distribution which does not depend on the parameters of our model. It follows that we can disregard the terms corresponding to U in the likelihood calculation. The likelihood contribution from an observation (Y, J) will therefore be as follows (see Section 3.2):

$$\begin{aligned} f(y,0) &= (1-q)\,\omega(y)\,\exp\left(-\Omega(y)\right) \\ f(y,1) &= q\,\omega(y)\,\mathrm{Ie}(\Omega(y)) \\ f(y,2) &= \exp\left(-\Omega(y)\right) - q\,\Omega(y)\,\mathrm{Ie}(\Omega(y))\ . \end{aligned} \tag{8}$$

The last expression follows from (7) and corresponds to the case where all we know is that $\max(X, Z) > y$.

To the model given above we now add imperfect repair. Recall that in the BP model there is a probability p of perfect repair ($D_i = 0$) after each event. We shall

here distinguish between preventive maintenance and corrective maintenance by letting D_i equal 0 with probability p if the ith event is a preventive maintenance or an external event, and with probability π if the ith event is a critical failure. Moreover, we shall assume that for all i we have $D_1, \ldots, D_i$ conditionally independent given $y_1, \ldots, y_i, j_1, \ldots, j_i$.

From this we are able to write down the likelihood of the data as a product of the following conditional distributions. The derivation is a straightforward extension of the one in Section 3.1.

$$\begin{aligned} f\Big((y_i, j_i) \mid (y_1, j_1), \ldots, (y_{i-1}, j_{i-1})\Big) \\ &= \sum_{d_1,\ldots,d_{i-1}} f((y_i, j_i) \mid (y_1, j_1), \ldots, (y_{i-1}, j_{i-1}), d_1, \ldots, d_{i-1}) \\ &\qquad \times f(d_1, \ldots, d_{i-1} | (y_1, j_1), \ldots, (y_{i-1}, j_{i-1})) \\ &= \sum_{j=1}^{i} f\left((y_i, j_i) \middle| \xi_{i-1} = \sum_{k=j}^{i-1} y_k \right) \\ &\qquad \times P\left(D_{j-1} = 0, D_j = \cdots = D_{i-1} = 1 | j_1, \ldots, j_{i-1}\right) \ . \end{aligned}$$

Here $P(D_{j-1} = 0, D_j = \cdots = D_{i-1} = 1 \mid j_1, \ldots, j_{i-1})$ is a simple function of p and π. Thus, what remains to be defined are the conditional densities $f\left((y_i, j_i) | \xi_{i-1}\right)$, i.e. the conditional densities of (Y_i, J_i) given that the age of the component immediately after the $(i-1)$th event is ξ_{i-1}. We shall define these to equal the conditional densities given no event in $(0, \xi_{i-1})$, of the distribution given in (8). Thus we have

$$\begin{aligned} f\left((y_i, 0) \mid \xi_{i-1}\right) &= \frac{(1-q)\,\omega(\xi_{i-1} + y_i)\, \exp(-(\Omega(\xi_{i-1} + y_i)))}{\exp(-\Omega(\xi_{i-1})) - q\,\Omega(\xi_{i-1})\,\mathrm{Ie}(\Omega(\xi_{i-1}))} \\ f\left((y_i, 1) \mid \xi_{i-1}\right) &= \frac{q\,\omega(\xi_{i-1} + y_i)\,\mathrm{Ie}(\Omega(\xi_{i-1} + y_i))}{\exp(-\Omega(\xi_{i-1})) - q\,\Omega(\xi_{i-1})\,\mathrm{Ie}(\Omega(\xi_{i-1}))} \\ f\left((y_i, 2) \mid \xi_{i-1}\right) &= \frac{\exp(-\Omega(\xi_{i-1} + y_i)) - q\,\Omega(\xi_{i-1} + y_i)\,\mathrm{Ie}(\Omega(\xi_{i-1} + y_i))}{\exp(-\Omega(\xi_{i-1})) - q\,\Omega(\xi_{i-1})\,\mathrm{Ie}(\Omega(\xi_{i-1}))} \ . \end{aligned}$$

If we have data from several independent components, the complete likelihood is given as the product of the individual likelihoods.

The model for a single failure mechanism is displayed as a directed acyclic graph[11,12] in Figure 5. Due to the imperfect repair we do not have guaranteed renewals at each event, hence we have to use a time evolving model to capture the dynamics in the system. For clarity, only time-slice r (i.e., the time between event $r-1$ and r) is shown.

4.2. *Identifiability of parameters*

The present discussion of identifiability is inspired by the corresponding discussion by Whitaker and Samaniego[8], who considered the simple BP model.

Refer again to the model of the previous subsection. We assume here that, conditional on $(Y_1, J_1), (Y_2, J_2), \ldots, (Y_{i-1}, J_{i-1})$, the (potential) time to the next

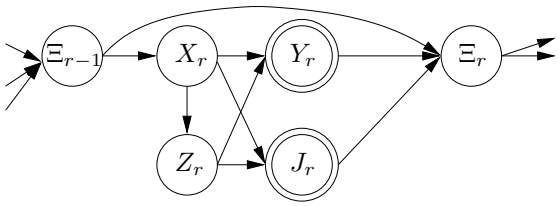


Figure 5: The model for a single failure mechanism, when only time-slice r is shown. The double-lined nodes represent the observable variables. Ξ_r is the effective age immediately after the rth repair, Ξ_r depends on Ξ_{r-1} together with what happens during the rth time-slice. X_r is the potential time to critical failure (given the history), and Z_r is the corresponding potential time to a degraded failure. Y_r is the rth inter-event time, and $J_r = I(Z_r < X_r)$.

external event is a random variable U with continuous distribution G and support on all of $(0, \tau]$ where τ as before is the regular maintenance interval. Moreover, the distribution G does not depend on the parameters of the model, and it is kept fixed in the following.

We also assume that $\omega(x) > 0$ for all $x > 0$ and that $0 < q < 1$. The parameters of the model are ω, q, p, π. These, together with G, determine a distribution of $(Y_1, J_1), \ldots, (Y_n, J_n)$ which we call $\mathcal{F}_{(\omega,q,p,\pi)}$. Here n is kept fixed.

The question of identifiability can be put as follows: Suppose

$$\mathcal{F}_{(\omega,q,p,\pi)} = \mathcal{F}_{(\omega^*,q^*,p^*,\pi^*)} \ , \tag{9}$$

which means that the two parameterizations lead to the same distribution of the observations $(Y_1, J_1), \ldots, (Y_n, J_n)$. Can we from this conclude that $\omega = \omega^*$, $q = q^*$, $p = p^*$, $\pi = \pi^*$?

First note that (9) implies that the distribution of (Y_1, J_1) is the same under the two parameterizations; $Y_1 = \min(X, Z, U)$. It is clear that each of the following two types of probabilities are the same under the two parameterizations,

$$P(x \le X \le x + dx, Z > x, U > x)$$
$$P(z \le Z \le z + dz, X > z, U > z).$$

By independence of (X, Z) and U, and since $P(U > x) > 0$ if and only if $x < \tau$, we conclude that each of the following two types of probabilities are equal under the two parameterizations,

$$P(x \le X \le x + dx, Z > x);\ x < \tau$$
$$P(z \le Z \le z + dz, X > z);\ z < \tau.$$

These probabilities can be written respectively

$$(1 - q)\, \omega(x)\, e^{-\Omega(x)}\, dx;\ x < \tau$$
$$q\, \omega(z)\, \mathrm{Ie}(\Omega(z))\, dz;\ z < \tau\ .$$

Thus, by integrating from 0 to x we conclude that (9) implies for $x \leq \tau$

$$(1-q)\left(1-e^{-\Omega(x)}\right) = (1-q^*)\left(1-e^{-\Omega^*(x)}\right) \quad (10)$$

$$q\left(1-e^{-\Omega(x)}+\Omega(x)\mathrm{Ie}(\Omega(x))\right) = q^*\left(1-e^{-\Omega^*(x)}+\Omega^*(x)\mathrm{Ie}(\Omega^*(x))\right). \quad (11)$$

We shall now see that this implies that $q = q^*$ and $\Omega(x) = \Omega^*(x)$ for all $x \leq \tau$. Suppose, for contradiction, that there is an $x_0 \leq \tau$ such that $\Omega(x_0) < \Omega^*(x_0)$. Then since both $1-\exp(-t)$ and $1-\exp(-t)+t\,\mathrm{Ie}(t)$ are strictly increasing in t, it follows from respectively (10) and (11) that $1-q > 1-q^*$ and $q > q^*$. But this is a contradiction. In the same manner we get a contradiction if $\Omega(x_0) > \Omega^*(x_0)$. Thus $\Omega(x) = \Omega^*(x)$ for all $x \leq \tau$ (so $\omega(x) = \omega^*(x)$ for all $x \leq \tau$) and hence also $q = q^*$.

We shall see below that in fact we have $\Omega(x) = \Omega^*(x)$ on the interval $(0, n\tau)$, but first we shall consider the identifiability of p and π. For this end we consider the joint distribution of $(Y_1, J_1), (Y_2, J_2)$. In the same way as already demonstrated we can disregard U in the discussion, by independence, but we need to restrict y_1, y_2 so that $y_1 + y_2 \leq \tau$. First, look at

$$\begin{aligned} &P\Big(y_1 \leq Y_1 \leq y_1 + dy_1, J_1 = 0, y_2 \leq Y_2 \leq y_2 + dy_2, J_2 = 0\Big) \quad (12)\\ &\quad = (1-q)\,\omega(y_1)\,e^{-\Omega(y_1)}\Big[\pi(1-q)\omega(y_2)e^{-\Omega(y_2)}\\ &\quad +(1-\pi)\,(1-q)\,\frac{\omega(y_1+y_2)\exp(-\Omega(y_1+y_2))}{\exp(-\Omega(y_1)) - q\,\Omega(y_1)\,\mathrm{Ie}(\Omega(y_1))}\Big]\,dy_1\,dy_2\ . \end{aligned}$$

This is a linear function of π with coefficient of π proportional to

$$\omega(y_2)\exp(-\Omega(y_2)) - \frac{\omega(y_1+y_2)\exp(-\Omega(y_1+y_2))}{\exp(-\Omega(y_1)) - q\,\Omega(y_1)\,\mathrm{Ie}(\Omega(y_1))}\ . \quad (13)$$

Using the assumption that $0 < q < 1$ we thus conclude that $\pi = \pi^*$ unless (13) equals 0 for all y_1 and y_2 with $y_1 + y_2 \leq \tau$. Making the similar computation, putting $J_2 = 1$ instead of $J_2 = 0$ in (12), we can similarly conclude that $\pi = \pi^*$ unless

$$\omega(y_2)\mathrm{Ie}(\Omega(y_2)) - \frac{\omega(y_1+y_2)\,\mathrm{Ie}(\Omega(y_1+y_2))}{\exp(-\Omega(y_1)) - q\,\Omega(y_1)\,\mathrm{Ie}(\Omega(y_1))} \quad (14)$$

equals 0 for all y_1 and y_2 with $y_1 + y_2 \leq \tau$. Now, if both (13) and (14) were 0 for all y_1 and y_2 with $y_1 + y_2 \leq \tau$, then we would necessarily have

$$\frac{\exp(-\Omega(y_2))}{\mathrm{Ie}(\Omega(y_2))} = \frac{\exp(-\Omega(y_1+y_2))}{\mathrm{Ie}(\Omega(y_1+y_2))} \quad (15)$$

for all y_1 and y_2 with $y_1 + y_2 \leq \tau$. Since we have assumed that $\omega(\cdot)$ is strictly positive, (15) would imply that $\exp(-t)/\mathrm{Ie}(t)$ is constant for t in some interval (a, b). This is of course impossible by the definition of $\mathrm{Ie}(\cdot)$, and it follows that not both of (13) and (14) can be identically zero. Hence π is identifiable.

Identifiability of p is concluded in the same way by putting $J_1 = 1$ instead of $J_1 = 0$ in (12).

So far we have concluded equality of the parameters q, p, π under the two parameterizations, while we have concluded that $\Omega(x) = \Omega^*(x)$ for all $x \leq \tau$. But then, putting $y_1 = \tau$ in (12), while letting y_2 run from 0 to τ, it follows that $\Omega(x) = \Omega^*(x)$ also for all $\tau < x \leq 2\tau$. By continuing we can eventually conclude that $\Omega(x) = \Omega^*(x)$ for all $0 < x \leq n\tau$.

If $\tau = \infty$, then of course the whole function $\omega(\cdot)$ is identifiable. However, even if $\tau < \infty$ we may have identifiability of all of $\omega(\cdot)$. For example, suppose $\Omega(x) = \mu x^\alpha$ with μ, α positive parameters. Then the parameters are identifiable since (10) in this case implies that

$$\mu x^\alpha = \mu^* x^{\alpha^*}$$

for all $x \leq \tau$. This clearly implies the pairwise equality of the parameters.

4.3. *Several failure mechanisms*

We now look at how to extend the model of Section 4.2 to $k > 1$ failure mechanisms and data given as in (1) or (3).

Our basic assumption is that the different failure mechanisms $M_1, \ldots, M_k$ act independently on the component. More precisely we let the complete likelihood for the data be given as the product of the likelihoods for each failure mechanism. Note that the set of events is the same for all failure mechanisms, and that failure due to one failure mechanism is treated as an external event for the other failure mechanisms.

The above assumption implies a kind of independence of the maintenance for each failure mechanism. Essentially we assume that the pairs (X, Z) are independent across failure mechanisms. This is appropriate if there are different maintenance crews connected to each failure mechanisms, or could otherwise mean that the "signals" of degradation emitted from the component are independent across failure mechanisms.

Another way of interpreting our assumption is that, conditional on

$$(y_1, k_1, j_1), \ldots, (y_{i-1}, k_{i-1}, j_{i-1})$$

the next vector (Y_i, K_i, J_i) corresponds to a competing risk situation involving m independent risks, one for each failure mechanism, and each with properties as for the model given in Section 4.1.

The parameters (ω, q, p, π) may (and will) in general depend on the failure mechanism. As regards identifiability of parameters, this will follow from the results for single failure mechanisms of Section 4.2 by the assumed independence of failure mechanisms.

If we have data from several independent components of the same kind, given as in (3), then the complete likelihood is given as the product of the likelihoods for each component.

Figure 6 depicts the complete model for time-slice r represented by a directed acyclic graph, confer also Figure 5.

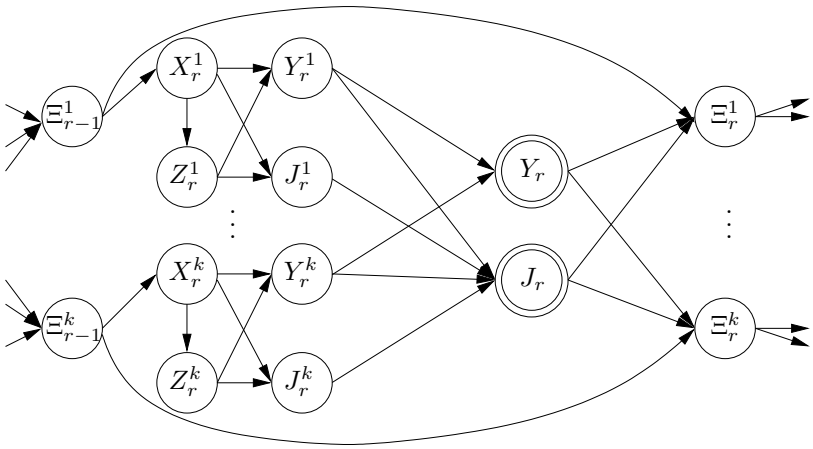


Figure 6: The complete model, but only showing time-slice r. The random variables are given a subscript index indicating the time-slice, and a superscript index showing the failure mechanism. For example, Ξ_r^m is the effective age of the m'th failure mechanism immediately after the r'th event. Only nodes drawn with double-line are observed.

	Deformation	Leakage	Breakage	Other
# Critical failures	4	1	1	2
# Degraded failures	8	2	0	4

Table 3: Number of failures per failure mechanism.

5. Parameter estimation

5.1. *Calculation scheme*

The complete model as described in Section 4 involves some important conditional independence properties that both special purpose maximum-likelihood estimator algorithms as well as Markov Chain Monte Carlo simulations can benefit from. In this section we have used maximum likelihood methods.

5.2. *A case study*

To exemplify the merits of the proposed model, we use Phase IV of the Gas Turbine dataset from the Offshore Reliability Database[13]. Only the Gas Generator subsystem is included in the study. We analyse data from a single offshore installation to ensure maximum homogeneity of the data sample. The dataset consists of 23 mechanical components, which are followed over a total of 603.690 operating hours. There are 22 failures, out of which 8 are classified as critical and 14 as degraded. The failures are distributed over four different failure mechanisms (so $k = 4$), namely deformation, leakage, breakage and other mechanical failure.

	Deformation	Leakage	Breakage	Other
Hazard (μ_m)	$2.5 \cdot 10^{-6}$	$1.3 \cdot 10^{-5}$	$8.3 \cdot 10^{-7}$	$5.6 \cdot 10^{-6}$
Preventive maint. (p_m)	0.6	0.3	1.0	0.8
Corrective maint. (p_m^κ)	1.0	1.0	1.0	1.0

Table 4: Estimated hazard rate and probability of successful maintenance.

	Deformation	Leakage	Breakage	Other
$\widehat{MTTFF}_{\text{Naked}}$	$4.0 \cdot 10^5$	$7.7 \cdot 10^4$	$1.2 \cdot 10^6$	$1.8 \cdot 10^5$
$\widehat{MTTFF}_{\text{OFR}}$	$6.0 \cdot 10^5$	$1.5 \cdot 10^5$	$6.0 \cdot 10^5$	$3.0 \cdot 10^5$

Table 5: Estimated MTTFF in our model and the "observed failure rate" model.

The PM history for the gas turbines consists of 78 PM events. The PM intervals ("τ") for the different components vary between 8 and 12 calendar months.

5.3. *Results*

The data can be put on the form (3) so the complete likelihood can be calculated as described in Section 4. Having a small number of critical failures, the estimates of $\pi_1, \ldots, \pi_4$ will not be reliable; the number of critical failures is simply too small. To reduce the number of parameters we introduce $\kappa > 0$ defined so that $\pi_m = p_m^\kappa$ for $m = 1, \ldots, 4$. Here κ indicates the difference between the effect of preventive and corrective maintenance. A small value of κ means that corrective maintenance is much more beneficial than the preventive, and a value close to 1 judges the two maintenance operations about equal. In the same way, we assume that $q_1 = \cdots = q_4$, and use q to denote these variables.

We also use a simple parametric forms of the $\omega_i(\cdot)$, namely the constant hazards $\omega_i(t) = \mu_i,\ i = 1, \ldots, 4$. The results of maximum likelihood estimation are presented in Table 4. The estimated value of q is $\hat{q} = .4$, while $\hat{\kappa} = 1 \cdot 10^{-2}$. The latter value indicates that corrective maintenance actions are highly effective.

It is also interesting to calculate the mean time to first failure ($MTTFF$) had there been no maintenance. This value, which we name $MTTFF_{\text{Naked}}$, shows the nature of the underlying failure process unbiased by the maintenance regime; it can be estimated directly by $1/\hat{\mu}_i$ in the present setting. In Table 5 we compare $\widehat{MTTFF}_{\text{Naked}}$ to the "observed failure rate" estimators given by

$$\widehat{MTTFF}_{\text{OFR}} = \frac{\#\text{Total Operating Time}}{\#\text{Critical Failures}}$$

to see the effect of including maintenance in the model.

It is worth noticing that the OFR-estimates are inclined to be more optimistic than the estimators from our model. This is because degraded failures tend to censor potential critical failures, and this influences the OFR-estimate.

6. Concluding remarks

In this paper we have proposed a simple but flexible model for maintained components which are subject to a variety of failure mechanisms. The proposed model has the standard models of perfect and minimal repair as special cases. Moreover, some of the parameters we estimate (namely p_m, π_m and q_m) can be used to examine the sufficiency of these smaller models. "Small" values of $\hat{q}_m$ accompanied by "extreme" values of all $\hat{p}_m$ and $\hat{\pi}_m$ (either "close" to one or zero) indicate that reduced models are detailed enough to capture the main effects in the data. Making specific model assumptions regarding the preventive maintenance we are able to prove identifiability of all parameters.

We note that many models simpler than ours may be useful if explicit notion of maintenance quality is considered unimportant[14,15,16]. In our experience, the model of Lawless and Thiagarajah[17],

$$\lambda(t \mid \mathcal{F}_{t^-}) = \exp\left(\alpha + \beta\, g_1(t) + \gamma\, g_2(t - T_{N(t^-)}\right), \tag{16}$$

where α, β and γ are unknown parameters, and g_1 and g_2 are known functions, offers good predictive ability in the setting corresponding to Section 3.2. Observe that the conditional intensity in (16) depends both on the age t and the time since last failure $t - T_{N(t^-)}$; hence it can be considered to be an imperfect repair model with perfect and minimal repair as special cases. However, the model is difficult to interpret with respect to the physical meaning of the parameters, and is therefore not satisfactory in our more general setting. Our motivation has been to build a model that could be used to estimate the effect of maintenance, where "effect" has been connected to the model parameters q_m, p_m and π_m. Here q_m is indicative of the crew's eagerness, their ability to perform maintenance at the correct times to try to stop evolving failures. The p_m and π_m indicate the crew's thoroughness; their ability to actually stop the failure development. The proposed model indirectly estimates the naked failure rate, and on a specific case using real life data these estimates are significantly different from those found by "traditional" models.

We make modest demands regarding data availability: Only the inter-failure times and the failure mechanisms leading to the failure accompanied by the preventive maintenance program are required. This information is available in most modern reliability data banks.

Acknowledgements

We would like to thank Tim J. Bedford for an interesting conversation about the model for PM versus critical failures and Roger M. Cooke for discussions regarding the applicability of random signs censoring with respect to the OREDA data. Previous short versions of this manuscript[18,19] were presented at the conferences ESREL'99 and MMR'02. The first author was supported by a grant from the Research Council of Norway.

References

1. M. Brown and F. Proschan. Imperfect repair. *Journal of Applied Probability*, 20:851–859, 1983.
2. R. M. Cooke. The design of reliability data bases, Part I and Part II. *Reliability Engineering and System Safety*, 52:137–146 and 209–223, 1996.
3. P. Andersen, Ø. Borgan, R. Gill, and N. Keiding. *Statistical models based on counting processes*. Springer, New York, 1992.
4. M. Kijima. Some results for repairable systems with general repair. *Journal of Applied Probability*, 26:89–102, 1989.
5. H. Block, W. Borges, and T. Savits. Age dependent minimal repair. *Journal of Applied Probability*, 22:370–385, 1985.
6. M. Hollander, B. Presnell, and J. Sethuraman. Nonparametric methods for imperfect repair models. *Annals of Statistics*, 20:879–896, 1992.
7. C. Dorado, M. Hollander, and J. Sethuraman. Nonparametric estimation for a general repair model. *Annals of Statistics*, 25:1140–1160, 1997.
8. L. R. Whitaker and F. J. Samaniego. Estimating the reliability of systems subject to imperfect repair. *Journal of American Statistical Association*, 84:301–309, 1989.
9. F. Proschan. Theoretical explanation of observed decreasing failure rate. *Technometrics*, 5:375–383, 1963.
10. M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover Publ., New York, 1965.
11. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA., 1988.
12. F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer Verlag, New York, 2001.
13. OREDA. *Offshore Reliability Data*. Distributed by Det Norske Veritas, P.O. Box 300, N-1322 Høvik, 3rd edition, 1997.
14. H. Pham and H. Wang. Multivariate imperfect repair. *European Journal of Operations Research*, 94:425–428, 1996.
15. P. A. Akersten. Imperfect repair models. In S. Lydersen, G. K. Hansen, and H. A. Sandtorv, editors, *Proceedings of the ninth European Conference on Safety and Reliability – ESREL'98*, pages 369–372, Rotterdam, 1998. A. A. Balkema.
16. B. H. Lindqvist. Repairable systems with general repair. In G. I. Schuëller and P. Kafka, editors, *Proceedings of the tenth European Conference on Safety and Reliability – ESREL'99*, pages 43–48, München, Germany, 1999. A. A. Balkema.
17. J. F. Lawless and K. Thiagarajah. A point process model incorporating renewals and time trends, with applications to repairable systems. *Technometrics*, 38:131–138, 1996.
18. H. Langseth. Modelling maintenance for components under competing risk. In G. I. Schuëller and P. Kafka, editors, *Proceedings of the tenth European Conference on Safety and Reliability – ESREL'99*, pages 179–184, München, Germany, 1999. A. A. Balkema.
19. H. Langseth and B. H. Lindqvist. Modelling imperfect maintenance and repair of components under competing risk. In H. Langseth and B. H. Lindqvist, editors, *Communuications of the Third International Conference on Mathematical Methods in Reliability – Methodology and Practice*, page 359, Trondheim, Norway, 2002.

II

Decision Theoretic Troubleshooting of Coherent Systems

Decision Theoretic Troubleshooting of Coherent Systems

Helge Langseth [1] and Finn V. Jensen

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

Abstract

We present an approach to efficiently generating an inspection strategy for fault diagnosis. We extend the traditional troubleshooting framework to model non-perfect repair actions, and we include questions. Questions are troubleshooting steps that do not aim at repairing the device, but merely are performed to capture information about the failed equipment, and thereby ease the identification and repair of the fault. We show how Vesely and Fussell's measure of component importance extends to this situation, and focus on its applicability to compare troubleshooting steps. We give an approximate algorithm for generating a "good" troubleshooting strategy in cases where the assumptions underlying Vesely and Fussell's component importance are violated, and discuss how to incorporate questions into this troubleshooting strategy. Finally, we utilize certain properties of the domain to propose a fast calculation scheme.

Key words: Repair strategies, Bayesian networks, fault diagnosis, Vesely and Fussell component importance.

Email addresses: helgel@math.ntnu.no (Helge Langseth), fvj@cs.auc.dk (Finn V. Jensen).

[1] Current affiliation: Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway.

To appear in Reliability Engineering and System Safety

1 Introduction

This paper describes a troubleshooting system which has been developed in the SACSO [2] project, and which is partly implemented in the BATS [3] tool. This is a troubleshooting (TS) system for performing efficient troubleshooting of electro-mechanical equipment, and it is currently employed in the printer domain. It is important to notice that the BATS tool is created to offer *printer users* a web-based interface to a decision-theoretic TS-system; it is not intended exclusively for maintenance personnel who are trained to handle the equipment that is to be repaired. The goal is that any user, however inexperienced, should be able to repair the failed equipment on his own instead of relying on professional help. By design the TS-system we describe therefore differs from other TS-systems (see e.g. [1–7]) in several aspects. Most importantly the users of the TS-system may be inexperienced with handling and repairing the failed equipment. Hence, they may fail to repair broken components, e.g., by seating a new network card incorrectly. Furthermore, this may even happen without the user realizing the mistake. It is therefore crucial for the TS-system to explicitly include the possibility that users perform prescribed repair actions incorrectly in the TS-model.

Secondly, the users are expected to have limited knowledge about (and interest in) the design of the malfunctioning equipment. They cannot be expected to be interested in finding the *cause* of a problem; they merely want to *repair* it. Focusing on the identification of the faulty minimal cutset, as in [4–7], is therefore not expected to be relevant for the foreseen group of users. The troubleshooting will thus be terminated as soon as the equipment is repaired; that is, we assume that the user is satisfied with a *minimal repair* of the failed equipment. *Perfect repair* is not necessarily accomplished by using our TS-system (and not by the methods in [4–7] either), but may be considered using other means.

Finally, as the faulty device can be located under a variety of external conditions, the TS-system can pose *questions* to the user in order to survey the faulty equipment's surroundings. Although these questions initially increase the cost of the troubleshooting, they may shed light on the situation, and ultimately decrease the overall cost of repairing the equipment.

To formalize, let the faulty equipment consist of K components $\mathcal{X} = \{X_1, \ldots, X_K\}$. Each component is either *faulty* ($X_i = \mathsf{faulty}$) or *operating* ($X_i = \mathsf{ok}$), and as the status of each component is unknown to the TS-system when the

[2] The SACSO (Systems for Automated Customer Support Operations) project constitutes joint work between the Research Unit for Decision Support Systems at Aalborg University and Customer Support R&D at Hewlett-Packard.

[3] BATS (Bayesian Automated Troubleshooting System) is available from DEZIDE over the internet: `http://www.dezide.dk/`.

troubleshooting starts, $\mathcal{X}$ is considered a set of random variables. The equipment consists of R Minimal Cut Sets (MCSs), and we use $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \ldots, \mathcal{C}_R\}$ for the collection of these. A MCS is *faulty* ($\mathcal{C}_i = \mathsf{faulty}$) if all its members are faulty. Otherwise it is *operating* ($\mathcal{C}_i = \mathsf{ok}$). The equipment is assumed to be faulty at the time when troubleshooting starts; troubleshooting is terminated as soon as the equipment is brought back to operating modus. We will assume that only one MCS is in its faulty state, and use $\mathcal{C}_F$ to denote the faulty MCS (named *the actual MCS* in [4]). This assumption is common for most TS-systems, and it is usually justified by considering systems that are used almost continuously, and thus (like a printer) tested frequently. It is unlikely that several components should fail approximately simultaneously. Common cause failures (due to, e.g., stroke of lightning, pouring coffee into the printer, etc.) are easily detected, and are handled separately. Note that if more than one MCS is faulty the proposed method will still repair the equipment, although not necessarily in an optimal fashion. [4] The TS-system may choose from a set of N possible actions and will ask the user to perform (some of) them to remedy the problem. The outcome of the actions are modelled by a set of random variables $\mathcal{A} = \{A_1, \ldots, A_N\}$ in the TS-system. There are also M predefined questions that the TS-system may pose to the user; the answers to these questions are modelled by the random variables $\mathcal{Q} = \{Q_1, \ldots, Q_M\}$. Since there is a one-to-one mapping between actions the TS-system can ask the user to perform and the random variables in the set $\mathcal{A}$, we will refer to $\mathcal{A}$ as the set of actions and say "perform $A \in \mathcal{A}$" when we really mean that the action associated with A is executed. In the same manner we call $\mathcal{Q}$ the questions, and use the term "pose the question $Q \in \mathcal{Q}$" when strictly speaking a question is posed, and the random variable Q models the *answer* to that question. A *TS-step* is a step in a TS-strategy, either a repair step (termed action) or an information-gathering step (termed question). To each TS-step B_i the associated cost is denoted by C_i. The system is informed about the outcome of each TS-step after it has been performed (i.e., the state of the associated random variable is observed).

The goal of a TS-system is to provide a "good" *TS-strategy*. Formally, a TS-strategy $\mathcal{S}$ is an ordering of TS-steps, such that new TS-steps are prescribed until the equipment is repaired or all steps have been performed. The ordering of steps may depend on the outcome of the steps already performed in the

[4] Srinivas [8] presents a modified algorithm to handle troubleshooting in serial systems where more than one component (and hence more than one MCS) may be faulty; it turns out that optimal troubleshooting in this case requires a balance between cost and probability of successful repair that is different from what is optimal in our situation. When presented with a system where more than one MCS is in its faulty state, our system may thus perform sub-optimally: The user may be asked to perform the repair in a way more expensive than had been required if we had not made this assumption.

strategy (there is for instance no need to examine the network connection if the printer test-page is printed correctly). Any TS-strategy can be represented by a *strategy tree*, see Fig. 1 for an example. The internal nodes in the strategy tree (depicted as ovals) represent chance nodes; TS-steps that we do not know the outcome of initially. Each possible outcome of a chance node corresponds to a unique sub-tree in the strategy tree, which is found by selecting the edge labelled with that particular outcome. The TS-strategy depicted in Fig. 1 starts by posing Q_S, and if the answer is $Q_S = \mathsf{yes}$, the TS-strategy prescribes to perform action A_2; if $Q_S = \mathsf{no}$, then the question Q_K should be posed. The terminal nodes (depicted as diamonds) signify that the troubleshooting strategy has ended, either because the problem is solved or because the set of actions has been exhausted. Note that this is a simple example, where we assume only two possible answers (yes and no) to the questions Q_S and Q_K.

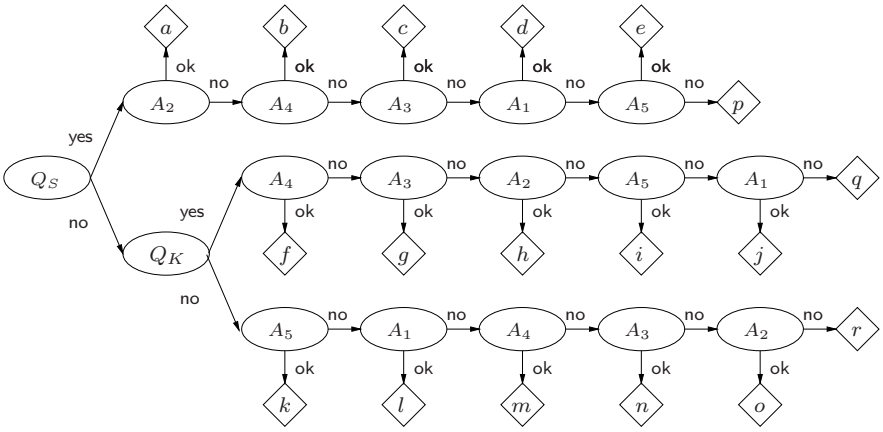


Fig. 1. A TS-strategy represented by a strategy tree; $\mathcal{A} = \{A_1, \ldots, A_5\}$ and $\mathcal{Q} = \{Q_S, Q_K\}$.

How "good" a TS-strategy is, is judged by its *expected cost of repair*, ECR, which is a function of the C_i. This is in accordance with the decision-theoretic formulation of the troubleshooting task: One should balance the cost of a TS-step with the likelihood of the step to be beneficial, so that the optimal TS-strategy can be found [3]. Breese and Heckerman [9] used Bayesian networks to model the troubleshooting domain, and Jensen et al. [10] report extensions to that framework. In [9,10] the domains under study were restricted to be serial systems, i.e., systems where all cutsets were singletons. In this paper we will extend these frameworks to work with any coherent system (represented by its cutsets). Finding the strategy which minimizes the ECR is NP-hard in general [11], so we our goal will be to approximate the optimal strategy.

The printer industry spends millions of dollars every year on customers support; mainly to provide telephone-support and on-site troubleshooting. This has sparked an interest for building automated troubleshooting systems which

can resolve some of the printer users' problems without requiring support from call agents. A printing system consists of several components: The application from which the printing command is sent, the printer driver, the network connection, the server controlling the printer, the printer itself, etc. It typically has about 40 different failure-modes, e.g., *Light print*. Each failure-mode can be caused by several component failures, and we have one TS-system for each of them.[5] The typical size of these TS-models is about 30 actions and 15 questions. We will not describe the printer model in further detail, as the TS-system we propose is general in nature; the interested reader is referred to [10,12].

The rest of the paper is outlined as follows: In Section 2 we describe the basic system model, and the formal language used to describe it. Section 3 is devoted to how the TS-system sequences actions, and handling of questions are described in Section 4. The calculation scheme is described in detail in Section 5, and we conclude in Section 6.

2 The troubleshooting model

In this section we will describe the troubleshooting model, and in particular focus on the modelling assumptions that we make. To do so, we start by introducing Bayesian networks (BNs), which constitute the representation language we employ. We then give a detailed description of how we generate a BN-representation of the troubleshooting domain.

2.1 Bayesian networks

Our system represents the TS-domain by a Bayesian network [13,14]. BNs have a long history of usage in the reliability and safety sciences, ranging from the early works [15,16] to the more recent contributions, see, e.g., [8–12,17–20]. BNs offer a flexible language to describe the TS-model, and we utilize this to make a realistic model of the interactions one can have with the failed equipment; specifically we can define repair steps including non-perfect repair, as well as information-gathering steps.

A Bayesian network over the discrete random variables $\boldsymbol{X}$ is a compact representation of the probability mass function $P(\boldsymbol{X} = \boldsymbol{x})$. A BN consists of a qualitative part; a directed acyclic graph, and a quantitative part; a set of conditional probability tables. More formally, a Bayesian network representing the probability mass function of a stochastic vector $\boldsymbol{X}$ is a 2-tuple $\left(\mathcal{G}, \boldsymbol{\Theta}_{\mathcal{G}}\right)$. $\mathcal{G}$ is a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes in the graph and $\mathcal{E}$ is the set of directed edges. There is a bijection between $\boldsymbol{X}$ and

[5] The first information the user enters into the system is the failure-mode he wants to troubleshoot. If some failure-modes are not easily distinguishable we have joined them into one TS-model.

$\mathcal{V}$, and the edges are used to represent dependence between the variables. In the TS-domain we only work with *discrete* BNs, where each node $V \in \mathcal{V}$ takes on values from a finite state-space denoted $\mathrm{sp}(V)$. We define the *parent set of* V, $\mathrm{pa}(V)$, as the set of nodes having outgoing edges directed into V. The graph is associated with the probability distributions $\boldsymbol{\Theta}_{\mathcal{G}}$ by letting each node $V \in \mathcal{V}$ be labelled with a conditional probability table $P(V \,|\, \mathrm{pa}(V))$. The full joint distribution over the variables $\mathcal{V}$ (and hence of $\boldsymbol{X}$) can now be calculated as $P(\mathcal{V}) = \prod_{V \in \mathcal{V}} P(V \,|\, \mathrm{pa}(V))$.

The essential property of the distribution function that is utilized in the BN representation of $P(\boldsymbol{X} = \boldsymbol{x})$ is the set of *conditional independencies* encoded in the distribution function: If $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ are vectors of random variables with joint probability distribution $P(\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{W})$, then we say that $\boldsymbol{Y}$ is conditionally independent of $\boldsymbol{Z}$ given $\boldsymbol{W}$, written $\boldsymbol{Y} \perp\!\!\!\perp \boldsymbol{Z} \,|\, \boldsymbol{W}$, if $P(\boldsymbol{Y} \,|\, \boldsymbol{Z}, \boldsymbol{W} = \boldsymbol{w}) = P(\boldsymbol{Y} \,|\, \boldsymbol{W} = \boldsymbol{w})$ for all $\boldsymbol{w}$ where $P(\boldsymbol{W} = \boldsymbol{w}) > 0$. If $\boldsymbol{Y} \perp\!\!\!\perp \boldsymbol{Z} \,|\, \{\emptyset\}$, then $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are (marginally) independent (written $\boldsymbol{Y} \perp\!\!\!\perp \boldsymbol{Z}$ for short).

An example of conditional independence from our domain is as follows. If the toner is low, then this can be detected in at least two ways: *i*) There may be an error message on the control panel, and *ii*) the last page may be printed lightly. There is a slight possibility for the error message not to show up, and for the last page not be visibly light-printed, even when the toner is low. If we learn that the last page was printed lightly, we may assume this was because the toner is low, and that will in turn increase our belief in finding the error message on the control panel; hence these two events are not (marginally) independent. On the other hand, if we know that the toner is low, then information about a message on the control panel will not change our belief regarding the last page being light-printed. The two events are conditionally independent given the toner's status.

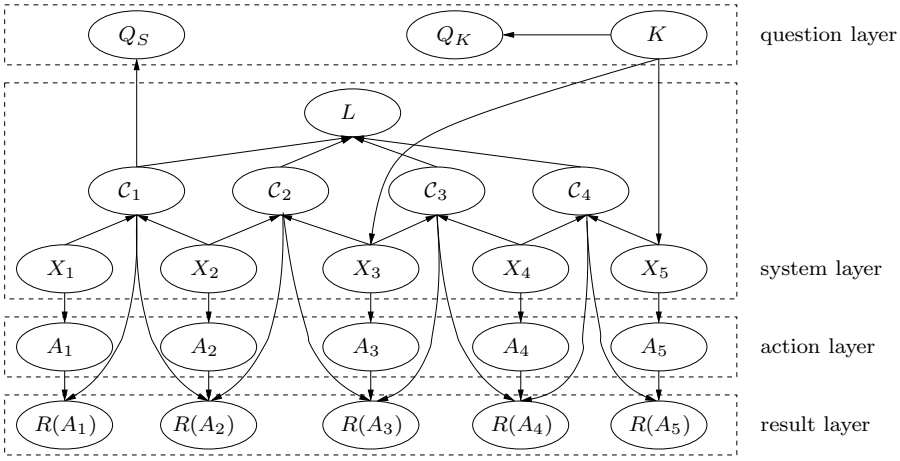


Fig. 2. The BN representation of the example model. Note that this model is extremely simple; more complex models in which, e.g., an action A_i can repair more than one component can easily be defined.

2.2 The basic troubleshooting model

The faulty equipment and the effect of interactions between the repair personnel and this equipment are modelled in a BN. As our starting point we use a BN model of the system generated from the MCS representation (see [20] for how this translation can be done). This part of the BN is denoted the *system layer* in Fig. 2; the system layer is the part of the BN that mimics the fault tree in Fig. 3. Note that we have introduced a *constraint node*[6] L to enforce the assumption that exactly one MCS is in its faulty state. All calculations are performed conditional on this assumption, and we will for simplicity of exposition not mention this conditioning explicitly in the following. Next, the MCSs are modelled by logical functions, such that $\mathcal{C}_i = \mathsf{faulty}$ if and only if all the components in the MCS are in the faulty state. Hence, $\mathrm{pa}(\mathcal{C}_i)$ are exactly those components that are members of the cutset $\mathcal{C}_i$, and $P(\mathcal{C}_i \,|\, \mathrm{pa}(\mathcal{C}_i))$ is used to encode this deterministic relationship. Note that the cutset nodes of the system layer are not really required to encode the equipment model; the probabilistic relationship could have been encoded in the constraint node L. There are however at least two reasons to include the cutset nodes in the model: Firstly, reliability engineers are used to working with the notion of cutsets, and including the cutsets explicitly makes the model more understandable and easier to build. Secondly, including the cutset nodes in the model typically makes the overall model more compact (i.e., the total number of required parameters is reduced).[7] $P(X_\ell = \mathsf{faulty})$ is given as the *a priori* probability for the component to have failed, i.e., the probability unconditioned on the equipment failure. After a *propagation* in the Bayesian network (see [21] for a description of how this is done) the posterior probability for a component failure given that the equipment is faulty (enforced by using the constraint node) can be read off the node representing that component in the BN, and the probability for each MCS to be the actual MCS can be found in the corresponding nodes.

The system model is extended by an explicit model of the effect of the interaction between the equipment and the repair personnel. These interactions are limited to the predefined sets of actions $\mathcal{A}$ and questions $\mathcal{Q}$. First, we look

[6] A constraint node is a node which is used to enforce other variables into specific configurations. In the example model we use L to enforce that exactly one cutset is faulty. This is done by defining $L = \mathsf{yes}$ if exactly one of the cutsets is faulty and $L = \mathsf{no}$ otherwise. The evidence $\{L = \mathsf{yes}\}$ is entered into the system before the calculations are performed, and the cutset nodes are thereby constrained s.t. the MCS assumption is fulfilled.

[7] If we choose not to include the cutset nodes in the BN representation we can, at the cost of a larger model, relax the binary system-model we employ. This can be utilized to create multi-state systems to, e.g., model "degrees of failure". We have nevertheless chosen to work with the MCS representation, primarily to render the fast calculation scheme of Section 5 possible.

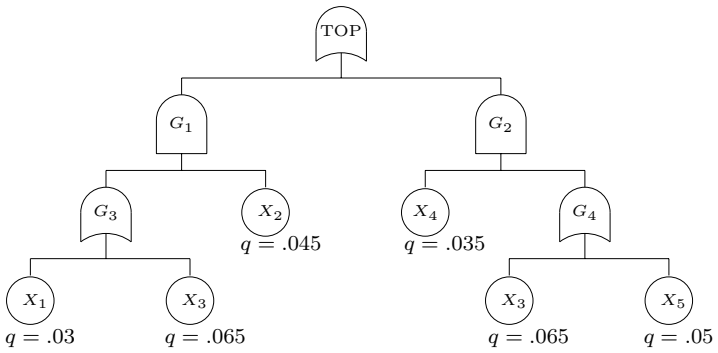


Fig. 3. A fault tree describing our example model.

at how the actions are modelled (see the *action layer* in Fig. 2).

Actions are connected to the system layer by making them children of the components they can repair, that is, $\mathrm{pa}(A_i) \subseteq \mathcal{X}$. We explicitly describe the joint effect an action A has on all the components it can repair. This is done by extending the state space of A. For the state space we use the notation $+rX$ for the event that A repairs X and $-rX$ otherwise; note that this notation is unconditioned on the state of X. For an example see Fig. 4, where action A can repair the components X_k and X_ℓ. Then, $\mathrm{pa}(A) = \{X_k, X_\ell\}$, and the state-space of A is $\mathrm{sp}(A) = \{+rX_k + rX_\ell, +rX_k - rX_\ell, -rX_k + rX_\ell, -rX_k - rX_\ell\}$. Without referring to $\mathrm{sp}(A)$ we use the notation $\{A^{\downarrow X} = \mathsf{yes}\}$ for the event that A repairs X, and $\{A^{\downarrow X} = \mathsf{no}\}$ otherwise. Thus, in the current example the shorthand $\{A^{\downarrow X_k} = \mathsf{yes}\}$ denotes the event $\{A = +rX_k + rX_\ell \vee A = +rX_k - rX_\ell\}$.

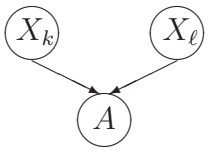


Fig. 4. Action A can repair both X_k and X_ℓ.

We make a number of assumptions about the TS-domain. Some are made to simplify the model definition, whereas others turn out to be beneficial when we perform calculations in the BN:

- We disregard component failure induced by troubleshooting personnel;[8] note that this is related to the assumption that only one MCS is faulty.
- By construction of the model it is made sure that an action only can repair components in its parent set, $P(A^{\downarrow X} = \mathsf{yes} \,|\, X = \mathsf{faulty}) = 0$ whenever $X \notin \mathrm{pa}(A)$.

[8] In the models underlying the BATS tool we have increased the cost of an action to partly reflect the *risk* of performing it. If the probability of introducing new component failures into the domain is high, then the risk is high, and the cost will be increased to reflect this potential danger.

- The state of a component X_ℓ does not influence the user's ability to repair component X_k, $A^{\downarrow X_k} \perp\!\!\!\perp X_\ell \,|\, X_k$, $k \neq \ell$. That is, we assume for instance that it is not more difficult to replace an MIO card when the toner cartridge is faulty than it would have been had the toner cartridge been operating.
- If we were to receive information about a user's failure to perform one repair action, then this would not influence our beliefs about his ability to perform other actions. Thus, when the user fails to repair some component we assume it is due to "bad luck" and not "clumsiness". Formally we write $A_i^{\downarrow X_k} \perp\!\!\!\perp A_j^{\downarrow X_\ell} \,|\, \{X_k, X_\ell\}$ whenever $i \neq j$. This assumption can be problematic if the group of users in is not homogeneous, in which case it may be beneficial to infer if the user is "skilled" or not. In real-world applications, where we typically have "novice" and "expert" users, it can be beneficial to maintain two TS-systems; one for the "novices" and one for the "experts".
- We use the convention that an action cannot repair a component that is already operating, $P(A^{\downarrow X} = \mathsf{yes} \,|\, X = \mathsf{ok}) = 0$. This may seem counterintuitive, but we use $A^{\downarrow X} = \mathsf{yes}$ to denote the event that the user has *improved* the system, it is not used to describe the *state* of the system.

These assumptions suffice for the TS-system to be operational, and for the calculation scheme (Section 5) to work. For simplicity we may also make the additional assumption that $A^{\downarrow X_k} \perp\!\!\!\perp A^{\downarrow X_\ell} \,|\, \{X_k, X_\ell\}$ whenever $k \neq \ell$. This means that a conditional probability $P(A \,|\, \mathrm{pa}(A))$ is fully specified by the collection of probabilities $\{P(A^{\downarrow X_k} = \mathsf{yes} \mid X_k = \mathsf{faulty}) : X_k \in \mathrm{pa}(A)\}$; this is often referred to as *independence of causal influence* [22]. Hence, if A can repair t components, then it is enough to enter only t conditional probabilities to describe $P(A \,|\, \mathrm{pa}(A))$. This should be compared to the 2^t numbers needed if this independence assumptions had not been made. Note that we do not require the repair actions to be perfect; non-perfect repair is modelled by $P(A^{\downarrow X_\ell} = \mathsf{yes} \,|\, X_\ell = \mathsf{faulty}) = \gamma$, $0 \leq \gamma < 1$.

There is an important difference between what is modelled in the action layer and what is actually observed. The action layer describes the events $\{A^{\downarrow X} = \mathsf{yes} \,|\, X = \mathsf{faulty}\}$, however, we may only observe whether the equipment is repaired or not, i.e., if the event $\{A^{\downarrow X} = \mathsf{yes} \wedge X \in \mathcal{C}_F\}$ occurs. To be able to work with the actual observations as evidence, we extend the model with a *result layer* consisting of a set of nodes $R(A)$, one for each $A \in \mathcal{A}$. $R(A)$, the *result of* A at the system level, is defined as $R(A) = \mathsf{ok}$ if $A^{\downarrow X} = \mathsf{yes}$ for some $X \in \mathcal{C}_F$ and $R(A) = \mathsf{no}$ otherwise.

The probability that action A repairs the equipment, $P(R(A) = \mathsf{ok})$, naturally extends Vesely and Fussell's measure of component importance [9] [23] when A

[9] $I^{\mathrm{VF}}(X)$ is defined as the probability that at least one minimal cutset which contains component X is faulty, given that the system is faulty. Under the assumptions in this paper this is simply $I^{\mathrm{VF}}(X) = P(X \in \mathcal{C}_F)$.

can repair only one component X. Let $I^{\mathrm{VF}}(X)$ be defined as the probability for X to be *critical*, i.e., $X \in \mathcal{C}_F$, given that the equipment is faulty. Then

$$\begin{aligned} P(R(A) = \mathsf{ok}) &= P(X \in \mathcal{C}_F) \cdot P(A^{\downarrow X} = \mathsf{yes} \,|\, X = \mathsf{faulty}) \\ &= I^{\mathrm{VF}}(X) \cdot P(A^{\downarrow X} = \mathsf{yes} \,|\, X = \mathsf{faulty}). \end{aligned} \tag{1}$$

When A can repair a set of components, we have (with a slight abuse of notation):

$$\begin{aligned} P(R(A) = \mathsf{ok}) &= P\left(\bigvee_{X \in \mathcal{X}} \left\{ A^{\downarrow X} = \mathsf{yes} \bigwedge X \in \mathcal{C}_F \right\} \right) \\ &= \sum_{\mathcal{C}_\ell \in \boldsymbol{\mathcal{C}}} I^{\mathrm{VF}}(\mathcal{C}_\ell) \prod_{X \in \mathcal{C}_\ell \cap \mathrm{pa}(A)} P(A^{\downarrow X} = \mathsf{yes} \,|\, X = \mathsf{faulty}), \end{aligned}$$

where $I^{\mathrm{VF}}(\mathcal{C}_\ell)$ is the probability that *all* components in $\mathcal{C}_\ell$ are critical, i.e., $I^{\mathrm{VF}}(\mathcal{C}_\ell)$ equals the probability that $\mathcal{C}_\ell$ is the actual cutset.

Regarding questions, we distinguish between *symptom questions* and *configuration questions*. Symptom questions are used to examine possible failure manifestations; an example from the printer-domain is "`Does the printer test-page print correctly?`". These questions are designed to shed light on the fault at the cutset level, e.g., by trying to replicate the equipment's faulty modus in other slightly different situations. (If the test-page prints correctly the problem is probably related to the application generating the print job.) Symptom questions are connected to the MCS nodes in the domain, see the node Q_S in Fig. 2. The edges are pointing in the direction of the causal influence, i.e., from the MCS nodes to the questions. The parent set of a symptom question Q_S, $\mathrm{pa}(Q_S) \subseteq \boldsymbol{\mathcal{C}}$, determines the set of MCSs that directly influences the likelihood of the different answers to the question.

Configuration questions are used to uncover the environment in which the equipment is embedded, by trying to reveal any configuration settings that are applied. An example from our domain is "`What operating system do you use?`". Configuration settings does not directly relate to a given MCS, but may change the likelihood for components to be operating. (If the operating system is Linux, the printing problem is not related to the Windows printer drivers.) The edges connecting a configuration node to the system layer are therefore directed from the configuration node to the components, see K in Fig. 2. The user may be unable to correctly answer questions regarding the configuration settings. The answer to a configuration question is therefore modelled as a random variable, see Q_K in Fig. 2. That is, we will receive information about Q_K (and not K directly) when the model is used, and Q_K is therefore needed explicitly in the model together with K.

2.3 Building the TS-models

The theme of this paper is to find a close to optimal TS-strategy in a *given* TS-model, but we will close this section by briefly touching upon the knowledge acquisition process used to generate the TS-model.[10] Building BNs has traditionally been considered such a complex task that BN experts would have to be deeply involved in the process. The BATS system consists of about 40 separate Bayesian network models, each representing a specific failure-mode. Some models are quite small, but the largest contains about 80 actions and 40 questions. To build these models we solely relied on a team of 6–7 experts. The number of models made it necessary to build a special tool for knowledge acquisition, see [24]. This tool, which is termed *BATS Author*, is designed to ensure that no knowledge about BNs is required to build the TS-models. The information required to generate the models can be specified in a terminology close to the experts' own, and the conditional probabilities can be expressed in the direction most natural for the expert. The BN structure is made s.t. the conditional independence statements encoded in the graph are easily verified. Skaanning [24] reports that all the models required to describe the failure-modes for another printer was built and validated in one man-month using this tool.

3 Action sequences

In this section we look at the situation where the only available troubleshooting steps are actions. In this case the TS-strategy is simply a *TS-sequence*, i.e., a string of actions performed one after another until the equipment is repaired. Let $\boldsymbol{\epsilon}$ denote arbitrary evidence collected so far during troubleshooting, i.e., a list of actions that all have failed to repair the equipment. To be more specific, we use $\boldsymbol{e}_j$ to denote the evidence that the first j actions in the sequence $\mathcal{S} = \langle A_1, \ldots, A_N \rangle$ have all failed to repair the equipment, $\boldsymbol{e}_j = \{R(A_i) = \mathsf{no} : i = 1, \ldots, j\}$. If A_k solves the problem with certainty, then $P(\boldsymbol{e}_k) = 0$, which reflects the fact that the TS-sequence is terminated after the k'th step. Note that $\boldsymbol{e}_0 = \{\emptyset\}$, and $P(\boldsymbol{e}_0) = 1$ as the equipment is assumed to be faulty at the beginning of the troubleshooting.

The *expected cost of repair* of a troubleshooting sequence $\mathcal{S} = \langle A_1, \ldots, A_n \rangle$, where action A_i is allocated the cost C_i, is the mean cost until an action succeeds or all actions have been performed:

$$\mathrm{ECR}(\mathcal{S}) = \sum_{i=1}^{N} C_i \cdot P(\boldsymbol{e}_{i-1}) \ . \tag{2}$$

[10] This outline is based on Skaanning [24] and Jensen et al. [10]; further details can be found in those papers.

A TS-sequence is said to be *optimal* if it achieves the minimum ECR of all TS-sequences. Note that it might be slightly misleading to use the term "Expected cost of repair" as we consider a situation where a repair sequence may fail to repair the equipment (since some actions may be imperfect, and therefore fail to fix the critical components). Thus, a repair sequence $\mathcal{S}$ may leave the equipment faulty, and the ECR is in this case the expected cost of *performing the sequence* and not of *repairing the equipment* (see the terminal nodes p, q and r in Fig. 1). The probability of a sequence failing to repair the equipment is however determined by the set $\mathcal{A}$ only, and does not depend on the sequencing of the actions. Hence, as we are only interested in finding the *cheapest sequence*, we will disregard this slight twist.[11] In this paper we focus our attention towards the cost of *performing* the TS-strategy, and we will continue to call this cost the ECR.

3.1 The greedy approach

Vesely and Fussell's component importance is commonly regarded as the best search heuristic when each component is repaired by a perfect action, and all repair actions have the same cost. Furthermore, when the costs are unequal the Vesely and Fussell's component importance can be scaled by the action's cost. The idea of using $\mathrm{I}^{\mathrm{FV}}(\cdot)$ to sequence the actions generalizes to our situation, see Equation 1, and we therefore define an action's *efficiency* in the following way:

Definition 1. Let $A \in \mathcal{A}$ be a repair action, let C_A be the cost of performing A, and let $\boldsymbol{\epsilon}$ be the evidence compiled so far during troubleshooting. The efficiency of A given $\boldsymbol{\epsilon}$ is defined as

$$\mathrm{ef}(A \,|\, \boldsymbol{\epsilon}) = \frac{P(R(A) = \mathsf{ok} \,|\, \boldsymbol{\epsilon})}{C_A} \ .$$

The efficiency has an important property when verifying that a TS-sequence $\mathcal{S}$ is sub-optimal:

Proposition 2. *Let $\mathcal{S} = \langle A_1, \ldots, A_N \rangle$ be an optimal TS-sequence of actions for which the cost of each action is independent of the other actions taken. Then it must hold that* $\mathrm{ef}(A_i \,|\, \boldsymbol{e}_{i-1}) \geq \mathrm{ef}(A_{i+1} \,|\, \boldsymbol{e}_{i-1})$.

[11] If, on the other hand, we were interested in the monetary value of the *expected cost* of the cheapest sequence, our approach would be misleading. To work in such situations, Breese and Heckerman [9] propose to introduce a new action named *Call Service* as the final act in a TS-sequence. Performing this action will put the equipment back in operating modus, but presumably at a high cost since external personnel is involved in fixing the problem.

Proof. Examine the two TS-sequences $\mathcal{S} = \langle A_1, \ldots, A_i, A_{i+1}, \ldots, A_N \rangle$ and $\mathcal{S}' = \langle A_1, \ldots, A_{i+1}, A_i, \ldots, A_N \rangle$. From Equation 2 we get

$$\begin{aligned}\mathrm{ECR}(\mathcal{S}) - \mathrm{ECR}(\mathcal{S}') = &(C_i \cdot P(\boldsymbol{e}_{i-1}) + C_{i+1} \cdot P(\boldsymbol{e}_{i-1}, R(A_i) = \mathsf{no})) \\ &- \Big(C_{i+1} \cdot P(\boldsymbol{e}_{i-1}) + C_i \cdot P(\boldsymbol{e}_{i-1}, R(A_{i+1}) = \mathsf{no})\Big)\end{aligned}$$

hence, $\mathrm{ECR}(\mathcal{S}) - \mathrm{ECR}(\mathcal{S}') \leq 0$ iff

$$\frac{P(R(A_i) = \mathsf{ok} \,|\, \boldsymbol{e}_{i-1})}{C_i} \geq \frac{P(R(A_{i+1}) = \mathsf{ok} \,|\, \boldsymbol{e}_{i-1})}{C_{i+1}} \,.$$

□

Note that Proposition 2 can in general not be used to decide whether a TS-sequence $\mathcal{S}$ is *optimal*, it is merely a characterization of some sub-optimal sequences.

A direct corollary of Proposition 2 is that if action A_i has the highest efficiency amongst all remaining actions given the aggregated evidence $\boldsymbol{\epsilon}$, and no evidence $\boldsymbol{\epsilon}' \supset \boldsymbol{\epsilon}$ excluding A_i exists such that this changes, then it is optimal to perform A_i before any other action. Some situations where this formulation is useful is given in the following Proposition, which is a simple reformulation of [10, Proposition 1]:

Proposition 3. *Assume that the following holds*

(1) The equipment has N components and N actions.
(2) There are no questions.
(3) Exactly one MCS is faulty.
(4) Each action has a specific probability of repairing the components. It is given by $P(A_i^{\downarrow X_i} = \mathsf{yes} \,|\, X_i = \mathsf{faulty}) > 0$, $P(A_i^{\downarrow X_j} \,|\, X_j = \mathsf{faulty}) = 0$ for $i \neq j$.
(5) The cost C_i of action A_i does not depend on the sequencing of the actions.
(6) The equipment is designed as a serial system, i.e., the MCSs are singletons: $\mathcal{C}_i = \{X_i\}$, $i = 1, \ldots, N$.

Then we have:
If $\mathrm{ef}(A_j \,|\, \boldsymbol{e}_0) \leq \mathrm{ef}(A_k \,|\, \boldsymbol{e}_0)$ then $\mathrm{ef}(A_j \,|\, \boldsymbol{\epsilon}) \leq \mathrm{ef}(A_k \,|\, \boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ is any evidence of the type "Actions $\mathcal{A}' \subseteq \mathcal{A} \setminus \{A_j, A_k\}$ have failed".

Propositions 2 and 3 motivate the *greedy approach*:

Algorithm 1 (Greedy approach).
`(1)` `For all` $A_j \in \mathcal{A}$ `Calculate` $\mathrm{ef}(A_j \,|\, \boldsymbol{e}_0)$;

(2) Let $\mathcal{S}$ be the list of actions ordered according to $\mathrm{ef}(\cdot \,|\, \boldsymbol{e}_0)$;
(3) Return $\mathcal{S}$;

It follows that the greedy approach is optimal under the assumptions of Proposition 3. Note that it is not always optimal to sequence the actions based on the efficiencies. A counter-example is given below:

Example 4. Consider the domain described in Fig. 2 (with failure data from Fig. 3). We assume perfect repair actions, let $C_i = 1$ for all actions, and disregard the questions Q_S and Q_K. The greedy approach selects the sequence $\langle A_3, A_2, A_4 \rangle$ with ECR = 1.58. The optimal sequence found by exhaustive search is $\langle A_2, A_4 \rangle$, with ECR = 1.47. (Note that this result is not contradictory to Proposition 2; the efficiencies are calculated as $\mathrm{ef}(A_2 \,|\, \boldsymbol{e}_0) = .529$, $\mathrm{ef}(A_3 \,|\, \boldsymbol{e}_0) = .624$ and $\mathrm{ef}(A_4 \,|\, \boldsymbol{e}_0) = .486$, hence it is in accordance with Proposition 2 to start with A_2 as long as it is not followed by A_3.)

An obvious attempt to improve the results of Example 4 is to recalculate the efficiencies each time new evidence comes in. In this way we make sure that all information available when the i'th step is to be chosen is actually taken into account; recall that we use B_j to denote the j'th step in the strategy $\mathcal{S}$:

Algorithm 2 (Greedy approach with recalculations).
(1) $\boldsymbol{\epsilon} \leftarrow \{\emptyset\}$; $\mathcal{A}' \leftarrow \{A_1, \ldots, A_N\}$; $\mathcal{S} = \langle \cdot \rangle$;
(2) For $i = 1$ to N
  (a) For all $A_j \in \mathcal{A}'$ Calculate $\mathrm{ef}(A_j \,|\, \boldsymbol{\epsilon})$;
  (b) Select $A_k \in \mathcal{A}'$ s.t. $\mathrm{ef}(A_k \,|\, \boldsymbol{\epsilon})$ is maximized;
  (c) $B_i \leftarrow A_k$; $\mathcal{A}' \leftarrow \mathcal{A}' \setminus \{A_k\}$; $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon} \cup \{R(A_k) = \mathsf{no}\}$.
(3) Return $\mathcal{S}$;

Applied to the model in Example 4 this algorithm generates the sequence $\mathcal{S} = \langle A_3, A_4, A_2 \rangle$ with ECR = 1.53. This is better than the greedy approach, but still not optimal. A result similar to Proposition 3 can be shown for arbitrary sized but disjoint MCSs if we assume that all actions are perfect:

Proposition 5. *Let $\mathcal{S} = \langle A_1, \ldots, A_n \rangle$ be a repair sequence for a troubleshooting problem fulfilling conditions 1 – 5 in Proposition 3. The MCSs are disjoint, $\mathcal{C}_i \cap \mathcal{C}_j = \{\emptyset\}$, $i \neq j$, and all repair actions are perfect, i.e., $P(A_i^{\downarrow X_i} = \mathsf{ok} \,|\, X_i = \mathsf{faulty}) = 1$ for $i = 1, \ldots, N$. Let $\mathcal{S}$ be the output of Algorithm 2. Then $\mathcal{S}$ is an optimal repair sequence.*

It should be emphasized that the actions are assumed to be perfect in Proposition 5. When the actions are non-perfect, optimality is no longer assured, as can be seen from the example below:

Example 6. Consider a TS-model with two cutsets $\mathcal{C}_1 = \{X_1, X_2\}$ and $\mathcal{C}_2 = \{X_3, X_4, X_5\}$. Let $P(X_i = \mathsf{faulty}) = 3 \cdot 10^{-6}$ for $i = 1, 3, 4, 5$ and $P(X_2 =$

$\mathsf{faulty}) = 7 \cdot 10^{-6}$. Each component X_i is repaired by a dedicated action A_i. Let the cost of the actions be $C_1 = 9$, $C_2 = 12$, and $C_i = 10$ for $i = 3, 4, 5$. Finally, $P(X_1^{\downarrow A_1} = \mathsf{ok} \,|\, X_1 = \mathsf{faulty}) = .9$, $P(X_i^{\downarrow A_i} = \mathsf{ok} \,|\, X_i = \mathsf{faulty}) = .98$ for $i = 2, 3$, and $P(X_i^{\downarrow A_i} = \mathsf{ok} \,|\, X_i = \mathsf{faulty}) = .95$ for $i = 4, 5$. Then Algorithm 2 returns $\mathcal{S}_1 = \langle A_5, A_1, A_3, A_4, A_2 \rangle$ with $\mathrm{ECR}(\mathcal{S}_1) = 14.95$, whereas the optimal sequence is $\mathcal{S}_2 = \langle A_5, A_3, A_4, A_1, A_2 \rangle$ with $\mathrm{ECR}(\mathcal{S}_2) = 14.84$.

3.2 Dependent actions

The crucial step when optimality is proven in the setting of Propositions 3 is the fact that no evidence obtained during troubleshooting can change the ordering of the remaining actions under consideration; the residual probability mass, i.e., the probability $P(R(A_i) = \mathsf{ok} \,|\, \boldsymbol{e}_{i-1})$, is absorbed uniformly by all these actions. Hence, the initial ordering of two actions, $A_i \prec A_j$, say, cannot change when some new evidence $R(A_k) = \mathsf{no}$, $A_k \notin \{A_i, A_j\}$ arrives. In the general case, however, the ordering of a subset of actions $\mathcal{A}' \subset \mathcal{A}$ may depend on what evidence $\boldsymbol{\epsilon}$ is collected, even if $\boldsymbol{\epsilon}$ does not contain explicit information about any of the actions in $\mathcal{A}'$. We call this situation *dependent actions* [25].

A domain for which the cost of an action does not depend on the sequence of actions taken is said to have dependent actions whenever there exists actions A_i, A_j and A_k s.t.

$$\frac{\mathrm{ef}(A_i \,|\, \emptyset)}{\mathrm{ef}(A_j \,|\, \emptyset)} \neq \frac{\mathrm{ef}(A_i \,|\, R(A_k) = \mathsf{no})}{\mathrm{ef}(A_j \,|\, R(A_k) = \mathsf{no})} \; .$$

A domain has dependent actions if there exists two actions A_i and A_j s.t. $\mathrm{pa}(A_i) \cap \mathrm{pa}(A_j) \neq \emptyset$ or there exists two actions A_i and A_j, two components $X_k \in \mathrm{pa}(A_i)$ and $X_\ell \in \mathrm{pa}(A_j)$, and an MCS $\mathcal{C}_m$ s.t. $\{X_k, X_\ell\} \subseteq \mathcal{C}_m$. An example from the printer domain is the action-pair "`Reseat toner cartridge.`" and "`Change toner cartridge.`" as both may solve problems related to bad seating of the cartridge.

Examples 4 and 6 showed that Vesely and Fussell's component importance is not optimal in general when the domain has dependent actions. This is hardly a surprise, since the problem of finding an optimal troubleshooting strategy is known to be NP-hard in this case [11]. To try to improve a suboptimal strategy we employ an adapted version of a standard algorithm for combinatorial optimization (similar to the algorithm presented by Norstrøm et al. [7]). This algorithm starts from an initial seed, and iteratively improves this sequence until it converges to a local optimum. Note that $B_k^{(i)}$ (Step 2a) denotes the k'th TS-step in the action sequence $\mathcal{S}$ when starting the i'th step of the iteration. Note also that the algorithm is said to converge (Step 3) when the ECR of the found sequence is not lower than the ECR of the sequence found previously.

Algorithm 3 (Discrete optimization).

(1) Initialization: $\mathcal{S} \leftarrow \langle B_1, \ldots, B_N \rangle$ for some ordering of $\mathcal{A}$;
(2) For $i = 1$ to N
  (a) For $j = i$ to N
    $\mathcal{R}_j \leftarrow \langle B_1^{(i)}, \ldots, B_{i-1}^{(i)}, B_j^{(i)}, B_i^{(i)}, \ldots, B_{j-1}^{(i)}, B_{j+1}^{(i)}, \ldots, B_N^{(i)} \rangle$;
  (b) Select $j_0 \in [i \ldots N]$ s.t. $\mathrm{ECR}(\mathcal{R}_{j_0})$ is minimized;
  (c) $\mathcal{S} \leftarrow \mathcal{R}_{j_0}$;
(3) If not converged then goto 2;
(4) Return $\mathcal{S}$;

A sequence $\mathcal{S} = \langle A_1, A_2, \ldots, A_i, \ldots, A_j, \ldots, A_N \rangle$ is a local optimum if, whenever we insert A_j before A_i ($j > i$) in $\mathcal{S}$ to obtain $\mathcal{S}' = \langle A_1, A_2, \ldots, A_{i-1}, A_j, A_i, \ldots, A_{j-1}, A_{j+1}, \ldots, A_N \rangle$, then $\mathrm{ECR}(\mathcal{S}) \leq \mathrm{ECR}(\mathcal{S}')$. It is obvious that Algorithm 3 converges to a local optimum since $\mathrm{ECR}(\mathcal{S})$ is guaranteed to be non-increasing after each loop of the algorithm (the algorithm can decide to stay put by selecting j_0 s.t. $\mathcal{R}_{j_0} = \mathcal{S}$ in Step 2b). It is however not guaranteed that the algorithm converges to the globally optimal sequence. The crucial choice to be made in Algorithm 3 is the initialization of $\mathcal{S}$ in Step 1. To ensure quick convergence to an approximately optimal solution, it can be beneficial to select a seed sequence that is close to the optimum. A natural choice is to initialize $\mathcal{S}$ as found by Algorithm 2. It is however easy to see that this sequence is a local optimum itself (confer Proposition 2), and it will therefore not be improved by Algorithm 3. Instead, we suggest to initialize the action sequence by ordering wrt. the *observation-based efficiency* (obef). We outline the derivation of the observation-based efficiency [25] below.

Consider a situation where the evidence $\boldsymbol{\epsilon}$ has been collected and it has been decided that the next action to perform is A. To calculate the observation-based efficiency, the TS-system should consider what information can be gained about the failed equipment by just getting to know that A does not solve the problem, and more importantly, the *value* of this information. It is natural to quantify this value as the difference in ECR between two degenerate models: *i*) The TS-system where the collected evidence is $\boldsymbol{\epsilon}' = \{\boldsymbol{\epsilon}, R(A = \mathsf{no})\}$ and *ii*) The TS-system where A has been made unavailable, but where the collected evidence is $\boldsymbol{\epsilon}'' = \boldsymbol{\epsilon}$. Assume that the sequence of remaining actions when given evidence $\boldsymbol{\epsilon}'$ is $\mathcal{S}(\boldsymbol{\epsilon}')$ and that the sequence of the actions when given evidence $\boldsymbol{\epsilon}''$ and A is unavailable is $\mathcal{S}(\boldsymbol{\epsilon}'')$. We define the *conditional* ECR *of the sequence* $\mathcal{S} = \langle A_1, \ldots, A_N \rangle$ *given* $\boldsymbol{\epsilon}'$ as $\mathrm{ECR}(\mathcal{S} \,|\, \boldsymbol{\epsilon}') = \sum_{j=1}^{N} C_j \cdot P(\boldsymbol{e}_{j-1} \,|\, \boldsymbol{\epsilon}')$. Finally, we define the value of the information contained in the event that $R(A) = \mathsf{no}$ given the current evidence $\boldsymbol{\epsilon}$ as

$$\mathrm{VOI}(R(A) = \mathsf{no} \,|\, \boldsymbol{\epsilon}) = \mathrm{ECR}(\mathcal{S}(\boldsymbol{\epsilon}') \,|\, \boldsymbol{\epsilon}') - \mathrm{ECR}(\mathcal{S}(\boldsymbol{\epsilon}'') \,|\, \boldsymbol{\epsilon}'),$$

i.e., $\mathrm{VOI}(R(A) = \mathsf{no} \,|\, \boldsymbol{\epsilon})$ is the difference of the expected cost of the strategies $\mathcal{S}(\boldsymbol{\epsilon}')$ and $\mathcal{S}(\boldsymbol{\epsilon}'')$. Note that both expected costs are calculated conditioned on

$\boldsymbol{\epsilon}'$, the evidence actually collected as the two strategies are considered to be employed.

To recapitulate, we want to consider the value of information an action *that fails* has to offer when we determine how to sequence the actions. This amount is calculated as $\mathrm{VOI}(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon})$, and we receive this gain with probability $P(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon})$. If we regard this amount as a refund, it is natural to approximate the "real" cost of action A as

$$\tilde{C}_A = C_A - P(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon}) \cdot \mathrm{VOI}(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon}).$$

$\tilde{C}_A$ is the cost we "spend" by performing A; $C_A - \tilde{C}_A$ is the expected reduction in ECR of the remaining sequence of action, which is obtained by learning that A fails. It is argued by Langseth and Jensen [25] that if one couples Definition 1 with Algorithm 2, one implicitly assumes that $\mathrm{VOI}(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon}) = 0$. On the other hand, if $\tilde{C}_A$ is used as the cost of A in the efficiency calculation, this will change the troubleshooting strategy in a way that attempts to incorporate the actual value of the information we receive. This leads to the definition of the observation-based efficiency:

Definition 7. Let $A \in \mathcal{A}$ be a repair action, let the cost of A be C_A, and let $\boldsymbol{\epsilon}$ be the evidence compiled so far during troubleshooting (i.e., not containing A). Let $\mathrm{VOI}(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon})$ be the value of information A will have if it fails (by altering the sequencing of the remaining actions). Then the observation-based efficiency of A given $\boldsymbol{\epsilon}$ is:

$$\mathrm{obef}(A \mid \boldsymbol{\epsilon}) = \frac{P(R(A) = \mathsf{ok} \mid \boldsymbol{\epsilon})}{C_A - P(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon}) \cdot \mathrm{VOI}(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon})}.$$

An algorithm that orders the actions according to the observation-based efficiency does in general not offer an optimal solution; a sequence ordered in this way may even violate the optimality check of Proposition 2. This is however of minor importance, as we only use the sequence as a seed to Algorithm 3 and do not regard it as a final solution on its own. Note however, that the probability update is proportional under the assumptions in Proposition 3, which means that $\mathrm{VOI}(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon}) = 0$ in this case. The observation-based efficiency is therefore exact under the assumptions of Proposition 3. "`Cycle power.`" is an example of an action from our domain which has high value of information. Power cycling repairs many temporal problems, and ruling these out can be very beneficial for the future troubleshooting.

A problem with Definition 7 is that $\mathrm{VOI}(R(A) = \mathsf{no} \mid \boldsymbol{\epsilon})$ cannot be calculated unless one is able to correctly sequence all remaining actions (after performing A) in order to calculate $\mathrm{ECR}(\mathcal{S}(\boldsymbol{\epsilon}') \mid \boldsymbol{\epsilon}')$ and $\mathrm{ECR}(\mathcal{S}(\boldsymbol{\epsilon}'') \mid \boldsymbol{\epsilon}')$; a computationally prohibitive task. Langseth and Jensen discuss two approximations of

VOI($R(A) = \mathsf{no} \mid \boldsymbol{\epsilon}$): One based on the Shannon entropy of the efficiencies of the remaining actions, and the computationally simpler approach to use the *myopic* ordering of actions (i.e., based on Definition 1), see [25] for details.

Table 1 shows results of a small simulation study. Three troubleshooting models have been used: The example model of Fig. 3 (with $N = 5$ actions and $R = 4$ cutsets), the CPQRA model [26] ($N = 25$, $R = 20$) and Norstrøm et al.'s example [7] ($N = 6$, $R = 4$). For each model the actions' costs and the failure probabilities of the components have been randomized. Additionally, the probability of an action to successfully repair a component in its parent set was randomly selected in the interval $[0.9, 1.0]$. Then Algorithm 2 and Algorithm 3 were run, and compared by difference in ECR. [12] The simulations were run for 500 iterations. The reported numbers give the relative number of times Algorithm 2 found a result inferior to that of Algorithm 3 (Rel.num.), the average relative difference in ECR in those runs (Avg.rel.diff.), and the maximum relative difference in ECR (Max.rel.diff.).

	Rel.num.	Avg.rel.diff.	Max.rel.diff.
Example model of Fig. 3	8.2%	4.0%	7.5%
The CPQRA model [26]	9.4%	4.2%	8.2%
Norstrøm et al.'s example [7]	4.0%	4.9%	9.2%

Table 1
Algorithm 2 and Algorithm 3 are compared through a small simulation study.

The results in Table 1 show that even for the relatively small models we have considered, a strategy generated by the Vesely and Fussell's component importance (Algorithm 2) fails fairly frequently, and the additional cost of following an inferior sequence may be considerable.

As it is NP-hard to find the optimal repair sequence, Algorithm 3 is not infallible; it may sometimes be stuck in sub-optimal solutions. This did for instance happen for the CPQRA model (see Table 1), where Algorithm 3 even was inferior to Algorithm 2 in 1.2% of the simulations, with maximum relative cost difference equal to 2.1%.

4 Questions

When we add questions to our TS-model, the strategy is represented by a strategy tree, see Fig. 1. Note that the ECR cannot be calculated by Equation

[12] Algorithm 3 was initialized by the sequence obtained when the actions were ordered according to the observation-based efficiency; the value of information was approximated by employing a myopic strategy.

2 in this case, instead we use a recursive calculation scheme to compute the expected cost of repair:

Proposition 8. *Let $\mathcal{S}$ be a TS-strategy which starts with the step $B_{(1)}$ and then continues with the strategy conditioned on the possible outcomes of $B_{(1)}$. Then the* ECR *of $\mathcal{S}$ can be calculated recursively as:*

$$\mathrm{ECR}(\mathcal{S}) = C_{(1)} + \sum_{b_{(1)} \in \mathrm{sp}\left(B_{(1)}\right)} P\left(B_{(1)} = b_{(1)}\right) \cdot \mathrm{ECR}\left(\mathcal{S} \,|\, B_{(1)} = b_{(1)}\right) \ . \tag{3}$$

where $C_{(1)}$ is the cost of step $B_{(1)}$ and $\mathrm{ECR}\left(\mathcal{S} \,|\, B_{(1)} = b_{(1)}\right)$ is the ECR *of the sub-tree $\mathcal{S}$ following the branch for which $B_{(1)} = b_{(1)}$. The recursion is terminated by $\mathrm{ECR}\left(\emptyset \,|\, \cdot\right) = \mathrm{ECR}\left(\cdot \,|\, R(A_j) = \mathsf{ok}\right) = 0$.*

The obvious way to decide whether it pays to pose a question Q, is to calculate the value of information for that particular question. Let the strategy be defined as $\langle Q, \mathcal{S} \rangle$, where $\mathcal{S}$ is the optimal strategy conditioned on the answer to the question Q, and let $\mathcal{S}'$ be the optimal strategy when we are refused to pose Q. We define $\mathrm{VOI}(Q)$ as:

$$\mathrm{VOI}(Q) = \mathrm{ECR}(\mathcal{S}') - \sum_{q \in \mathrm{sp}(Q)} P(Q = q) \cdot \mathrm{ECR}(\mathcal{S} \,|\, Q = q) \ .$$

The system should pose the question if $\mathrm{VOI}(Q) > C_Q$.

A problem with this approach is that we must correctly position all other questions in the strategy before we can calculate $\mathrm{ECR}(\mathcal{S}')$ and $\mathrm{ECR}(\mathcal{S} \,|\, Q = q)$; this will lead to a too expensive recursion. Breese and Heckerman [9] propose to use a *myopic* approach to this problem: Assume that it is sufficient to sequence only *actions* when $\mathrm{VOI}(Q)$ is to be calculated, i.e., one should consider the effect of the question Q only on the sequencing of actions, and disregard the effect of the other questions. The two action sequences $\mathcal{S}$ and $\mathcal{S}'$ are then approximated by ordering the actions according to their efficiencies by Algorithm 2. In [10] it is argued that this approach will over-rate the effect of the question, because one in this case only compares the effect of asking the question *now*, with $\mathrm{ECR}_{\mathrm{Now}} = C_Q + \sum_{q \in \mathrm{sp}(Q)} P(Q = q) \cdot \mathrm{ECR}(\mathcal{S} \,|\, Q = q)$, or *never*, $\mathrm{ECR}_{\mathrm{Never}} = \mathrm{ECR}(\mathcal{S}')$. The decision rule is to pose the question iff

$$\mathrm{ECR}_{\mathrm{Now}} < \mathrm{ECR}_{\mathrm{Never}}. \tag{4}$$

Jensen et al. [10] argue that one should also compare $\mathrm{ECR}_{\mathrm{Now}}$ to the ECR of a strategy starting with what appears to be the best action, followed by Q, and thereafter a TS-sequence $\mathcal{S}''$, which depends on the outcome of Q. This approach has ECR given by

$$\mathrm{ECR}_{A,Q}(A, Q, \mathcal{S}'') = C_A + P(R(A) = \mathsf{no}) \cdot C_Q + \sum_{q \in \mathrm{sp}(Q)} P(Q = q, R(A) = \mathsf{no}) \cdot \mathrm{ECR}(\mathcal{S}'' \mid Q = q, R(A) = \mathsf{no}) \ .$$

The question should be posed iff

$$\mathrm{ECR}_{\mathrm{Now}} < \min\{\mathrm{ECR}_{A,Q}(A, Q, \mathcal{S}''), \mathrm{ECR}_{\mathrm{Never}}\}. \tag{5}$$

N	M	Optimal	Alg. 2 + Eq. 5	Alg. 2 + Eq. 4	Alg. 2
6	2	433.24	442.39	442.43	444.54
9	3	129.21	129.21	205.54	155.10
11	3	106.20	108.07	111.75	116.80
12	3	38.38	40.01	52.86	43.05
13	4	124.32	125.56	125.94	300.85
14	4	115.41	115.86	116.74	236.58
9	9	70.67	77.67	76.53	121.10
16	5	161.38	162.25	162.49	286.75
10	10	250.45	256.96	445.93	479.96
Avg. rel. diff. from opt.			2.51%	21.5%	59.16%

Table 2
Empirical comparison of the effect of including questions into 9 of the BATS TS-models. The results are extended from those reported by Vomlel [27].

To emphasize the importance of including questions in the troubleshooter system we reproduce and extend a set of experimental results from Vomlel [27]. We have examined 9 of the troubleshooter models included in the BATS tool; for each of them we calculated the ECR of the optimal TS-strategy, the ECR of the TS-strategy produced when combining Algorithm 2 with Equation 5, the ECR of the TS-strategy produced when combining Algorithm 2 with Equation 4, and finally the ECR of the TS-sequence generated from Algorithm 2 when questions were disregarded. The models we used for testing were moderately sized with N (number of actions) ranging from 6 to 16 and M (number of questions) in the interval from 2 to 10. The results clearly show how important the questions are in these real-life TS-models, and they also indicate that the approximations made in Algorithm 2 combined with the decision rule of Equation 5 may be quite reasonable.

5 Calculation scheme

In this section we will consider how to perform the required calculation in the model. As the TS-system will continuously interact with the user, it is important that the system can perform its calculations in "real time" (that is, the calculations should be performed using an amount of time that seems negligible to the user). The important point to make is that performing calculations is in principle of time complexity exponential in the number of components in the model. It is therefore crucial to identify the "idle time" of the system (i.e., the time when the user is not interacting with it), and use those points in time to perform the calculations. Idle time is available before the system is put into use, and at times when the user is busy performing an action or trying to find information to answer a question. Before the system is ready to be used it has to go through an initialization phase, which basically amounts to calculating the initial probabilities for each component to be in its faulty state given that the equipment is faulty, the probabilities for the actions to be successful, and the initial beliefs regarding possible answers to the different questions. These calculations can be performed off-line and are thus not subject to speed requirements. In the following we will therefore focus on how to incorporate information from the performed TS-steps into the system, that is, how to update the probability distributions when the compiled evidence $\boldsymbol{\epsilon}$ is extended.

5.1 Action sequences

First we will look at TS-systems that only consists of actions, and describe a method for calculating $P(R(A) = \mathsf{ok} \,|\, \boldsymbol{\epsilon})$ for an action $A \in \mathcal{A}$ where $\boldsymbol{\epsilon}$ is some evidence not involving A. Next, we will describe a method to calculate $P(\boldsymbol{\epsilon})$ (required by the ECR-calculations, see Equation 2). Note that the evidence $\boldsymbol{\epsilon}$ will contain only a list of *failed* actions, i.e., $\boldsymbol{\epsilon} = \{R(A) = \mathsf{no} : A \in \mathcal{A}'\}$. If an action is successful, the troubleshooting ends, and there is no need to incorporate that evidence into the system.

The key point during the calculations is that of conditional independence. Let $\mathrm{nd}(V)$ be the *non-descendants* of V in a directed graph $\mathcal{G}$; $Y \in \mathrm{nd}(X)$ iff there is no directed path $X \to \ldots \to Y$ in $\mathcal{G}$. An important result we shall use frequently is that $V \perp\!\!\!\perp \mathrm{nd}(V) \,|\, \mathrm{pa}(V)$ for any variable $V \in \mathcal{V}$.

The backbone of our calculating scheme is the observation that *if we know that $\mathcal{C}_i$ is the actual cutset*, then it is easy to calculate the success probabilities given the evidence $\boldsymbol{\epsilon}$. It turns out that $P(R(A) = \mathsf{ok} \,|\, \mathcal{C}_i = \mathsf{faulty}, \boldsymbol{\epsilon}) = P(R(A) = \mathsf{ok} \,|\, \mathcal{C}_i = \mathsf{faulty})$, see Lemma 9 below. Since the actual cutset is not known during troubleshooting, we use

$$P(R(A) = \mathsf{ok} \,|\, \boldsymbol{\epsilon}) = \sum_{\mathcal{C}_\ell \in \boldsymbol{\mathcal{C}}} P(R(A) = \mathsf{ok} \,|\, \mathcal{C}_\ell = \mathsf{faulty}, \boldsymbol{\epsilon}) \cdot P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{\epsilon})$$
$$= \sum_{\mathcal{C}_\ell \in \boldsymbol{\mathcal{C}}} P(R(A) = \mathsf{ok} \,|\, \mathcal{C}_\ell = \mathsf{faulty}) \cdot P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{\epsilon})$$

to calculate $P(R(A) = \mathsf{ok} \,|\, \boldsymbol{\epsilon})$. Next, we formalize the above statement:

Lemma 9. *Let $A \in \mathcal{A}$ be a repair action, and let evidence compiled during troubleshooting be denoted by $\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} = \{R(A_i) = \mathsf{no} : A_i \in \mathcal{A}'\}$ ($A \notin \mathcal{A}'$). Assume that the user's ability to repair one component X does not depend on the state of the other components, $A^{\downarrow X} \perp\!\!\!\perp X' \,|\, X$ for all $X' \in \mathcal{X} \backslash \{X\}$, and that information about the user failing to perform one repair action will not influence our beliefs about his ability to perform other actions, $A_i^{\downarrow X_k} \perp\!\!\!\perp A_j^{\downarrow X_\ell} \,|\, \{X_k, X_\ell\}$ whenever $i \neq j$. Then $P(R(A) \,|\, \mathcal{C}_m = \mathsf{faulty}, \boldsymbol{\epsilon}) = P(R(A) \,|\, \mathcal{C}_m = \mathsf{faulty})$. That is, the evidence $\boldsymbol{\epsilon}$ does not influence $R(A)$ when conditioning on the actual MCS.*

Proof. First, notice that if $P(R(A) = \mathsf{ok} \,|\, \mathcal{C}_m = \mathsf{faulty}) = 0$, then no evidence $\boldsymbol{\epsilon}$ can change this belief. Hence, $P(R(A) \,|\, \mathcal{C}_m = \mathsf{faulty}, \boldsymbol{\epsilon}) = P(R(A) \,|\, \mathcal{C}_m = \mathsf{faulty})$ if A cannot repair any component in $\mathcal{C}_m$. Next, assume that the action A can repair components in only one MCS, $\mathcal{C}_\ell$. If $\mathcal{C}_\ell = \mathsf{faulty}$, then all components $X_j \in \mathcal{C}_\ell$ are in their faulty state. Hence, we have evidence on the set $\mathrm{pa}(A)$, and since $\boldsymbol{\epsilon}$ only contains non-descendant of A by construction of the domain model, $A \perp\!\!\!\perp \boldsymbol{\epsilon} \,|\, \{\mathcal{C}_\ell = \mathsf{faulty}\}$. It follows that $R(A) \perp\!\!\!\perp \boldsymbol{\epsilon} \,|\, \{\mathcal{C}_\ell = \mathsf{faulty}\}$, and therefore $P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty}, \boldsymbol{\epsilon}) = P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty})$. (See A_1 in Fig. 5; the probability for A_1 to repair the equipment is determined by the state of $\mathcal{C}_1$ only. If $\mathcal{C}_1$ is the actual MCS then A_1 repairs the equipment with probability $P(A_1^{\downarrow X_1} = \mathsf{yes} \,|\, X_1 = \mathsf{faulty})$ no matter what actions have earlier been performed; if $\mathcal{C}_1$ is not faulty, then A_1 can never repair the equipment.)

In the general case action A can repair more than one MCS. To see that the Lemma holds also in this case, we introduce the random variable $\zeta(\mathcal{C}_\ell)$, which is defined s.t. $\zeta(\mathcal{C}_\ell) = \mathsf{yes}$ if $\{X_i = \mathsf{faulty} : X_i \in \mathcal{C}_\ell \wedge X_j = \mathsf{ok} : X_j \notin \mathcal{C}_\ell\}$; $\zeta(\mathcal{C}_\ell) = \mathsf{no}$ otherwise. Notice that the effect of conditioning on the event $\zeta(\mathcal{C}_\ell) = \mathsf{yes}$ is that all $X \in \mathcal{X}$ are given evidence, and by construction of the domain model, the set $\mathrm{pa}(A) \subseteq \mathcal{X}$ is instantiated. Hence $P(R(A) = \mathsf{ok} \,|\, \zeta(\mathcal{C}_\ell) = \mathsf{yes}, \boldsymbol{\epsilon}) = P(R(A) = \mathsf{ok} \,|\, \zeta(\mathcal{C}_\ell) = \mathsf{yes})$. Since $A^{\downarrow X} \perp\!\!\!\perp X' \,|\, X$ for $X' \in \mathcal{X} \setminus \{X\}$, we have $P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty}, \boldsymbol{\epsilon}) = P(R(A) \,|\, \zeta(\mathcal{C}_\ell) = \mathsf{yes}, \boldsymbol{\epsilon})$. Finally, it follows that $P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty}, \boldsymbol{\epsilon}) = P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty})$. (Look at action A_3 in Fig. 5, and assume that $\mathcal{C}_3$ is known to be faulty, which means that $X_3 = X_4 = \mathsf{faulty}$. The event $\{R(A_3) = \mathsf{ok}\}$ is in this case equivalent to $\{A_3^{\downarrow X_3} = \mathsf{yes} \vee A_3^{\downarrow X_4} = \mathsf{yes}\}$. So far we only have observations on X_3 and X_4; X_1 and X_2 are not instantiated. Hence, information may flow from $R(A_1)$ to $R(A_3)$, and

thereby break the required independence (which is problematic if $\{R(A_1) = \mathsf{no}\}$ has been observed). The assumption $A_i^{\downarrow X_j} \perp\!\!\!\perp X_k \,|\, X_j$ does however justify that we may set $X_1 = X_2 = \mathsf{ok}$ without changing the required probability $P(R(A_3) \,|\, \mathcal{C}_3 = \mathsf{faulty}, \boldsymbol{\epsilon})$. All flow of information from any compiled evidence $\boldsymbol{\epsilon}$ to $R(A_3)$ is blocked when these stochastic variables are instantiated, and the desired conditional independence follows.) □

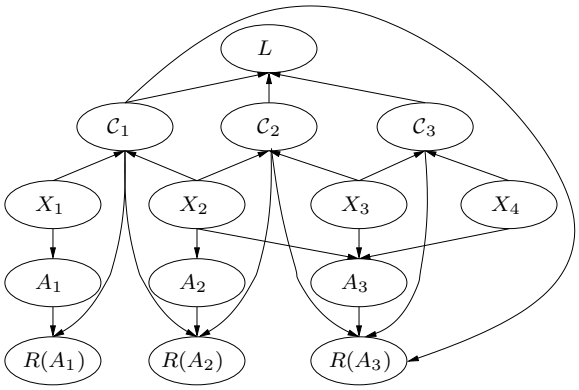


Fig. 5. Example TS-model to exemplify the proof of Lemma 9.

We utilize Lemma 9 to calculate the probability that an action $A \in \mathcal{A}$ repairs the equipment:

$$P(R(A) \,|\, \boldsymbol{\epsilon}) = \sum_{\mathcal{C}_\ell \in \boldsymbol{\mathcal{C}}} P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty}) \cdot P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{\epsilon}) \ . \tag{6}$$

That is, calculating $P(R(A) \,|\, \boldsymbol{\epsilon})$ amounts to finding $P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty})$ and $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{\epsilon})$ for all $\mathcal{C}_\ell \in \boldsymbol{\mathcal{C}}$. The values of $P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty})$ can easily be calculated from the model description before the troubleshooting starts, whereas $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{\epsilon})$ must be calculated in each case.

We now show that Lemma 9 can be used also to calculate $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_i)$ rather efficiently; recall that $\boldsymbol{e}_i$ is used to denote the evidence that the first i actions in the sequence $\mathcal{S} = \langle A_1, \ldots, A_N \rangle$ have all failed to repair the equipment. We first use Bayes' rule to investigate how to update this probability when new evidence $\{R(A_i) = \mathsf{no}\}$ is received and appended to the compiled knowledge $\boldsymbol{e}_{i-1}$:

$$\begin{aligned} P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_i) &= P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_{i-1}, R(A_i) = \mathsf{no}) \\ &= \frac{P(R(A_i) = \mathsf{no} \,|\, \mathcal{C}_\ell = \mathsf{faulty},\, \boldsymbol{e}_{i-1}) \cdot P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_{i-1})}{P(R(A_i) = \mathsf{no} \,|\, \boldsymbol{e}_{i-1})} \\ &= \frac{P(R(A_i) = \mathsf{no} \,|\, \mathcal{C}_\ell = \mathsf{faulty}) \cdot P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_{i-1})}{P(R(A_i) = \mathsf{no} \,|\, \boldsymbol{e}_{i-1})}. \end{aligned} \tag{7}$$

$P(R(A_i) = \mathsf{no} \,|\, \boldsymbol{e}_{i-1})$ is just a normalization constant in this calculation, which can be found by

$$P(R(A_i) = \mathsf{no} \,|\, \boldsymbol{e}_{i-1}) = \sum_{\mathcal{C}_k \in \boldsymbol{\mathcal{C}}} P(R(A_i) = \mathsf{no} \,|\, \mathcal{C}_k = \mathsf{faulty}) \cdot P(\mathcal{C}_k = \mathsf{faulty} \,|\, \boldsymbol{e}_{i-1}) \,.$$

Hence $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_i)$ can be calculated by expanding the evidence iteratively. The first step of this procedure requires the *a priori* distribution over the MCSs, $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_0)$. This distribution should be calculated by a full propagation in the Bayesian network, see [21]; remember that this propagation can be performed off-line (i.e., before troubleshooting starts). The evidence $\boldsymbol{e}_i$ is then incorporated by using Equation 7 until we obtain $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_i)$. This means that calculating $P(R(A) \,|\, \boldsymbol{e}_i)$ is of complexity $\mathcal{O}(R)$, where R is the number of MCSs in the domain if we have stored $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{e}_{i-1})$. As a consequence, the complexity of Algorithm 1 is $\mathcal{O}(NR + N\log(N))$ and the complexity of Algorithm 2 is $\mathcal{O}(N(NR + N)) = \mathcal{O}(N^2R)$.

Next, we look at how to calculate $P(\boldsymbol{e}_i)$; a number required by the ECR calculations, see Equation 2. This can be done by using the identity $P(\boldsymbol{e}_i) = P(R(A_i) = \mathsf{no} \,|\, \boldsymbol{e}_{i-1})P(\boldsymbol{e}_{i-1})$ and make the calculations iteratively; $P(R(A_i) = \mathsf{no} \,|\, \boldsymbol{e}_{i-1})$ is given by Equation 6; $P(\boldsymbol{e}_0) = 1$ by convention. Calculating $P(\boldsymbol{e}_i)$ is therefore of complexity $\mathcal{O}(R)$ if we store the values $P(\boldsymbol{e}_{i-1})$. In total, the calculation of ECR is thus of time complexity $\mathcal{O}(NR)$.

The time complexity of generating a full action sequence based on the observation-based efficiency (Definition 7) is dominated by the expensive calculations required to find $\mathrm{VOI}(\cdot \,|\, \boldsymbol{\epsilon})$. If this value is approximated by calculating the ECR of the sequence generated by Algorithm 2, then the time complexity of generating a complete action sequence by the observation-based efficiency is $\mathcal{O}(N^3R)$. If one settles for the cruder approximation offered by Algorithm 1 the time complexity of generating the sequence is reduced to $\mathcal{O}\left(N^2(\log(N) + R)\right)$.

The time complexity of Algorithm 3 is given by the complexity of the initialization and the cost of $\mathcal{O}(N^2)$ calculations of ECR. This means that the total complexity of Algorithm 3 when initialized according to the obef-sequence is $\mathcal{O}(N^3R)$. This should be compared to the corresponding calculations performed in a fault tree, which Norstrøm et al. [7] report to be $\mathcal{O}(N^2 3^N)$.

5.2 Questions

In this section we consider the cost of belief updating when the TS-model is extended to incorporate questions.

5.2.1 Symptom questions

We start the treatment of questions by considering symptom questions. Recall that symptom questions are used to examine possible failure symptoms; they are connected to the system layer at the MCS level, with edges directed from problem causes to the questions, see Q_S in Fig 2. By construction, the parent set of a symptom question Q_S in our BN representation is therefore restricted to the MCS nodes, $\text{pa}(Q_S) \subseteq \boldsymbol{\mathcal{C}}$. Furthermore, symptom questions do not have descendants in the graph. It follows that $Q_S \perp\!\!\!\perp \mathcal{V} \setminus \{\boldsymbol{\mathcal{C}}, Q_S\} \,|\, \boldsymbol{\mathcal{C}}$. Therefore, to calculate the effect of a symptom question on the remaining strategy, it is only required to calculate the effect on the distribution over the MCSs, $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, Q_S = q, \boldsymbol{\epsilon})$. This can be done by using Bayes' rule:

$$\begin{aligned} P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, Q_S = q, \boldsymbol{\epsilon}) &= \frac{P(Q_S = q \,|\, \mathcal{C}_\ell = \mathsf{faulty}, \boldsymbol{\epsilon}) \cdot P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{\epsilon})}{P(Q_S = q \,|\, \boldsymbol{\epsilon})} \\ &= \frac{P(Q_S = q \,|\, \mathcal{C}_\ell = \mathsf{faulty}) \cdot P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{\epsilon})}{P(Q_S = q \,|\, \boldsymbol{\epsilon})}\,, \end{aligned} \tag{8}$$

where $P(Q_S = q \,|\, \boldsymbol{\epsilon}) = \sum_{\mathcal{C}_k \in \boldsymbol{\mathcal{C}}} P(Q_S = q \,|\, \mathcal{C}_k = \mathsf{faulty}) \cdot P(\mathcal{C}_k = \mathsf{faulty} \,|\, \boldsymbol{\epsilon})$. Hence, the complexity of calculating $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, Q_S = q, \boldsymbol{\epsilon})$ from $P(\mathcal{C}_\ell = \mathsf{faulty} \,|\, \boldsymbol{\epsilon})$ is $\mathcal{O}(R)$. If we assume that the ordering of actions needed to calculate the ECR values in the decision rule of Equation 5 is based on Algorithm 2, then a question can be evaluated in time complexity $\mathcal{O}(N^2 R)$. Note that the calculations will require the computation of ECR for several action sequences (described in Section 5.1); one for each possible answer to the question.

Note that $Q_S \perp\!\!\!\perp \mathcal{V} \setminus \{\boldsymbol{\mathcal{C}}, Q_S\} \,|\, \boldsymbol{\mathcal{C}}$ implies that symptom questions will not corrupt the calculations of $R(A \,|\, \boldsymbol{\epsilon})$ in Equation 6; we can use that calculation scheme to calculate $R(A \,|\, \boldsymbol{\epsilon})$ even when the evidence $\boldsymbol{\epsilon}$ contains answers to symptom questions.

5.2.2 Configuration questions

Configuration questions are designed to highlight the likelihood of component failures by uncovering the environment in which the failed equipment is embedded. Configuration nodes are connected to the system layer via the component layer, with edges directed from question to components, see K in Fig. 2. The answer to the question is modelled as a random variable dependent on the configuration, see Q_K in Fig. 2.

As for symptom questions, we are interested in evaluating Q_K according to Equation 5. First, however, we note that $R(A) \perp\!\!\!\perp \boldsymbol{\epsilon} \,|\, \{\mathcal{C}_\ell = \mathsf{faulty}\}$ also when configuration questions have been posed, $\{Q_K = q\} \subseteq \boldsymbol{\epsilon}$. Recall that $P(R(A) \,|\, \boldsymbol{\epsilon}',\ \mathcal{C}_\ell = \mathsf{faulty}) = P(R(A) \,|\, \mathcal{C}_\ell = \mathsf{faulty})$ when $\boldsymbol{\epsilon}'$ is a list of actions (not containing A) that have failed. This result trivially extends to the

case where $\boldsymbol{\epsilon}$ contain answers to questions because configuration questions are non-descendants of the actions' result nodes. We can therefore calculate the efficiency of an action using Equation 6 also in the case when configuration questions have been answered. Similarly, we can calculate the ECR-values required to evaluate a configuration question Q_K (according to Equation 5) efficiently by incorporating the effect of a question Q_K at the cutset nodes by using Equation 8.

Special attention is however required for the case when one configuration question Q_{K1} is evaluated, and the evidence $\boldsymbol{\epsilon}$ already contains the answer to another configuration question Q_{K2} together with a list of failed actions $\boldsymbol{\epsilon}'$, $\boldsymbol{\epsilon} = \{Q_{K2} = q, \boldsymbol{\epsilon}'\}$. The answer to the two configuration questions Q_{K1} and Q_{K2} are not independent given the actual cutset; we have $P(Q_{K1} = q \,|\, \boldsymbol{\epsilon}, \mathcal{C}_\ell = \mathsf{faulty}) = P(Q_{K1} = q \,|\, Q_{K2} = q, \mathcal{C}_\ell = \mathsf{faulty})$. Hence, we must take the answers to all earlier configuration questions into account when we want to calculate $P(Q_{K1} = q \,|\, \boldsymbol{\epsilon})$. A consequence of this conditional dependence is that the fast rules to incorporate new evidence into the system, see Equations 7 and 8, cannot be generalized to evidence containing configuration questions if the distribution of other configuration questions should be updated correctly. We therefore have to perform a propagation in the model as soon as a configuration question is *answered*; note that it is not required to perform any propagations as long as the TS-system just considers to *pose* the question. The complexity of *evaluating* a configuration question is therefore $\mathcal{O}(N^2R)$; the time complexity of incorporating the answer into the system is exponential in the number of components.

6 Concluding remarks

We have described a decision-theoretic troubleshooting system, which builds on a Bayesian network describing the faulty equipment and its surroundings. The expressive power of the BN framework outperforms that of more commonly used model description paradigms as, e.g., fault trees, see [20]. We utilized this to make a rich description of the troubleshooting domain, which may include, e.g., non-perfect actions and information-gathering troubleshooting steps. Finally, we showed how our BN models allow fast calculation of the probabilities required to generate a reasonable troubleshooting strategy.

Acknowledgements

We would like to thank our project coworkers, in particular Claus Skaanning, Jiří Vomlel, and Olav Bangsø, for interesting discussions. Jiří Vomlel also supplied the software used to generate the results in Table 2. An anonymous referee gave comments that helped improving the paper.

References

[1] W. E. Vesely, Fault tree handbook, Tech. Rep. NUREG-0492, US Nuclear Regulatory Committee, Washington DC (1981).

[2] Q. Zhang, Q. Mei, A sequence of diagnosis and repair for a 2-state repairable system, IEEE Transactions on Reliability R-36 (1) (1987) 32–33.

[3] J. Kalagnanam, M. Henrion, A comparison of decision analysis and expert rules for sequential analysis, in: Uncertainty in Artificial Intelligence 4, North-Holland, New York, 1990, pp. 271–281.

[4] W. Xiaozhong, Fault tree diagnosis based on Shannon entropy, Reliability Engineering and System Safety 34 (1991) 143–167.

[5] W. Xiaozhong, R. M. Cooke, Optimal inspection sequence in fault diagnosis, Reliability Engineering and System Safety 37 (1992) 207–210.

[6] R. Reinertsen, W. Xiaozhong, General inspection strategy for fault diagnosis–minimizing the inspection costs, Reliability Engineering and System Safety 48 (3) (1995) 191–197.

[7] J. Norstrøm, R. M. Cooke, T. J. Bedford, Value of information based inspection-strategy of a fault-tree, in: Proceedings of the tenth European Conference on Safety and Reliability, A. A. Balkema, Munich, Germany, 1999, pp. 621–626.

[8] S. Srinivas, A polynomial algorithm for computing the optimal repair strategy in a system with independent component failures, in: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, San Francisco, CA., 1995, pp. 515–522.

[9] J. S. Breese, D. Heckerman, Decision-theoretic troubleshooting: A framework for repair and experiment, in: Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Francisco, CA., 1996, pp. 124–132.

[10] F. V. Jensen, U. Kjærulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, M. Vomlelová, The SACSO methodology for troubleshooting complex systems, Artificial Intelligence for Engineering, Design, Analysis and Manufacturing 15 (5) (2001) 321–333.

[11] M. Sochorová, J. Vomlel, Troubleshooting: NP-hardness and solution methods, in: The Proceedings of the Fifth Workshop on Uncertainty Processing, WUPES'2000, Jindřichův Hradec, Czech Republic, 2000, pp. 198–212.

[12] C. Skaanning, F. V. Jensen, U. Kjærulff, P. Pelletier, L. Ropstrup-Jensen, Printing system diagnosis: A Bayesian network application, Workshop on Principles of Diagnosis, Cape God, MA. (2000).

[13] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, San Mateo, CA., 1988.

[14] F. V. Jensen, Bayesian Networks and Decision Graphs, Springer Verlag, New York, 2001.

[15] R. E. Barlow, Using influence diagrams, in: C. A. Clarotti, D. V. Lindley (Eds.), Accelerated life testing and experts' opinions in reliability, 1988, pp. 145–157.

[16] H. J. Call, W. A. Miller, A comparison of approaches and implementations for automating decision analysis, Reliability Engineering and System Safety 30 (1990) 115–162.

[17] J. G. Torres-Toledano, L. E. Sucar, Bayesian networks for reliability analysis of complex systems, Lecture Notes in Artificial Intelligence 1484 (1998) 195–206.

[18] N. Fenton, B. Littlewood, M. Neil, L. Strigini, A. Sutcliffe, D. Wright, Assessing dependability of safety critical systems using diverse evidence, IEE Proceedings Software Engineering 145 (1) (1998) 35–39.

[19] P. H. Ibargüengoytia, L. E. Sucar, E. Morales, A probabilistic model approach for fault diagnosis, in: Eleventh International Workshop on Principles of Diagnosis, Morelia, Mexico, 2000, pp. 79–86.

[20] A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla, Improving the analysis of dependable systems by mapping fault trees into Bayesian networks, Reliability Engineering and System Safety 71 (3) (2001) 249–260.

[21] F. V. Jensen, S. L. Lauritzen, K. G. Olesen, Bayesian updating in causal probabilistic networks by local computations, Computational Statistics Quarterly 4 (1990) 269–282.

[22] D. Heckerman, J. S. Breese, A new look at causal independence, in: Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Francisco, CA., 1994, pp. 286–292.

[23] W. E. Vesely, A time-dependent methodology for fault tree evaluation, Nuclear Engineering and design 13 (1970) 339–360.

[24] C. Skaanning, A knowledge acquisition tool for Bayesian-network troubleshooters, in: Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference, Morgan Kaufmann Publishers, San Francisco, CA., 2000, pp. 549–557.

[25] H. Langseth, F. V. Jensen, Heuristics for two extensions of basic troubleshooting, in: Seventh Scandinavian Conference on Artificial Intelligence, SCAI'01, Frontiers in Artificial Intelligence and Applications, IOS Press, Odense, Denmark, 2001, pp. 80–89.

[26] Center for Chemical Process Safety, Guidelines for Chemical Process Quantitative Risk Analysis, American Institute of Chemical Engineers, New York, 1989.

[27] J. Vomlel, On quality of BATS troubleshooter and other approximative methods, Technical report, Department of Computer Science, Aalborg University, Denmark (2000).

III

Classification using Hierarchical Naïve Bayes models

Classification using Hierarchical Naïve Bayes models

Helge Langseth
Dept. of Mathematical Sciences
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
helgel@math.ntnu.no

Thomas D. Nielsen
Dept. of Computer Science
Aalborg University
DK-9220 Aalborg Øst, Denmark
tdn@cs.auc.dk

Abstract

Classification problems have a long history in the machine learning literature. One of the simplest, and yet most consistently well performing set of classifiers is the Naïve Bayes models. However, an inherent problem with these classifiers is the assumption that all attributes used to describe an instance are conditionally independent given the class of that instance. When this assumption is violated (which is often the case in practice) it can reduce classification accuracy due to "information double-counting" and interaction omission.

In this paper we focus on a relatively new set of models, termed Hierarchical Naïve Bayes models. Hierarchical Naïve Bayes models extend the modelling flexibility of Naïve Bayes models by introducing latent variables to relax some of the independence statements in these models. We propose a simple algorithm for learning Hierarchical Naïve Bayes models in the context of classification. Experimental results show that the learned models can significantly improve classification accuracy as compared to other frameworks. Furthermore, the algorithm gives an explicit semantics for the latent structures (both variables and states), which enables the user to reason about the classification of future instances and thereby boost the user's confidence in the model used.

1 Introduction

Classification is the task of predicting the class of an instance from a set of attributes describing that instance, i.e., to apply a mapping from the attribute space into a predefined set of classes. When learning a classifier we seek to generate such a mapping based on a database of labelled instances. Classifier learning, which has been an active research field over the last decades, can therefore be seen as a model selection process where the task is to find the single model, from some set of models, with the highest classification accuracy. The *Naïve Bayes* (NB) models (Duda and Hart 1973) is a set of particularly simple models which

has shown to offer very good classification accuracy. NB models assume that all attributes are conditionally independent given the class, but this assumption is clearly violated in many real world problems; in such situations overlapping information is counted twice by the classifier. To resolve this problem, methods for handling the conditional dependence between the attributes have become a lively research area; these methods are typically grouped into three categories: *Feature selection* (Kohavi and John 1997), *feature grouping* (Kononenko 1991; Pazzani 1995), and *correlation modelling* (Friedman et al. 1997).

The approach taken in this paper is based on correlation modelling using Hierarchical Naïve Bayes (HNB) models, see (Zhang et al. 2002). HNBs are tree-shaped Bayesian networks, with latent variables between the class node (the root of the tree) and the attributes (the leaves), see Figure 1. The latent variables are introduced to relax some of the independence statements of the NB classifier. For example, in the HNB model shown in Figure 1, the attributes A_1 and A_2 are not independent given C because the latent variable L_1 is unobserved. Note that if there are no latent variables in the HNB, it reduces to an NB model.

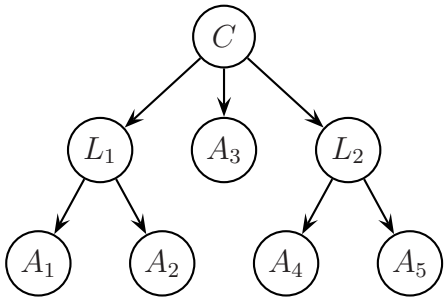


Figure 1: An HNB designed for classification. The class attribute C is in the root, and the attributes $\mathcal{A} = \{A_1, \ldots, A_5\}$ are leaf nodes. L_1 and L_2 are latent variables.

The idea to use HNBs in classification was first explored by Zhang et al. (2002). Zhang et al. (2002) search for the model maximizing the BIC score, which is a form of penalized log likelihood, see (Schwarz 1978); hence they look for a *scientific model* (Cowell et al. 1999) where the key is to find an interesting latent structure. In this paper we take the *technological modelling approach*: Our goal is mainly to build an accurate classifier. As a spin-off we also provide the latent variables with an explicit semantics, including a semantics for the state-spaces: Informally, a latent variable can be seen as aggregating the information from its children which is relevant for classification. Such a semantic interpretation is extremely valuable for a decision maker employing a classification system, as she can inspect the classification model and extract the "rules" which the system uses for the classification task.

The remainder of this paper is organized as follows: In Section 2 we give a brief overview of some approaches to Bayesian classification, followed by an introduction to HNB models in Section 3. In Section 4 we present an algorithm for learning HNB classifiers form data,

and Section 5 is devoted to empirical results. We discuss some aspects of the algorithm in further detail in Section 6 and conclude in Section 7.

2 Bayesian classifiers

A Bayesian network (BN) (Pearl 1988; Jensen 2001) is a powerful tool for knowledge representation, as it provides a compact representation of a joint probability distribution over a set of variables. Formally, a BN over a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_m\}$ is denoted by $B = (B_S, \boldsymbol{\Theta}_{B_S})$, where B_S is a directed acyclic graph and $\boldsymbol{\Theta}_{B_S}$ is the set of conditional probabilities. To describe B_S, we let $\mathrm{pa}\,(X_i)$ denote the parents of X_i in B_S, we use $\mathrm{sp}\,(X_i)$ to denote the state-space of X_i, and for a set of variables we have $\mathrm{sp}\,(\mathcal{X}) = \times_{X \in \mathcal{X}} \mathrm{sp}\,(X)$. In the context of classification, we shall use C to denote the class variable ($\mathrm{sp}\,(C)$ is the set of possible classes), and $\mathcal{A} = \{A_1, \ldots, A_n\}$ is the set of attributes describing the possible instances to be classified.

When doing classification in a probabilistic framework, a new instance (described by $\boldsymbol{a} \in \mathrm{sp}\,(\mathcal{A})$) is classified to class c^* according to:

$$c^* = \arg \min_{c \in \mathrm{sp}\,(C)} \sum_{c' \in \mathrm{sp}\,(C)} L(c, c') P(C = c' \,|\, \boldsymbol{a}),$$

where $L(\cdot, \cdot)$ defines the *loss function*, i.e., $L(c, c')$ is the cost of classifying an instance to class c when the correct class is c'. The two most commonly used loss functions are the 0/1-loss and the log-loss: The 0/1-loss is defined s.t. $L(c, c') = 0$ if $c' = c$ and 1 otherwise, and the log-loss is given by $L(c, c') = \log(P(c' \,|\, \boldsymbol{a}))$ independently of c.

Since we rarely have access to $P(C = c \,|\, \boldsymbol{A})$, learning a classifier amounts to estimating this probability distribution from a set of labelled training samples which we denote by $\mathcal{D}_N = \{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_N\}$; N is the number of training instances and $\boldsymbol{D}_i = \left(c^{(i)}, a_1^{(i)}, \ldots, a_n^{(i)}\right)$ is the class and attributes of instance i, $i = 1, \ldots, N$. Let $P(C = c \,|\, \boldsymbol{A}, \mathcal{D}_N)$ be the *a posteriori* conditional probability for $C = c$ given $\boldsymbol{A}$ after observing $\mathcal{D}_N$. Then an optimal Bayes classifier will classify a new instance with attributes $\boldsymbol{a}$ to class c^* according to (see e.g. (Mitchell 1997)):

$$c^* = \arg \min_{c \in \mathrm{sp}\,(C)} \sum_{c' \in \mathrm{sp}\,(C)} L(c, c') P(C = c' \,|\, \boldsymbol{a}, \mathcal{D}_N). \tag{1}$$

An immediate approach to estimate $P(C = c \,|\, \boldsymbol{A})$ is to use a standard BN learning algorithm, where the training data is used to give each possible classifier a *score* which signals its appropriateness as a classification model. One such scoring function is based on the

minimum description length (MDL) principle (Rissanen 1978; Lam and Bacchus 1994):

$$\mathrm{MDL}(B \mid \mathcal{D}_N) = \frac{\log N}{2} \left| \widehat{\boldsymbol{\Theta}}_{B_S} \right| - \sum_{i=1}^{N} \log \left(P_B \left(c^{(i)}, \boldsymbol{a}^{(i)} \,\middle|\, \widehat{\boldsymbol{\Theta}}_{B_S} \right) \right). \tag{2}$$

That is, the best scoring model is the one that minimizes $\mathrm{MDL}(\cdot \mid \mathcal{D}_N)$, where $\widehat{\boldsymbol{\Theta}}_{B_S}$ is the maximum likelihood estimate of the parameters in the model, and $\left| \widehat{\boldsymbol{\Theta}}_{B_S} \right|$ is the dimension of the parameter space (i.e., the number of free parameters in the model). However, as pointed out in (Greiner et al. 1997; Friedman et al. 1997) a "global" criteria like MDL may not be well suited for learning a classifier, as:

$$\sum_{i=1}^{N} \log \left(P_B \left(c^{(i)}, \boldsymbol{a}^{(i)} \right) \right) = \sum_{i=1}^{N} \log \left(P_B \left(c^{(i)} \mid \boldsymbol{a}^{(i)} \right) \right) + \sum_{i=1}^{N} \log \left(P_B \left(a_1^{(i)}, \ldots, a_n^{(i)} \right) \right).$$

In the equation above, the first term on the right-hand side measures how well the classifier performs on $\mathcal{D}_N$, whereas the second term measures how well the classifier estimates the joint distribution over the attributes. Thus, only the first term is related to the classification task, and the latter term will therefore merely bias the model search; in fact, the latter term will dominate the score if n is large. To overcome this problem, Friedman et al. (1997) propose to replace MDL with *predictive* MDL, MDL_p, defined as:

$$\mathrm{MDL}_p(B \mid \mathcal{D}_N) = \frac{\log N}{2} \left| \widehat{\boldsymbol{\Theta}}_{B_S} \right| - \sum_{i=1}^{N} \log \left(P_B \left(c^{(i)} \mid \boldsymbol{a}^{(i)} \,,\, \widehat{\boldsymbol{\Theta}}_{B_S} \right) \right). \tag{3}$$

However, as also noted by Friedman et al. (1997), $\sum_{i=1}^{N} \log \left(P_B \left(c^{(i)} \,\middle|\, \boldsymbol{a}^{(i)}, \widehat{\boldsymbol{\Theta}}_{B_S} \right) \right)$ cannot be calculated efficiently in general.

The argument leading to the use of predictive MDL as a scoring function rests upon the asymptotic theory of statistics. That is, model search based on MDL_p is guaranteed to select the best classifier w.r.t. both log-loss and 0/1-loss when $N \to \infty$. Unfortunately, though, the score may not be successful for finite data sets (Friedman 1997). To overcome this potential drawback, Kohavi and John (1997) describe the *wrapper approach*. Informally, this method amounts to estimating the accuracy of a given classifier by cross validation (based on the *training data*), and to use this estimate as the scoring function. The wrapper approach relieves the scoring function from being based on approximations of the classifier design, but at the potential cost of higher computational complexity. In order to reduce this complexity when learning a classifier, one approach is to focus on a particular sub-class of BNs. Usually, these sub-classes are defined by the set of independence statements they encode. For instance, one such restricted set of BNs is the Naïve Bayes models which assume that $P(C|\boldsymbol{A}) \propto P(C) \prod_{i=1}^{n} P(A_i|C)$, i.e., that $A_i \perp\!\!\!\perp A_j \mid C$.

Even though the independence statements of the NB models are often violated in practice, these models have shown to provide surprisingly good classification results. Resent research

into explaining the merits of the NB model has emphasized the difference between the 0/1-loss function and the log-loss, see e.g. (Friedman 1997; Domingos and Pazzani 1997). Friedman (1997, p. 76) concludes:

> Good probability estimates are not necessary for good classification; similarly, low classification error does not imply that the corresponding class probabilities are being estimated (even remotely) accurately.

The starting point of Friedman (1997) is that a classifier learned for a particular domain is a function of the training set. As the training set is considered a random sample from the domain, the classifier generated by a learner can be seen as a random variable; we shall use $\hat{P}(C = c \,|\, \boldsymbol{A})$ to denote the learned classifier. Friedman (1997) characterizes a classifier based on its bias (i.e., $\mathbb{E}_{\mathcal{D}_N}\left[P(C \,|\, \boldsymbol{A}) - \hat{P}(C \,|\, \boldsymbol{A})\right]^2$) and its variance (i.e., $\mathrm{Var}_{\mathcal{D}_N}\left(\hat{P}(C \,|\, \boldsymbol{A})\right)$); the expectations are taken over all possible training sets of size N. Friedman (1997) shows that in order to learn classifiers with low 0/1-loss it may not be sufficient to simply focus on finding a model with low classifier bias; robustness in terms of low classifier variance can be just as important.

An example of a class of models where low bias (i.e., fairly high model expressibility) is combined with robustness is the *Tree Augmented Naïve Bayes* (TAN) models, see (Friedman et al. 1997). TAN models relax the NB assumption by allowing a more general correlation structure between the attributes. More specifically, a Bayesian network model is initially created over the variables in $\mathcal{A}$, and this model is designed s.t. each variable A_i has at most one parent (that is, the structure is a *directed tree*). Afterwards, the class attribute is included in the model by making it the *parent* of each attribute. Friedman et al. (1997) use an adapted version of the algorithm by Chow and Liu (1968) to learn the classifier, and they prove that the structure they find is the TAN which maximizes the likelihood of $\mathcal{D}_N$; the algorithm has time complexity $O\left(n^2(N + \log(n))\right)$.

3 Hierarchical Naïve Bayes models

A special class of Bayesian networks is the so-called Hierarchical Naïve Bayes (HNB) models, a concept first introduced by Zhang et al. (2002), see also (Zhang 2002; Kočka and Zhang 2002). An HNB is a tree-shaped Bayesian network, where the variables are partitioned into three disjoint sets: $\{C\}$ is the class variable, $\mathcal{A}$ is the set of attributes, and $\mathcal{L}$ is a set of *latent* (or *hidden*) variables. In the following we use A to represent an attribute, whereas L is used to denote a latent variable; X and Y denote variables that may be either attributes or latent variables. In an HNB the class variable C is the root of the tree ($\mathrm{pa}(C) = \emptyset$) and the attributes are at the leaves ($\mathrm{ch}(A) = \emptyset, \forall A \in \mathcal{A}$); the latent variables are all internal ($\mathrm{ch}\,(L) \neq \emptyset$, $\mathrm{pa}\,(L) \neq \emptyset$, $\forall\, L \in \mathcal{L}$). The use of latent variables allows

conditional dependencies to be encoded in the model (as compared to e.g. the NB model). For instance, by introducing a latent variable as a parent of the attributes A_i and A_j, we can represent the (local) dependence statement $A_i \not\perp\!\!\!\perp A_j | C$. Being able to model such local dependencies is particularly important for classification, as overlapping information would otherwise be double-counted. Note that the HNB model reduces to the NB model in the special case when there are no latent variables.

When learning an HNB we can restrict our attention to the *parsimonious* HNB models; we need not consider models which encode a probability distribution that is also encoded by another model which has fewer parameters. Formally, an HNB model, $H = (B_S, \Theta_{B_S})$, with class variable C and attribute variables $\mathcal{A}$ is said to be parsimonious if there does not exist another HNB model, $H' = (B'_S, \Theta'_{B_S})$, with the same class and attribute variables s.t.:

i) H' has fewer parameters than H, i.e., $|\Theta_{B_S}| > |\Theta'_{B_S}|$.

ii) The probability distributions over the class and attribute variables are the same in the two models, i.e., $P(C, \mathcal{A}|B_S, \Theta_{B_S}) = P(C, \mathcal{A}|B'_S, \Theta'_{B_S})$.

In order to obtain an operational characterization of these models, Zhang et al. (2002) define the class of *regular* HNB models. An HNB model is said to be regular if for any latent variable L, with neighbours (parent and children) $X_1, X_2, \ldots X_n$, it holds that:

$$|\mathrm{sp}(L)| \leq \frac{\prod_{i=1}^{n} |\mathrm{sp}(X_i)|}{\max_{i=1,\ldots,n} |\mathrm{sp}(X_i)|},$$

and strict inequality holds when L has only two neighbours and at least one of them is a latent node.

Zhang et al. (2002) show that *i)* any parsimonious HNB model is regular, and *ii)* for a given set of class and attribute variables, the set of regular HNB model structures is finite. Observe that these two properties ensure that when searching for an HNB model we only need to consider regular HNB models and we need not deal with infinite search spaces.

As opposed to other frameworks, such as NB or TAN models, an HNB can model any correlation among the attribute variables by simply choosing the state-spaces of the latent variables large enough (although the encoding is not necessarily done in a cost-effective manner in terms of model complexity); note that the independence statements are not always represented explicitly in the graphical structure, but are sometimes only encoded in the conditional probability tables. On the other hand, the TAN model, for instance, is particular efficient for encoding such statements but may fail to represent certain types of dependence relations among the attribute variables. A TAN model is, e.g., not able to represent the statement "$C = 1$ if and only if exactly two out of the three attributes A_1, A_2 and A_3 are in state 1".

4 Learning HNB classifiers

4.1 The main algorithm

Our search algorithm is based on a greedy search over the space of all HNBs; we initiate the search with an HNB model, H_0, and learn a sequence $\{H_k\}$, $k = 1, 2 \ldots$ of HNB models. The search is conducted s.t. at each step we investigate the *search boundary* of the current model (denoted $\mathcal{B}(H_k)$), i.e., the set of models that can be reached from H_k in a single step. From this set of models the algorithm always selects a model with a higher score than the current one; if no such model can be found, then the current model is returned (see Algorithm 1).

Algorithm 1 (Greedy search)

1. Initiate model search with H_0;
2. **For** $k = 0, 1, \ldots$
   (a) Select $H' = \arg \max_{H \in \mathcal{B}(H_k)} \text{Score}(H \mid \mathcal{D}_N)$;
   (b) **If** $\text{Score}(H' \mid \mathcal{D}_N) > \text{Score}(H_k \mid \mathcal{D}_N)$ **then**:
   $H_{k+1} \leftarrow H'$; $k \leftarrow k + 1$;
   else
   return H_k;

In order to make the above algorithm operational we need to specify the score function $\text{Score}(\cdot \mid \mathcal{D}_N)$ as well as the search operator (which again defines the search boundary).

The score-function is defined s.t. a high value corresponds to what is thought to be a structure with good classification qualities (as measured by the average loss on unseen data), i.e., $\text{Score}(H \mid \mathcal{D}_N)$ measures the "goodness" of H. Note that the algorithm makes sure that $\text{Score}(H_{k+1} \mid \mathcal{D}_N) > \text{Score}(H_k \mid \mathcal{D}_N)$ for $k = 0, 1, \ldots$ which ensures convergence as long as the score is finite for all models. In order to apply a score metric that is closely related to what the search algorithm tries to achieve, we use the wrapper approach by Kohavi and John (1997). That is, we use cross validation (over the training set $\mathcal{D}_N$) to estimate an HNB's classification accuracy on unseen data; notice that the test-set (if defined) is *not* used when the score is calculated.

The search operator is defined s.t. the HNB structure is grown incrementally. More specifically, if $\mathcal{L}_k$ is the set of latent variables in model H_k, then the set of latent variables in H_{k+1}, is enlarged s.t. $\mathcal{L}_{k+1} = \mathcal{L}_k \cup \{L\}$, where L is a new latent variable. We restrict ourself to only considering candidate latent variables which are parents of two variables X and Y where $\{X, Y\} \subseteq \text{ch}(C)$ in H_k. Hence, we define H_{k+1} as the HNB which is produced from

H_k by including a latent variable L s.t. $\mathrm{pa}(L) = \{C\}$ and $\mathrm{pa}(X) = \mathrm{pa}(Y) = \{L\}$; H_{k+1} is otherwise identical to H_k. Thus, the search boundary $\mathcal{B}(H_k)$ consists of all models where exactly one latent variable has been added to H_k; there is one model in $\mathcal{B}(H_k)$ for each possible definition of the state-space of each possible new latent variable. Finally, as our starting point, H_0, we use the NB model structure; this implies that each H_k is a tree with a binary internal structure, i.e., any latent node $L' \in \mathcal{L}_k$ has exactly two children but the class node C may have up to n children. It is obvious that any distribution is in principle reachable by the search algorithm but, as the score function is multi-modal over the search space, the search will in general only converge towards a local optimum.

4.2 Restricting the search boundary

Unfortunately, $\mathcal{B}(H_k)$ is too large for the search algorithm to efficiently examine all models. To overcome this problem we shall instead focus the search by only selecting a subset of the models in $\mathcal{B}(H_k)$, and these models are then used to represent the search boundary. The idea is to pinpoint a few promising candidates in the search boundary without examining all models available. Basically the algorithm proceeds in two steps by first deciding where to include a latent variable, and then defining the state-space of the new latent variable:[1]

1. Find a candidate latent variable.
2. Select the state-space of the latent variable.

Note that when using this two-step approach for identifying a latent variable, we cannot use scoring functions such as the wrapper approach, MDL, or MDL_p in the *first* step; this step does not select a completely specified HNB.

Before describing the two steps in detail, recall that the algorithm starts out with an NB model, and that the goal is to introduce latent variables to improve upon that structure, i.e., to avoid "double-counting" of information when the independence statements of the NB model are violated.

4.2.1 Step 1: Finding a candidate latent variable

To facilitate the goal of the algorithm, a latent variable L is proposed as the parent of $\{X, Y\} \subseteq \mathrm{ch}(C)$ if the data points towards $X \not\perp\!\!\!\perp Y \mid C$. That is, we consider variables that are strongly correlated given the class variable as indicating a promising position for including a latent variable; from this perspective there is no reason to introduce a latent variable as a parent of X and Y if $X \perp\!\!\!\perp Y \mid C$. Hence, the variables that have the highest

[1] Ideally, a candidate latent variable should be selected directly (that is, defining location *and* state-space at the same time), but this is computationally prohibitive.

correlation given the class variable may be regarded as the most promising candidate-pair. More specifically, we calculate the conditional mutual information given the class variable, $I(\cdot, \cdot \,|\, C)$, for all (unordered) pairs $\{X, Y\} \subseteq \mathrm{ch}\,(C)$. However, as $I(X, Y \,|\, C)$ is increasing in both $|\mathrm{sp}\,(X)|$ and $|\mathrm{sp}\,(Y)|$ we cannot simply pick the pair $\{X, Y\}$ that maximizes $I(X, Y \,|\, C)$; this strategy would unintentionally bias the search towards latent variables with children having large domains. Instead we utilize that:

$$2N \cdot I(X, Y \,|\, C) \xrightarrow{\mathcal{L}} \chi^2_{|\mathrm{sp}\,(C)|\left(|\mathrm{sp}\,(X)|-1\right)\left(|\mathrm{sp}\,(Y)|-1\right)},$$

where $\xrightarrow{\mathcal{L}}$ means convergence in distribution as $N \to \infty$, see e.g. (Whittaker 1990). Finally, we calculate

$$Q(X, Y \,|\, \mathcal{D}_N) = P\left(Z \leq 2N \cdot I(X, Y \,|\, C)\right), \tag{4}$$

where Z is χ^2 distributed with $|\mathrm{sp}\,(C)|\left(|\mathrm{sp}\,(X)|-1\right)\left(|\mathrm{sp}\,(Y)|-1\right)$ degrees of freedom. The pairs $\{X, Y\}$ are ordered according to these probabilities, s.t. the pair with the highest probability is picked out. By selecting the pairs of variables according to $Q(X, Y \,|\, \mathcal{D}_N)$, the correlations are normalized w.r.t. the size differences in the state-spaces.

Unfortunately, to greedily select a pair of highly correlated variables as the children of a new latent variable is not always the same as improving classification accuracy, as can be seen from the example below:[2]

Example 1 Consider a classifier with binary attributes $\mathcal{A} = \{A_1, A_2, A_3\}$ (all with uniform marginal distributions) and target concept $C = 1 \Leftrightarrow \{A_1 = 1 \wedge A_2 = 1\}$. Assume that A_1 and A_2 are marginally independent but that $P(A_2 = A_3) = 0.99$. It then follows that:

$$P\left(Q(A_2, A_3 \,|\, \mathcal{D}_N) > Q(A_1, A_2 \,|\, \mathcal{D}_N)\right) \to 1$$

as N grows large (the uncertainty is due to the random nature of $\mathcal{D}_N$). Hence, the heuristic will not pick out $\{A_1, A_2\}$ which is most beneficial w.r.t.˙ classification accuracy, but will propose to add a variable L' with children $\mathrm{ch}\,(L') = \{A_2, A_3\}$.

4.2.2 Step 2: Selecting the state-space

To find the cardinality of a latent variable L, we use an algorithm similar to the one by Elidan and Friedman (2001): Initially, the latent variable is defined s.t. $|\mathrm{sp}\,(L)| = \prod_{X \in \mathrm{ch}(L)} |\mathrm{sp}\,(X)|$, where each state of L corresponds to exactly one combination of the states of the children of L. Let the states of the latent variable be labelled $l_1, \ldots, l_t$. We then iteratively collapse two states l_i and l_j into a single state l^* as long as this is "beneficial". Ideally, we would measure this benefit using the wrapper approach, but as this is computationally expensive we shall instead use the MDL_p score to approximate the

[2]This issue is also discussed in Section 6.

classification accuracy. Let $H' = (B'_S, \boldsymbol{\Theta}_{B'_S})$ be the HNB model obtained from a model $H = (B_S, \boldsymbol{\Theta}_{B_S})$ by collapsing states l_i and l_j. Then l_i and l_j should be collapsed if and only if $\Delta_L(l_i, l_j \,|\, \mathcal{D}_N) = \mathrm{MDL}_p\,(H \,|\, \mathcal{D}_N) - \mathrm{MDL}_p\,(H' \,|\, \mathcal{D}_N) > 0$. For each pair (l_i, l_j) of states we therefore compute:

$$\begin{aligned}\Delta_L(l_i, l_j \,|\, \mathcal{D}_N) &= \mathrm{MDL}_p(H|\mathcal{D}_N) - \mathrm{MDL}_p(H'|\mathcal{D}_N) \\ &= \frac{\log(N)}{2}\left(|\boldsymbol{\Theta}_{B_S}| - \left|\boldsymbol{\Theta}_{B'_S}\right|\right) + \sum_{i=1}^{N}\left[\log\left(P_{H'}(c^{(i)}|a^{(i)})\right) - \log\left(P_H(c^{(i)}|a^{(i)})\right)\right].\end{aligned}$$

For the second term we first note that:

$$\begin{aligned}\sum_{i=1}^{N}\left[\log\left(P_{H'}(c^{(i)}|a^{(i)})\right) - \log\left(P_H(c^{(i)}|a^{(i)})\right)\right] &= \sum_{i=1}^{N}\log\frac{P_{H'}(c^{(i)}|a^{(i)})}{P_H(c^{(i)}|a^{(i)})} \\ &= \sum_{D\in\mathcal{D}_N : f(D,l_i,l_j)}\log\frac{P_{H'}\left(c^D|a^D\right)}{P_H\left(c^D|a^D\right)},\end{aligned}$$

where $f(D, l_i, l_j)$ is true if case D includes either $\{L = l_i\}$ or $\{L = l_j\}$; cases which does not include these states cancel out. This is also referred to as *local decomposability* in (Elidan and Friedman 2001), i.e., the gain of collapsing two states l_i and l_j is local to those states and it does not depend on whether or not other states have been collapsed. In order to avoid considering all possible combinations of the attributes we approximate the difference in predictive MDL as the difference w.r.t. the relevant subtree. The relevant subtree is defined by C together with the subtree having L as root:[3]

$$\begin{aligned}&\sum_{D\in\mathcal{D}_N : f(D,l_i,l_j)}\log\frac{P_{H'}\left(c^D|a^D\right)}{P_H\left(c^D|a^D\right)} \qquad (5)\\ &\approx \log\prod_{c\in\mathrm{sp}\,(C)}\left[\left(\frac{N(c,l_i)}{N(l_i)}\right)^{N(c,l_i)}\cdot\left(\frac{N(c,l_j)}{N(l_j)}\right)^{N(c,l_j)} \Big/ \left(\frac{N(c,l_i)+N(c,l_j)}{N(l_i)+N(l_j)}\right)^{N(c,l_i)+N(c,l_j)}\right],\end{aligned}$$

where $N(c, s)$ and $N(s)$ are the sufficient statistics, e.g., $N(c, s) = \sum_{i=1}^{N}\gamma(C = c, L = s : \boldsymbol{D}_i)$; $\gamma(C = c, L = s : \boldsymbol{D}_i)$ takes on the value 1 if $(C = c, L = s)$ appears in case $\boldsymbol{D}_i$, and 0 otherwise; $N(s) = \sum_{c\in\mathrm{sp}\,(C)} N(c, s)$. Note that Equation 5 is in fact an equality if the relationship between C and $\mathrm{ch}\,(C)$ satisfy *independence of causal influence* (Heckerman and Breese 1994).

States are collapsed in a greedy manner, i.e., we find the pair of states with highest $\Delta_L(l_i, l_j \,|\, \mathcal{D}_N)$ and collapse those two states if $\Delta_L(l_i, l_j \,|\, \mathcal{D}_N) > 0$. This is repeated (making use of local decomposability) until no states can be collapsed, see also Algorithm 2.

[3]The relevant subtree can also be seen as the part of the classifier structure that is directly affected by the potential collapse of the states l_i and l_j.

Algorithm 2 (Determine state-space of L)

1. Initiate state-space s.t. $|\mathrm{sp}(L)| = \prod_{X \in \mathrm{ch}(L)} |\mathrm{sp}(X)|$;
   Label the states s.t. each state corresponds to a unique combination of $\mathrm{ch}(L)$;
2. **For each** $\{l_i, l_j\} \subseteq \mathrm{sp}(L)$ **do**:
   Calculate $\Delta_L(l_i, l_j \mid \mathcal{D}_N)$;
3. Select $\{l'_i, l'_j\} \subseteq \mathrm{sp}(L)$ s.t. $\Delta_L(l'_i, l'_j \mid \mathcal{D}_N)$ is maximized;
4. **If** $\Delta_L(l'_i, l'_j \mid \mathcal{D}_N) > 0$ **then**:
   Collapse states l'_i and l'_j; **goto** 2;
5. **Return** state-space of L.

It should be noted that Elidan and Friedman (2001) initialize their search with one state in L for each combination of the variables in the *Markov blanket* of L, whereas we use the smaller set of variables defined by $\mathrm{ch}(L)$. This is done to facilitate a semantic interpretation of the latent variables (described below), and it does not exclude any regular HNB models.[4]

Example 2 (Example 1 cont'd) The state-space of L' with $\mathrm{ch}(L') = \{A_2, A_3\}$ is collapsed by Algorithm 2 after L' is introduced. For large N the penalty term in MDL_p ensures that the state-space will be collapsed to two states mirroring the states of A_2 because L' will not significantly change the predictive likelihood from what the model previously held (note that $P(C = c \mid A_2, A_3, \mathcal{D}_N) \approx P(C = c \mid A_2, \mathcal{D}_N)$). Hence, by introducing L' we get a more robust classifier, where the classification noise introduced by A_3 is removed. The latent variable L'' with children $\mathrm{ch}(L'') = \{L', A_1\}$ will be introduced in the next iteration of Algorithm 1, and the target concept can eventually be learned.

An important side-effect of Algorithm 2 is that we can give a semantic interpretation to the state-spaces of the latent variables: $L \in \mathcal{L}$ *aggregates* the information from its children which is relevant for classification. If, for example, L is the parent of two binary variables A_1 and A_2, then Algorithm 2 is initiated s.t. L's state-space is $\mathrm{sp}(L) = \{A_1 = 0 \wedge A_2 = 0, A_1 = 0 \wedge A_2 = 1, A_1 = 1 \wedge A_2 = 0, A_1 = 1 \wedge A_2 = 1\}$. When the algorithm collapses states, we can still maintain an explicit semantics over the state-space, e.g., if the first and second state is collapsed we obtain a new state defined as $(A_1 = 0 \wedge A_2 = 0) \vee (A_1 = 0 \wedge A_2 = 1)$, i.e., $A_1 = 0$. Having such an interpretation can be of great importance when the model is put into use: The semantics allows a decision maker to inspect the "rules" that form the basis of a given classification. Through this insight she can consider whether the classification of the system should be overruled or accepted.

Another important aspect of the semantic interpretation, is that it allows us to *infer* data for the latent variables due to the deterministic relations encoded in the model. This

[4]Note that we do not consider regular HNB models with singly connected latent variables.

fact provides us with a fast calculation scheme, as we "observe" all the variables in $\mathcal{A}$ *and* $\mathcal{L}$. Therefore, it also follows that we can represent the HNB classifier using only the class variable and its children. Hence, the representation we will utilize is a Naïve Bayes structure where the "attributes" are represented by the variables which occur as children of the class variable in the HNB model. It is simple to realize that the number of free parameters required to represent this structure equals:

$$|\boldsymbol{\Theta}_{B_S}| = (|\mathrm{sp}\,(C)| - 1) + |\mathrm{sp}\,(C)| \sum_{X \in \mathrm{ch}\,(C)} (|\mathrm{sp}\,(X)| - 1)\,, \tag{6}$$

see also (Kočka and Zhang 2002). Hence, the difference in predictive MDL (used in Algorithm 2) can be approximated by:

$$\begin{aligned} \Delta_L(l_i, l_j) \;\approx\; & \log_2(N)\frac{|\mathrm{sp}\,(C)|}{2} \\ & - \sum_{c \in \mathrm{sp}\,(C)} N(c, l_i) \log_2 \left(\frac{N(c, l_i)}{N(c, l_i) + N(c, l_j)} \right) \\ & - \sum_{c \in \mathrm{sp}\,(C)} N(c, l_j) \log_2 \left(\frac{N(c, l_j)}{N(c, l_i) + N(c, l_j)} \right) \\ & + N(l_i) \log \left(\frac{N(l_i)}{N(l_i) + N(l_j)} \right) + N(l_j) \log \left(\frac{N(l_j)}{N(l_i) + N(l_j)} \right). \end{aligned} \tag{7}$$

Note again that the approximation is exact if the relationship between C and the children of C can be modelled using independence of causal influence.

4.2.3 The search boundary

By following the two-step procedure described above, the focusing algorithm produces a single candidate model $H' \in \mathcal{B}(H_k)$ to represent the search boundary. However, from our experiments we have found that picking out a single model to represent the search boundary is not an adequate representation of $\mathcal{B}(H_k)$. We can easily solve this drawback in at least two different ways:

i) Go through the candidate latent nodes one at a time in order of decreasing $Q(\cdot, \cdot \,|\, \mathcal{D}_N)$, and accept the first candidate model $H'' \in \mathcal{B}(H_k)$ for which $\mathrm{Score}(H'' \,|\, \mathcal{D}_N) > \mathrm{Score}(H_k \,|\, \mathcal{D}_N)$ in Step 2b of Algorithm 1.

ii) Limit the number of candidates used to represent the boundary to $\kappa > 1$ models, and do a greedy search over these models.

The first approach can be seen as a *hill-climbing* search, where we use Equation 4 to guide the search in the right direction. Step 2a will in this case not be a maximization over

$\mathcal{B}(H_k)$, but merely a search for a model which can be accepted in Step 2b. In Step 2a the algorithm may have to visit all models in the boundary $\mathcal{B}'(H_k) \subset \mathcal{B}(H_k)$ where $\mathcal{B}'(H_k)$ is defined s.t. each possible latent node is represented by exactly one state-space specification, i.e., a total of $O(n^2)$ models. On the other hand, the second approach will only examine κ models in Step 2a. It follows that alternative i) has higher computational complexity; in fact we may have to inspect $O(n^3)$ candidates before the algorithm terminates (Step 2 may be repeated $n-1$ times), and since inspecting each candidate latent variable involves costly calculations it may be computationally expensive. For the results reported in Section 5 we have therefore used the second alternative: A fixed number of candidate models ($\kappa = 10$) are selected from the search boundary, and the search proceeds as in Algorithm 1. The computational complexity of this approach is detailed in Section 4.3.

An immediate approach for implementing this refined algorithm would be to: 1) pick out the κ node pairs that have the strongest correlation (according to Equation 4), 2) find the associated state-spaces, and 3) select the model with the highest score in Step 2a. However, to increase the robustness of the algorithm, we do it slightly differently: Initially, we randomly partition the training data $\mathcal{D}_N$ in κ partly overlapping subsets, each containing $(\kappa - 1)/\kappa$ of the training data, and then each of these subsets are used to approximate the *best* model in the search boundary; this results in a list of up to κ different candidate models. We let these models represent $\mathcal{B}(H_k)$, and continue as if this was the whole boundary: If the best model amongst them (the one with the highest accuracy estimated by cross validation over the training data) is better than the current model candidate, we select that one and start all over again. If the best model is inferior to the current model, the search algorithm terminates, and the current model is returned (see Algorithm 3).

Algorithm 3 (Find HNB classifier)

1. Initiate model search with H_0;
2. Partition the training-set into κ partly overlapping subsets $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(\kappa)}$;
3. **For** $k = 0, 1, \ldots, n-1$
   (a) **For** $i = 1, \ldots, \kappa$
      i. Let $\{X^{(i)}, Y^{(i)}\} = \arg\max_{\{X,Y\} \subseteq \mathrm{ch}(C)} Q\left(X, Y \mid \mathcal{D}^{(i)}\right)$ (i.e., $\left\{X^{(i)}, Y^{(i)}\right\} \subseteq \mathrm{ch}(C)$ in H_k), and define the latent variable $L^{(i)}$ with children $\mathrm{ch}\left(L^{(i)}\right) = \left\{X^{(i)}, Y^{(i)}\right\}$;
      ii. Collapse the state-space of $L^{(i)}$ (Algorithm 2 with $\mathcal{D}^{(i)}$ used in place of $\mathcal{D}_N$);
      iii. Define $H^{(i)}$ by introducing $L^{(i)}$ into H_k;
   (b) $H' = \arg\max_{i=1,\ldots,\kappa} \mathrm{Score}\left(H^{(i)} \mid \mathcal{D}_N\right)$;

(c) **If** $\text{Score}(H' \mid \mathcal{D}_N) > \text{Score}(H_k \mid \mathcal{D}_N)$ **then**:
$H_{k+1} \leftarrow H'$; $k \leftarrow k+1$;
else
return H_k;

4. **Return** H_n;

4.3 Complexity analysis

When analyzing the complexity of the algorithm we can divide the description into three steps:

1) Find a candidate latent variable.

2) Find the state-space of a candidate latent variable, and check if it is useful.

3) Iterate until no more candidate latent variables are accepted.

Part 1
Proposing a candidate latent variable corresponds to finding the pair (X, Y) of variables having the strongest correlation (Equation 4). There are at most $(n^2 - n)/2$ such pairs, where n is the number of attribute variables. Calculating the conditional mutual information for a pair of variables can be done in time $O(N)$ (N being the number of cases in the database) hence, calculating the correlation measure for each pair of variables can be done in time $O(N \cdot n^2)$. Finally, the list is sorted (to accommodate future iterations), and the resulting time complexity is $O(n^2 \cdot (N + \log(n)))$.

Part 2
When determining the cardinality of a latent variable, L, we consider the gain of collapsing two states as compared to the current model; the gain is measured as the difference in predictive MDL. The time complexity of calculating the gain of collapsing two states is simply $O(N)$, see Equation 7. Due to local decomposability, the gain of collapsing two states has no effect on collapsing two other states, and there are therefore at most $(|\text{sp}(L)|^2 - |\text{sp}(L)|)/2$ possible combinations, i.e., $O(|\text{sp}(L)|^2 \cdot N)$. When two states are collapsed, $\Delta_L(\cdot, \cdot)$ must be calculated for $|\text{sp}(L)| - 1$ new state combinations, next time $|\text{sp}(L)| - 2$ state combinations are evaluated, and so on; the collapsing is performed at most $|\text{sp}(L)| - 1$ times. The time complexity of finding the state-space of a candidate latent variable is therefore $O\left(N \cdot |\text{sp}(L)|^2 + N \cdot |\text{sp}(L)| \left(|\text{sp}(L)| - 1\right)/2\right) = O(|\text{sp}(L)|^2 \cdot N)$.

Having found the cardinality of a candidate variable, say L, we test whether it should be included in the model using the wrapper approach. From the rule-based propagation method it is easy to see that the time complexity of this task is $O(n \cdot N)$. Thus, the time complexity of Part 2 is $O((n + |\text{sp}(L)|^2) \cdot N)$.

Part 3
Each time a latent variable is introduced we would in principle need to perform the above steps again, and the time complexity would therefore be $n-1$ times the time complexities above. However, as described below some of the previous calculations can be reused.

First of all, as $Q(X, Y|\mathcal{D})$ is a local measure we only need to calculate $Q(L, Z|\mathcal{D})$, $Z \in \text{ch}(C)$, where L is the latent variable introduced in the previous iteration. Moreover, since we need to calculate $Q(L, \cdot|\mathcal{D})$ at most $n-2$ times, the time complexity will be $O(n \cdot N)$, and, as the pairs (X, Y) are still sorted according to $Q(X, Y|\mathcal{D})$, we only need to sort $n-2$ pairs, i.e., after having included a latent variable the re-initialization of step 1 has complexity $O(n \cdot N + (n-1) \cdot \log(n-1)) = O(n \cdot (N + \log(n)))$.

Moreover, after having introduced a latent variable L with children X and Y, we cannot create another latent variable having either X or Y as a child (due to the structure of the HNB model). Thus, after having included a latent variable the cardinality of the resulting set of candidate pairs is reduced by $n-1$. This implies that we will perform at most $n-2$ re-initializations, thereby giving the overall time complexity $O(n^2 \cdot N + n \cdot (n \cdot (N + \log(n)) + (|\text{sp}(L)|^2 \cdot N))) = O(n^2 \cdot (\log(n) + |\text{sp}(L)|^2 \cdot N))$.

5 Empirical results

In this section we will investigate the merits of the proposed learning algorithm by using it to learn classifiers for a number of different domains. All data-sets are taken from the Irvine Machine Learning Repository (Blake and Merz 1998), see Table 1 for a summary of the 22 datasets used in this empirical study.

We have compared the results of the HNB classifier to those of the Naïve Bayes model (Duda and Hart 1973), the TAN model (Friedman et al. 1997), C5.0 (Quinlan 1998), and a standard implementation of neural networks with one hidden layer trained by back-propagation.[5] As some of the learning algorithms require discrete variables, the attributes were discretized using the entropy-based method of (Fayyad and Irani 1993). In addition, instances containing missing attribute-values were removed; all pre-processing was performed using MLC++ (Kohavi et al. 1994).

The accuracy-results are given in Table 2. For each dataset we have estimated the accuracy of each classifier (in percentage of instances which are correctly classified), and give a standard deviation of this estimate. The standard deviations are the theoretical values calculated according to (Kohavi 1995), and are not necessarily the same as the empirical standard deviations observed during cross validation. For comparison of the algorithms

[5]We used Clementine (SPSS Inc. 2002) to generate the C5.0 and neural network models. We have not compared our system to that of (Zhang et al. 2002) because of the high computational complexity of Zhang et al.'s algorithm. However, the numerical results reported by Zhang et al. (2002) point towards our model offering significantly better classification accuracy.

			#Inst					#Inst	
Database	#Att	#Cls	Train	Test	Database	#Att	#Cls	Train	Test
postop	8	3	90	CV(5)	cleve	13	2	296	CV(5)
iris	4	3	150	CV(5)	wine	13	3	178	CV(5)
monks-1	6	2	124	432	thyroid	5	3	215	CV(5)
monks-2	6	2	124	432	ecoli	7	8	336	CV(5)
monks-3	6	2	124	432	breast	10	2	683	CV(5)
glass	9	7	214	CV(5)	vote	16	2	435	CV(5)
glass2	9	2	163	CV(5)	crx	15	2	653	CV(5)
diabetes	8	2	768	CV(5)	australian	14	2	690	CV(5)
heart	13	2	270	CV(5)	chess	36	2	2130	1066
hepatitis	19	2	155	CV(5)	vehicle	18	4	846	CV(5)
pima	8	2	768	CV(5)	soybean-large	35	19	562	CV(5)

Table 1: A summary of the 22 databases used in the experiments: #Att indicates the number of attributes; #Cls is the number of classes; #Inst is the number of instances (given separately for training and test sets). CV(5) denotes 5-fold cross validation. Further details regarding the datasets can be found at the UCI Machine Learning Repository.

we made sure that the same cross validation folds were used for all the different learning methods. The best result for each dataset is given in boldface. We note that the HNB classifier achieves the best result for 10 of the 22 datasets, comes top-two for all but 5 datasets, and also has the best performance averaged over all datasets.

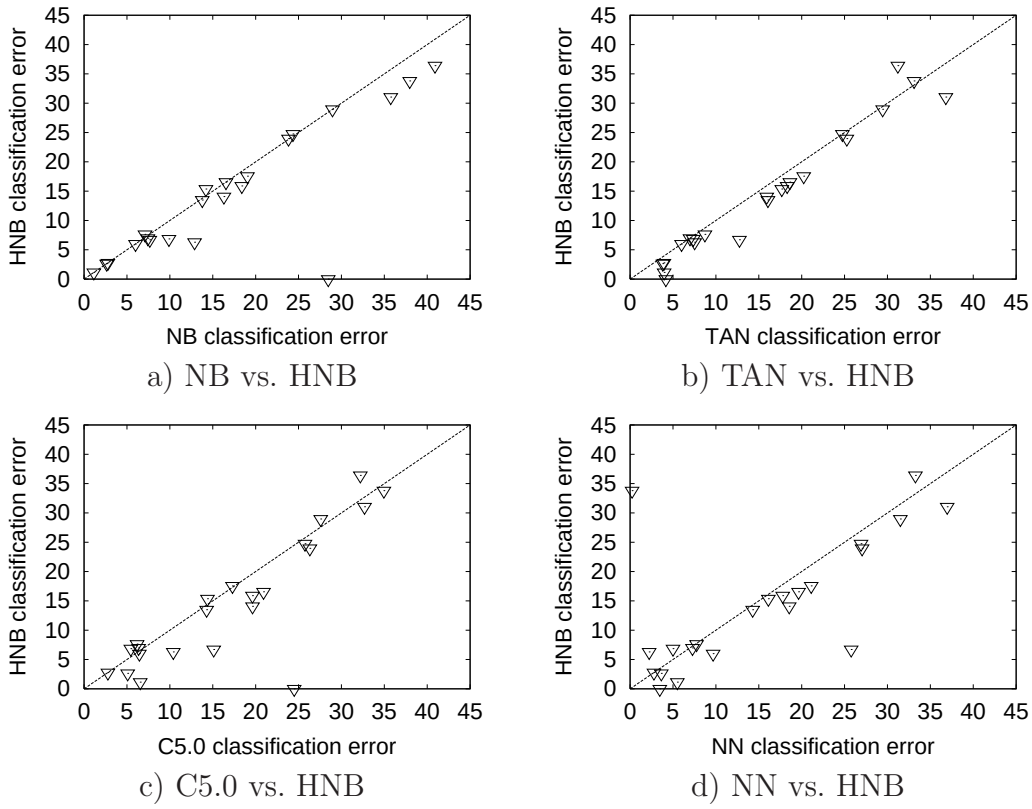
To quantify the difference between the HNB classifier and the other classifiers we advocate the method of (Kohavi 1995); Kohavi (1995) argues that the true merit of a classifier cannot be found by calculating the accuracy on a finite test-set. Instead we define α as the true accuracy of a classifier (only to be found if the target concept of the domain is known or fully described by an infinite test set), and we use $\hat{\alpha}$ to denote the estimate of α based on a test set of size N. Kohavi (1995) argues that $\hat{\alpha}$ is approximately Gaussian distributed with expectation α and variance $\alpha \cdot (1-\alpha)/N$ for large N. In our setting we have several datasets (indexed by $i = 1, \ldots, t$; t is the number of datasets, i.e., $t = 22$ in this study) and several classifier algorithms (indexed by j), and with this notation Kohavi's approximation can be written as $\hat{\alpha}_{ij} \sim \mathcal{N}(\alpha_{ij}, \alpha_{ij} \cdot (1-\alpha_{ij})/N_i)$. To simplify, we assume $\hat{\alpha}_{ij} \perp\!\!\!\perp \hat{\alpha}_{ik}$ for $j \neq k$ and $\hat{\alpha}_{ij} \perp\!\!\!\perp \hat{\alpha}_{\ell j}$ for $i \neq \ell$. Finally, we use the estimated standard deviation s_{ij} (given in Table 2) as if it was known. It follows that under the hypothesis that classifiers j and k are equally capable ($\alpha_{ij} = \alpha_{ik}$, $i = 1, \ldots, t$) then:

$$\Lambda_i(j,k) = \hat{\alpha}_{ij} - \hat{\alpha}_{ik} \sim \mathcal{N}(0, s_{ij}^2 + s_{ik}^2) \ , \quad \Lambda(j,k) = \sum_{i=1}^{t} \frac{\Lambda_i}{t} \sim \mathcal{N}\left(0, \sum_{i=1}^{t} \frac{s_{ij}^2 + s_{ik}^2}{t^2}\right) \ .$$

This enables us to test the hypothesis that the HNB classifier is not better than the other classifiers; more precisely we test the hypothesis H_0: $\Lambda(\cdot,\cdot) \leq 0$ against H_1: $\Lambda(\cdot,\cdot) > 0$,

Database	NB	TAN	C5.0	NN	HNB
postop	64.25+/-5.0	63.20+/-5.1	67.31+/-4.9	63.04+/-5.1	**68.95+/-4.9**
iris	**94.00+/-2.0**	**94.00+/-2.0**	93.55+/-2.0	90.32+/-2.4	**94.00+/-2.0**
monks-1	71.53+/-2.2	95.83+/-1.0	75.50+/-2.1	96.54+/-0.9	**100.0+/-0.1**
monks-2	62.04+/-2.3	66.90+/-2.3	65.05+/-2.3	**99.77+/-0.3**	66.20+/-2.0
monks-3	**97.22+/-0.8**	96.06+/-0.9	**97.22+/-0.8**	**97.22+/-0.8**	**97.22+/-0.8**
glass	71.04+/-3.1	70.56+/-3.1	**72.42+/-3.1**	68.50+/-3.2	71.04+/-3.1
glass2	81.61+/-3.0	81.69+/-3.0	80.37+/-3.1	82.21+/-3.0	**84.11+/-3.1**
diabetes	**75.65+/-1.5**	75.25+/-1.6	74.25+/-1.6	73.08+/-1.6	75.25+/-1.5
heart	83.70+/-2.2	84.07+/-2.2	80.36+/-2.4	81.45+/-2.4	**85.93+/-2.3**
hepatitis	92.34+/-2.1	87.25+/-2.7	84.89+/-2.9	74.23+/-3.5	**93.29+/-2.1**
pima	**76.17+/-1.5**	74.74+/-1.6	73.68+/-1.6	72.96+/-1.6	76.04+/-1.5
cleve	**83.46+/-2.1**	81.38+/-2.2	79.08+/-2.4	80.36+/-2.3	83.45+/-2.2
wine	**98.86+/-0.8**	96.03+/-1.5	93.45+/-1.9	94.49+/-1.7	**98.86+/-0.8**
thyroid	92.56+/-1.8	93.02+/-1.7	**93.64+/-1.7**	92.73+/-1.8	93.02+/-1.7
ecoli	80.95+/-2.1	79.76+/-2.2	**82.70+/-2.1**	78.89+/-2.2	82.44+/-2.1
breast	**97.36+/-0.6**	96.19+/-0.7	94.92+/-0.8	96.36+/-0.7	**97.36+/-0.6**
vote	90.11+/-1.4	92.64+/-1.3	94.55+/-1.1	**95.00+/-1.1**	93.15+/-1.3
crx	86.22+/-1.3	83.93+/-1.4	85.71+/-1.4	85.71+/-1.4	**86.51+/-1.3**
australian	**85.80+/-1.3**	82.32+/-1.5	85.61+/-1.3	83.88+/-1.4	84.64+/-1.4
chess	87.12+/-1.0	92.48+/-0.8	89.60+/-0.9	**97.78+/-0.5**	93.71+/-0.7
vehicle	59.09+/-1.7	**68.79+/-1.6**	67.80+/-1.6	66.74+/-1.6	63.59+/-1.7
soybean-large	92.90+/-1.0	91.28+/-1.1	**93.82+/-1.0**	92.25+/-1.1	92.36+/-1.1
Average	82.91	83.97	82.98	84.71	85.52

Table 2: Calculated accuracy for the 22 datasets used in the experiments. The results are given together with their theoretical standard deviation.



a) NB vs. HNB b) TAN vs. HNB

c) C5.0 vs. HNB d) NN vs. HNB

Figure 2: Scatter plot of classification error for HNB and a selection of other classification systems. In each plot, a point represents a dataset. The HNB's classification error is given on the x-axis, whereas the other system's error is given on the y-axis. Hence, data points below the diagonal corresponds to datasets where the HNB is superior, whereas points above the diagonal are datasets where the HNB classifier is inferior to the other system.

where the classifiers are labelled s.t. higher average accuracy for the HNB classifier coincides with a positive value of $\Lambda(\cdot, \cdot)$. With this setup H_0 is rejected at level $p = 5 \cdot 10^{-12}$ (NB), $p = 6 \cdot 10^{-6}$ (TAN), $p = 6 \cdot 10^{-11}$ (C5.0) and $p = .02$ (NN).

Finally, we note that in some of the domains the HNB models come up with an interesting latent structure. We are not experts to tell whether these structures are in fact meaningful, but some of them are at least worth attention. For example, in the heart model the HNB aggregates information about "Chest pain" and "Training induced angina". The probability of a heart disease increases slightly when chest pain is of a certain type; this probability can then again be increased dramatically if the instance also contains information about a training induced angina. Training induced angina has no effect in the model if chest pain is not of this particular type. Note that the classifier in this example uses the latent variable to encode *context specific independence* (Boutilier et al. 1996).

6 Discussion

6.1 Parameter learning

The parameters in the model are estimated by their maximum likelihood values. This may not be optimal for classification, and recent research has shown some improvement in classification accuracy when the parameters are chosen otherwise (Wettig et al. 2002). However, to support the interpretation of the empirical results in Section 5 we have deliberately not taken the opportunity of improving the classification accuracy further in this way. Optimization of the model is left for future work.

6.2 Finding candidate latent variables

As described by Example 1 and Example 2 the search for candidate latent variables may introduce a latent variable for a pair of variables which are marginally dependent, but where only one of the variables is actually dependent on the class variable C; as also shown is the examples, this does not jeopardize classification accuracy (actually it can be seen as a form of feature selection). Similarly, if several attributes are marginally dependent but independent of the class variable, the algorithm performs some redundant computations: For each such pair of attributes we include a latent variable, but as these attributes are independent of the class variable all states of such a latent variable are collapsed and the effect of the attributes on the classification result is removed.

Obviously both of the above mentioned problems can be overcome by simply performing a feature selection before initializing the learning algorithm. However, another approach would be to apply a correlation measure which directly considers the probability distribution over the class variable conditioned on the two variables X and Y in question. That is, the difference between the probability distribution $P(C|X,Y)$ and the probability distribution $P'(C|X,Y)$, where the latter is encoded by the model where $X \perp\!\!\!\perp Y|C$. This distance can be described using the well-known Kullback-Leibler (KL) divergence (Kullback and Leibler 1951) averaged over the possible states of X and Y:

$$\mathbb{E}(KL(P;P')|X,Y) = \sum_{x,y} P(x,y) \sum_{c} P(c|x,y) \log \left(\frac{P(c|x,y)}{P'(c|x,y)} \right).$$

In the context of classification, this distance measure can also be given another interpre-

tation by observing that:

$$\begin{aligned}
\mathbb{E}(KL(P;P')|X,Y) &= \sum_{c,x,y} P(c,x,y)\log\left(\frac{P(c,x,y)}{P(x,y)}\cdot\frac{1}{P'(c|x,y)}\right) \\
&= \sum_{c,x,y} P(c,x,y)\log\left(\frac{P(c,x,y)}{P(x,y)}\cdot\frac{\sum_c(P(x|c)P(y|c)P(c))}{P(x|c)P(y|c)P(c)}\right) \\
&= \sum_{x,y,c} P(x,y,c)\log\left(\frac{P(x,y|c)}{P(x|c)P(y|c)}\right) \\
&\qquad - \sum_{x,y} P(x,y)\log\left(\frac{P(x,y)}{\sum_c P(x|c)P(y|c)P(c)}\right) \\
&= I(X,Y|C) - KL(P(X,Y),P'(X,Y)).
\end{aligned}$$

Thus, the expected KL-divergence can be interpreted as the difference in conditional mutual information between X and Y conditioned on C, and the KL-divergence between $P(X,Y)$ in the unconstrained model and the model where $X \perp\!\!\!\perp Y|C$. In particular, if X and Y are marginally dependent but independent of the class variable C, we would have $\mathbb{E}(KL(P;P')|X,Y) = 0$ whereas $I(X,Y|C) > 0$ would have suggested that a latent variable should be introduced. Thus, this distance measure also takes into account that variables may be marginally dependent but independent of the class variable.

6.3 Inference and model structure

The algorithm for collapsing the state-space of a latent variable is the source of the semantics for these nodes, and in turn the reason why we can represent the HNB as a Naïve Bayes model with aggregations in place of the attributes. This compact representation requires a "deterministic inference engine" to calculate $P(C \mid \boldsymbol{a})$, because the aggregations defined by the semantics of the latent variables can in general not be encoded by the conditional probability tables for the variables. Assume, for instance, that we have three binary variables L, X, Y, $\mathrm{ch}(L) = \{X, Y\}$, and "$L = 1$ if and only if $X = Y$". This relationship cannot be encoded in the model $X \leftarrow L \rightarrow Y$, and to infer the state of the latent variable L from X and Y we would therefore need to design a special inference algorithm which explicitly uses the semantics of L. To alleviate this potential drawback we can simply redefine the network-structure: Introduce a new latent variable L', and change the network structure s.t. $\mathrm{ch}(L) = \mathrm{pa}(X) = \mathrm{pa}(Y) = \{L'\}$; L' is equipped with at most one state for each possible combination of its children's states. This enlarged structure is capable of encoding any relation between $\{X, Y\}$ and L using the conditional probability tables only. Hence, the enlarged structure can be handled by any standard BN propagation algorithm and, since the structure is still an HNB, the inference can be performed extremely fast.

7 Concluding remarks

In this paper we have used Hierarchical Naïve Bayes models for classification, and through experiments we have shown that the HNB classifiers offer results that are significantly better than those of other commonly used classification methods. Moreover, a number of existing tools may be able to improve the classification accuracy even further. These include feature selection (Kohavi and John 1997), smoothing (significant improvements reported by (Friedman et al. 1997) for some model classes), and supervised learning of the probability parameters (Wettig et al. 2002). We leave the investigation of these sources of potential improvements for future work. Finally, the proposed learning algorithm also provides an explicit semantics for the latent structure of a model. This allows a decision maker to easily deduce the rules which govern the classification of some instance hence, the semantics may also increase the user's confidence in the model.

Acknowledgements

We have benefited from interesting discussions with the members of the Decision Support Systems group at Aalborg University, in particular Tomás Kočka, Nevin L. Zhang, and Jiří Vomlel. We would like to thank `Hugin Expert` (www.hugin.com) for giving us access to *Hugin Decision Engine* which forms the basis for our implementation. The first author was supported by a grant from the Research Council of Norway.

References

Blake, C. and C. Merz (1998). UCI repository of machine learning databases. URL: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Boutilier, C., N. Friedman, M. Goldszmidt, and D. Koller (1996). Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, Portland, OR., pp. 115–123.

Chow, C. K. and C. Liu (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory 14*, 462–467.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Sciences. New York: Springer Verlag.

Domingos, P. and M. Pazzani (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning 29*(2–3), 103–130.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons.

Elidan, G. and N. Friedman (2001). Learning the dimensionality of hidden variables. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 144–151. Morgan Kaufmann Publishers.

Fayyad, U. M. and K. B. Irani (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, San Mateo, CA., pp. 1022–1027. Morgan Kaufmann Publishers.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse of dimensionality. *Data Mining and Knowledge Discovery 1*(1), 55–77.

Friedman, N., D. Geiger, and M. Goldszmidt (1997). Bayesian network classifiers. *Machine Learning 29*(2–3), 131–163.

Greiner, R., A. J. Grove, and D. Schuurmans (1997). Learning Bayesian nets that perform well. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 198–207. Morgan Kaufmann Publishers.

Heckerman, D. and J. S. Breese (1994). A new look at causal independence. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 286–292. Morgan Kaufmann Publishers.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. New York: Springer Verlag.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, San Mateo, CA., pp. 1137–1143. Morgan Kaufmann Publishers.

Kohavi, R., G. John, R. Long, D. Manley, and K. Pfleger (1994). MLC++: A machine learning library in C++. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, pp. 740–743. IEEE Computer Society Press.

Kohavi, R. and G. H. John (1997). Wrappers for feature subset selection. *Artificial Intelligence 97*(1–2), 273–324.

Kononenko, I. (1991). Semi-naive Bayesian classifier. In *Proceedings of Sixth European Working Session on Learning*, Berlin. Springer Verlag.

Kočka, T. and N. L. Zhang (2002). Dimension correction for hierarchical latent class models. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 267–274. Morgan Kaufmann Publishers.

Kullback, S. and R. A. Leibler (1951). On information and sufficiency. *Annals of Mathematical Statistics 22*, 79–86.

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

Mitchell, T. M. (1997). *Machine Learning*. Boston, MA.: McGraw Hill.

Pazzani, M. (1995). Searching for dependencies in Bayesian classifiers. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA.: Morgan Kaufmann Publishers.

Quinlan, R. (1998). C5.0: An informal tutorial. Available from the internet at URL: http://www.rulequest.com/see5-unix.html.

Rissanen, J. (1978). Modelling by shortest data description. *Automatica 14*, 465–471.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics 6*, 461–464.

SPSS Inc. (2002). Clementine v6.5. http://www.spss.com/spssbi/clementine/index.htm.

Wettig, H., P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri (2002). On supervised learning of Bayesian network parameters. HIIT Technical Report 2002-1, Helsinki Institute for Information Technology.

Whittaker, J. (1990). *Graphical models in applied multivariate statistics*. Chichester: John Wiley & Sons.

Zhang, N. (2002). Hierarchical latent class models for cluster analysis. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, Menlo Park, CA., pp. 230–237. AAAI Press.

Zhang, N., T. D. Nielsen, and F. V. Jensen (2002). Latent variable discovery in classification models. Available from the first author upon request.

IV

Parameter Learning in Object Oriented Bayesian Networks


Annals of Mathematics and Artificial Intelligence **32:** 221–243, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

Parameter learning in object-oriented Bayesian networks

Helge Langseth [a,b] and Olav Bangsø [b]

[a] *Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway*
E-mail: helgel@math.ntnu.no
[b] *Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Øst, Denmark*
E-mail: {hl, bangsy}@cs.auc.dk

This paper describes a method for parameter learning in Object-Oriented Bayesian Networks (OOBNs). We propose a methodology for learning parameters in OOBNs, and prove that maintaining the object orientation imposed by the prior model will increase the learning speed in object-oriented domains. We also propose a method to efficiently estimate the probability parameters in domains that are *not* strictly object oriented. Finally, we attack type uncertainty, a special case of model uncertainty typical to object-oriented domains.

Keywords: Bayesian networks, object orientation, learning

AMS subject classification: 68T05

1. Introduction

Bayesian Networks (BNs) [21,32] have established themselves as a powerful tool in many areas of artificial intelligence, including planning, vision, decision support systems and robotics. However, one of the main obstacles is to create and maintain very large domain models. To remedy this problem, object-oriented versions of the BN framework have been proposed in the literature [4,22]. Object-Oriented BNs (OOBNs) as defined in these papers offer an easy way of creating BNs, but the problem of assessing and maintaining the probability estimates still remain; conventional learning algorithms like [6] do not exploit that the domain is object oriented while learning.

In this paper we propose a learning method that is applied directly to the OOBN specification. It is proven that this learning method is superior to conventional learning methods in object oriented domains, and a method to efficiently estimate the probability parameters in domains that are *not* strictly object oriented is also proposed.

This paper is organized as follows: The rest of this section will create a starting point for our analysis by introducing OOBNs and the required notation and assumptions. In section 2 we outline the proposed learning method, and in section 3 we propose a framework for learning in domains that are only approximately object oriented. A special case of model uncertainty, typical to object-oriented domains, is handled in section 4, and we conclude in section 5.

1.1. Object-oriented Bayesian networks

Using small and "easy-to-read" pieces of a complex model is an already applied technique for constructing large Bayesian networks. For instance, [34] introduces the concept of sub-networks which can be viewed and edited separately even if they are different pieces of the same network; [37] adds levels of integration of fragments (using an analogy with Boolean circuits); [25] is concerned with the combination of fragments (using conditional noisy-MIN). Frameworks for such representations called Object-Oriented Bayesian Networks are presented in [4,22]. An introduction to the framework of [4] will be given in this section, as it is the foundation for our work on learning in OOBNs.

OOBNs as defined by [4] will be described in the following by way of an example adapted from that paper. The example will be used throughout the paper to illustrate the proposed learning mechanism and to show how well it works. We limit our description of the framework to those parts that are most relevant for learning in OOBNs; further details can be found in [3,4]. **This font** will be used to describe classes, instantiations of classes are described using THIS FONT, and *this font* is employed when referring to variables.

Old McDonald (OMD) has a farm with 2 milk cows and 2 meat cows. A milk cow primarily produces milk and a meat cow primarily produces meat. OMD wants to model his stock using OOBN classes. OMD constructs a **Generic cow** as shown in figure 1. He knows that what a cow eats and who its mother is influences how much milk and meat it produces. OMD wants *Mother* and *Food* to be *input nodes*; an input node is a reference to a node outside the class. OMD wants *Milk* and *Meat* to be *output nodes*, nodes from a class usable outside the instantiations of the class. Dashed ellipses represent input nodes and shaded ellipses represent output nodes, see figure 1. Input and output nodes form the interface between an instantiation and the context in which the instantiation exists. Nodes in an instantiation that are neither input nor output nodes are termed *normal nodes*. A class may be instantiated several times with different nodes having influence on the different instantiations through the input nodes, so only the number of states of the input nodes is known at the time of specification (e.g., the cows might have different mothers).

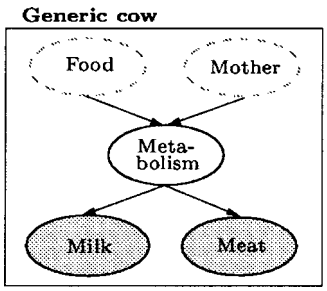


Figure 1. The **Generic cow** class as defined by OMD. The arrows are links as in normal BNs. The dashed ellipses are input nodes, and the shaded ellipses are output nodes.

OMD consults an expert that tells him that he might want to get specifications of both a **Milk cow** and a **Meat cow**, which OMD agrees to. The two new cow specifications, shown in figure 2, are subclasses of the **Generic cow** class (hence the "IS A **Generic cow**" in the top left of each of the class specifications). 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 tables (CPTs) 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 antisymmetric, so to avoid cycles it is required that a subclass of a class cannot be a superclass of this 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*. All trees from the class hierarchy forest can be arranged so that the unique node with no superclass is the root, and all other nodes of the tree have their superclass as parent. Such a tree is called a *class tree*.

OMD continues by constructing a **Stock** class representing his live-stock. In figure 3 the boxes are instantiations, e.g., Cow1 is an instantiation of the class **Meat cow**.

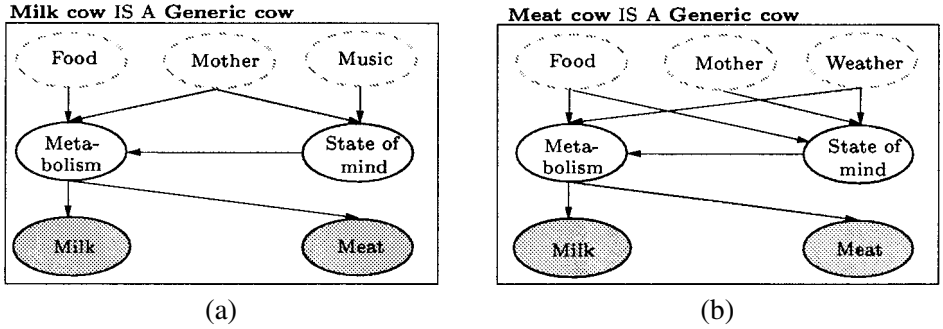


Figure 2. (a) The experts specification of a **Milk cow**. (b) The experts specification of a **Meat cow**. Note that their input sets are larger than the input set of the **Generic cow** (figure 1).

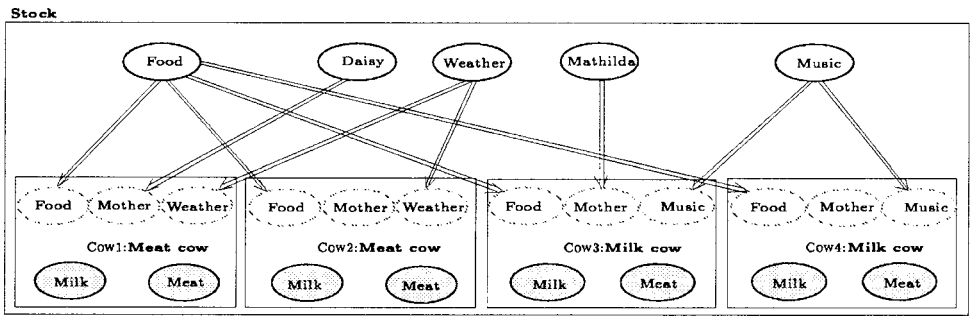


Figure 3. The **Stock** with two instantiations of the **Milk cow** class and two instantiations of the **Meat cow** class. Note that some input nodes are not referencing any nodes.

This 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 the instantiation available to the encapsulating class (**Stock**). The double arrows are *reference links,* where the leaf of a link is a reference to the root of that link;[1] e.g., the input node *Mother* of Cow1 is a reference to the node *Daisy*. This means that whenever the node *Mother* is to be used inside the instantiation Cow1, the node *Daisy* will be the node actually used.

As the subclasses in a class tree may have a larger set of nodes than their superclass, the input set of a subclass **S** might be larger than the input set of its superclass **C**. If an instantiation of **S** is used instead of an instantiation of **C**, the extra input nodes will not be referencing a node. To ensure that these nodes contain a potential, the notion of a *default potential* is introduced: a default potential is a probability distribution over the states of an input node, which is used when the input node is not referencing any node. A default potential can also be used when no reference link is specified, even if the reason for it is not subclassing. 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 noting that the structure of references is always a tree or a forest; cycles of reference links are not possible [3]. These trees consist of a unique root and one or more leaf-nodes; there are only two "layers" in these structures in our case.

Inference can be performed by translating the OOBN into a *multiply-sectioned Bayesian network* [41,42], see [3] for details on this translation, or by constructing the *underlying BN*. The underlying BN, BN_{I}, of an instantiation I is constructed using the following algorithm, assuming I to define a legal OOBN:

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 contained 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 the default potential. Figure 4 describes the underlying BN of OMD's instantiation of the **Stock**-class (figure 3) as found by the above algorithm.

[1] To avoid confusion with the normal links in the model we do not use the terms "parent" and "child" when referring to reference links.

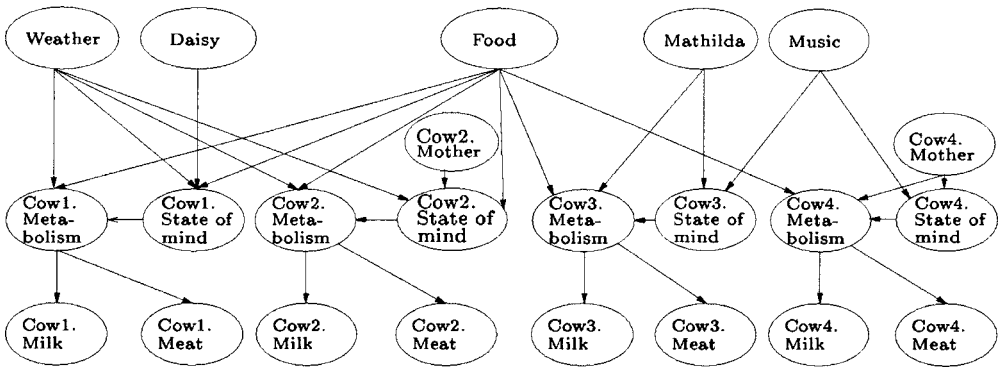


Figure 4. The underlying BN for OMD's instantiation of the **Stock** class.

1.2. Notation and assumptions

The following is a description of the most important assumptions we make throughout the paper, and an introduction to the distance measure we use to evaluate the learning methods we propose. We will use standard terminology from the learning community, and do not follow the OOBN terminology if not necessary.

The domain of interest is modeled by a stochastic vector $\boldsymbol{X} = (X_1, \ldots, X_m)$ of dimension m, where $\boldsymbol{X}$ is distributed according to an unknown distribution function $f(\boldsymbol{x}|\boldsymbol{\theta})$. $\boldsymbol{\theta}$ is the (unknown) vector of parameters determining the distribution. The vector $\boldsymbol{X}$ is sampled "regularly", and the observations are stored in a database D. The database is of size N, $D = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N\}$. We will assume that the cases in the database are identically and independently distributed given $f(\cdot|\boldsymbol{\theta})$

The distribution $f(\cdot|\boldsymbol{\theta})$ is assumed to belong to a known parametric distribution family $\mathcal{F}$, so the estimation problems boils down to estimating the parameters $\boldsymbol{\theta}$ of the distribution. Stated as a BN learning task, this assumption corresponds to assuming that the *structure* of the BN is known (see [2] for a description of learning in object-oriented domains when also the structure is unknown a priori). We use $\widehat{\boldsymbol{\theta}}$ to denote the estimate of $\boldsymbol{\theta}$. The domain of a variable is assumed to be *discrete* meaning that X_i takes its values in a finite universe $\mathcal{X}_i$, $i = 1, \ldots, m$, and $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_m = \mathcal{X}$; $\boldsymbol{x}$ is a configuration over $\mathcal{X}$. The probability distribution estimated from N samples will be denoted by $f(\boldsymbol{x}|\widehat{\boldsymbol{\theta}}_N)$ or simply $\widehat{f}_N$. The unknown "true" distribution function is called $f(\boldsymbol{x}|\boldsymbol{\theta})$ or f.

As this work is within the framework of discrete Bayesian networks, the family of distribution functions $\mathcal{F}$ can be characterized by the fact that $f(\boldsymbol{x}|\boldsymbol{\theta})$ takes the form of a product of m conditional probability tables $P(X_i = x_i|pa(X_i))$ where $pa(X_i)$ denotes X_i's parents in the Bayesian network. The event that $pa(X_i)$ takes on a particular configuration j in some enumeration of the possible configurations is denoted by $pa(X_i) = j$. Furthermore, we will use θ_{ijk} to denote the probability $P(X_i = k|pa(X_i) = j)$, and we

will assume $0 < \theta_{ijk} < 1$ to avoid trivial deterministic cases of learning.[2] We will let $|\boldsymbol{\theta}|$ denote the dimension of the parameter space, meaning the smallest possible number of free parameters that can encode $f(\boldsymbol{x}|\boldsymbol{\theta})$ correctly. This is not the same as the sum of the sizes of the CPTs, since one can, e.g., encode the distribution of a binary variable X by using only the one parameter p, i.e., $P(X = 1) = p,\ P(X = 0) = 1 - P(X = 1) = 1 - p$. $|\boldsymbol{\theta}|$ is, furthermore, not to be calculated directly from the dimension of $\mathcal{X}$, since a Bayesian network (that is not a complete graph) utilizes a more compact representation.

The work presented in this paper focuses on the *maximum likelihood* estimates of the parameters. To generate the maximum likelihood estimates we use the EM-algorithm [12]. The EM-algorithm is particularly easy to implement in graphical models [26], but there are problematic issues both regarding speed of convergence as well as convergence towards a local (sub-optimal) maximum of the likelihood function. The first of these problems can be overcome by different acceleration measures, see, e.g., [31,38], the second problem is typically managed by a series of random restarts of the iteration process after convergence of the EM algorithm.

The work described here does not consider the use of parameter priors in the learning algorithms. The reason for this is that we want to build our theory around the asymptotic properties of the estimators we find, i.e., when the sample-size $N \to \infty$. The focus on maximum likelihood estimators is not constraining our results, as Bayesian estimators will converge towards the maximum likelihood estimators if the priors are strictly positive over the parameter space, see e.g., [27, p. 512]. Note that we can also find the Bayesian maximum a posteriori estimators within the EM framework by following [16]. Note also that the convergence towards the estimators' large-sample distribution is quite rapid in our examples, so the focus on asymptotic results does not constrain the applicability of the results.

For simplicity we will assume the data to be *Missing Completely At Random* (MCAR), see [28]. Informally, this means that the observability of one variable is independent of the value of any other variable (both missing and observed). Note that variables that are always missing (so called "*hidden*") also obey the MCAR assumption. The extension to *Missing at Random* (MAR) [19], which informally means that the MCAR assumption is relaxed to allow the pattern of missingness to depend on the values of the observed variables, is immediate. The extension is, however, left out for clarity of exposition.

The quality of the learned distribution will be measured with respect to the Kullback–Leibler divergence (KL divergence) between the estimated and "true" distrib-

[2] This assumption is made for simplicity of exposition, and is not needed for the results to be valid. The learning speed in a domain with deterministic nodes, as measured the way we do in this paper, is the same as the learning speed in the same domain where deterministic nodes are considered fixed. Hence, including deterministic nodes only gives a more tedious notation, and does not jeopardize the underlying mathematics.

utions, $D_N(\widehat{f}\,\|\,f)$, which is calculated as

$$D\big(\widehat{f}_N\,\|\,f\big) = \sum_{\boldsymbol{x}\in\mathcal{X}} f\big(\boldsymbol{x}|\widehat{\boldsymbol{\theta}}_N\big)\cdot\log\left[\frac{f(\boldsymbol{x}|\widehat{\boldsymbol{\theta}}_N)}{f(\boldsymbol{x}|\boldsymbol{\theta})}\right] = \mathbb{E}_{\widehat{\boldsymbol{\theta}}}\log\left[\frac{f(\boldsymbol{X}|\widehat{\boldsymbol{\theta}}_N)}{f(\boldsymbol{X}|\boldsymbol{\theta})}\right]. \tag{1}$$

The expectation $\mathbb{E}_{\widehat{\boldsymbol{\theta}}}$ is taken with respect to the estimated distribution $\widehat{f}_N$. This expectation can be calculated without expanding the sum in equation (1), see [9, chapter 6].

There are many arguments for using this particular measurement for calculating the quality of the approximation, see [8]. One of them is the fact that the KL divergence bound the maximum error in the assessed probability for a particular event A [40, proposition 4.3.7],

$$\sup_A\left|\sum_{\boldsymbol{x}\in A} f\big(\boldsymbol{x}|\widehat{\boldsymbol{\theta}}_N\big) - \sum_{\boldsymbol{x}\in A} f(\boldsymbol{x}|\boldsymbol{\theta})\right| \leqslant \sqrt{\frac{1}{2}D\big(\widehat{f}_N\,\|\,f\big)}.$$

Similar results for the maximal error of the estimated conditional distribution are derived in [39]. These results have made the KL divergence the "distance measure"[3] of choice in Bayesian network learning, see e.g., [11,14,18,23,32]. We have chosen to use the empirical KL divergence $D(\widehat{f}_N\,\|\,f)$ instead of $D(f\,\|\,\widehat{f}_N)$ since the former is finite (with probability 1), and therefore simplifies the asymptotic expansion. Results similar to ours can be obtained for $D(f\,\|\,\widehat{f}_N)$ by use of *bounded approximations* [1] for the divergence measure.

For the OOBN learning to be meaningful, we will initially assume that the domain is in fact object oriented, such that the CPTs of one instantiation of a class are identical to the corresponding CPTs of any other instantiation of that class. We call this *the OO assumption*. In section 3 we will investigate what happens if this assumption is violated.

2. OOBN learning

As described in section 1.1 a class hierarchy is by definition a forest containing trees of classes that are subclasses of their parents in the tree. Given a class hierarchy, and data for some instantiations of the classes in the hierarchy, we want to learn from the data. The way this is done is described in the following.

The typical way to learn from data is to learn in the underlying BN, but this does not take advantage of the object oriented specification, and it will (probably) violate the OO assumption as well. According to this assumption, instantiations of a class are identical. To take advantage of the OOBN specification, the learning method we propose learns in the class specification instead of in each instantiation. This means that every observation of a class instantiation will be treated as a (virtual) case from the class.

The CPTs are only represented in a class if the CPT is different from that of the superclass (if one exists). As an example, consider the definition of **Generic cow** given

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

in figure 1, and its subclass **Milk cow** shown in figure 2(a). The CPTs for *Music* and *State of mind* must be defined in **Milk cow**, since these variables are not defined in **Generic cow**. Furthermore, since the parent set of *Metabolism* is different in the two class specifications, the CPT for *Metabolism* must be specified in both **Generic cow** and **Milk cow**. The CPTs for *Food, Mother, Milk* and *Meat* need only be specified in the **Generic cow** class (figure 1). It is possible for *Food, Mother, Milk* and *Meat* in **Milk cow** to differ from those of the **Generic cow** specification, and in that case the CPTs will be defined in both specifications.

The *scope* of a CPT specification associated with a node X_T is defined as follows. Let $\mathbf{C}_T$ be the class where the node X_T is defined for the first time (meaning that X_T is not defined in the superclass of $\mathbf{C}_T$, if one exists). Then, the scope of the CPT of X_T is a substructure of the class tree with $\mathbf{C}_T$ as the root. Each subclass of $\mathbf{C}_T$ is a member of the scope if and only if the CPT is not overwritten in that subclass. See figure 5 for an example class tree. Let $\mathcal{A}$ be the set of classes that are included in the scope. Then the subclasses of the members of $\mathcal{A}$ are evaluated for inclusion in $\mathcal{A}$ using the same rule, and this is done recursively throughout the class tree. For each of the subclasses of $\mathbf{C}_T$ that are not included in $\mathcal{A}$, the scope of their CPT specifications can be found in the same way. It is now easy to see that the scopes of the CPTs associated with X_T will partition the class tree into substructures that are trees. The intersection of the scopes are empty, and the union of the scopes is the whole substructure for which the variable of the CPT is defined, i.e., the class tree rooted at $\mathbf{C}_T$. When learning is to be performed, it will be done where the CPTs are specified. This means that learning of a given CPT based on data from an instantiation of a class will be performed in the root of the substructure defined by the scope of that CPT. As an example, consider the **Generic cow** and **Milk cow** classes in figures 1 and 2(a); **Generic cow** is the superclass, and **Milk cow** is the subclass. Assume we have observed some data from an instantiation of the **Milk cow** class, and want to update the CPTs of *Milk* and *Metabolism*. The scope of the

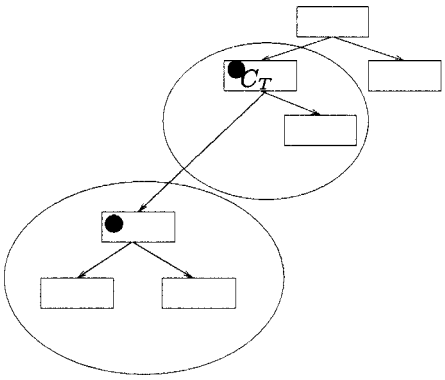


Figure 5. A class tree that shows the scope of the two definitions of the CPT for node X_T. Classes where a CPT for X_T is defined are marked with a filled circle. Since a CPT for X_T is defined twice in this class tree, there are two non-overlapping scope definitions that partition the class tree into three parts: One part where the first CPT is valid, one where the second is valid and one where the node X_T is not defined.

Milk specification in the class tree is equal to the whole tree (we assume that *Milk* is not overwritten in the subclass). Learning of the CPT for *Milk* will therefore be performed in the root of the class tree, i.e., in the **Generic cow** class. The CPT of *Metabolism* is overwritten in the **Milk cow** specification, so learning of *Metabolism* is performed in the **Milk cow** class. Note that no learning is performed in the instantiations; we do not update the CPTs of the underlying BN during learning. After a re-compilation of the OOBN, the CPTs from the class specifications are distributed to the instantiations as described in [4], and at that point the underlying BN is updated as well.

One of the consequences of this is that another subclass of **Generic cow**, say **Meat cow**, might be updated because of the learning performed in **Milk cow**. In figure 2(b) the class specification for **Meat cow** is shown. This class has the same CPTs in *Food, Mother, Milk* and *Meat* as **Generic cow** (we assume they are not overwritten in **Meat cow**). Hence, the data from the instantiation of **Milk cow** used to update *Milk* will also change the instantiations of **Meat cow**. If this is not desirable, the CPTs of **Generic cow** should be overwritten in the subclasses, e.g., the milk production of a milk cow could be different from a generic cow, and the meat production could be different for meat cows.

In addition to maintaining the OO assumption, the proposed learning algorithm also has another important effect. If at least one of the CPTs are shared by more than one instantiation, the number of parameters to learn is reduced. This is desirable, as shown in the following.

2.1. The case of no missing data

When the database D is complete, i.e., we have no missing values, the learning theory becomes particularly easy. To recapitulate, we have N independent realizations from a distribution with distribution function f. Since the data is complete we can find the maximum likelihood approximation $\widehat{f}_N$ by using closed-form equations instead of applying the iterative EM algorithm. To test the learning algorithms we thereafter calculate the KL divergence $D(\widehat{f}_N \| f)$ between the estimated distribution $\widehat{f}_N$ and the "true" distribution f. Let $\xrightarrow{\mathcal{L}}$ denote convergence in distribution, meaning that if we have an infinite sequence $\{X_1, X_2, \ldots\}$, then we write $X_n \xrightarrow{\mathcal{L}} X$ if and only if the distribution functions $F_n(x)$ of X_n converge to the distribution function $F(x)$ of X for any continuity point x of F, where $F(x) = \sum_{x' \leqslant x} f(x')$ [27, definition 2.3.2]. Using large sample theory it is easy (see [24] for details) to verify that when $\widehat{\boldsymbol{\theta}}$ is an unbiased estimator of $\boldsymbol{\theta}$, then

$$2N \cdot D\big(\widehat{f}_N \| f\big) \xrightarrow{\mathcal{L}} \sum_{i=1}^{|\boldsymbol{\theta}|} \left(\frac{\widehat{\theta}_i - \theta_i}{\tau_i} \right)^2,$$

where τ_i^2 is the Cramér–Rao lower bound for the variance of an unbiased estimator for θ_i, defined in [10, chapter 32]. Using this result, and the fact that we have complete data,

$2N \cdot D(\widehat{f}_N \| f)$ converges towards a particular χ^2 distribution:

$$2N \cdot D\big(\widehat{f}_N \| f\big) \xrightarrow{\mathcal{L}} X \sim \chi^2_p, \tag{2}$$

where $p = |\boldsymbol{\theta}|$ is the size of the parameter space of f. Hence, as N grows large, we have an easily interpretable relationship for the expected value of the KL divergence

$$\lim_{N\to\infty} 2N \cdot \mathbb{E}\big[D\big(\widehat{f}_N \| f\big)\big] = |\boldsymbol{\theta}|, \tag{3}$$

which may be formulated as

$$\mathbb{E}\big[D\big(\widehat{f}_N \| f\big)\big] \approx \frac{|\boldsymbol{\theta}|}{2N} \tag{4}$$

for large N. Thus, not surprisingly, having fewer parameters will increase the expected learning speed as measured by the empirical KL divergence. Object-oriented learning reduces the number of parameters to learn. Since learning is done in the class specification, we get fewer parameters to estimate (by constraining some of the existing parameters in the underlying BN to be identical).

We define $\widetilde{p}$, the *effective number of parameters* for the object-oriented learning, as the number of free parameters in the object oriented model. Hence, $\widetilde{p}$ is made up by the sum of the free parameters in the CPTs of the *class specifications* instantiated in the OOBN. Remember that the complete OOBN is also an instantiation of a class (OMD's **Stock**-class). The number of parameters in the instantiations are not counted, as they are forced to be identical to the parameters in the class definitions.

To see that equation (2) is valid in object oriented learning with $p = \widetilde{p}$, the key property we need is that for a class with k instantiations, *observing one case with all the k instantiations of the class has the same effect for learning the parameters in the object oriented model as observing k hypothetical cases of the class*. This follows trivially from the asymptotic theory of statistics, as outlined below. Note that we suppress all technicalities from this discussion and without notice make use of the smoothness and strict positivity of the distribution functions, and that all quantities involved are finite with probability 1. The presentation below is based on [27], and in particular, chapter 7 of that book.

In the current setting, it is well known that the Maximum Likelihood estimates $\widehat{\boldsymbol{\theta}}_N$ are asymptotically Gaussian distributed with mean $\boldsymbol{\theta}$ and some variance $\boldsymbol{\Sigma}$, i.e., $\widehat{\boldsymbol{\theta}}_N \xrightarrow{\mathcal{L}} \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma})$. The Fisher Information matrix $\boldsymbol{I}$ is the $|\boldsymbol{\theta}| \times |\boldsymbol{\theta}|$ matrix defined by

$$I_{ij} = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_i \partial\theta_j} \log f(\boldsymbol{X}|\boldsymbol{\theta})\right].$$

The asymptotic variance of the maximum likelihood estimator $\widehat{\boldsymbol{\theta}}_N$ can now be defined by the Fisher information, $\boldsymbol{\Sigma} = (1/N)\boldsymbol{I}^{-1}$ (given certain regularity conditions that are fulfilled in the setting of our work).

Let $\boldsymbol{Y}$ and $\boldsymbol{Z}$ be random variables distributed with density $f_{\boldsymbol{\theta}}(\cdot)$ and $g_{\boldsymbol{\theta}}(\cdot)$, respectively. Furthermore, let the information about $\boldsymbol{\theta}$ from $\boldsymbol{Y}$ and $\boldsymbol{Z}$ be denoted $\boldsymbol{I}_{\boldsymbol{Y}}$ and $\boldsymbol{I}_{\boldsymbol{Z}}$,

respectively. The information available by the sample $\{\boldsymbol{Y}, \boldsymbol{Z}\}$, called $\boldsymbol{I}_{\{\boldsymbol{Y},\boldsymbol{Z}\}}$, is by using [27, theorem 7.2.2] given as

$$\boldsymbol{I}_{\{\boldsymbol{Y},\boldsymbol{Z}\}} = \boldsymbol{I}_{\boldsymbol{Y}} + \boldsymbol{I}_{\boldsymbol{Z}} \tag{5}$$

when

$$\frac{\partial}{\partial\theta_i} \log f_{\boldsymbol{\theta}}(\boldsymbol{Y}) \perp\!\!\!\perp \frac{\partial}{\partial\theta_j} \log g_{\boldsymbol{\theta}}(\boldsymbol{Z}), \quad i, j = 1, \ldots, |\boldsymbol{\theta}|.$$

Since the maximum likelihood estimators are asymptotically efficient [27, section 7.6], and the empirical KL divergence is a function of $\widehat{\boldsymbol{\theta}}$ through the parameter variances only, see [24], the information about $\boldsymbol{\theta}$ in k instantiations equals the sum of the information in k imaginary cases of the class, as long as there are no missing data in the database. The fact that equation (2) is valid in object-oriented learning with $p = \widetilde{p}$ follows.

To test the object oriented learning method, consider the example of OMD's farm as described in section 1.1. Assume OMD measures all the variables of the domain regularly, and stores them in a database. He wishes to estimate the parameters in his domain, and uses both the conventional as well as the object-oriented learning methods. The results are displayed in figure 6, where the asymptotic values of the expected KL divergence of the two methods as a function of N according to equation (4) are indicated as well. The conventional learning algorithm has 634 parameters to estimate, whereas the object-oriented domain only has 322. Hence, according to equation (4) the KL divergence of the conventional learning algorithm is approximately $634/322 \approx 1.97$ times as large as the one of object-oriented learning for large N.

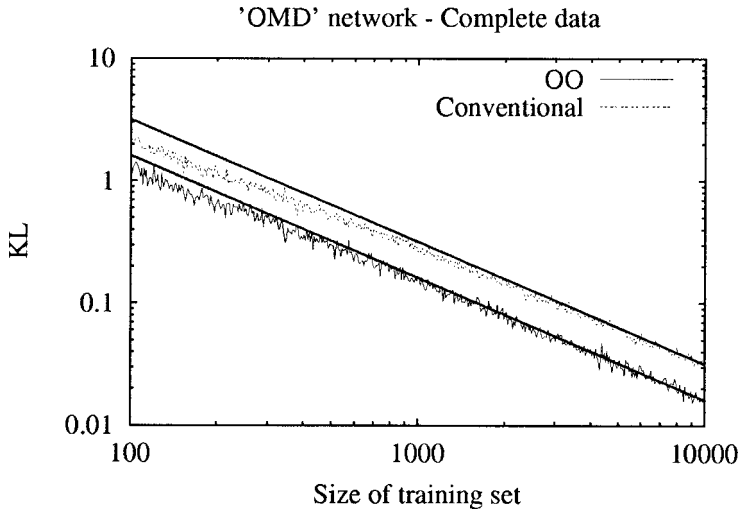


Figure 6. KL divergence between learned networks and the "true" distribution as a function of the size of the training set for the OMD network in figure 3 using complete data. The results from the OO learning are drawn with solid line, whereas the conventional learning results are dotted. The large-sample approximations from equation (4) are drawn with thick lines.

2.2. Missing data

When learning with missing data, the relation in equation (3) no longer holds. Assume that the data is missing completely at random, and let q denote the probability that a given variable X_i in a given data vector is missing. If q is "small" and the network is sparsely connected, then it is argued in [24] that for conventional learning we have

$$\lim_{N\to\infty} 2N(1-q)\cdot \mathbb{E}\big[D\big(\widehat{f}_N \| f\big)\big] \approx |\boldsymbol{\theta}|.$$

Hence, the expected value of $D(\widehat{f}_N \| f)$ is still approximately proportional to the number of parameters asymptotically. This does not, however, guarantee that object oriented learning is faster than conventional learning when some of the data is missing. To see the problem, consider the simple example domain in figure 7. The underlying BN of the OOBN is shown, and two instantiations of a class are framed. We follow, e.g., [36] and include the unknown probability parameters $\boldsymbol{\theta}$ in the model. The probability parameters are drawn as filled circles, the empty circles are domain variables. Assume that for a given data record from the domain in figure 7 we have observed $I_1.X_2 = x_2$ and $X_4 = x_4$. X_4 is the common child of $I_1.X_2$ and $I_2.X_2$. However, $I_2.X_2$ is missing from the data sample. In this case we get into trouble when we want to learn the probability $P(X_2 = x_2 | X_1 = x_1)$, as the two pieces of information used in learning this probability parameter are correlated (the observed value of $I_1.X_2$ influences both of them). Hence, the parameter estimates become *dependent* and thus the additivity of information in equation (5) is no longer valid. However, since the information matrix $\boldsymbol{I}$ is positive semi-definite [27, corollary 7.5.1], it follows that the information gain is always positive. Hence,

$$\boldsymbol{I}_{\{\boldsymbol{Y},\boldsymbol{Z}\}} \geqslant \boldsymbol{I}_{\boldsymbol{Y}}, \qquad \boldsymbol{I}_{\{\boldsymbol{Y},\boldsymbol{Z}\}} \geqslant \boldsymbol{I}_{\boldsymbol{Z}}.$$

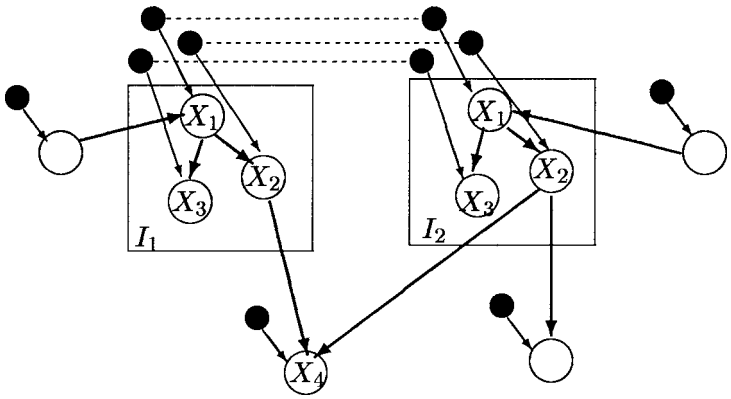


Figure 7. A simple example with two instantiations I_1 and I_2 of a class C. When doing object oriented learning some of the parameters are constrained to be equal. This is indicated by dotted lines.

Using the fact that the maximum likelihood estimators are asymptotically efficient, we have for large N

$$\mathrm{Var}_{\mathrm{OO}}(\widehat{\theta}_i) \leqslant \mathrm{Var}_{\mathrm{Conv}}(\widehat{\theta}_i)$$

for any parameter estimate $\widehat{\theta}_i$ where $\mathrm{Var}_{\mathrm{OO}}(\cdot)$ denotes the parameter variance obtained by object-oriented learning and $\mathrm{Var}_{\mathrm{Conv}}(\cdot)$ denotes the variance of the conventional learning estimates. The object-oriented learning will therefore not be *worse* than conventional learning in expectation as measured by the empirical KL divergence. However, as q grows large, the object-oriented learning may not be any *better* than the conventional one either.

To test the object-oriented learning with missing data, we assume that OMD does not have the time to measure all the available information every day. Therefore, at the beginning of the day he independently chooses to measure each variable with a probability $1-q$, or skip it that day (with probability q). This dataset is *missing completely at random*. The KL divergences that OMD achieves when learning both object oriented as well as conventionally are depicted in figure 8 for different values of q. Object-oriented learning is at least as good as the conventional one for all degrees of missing data, and for all sample sizes. The results for $q = 0.5$ and $q = 0.75$ were obtained by random

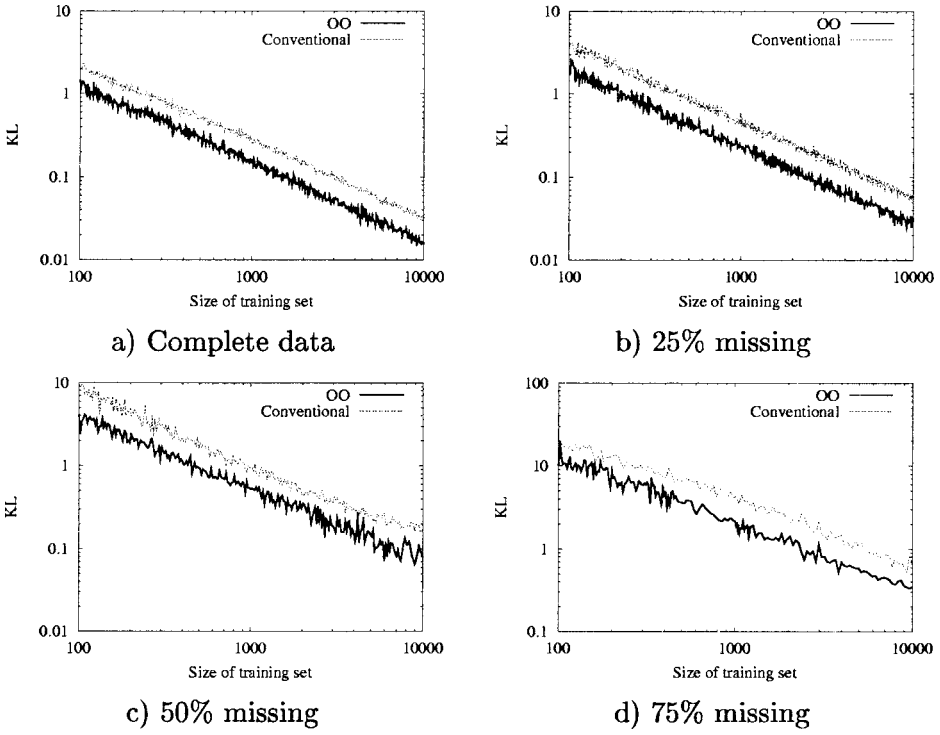


Figure 8. KL divergence between learned networks and the true distribution as a function of the size of the training set. Object-oriented learning offers a KL divergence that in expectation is at least as small as the one from conventional learning for all data sizes and all degrees of missing data.

restart of the EM algorithm up to 10 times, whereas the two other graphs were obtained by only one run of the EM algorithm.

When some of the data is missing we can not guarantee the increased learning speed that was obtained in the case of complete data. The method is, however, intuitively more appealing, and one will not loose information by using the object-oriented approach. The empirical results illustrated in figure 8 indicate that the object-oriented learning is strictly better even with large amount of missing data.

3. Violating the OO assumption

The results in figures 6 and 8 show that the OOBN approach indeed works better than the conventional approach on our example network. This is hardly a surprise, since we *know* that all instantiations are identical, and object-oriented learning simply takes this into account as part of its learning bias. More interesting is what happens if the instantiations of a class are slightly different[4] to each other. It may be reasonable to assume that the structure of all instantiations are identical, but that the parameters may be somewhat different. In papers on parameter learning the authors typically state that:

> "This [learning probability parameters in a BN with known structure and hidden variables] is an important problem, because structure is much easier to elicit from experts than numbers." [6, abstract]

A similar line of argument can be employed here: It is easy for an expert to say that the instantiations have identical structure. However, although the CPTs are about equal, there may be differences so small or subtle (e.g., due to variables not in the model that differ between the individual instantiations) that they are difficult to quantify. In OMD's case, for instance, no two cows are exactly alike, due to e.g., genetic differences.

We, therefore, propose a "relaxed OO" parameter learning, where differences between instantiations of the same class are penalized, but not totally rejected. Note that when applying "relaxed OO" learning the resulting network will not be object oriented any more. In this case the object orientation was merely a help during the network design, and not necessarily an anticipated property of the network during routine use.

The framework we propose to use for this calculation is Bayesian Model Averaging (BMA), see, e.g., [20]. In BMA one has a set of competing statistical models $\{M_1, M_2, \ldots, M_K\}$. To each model M_k a prior degree of belief, $P(M_k)$, is attached. The posterior degree of belief (given the database D) can be calculated in the standard Bayesian way,

$$P(M_k|D) = \frac{P(D|M_k) \cdot P(M_k)}{\sum_{\ell=1}^{K} P(D|M_\ell) \cdot P(M_\ell)}, \tag{6}$$

[4] If the instantiations are very different, a domain *expert* will not make the OO assumption. Proper modeling would instead imply the use of subclasses to fulfill the OO assumption. We therefore expect this situation to occur when the domain is "almost" object oriented, but the theory outlined will also work when the instantiations are very different, see the discussion leading to figure 10.

where

$$P(D|M_k) = \int_{\boldsymbol{\Theta}_k} P(D|\boldsymbol{\theta}_k, M_k) P(\boldsymbol{\theta}_k|M_k)\, \mathrm{d}\boldsymbol{\theta}_k. \tag{7}$$

$\boldsymbol{\theta}_k$ is the model parameters given model M_k, and the integration is performed over the whole parameter space $\boldsymbol{\Theta}_k$ of $\boldsymbol{\theta}_k$. If Δ is the property of interest, the posterior distribution of Δ according to BMA is

$$P(\Delta|D) = \sum_{k=1}^{K} P(\Delta|M_k, D) P(M_k|D). \tag{8}$$

In our application Δ will be the event that some variable takes on a particular value given the configuration of its parents, e.g., $\{X_i = k \mid pa(X_i) = j\}$. We use $\widehat{\theta}^O_{ijk}$ to denote the parameter estimate of $\theta_{ijk} = P(X_i = k|pa(X_i) = j)$ in the object oriented learning, and $\widehat{\theta}^C_{ijk}$ in the case of conventional learning. The BMA estimate $\widehat{\theta}^B_{ijk}$ will be given by

$$\widehat{\theta}^B_{ijk} = \widehat{\theta}^O_{ijk} P(M_O|D) + \widehat{\theta}^C_{ijk} P(M_C|D). \tag{9}$$

Here $P(M_O|D)$ and $P(M_C|D)$ are the posterior belief in the object oriented and conventional model, respectively. In [30] it is shown that when using a logarithmic scoring rule, averaging over all models provide better average predictive ability than using any single model M_j, conditioned on the set of models being considered.

The typical problem when implementing BMA is the computational complexity. First of all, the set of models can grow very large. Fortunately, this is not problematic in our case, as we limit the set of models to "Object oriented" and "*Not* object oriented". Secondly, the integration in equation (7) may be difficult to perform. This is cumbersome also in this work. As a first approximation one may crudely approximate the likelihood by using a distribution for $\boldsymbol{\theta}_k$ that is degenerated at the maximum likelihood estimate. Using $\boldsymbol{\Theta}_k = \{\widehat{\boldsymbol{\theta}}_k\}$ in equation (6), our posterior belief would be approximated by

$$P(M_k|D) \approx P\big(M_k|D,\ \boldsymbol{\theta}_k = \widehat{\boldsymbol{\theta}}_k\big) = \frac{P(D|M_k, \widehat{\boldsymbol{\theta}}_k) \cdot P(M_k)}{\sum_{\ell=1}^{K} P(D|M_\ell, \widehat{\boldsymbol{\theta}}_\ell) \cdot P(M_\ell)}. \tag{10}$$

Note that equation (10) will over-estimate the likelihood of the data, especially for larger models. Since the conventional model contains more parameters than the object-oriented one, we know that the likelihood of that model will be at least as large as the likelihood of the object-oriented model. This tendency for choosing the more complex model leads to the well-known problem of *over-fitting,* and is due to the higher flexibility of the more complex model. In our work we use an approximation to the log likelihood where a model is penalized for its size. The approximation is known as the Bayesian Information Criteria (BIC):

$$\log\big(P(D|M_k, \boldsymbol{\theta}_k)\big) \approx \log\big(P\big(D|M_k, \widehat{\boldsymbol{\theta}}_k\big)\big) - \frac{|\widehat{\boldsymbol{\theta}}_k|}{2} \log(N), \tag{11}$$

where $|\widehat{\boldsymbol{\theta}}_k|$ is the number of free parameters for model M_k, and N is the size of the data set. It is shown in [35] that the asymptotic size of the error in this approximation does not increase with N. The BIC has earlier been applied for learning in Bayesian networks, see e.g., [17,18]. We now use equation (11) to modify the likelihood calculations in equation (10), and get

$$P(M_k|D) \approx \frac{P(D|M_k, \widehat{\boldsymbol{\theta}}_k) \cdot N^{-|\widehat{\boldsymbol{\theta}}_k|/2} \cdot P(M_k)}{\sum_{\ell=1}^{K} P(D|M_\ell, \widehat{\boldsymbol{\theta}}_\ell) \cdot N^{-|\widehat{\boldsymbol{\theta}}_\ell|/2} \cdot P(M_\ell)} \tag{12}$$

as our posterior belief in model M_k.

The last problem of BMA is that of defining model priors. There is quite a lot of work available on generating model priors in the framework of Bayesian networks, both through knowledge elicitation [29,30] and non-informed methods as in, e.g., [18]. In our experience the domain experts find it difficult to assess priors for the two competing models at hand. Since the model he initially developed is object oriented he would like to believe that the OO assumption is justified, and therefore tends to hold a large belief in the object-oriented model. On the other hand, at a sufficiently detailed level the truly object-oriented real-world domains are very rare, and confronted by this fact the domain expert tends to be in trouble when the belief is to be quantified. In the end, the domain experts typically claim to be ignorant and give uniform priors, which is "... a *reasonable* '*neutral*' *choice* [*when there is little prior information about the relative plausibility of the models considered*]." [20, p. 390].

In the following we employ the BMA framework to a version of OMD's domain that is not object oriented: Without OMD's knowledge, two of his cows have been given hormones to produce more meat. Out of the two hormone-treated cows there is one **Meat cow** and one **Milk cow**. The effect of the hormone treatment (in our model, where food quality is not an issue) is that the treated cows produce significantly more meat. Hence, the true probability distributions over the *Meat* node has been changed for both cows. The rest of the domain is unchanged. The two **Milk cows** are thus not identical anymore, as their probability tables match for all but the *Meat* node; the same goes for the **Meat cows**. Since OMD does not know of this treatment, he models his stock in an object-oriented way, and wants to learn the probability tables in the domain from his data. He feels that his OO assumption is justified, and holds a prior belief of 75% for the object-oriented model. The results are shown in figure 9.

As the domain is not entirely object oriented, but still has some similarity to an object-oriented domain, the learning task of this example is a difficult one. The number of parameters in the conventional BN learning is almost twice that of the object-oriented model. By equation (12) this will give OMD a high posterior belief in the object-oriented model even when the observed data is carrying strong evidence against the OO assumption (i.e., the node *Meat* differs in the different instantiations). OMD could have used a larger model space describing the intermediate cases more specifically, e.g., by considering all models of the type "*Nodes* $\{X_k, \ldots, X_\ell\}$ *are different between instantiations, but otherwise the domain is object oriented*". In this case the learning method would have discovered the violation of the OO assumption faster. The correct model

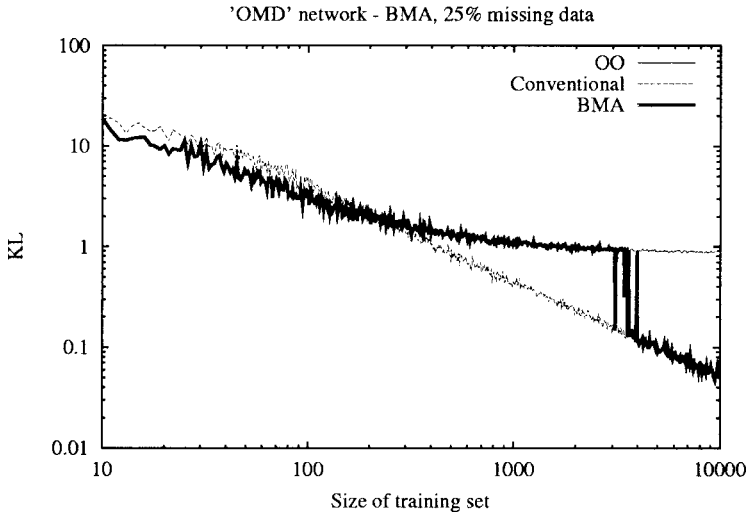


Figure 9. The empirical KL divergence versus size of the database is displayed for conventional learning, object-oriented learning and Bayesian model averaging. The object-oriented learning is better for smaller data sizes, but as the data size gets larger, the conventional learner is better (since the OO assumption is violated). The BMA follows the object-oriented model for small data sizes, but as the evidence against the OO assumption gets very outspoken, the conventional model is selected with weight 1.

would not have had any redundant parameters, and it would therefore not be so strongly penalized for its complexity. We have however not employed this enlarged model space in our calculations, as in most real-world situations the objects are very large, and fitting parameters to all models in a full enumeration of this extended model space is computationally prohibitive.

We could also have used a frequentistic hypothesis test to check whether the data indicate an object-oriented model or not. A test like Pearson's asymptotic χ^2-test [27, p. 325] can be employed. However, problems regarding the setting of the significance level and the interpretation of "large but not significantly large" test statistics made us choose the BMA setup.

To examine the effect of the BMA setup more closely, we performed a simple example with a class containing only one binary variable X. The class has two instantiations, with $P(X = 1) = (1 + \varepsilon)/2$ in the first instantiation, and $P(X = 1) = (1 - \varepsilon)/2$ in the other; $\varepsilon \in [0, 1]$ defines the difference between the two instantiations. Note that the OO assumption is violated as long as $\varepsilon \neq 0$. We calculated the degree of belief in the model to be object oriented by using equation (12). The results are shown for different data sizes in figure 10. The calculation scheme is able to detect that the OO assumption is violated as ε grows. For smaller values of ε, equation (12) is willing to assume that the domain is object oriented for small data sizes; the preference for the object oriented model vanishes as N grows larger. The effect of the BMA framework is thus that the estimators for one instantiation "borrows strength" from the other instantiations (by not rejecting that the domain is object oriented), so that the overall estimates become more

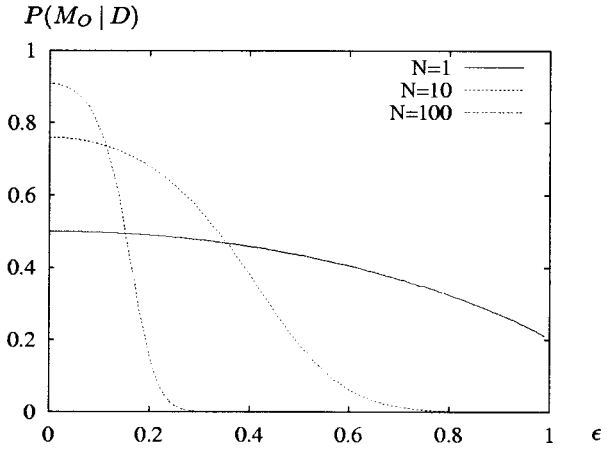


Figure 10. Posterior belief in the preposition that the domain is object oriented calculated by equation (12) for different values of ε and different data sizes N.

robust. When more data is present, or when the observed data clearly indicate that the OO assumption is violated, this "borrowing" does not take place to the same extent.

The same kind of result can be obtained by building a hierarchical Bayesian model. In this setting, we model θ_{ijk} in the different instantiations as random variables determined by an underlying distribution Γ_{ijk}. The posterior variance of Γ_{ijk} determines how equal the instantiations of the classes are, see, e.g., [5] for a case-study.

4. Type uncertainty

So far we have assumed that the domain expert is able to unambiguously classify each instantiation of the domain to a specific class. 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* in [33]: The expert is uncertain about the type (or class in our terminology) of an instantiation. As an example, assume OMD is unable to determine whether COWl is a **Milk cow** or a **Meat cow**. Even though he is not able to determine the class of COWl, he would like to learn from the available data. This section is devoted to showing how we treat type uncertainty within our framework.

Let the candidate classes of an instantiation I in an OOBN be given by the set $\mathcal{S}_{\mathrm{I}}$. The expert encodes his prior beliefs about the class of the instantiation I as a distribution over $\mathcal{S}_{\mathrm{I}}$. We assume that the probability distributions for the different instantiations are independent a priori. Recall that we use the notation I.X to denote the variable X in the instantiation I. $\mathcal{Z}_{\mathrm{I}}$ is the set of nodes that are defined inside the instantiation I (that is, not including those input nodes of the instantiation that reference nodes outside I). Let $\mathcal{I}$ denote the set of all instantiations in the OOBN. We use $\mathbf{T}(\mathrm{I})$ to denote the class of an instantiation I, and $\mathbf{T}(\mathcal{I})$ to denote a classification of all the instantiations in the domain. If we have a classification $\mathcal{C} = \mathbf{T}(\mathcal{I})$, then $\mathcal{C}^{\downarrow \mathrm{I}}$ is the induced classification of a given instantiation $\mathrm{I} \in \mathcal{I}$. We use $\alpha_{\mathrm{I},\mathbf{C}_\ell}$ for $P(\mathbf{T}(\mathrm{I}) = \mathbf{C}_\ell)$. Furthermore, $pa(\mathrm{I}.X|\mathbf{T}(\mathrm{I}))$

is used to denote the set of parents of I.X given the class of I. If $X_i \in \mathcal{Z}_{\mathrm{I}}$, we use $\theta_{\ell,ijk}$ for the probability $P(\mathrm{I}.X_i = k|\mathbf{T}(\mathrm{I}) = \mathbf{C}_\ell,\ pa(\mathrm{I}.X_i|\mathbf{C}_\ell) = j)$. To avoid problems with overfitting, we will assume that we have instantiations that are allocated to all classes in the OOBN model. If this is not the case, penalization of model complexity as in equation (11) should be introduced.

Let $\boldsymbol{X}$ denote the variables contained in the underlying BN. By means of the fundamental factorization of a probability distribution encoded by a BN, and hence by an OOBN, we get:

$$\begin{aligned} P\big(\boldsymbol{X}, \mathbf{T}(\mathcal{I})\big) &= P\big(\mathbf{T}(\mathcal{I})\big) \cdot P\big(\boldsymbol{X}|\mathbf{T}(\mathcal{I})\big) \\ &= \prod_{\mathrm{I}\in\mathcal{I}} P\big(\mathbf{T}(\mathrm{I})\big) \cdot \Bigg[\prod_{X_\ell\in\mathcal{Z}_{\mathrm{I}}} P\big(\mathrm{I}.X_\ell|pa\big(\mathrm{I}.X_\ell|\mathbf{T}(\mathrm{I})\big),\ \mathbf{T}(\mathrm{I})\big)\Bigg]. \end{aligned} \tag{13}$$

Note that for each choice of the classifications $\mathbf{T}(\mathcal{I})$ we have a different OOBN. The possible OOBNs are structurally identical everywhere except for the local models of the instantiations where the expert is uncertain. The correct OOBN is unknown, but we hold a prior distribution over the possible candidates. A priori the different OOBN models are conditionally independent given the classification. The overall model can therefore be modeled as an object oriented version of a Bayesian multinet; Bayesian multinets were introduced in [15].

Our goal is to employ a learning algorithm that learns the parameters of a domain, without specifying the class of I more precisely than by a prior distribution over $\mathcal{S}_{\mathrm{I}}$. This can be done by standard use of the EM algorithm.[5] In the following, we let $\widehat{\alpha}^{(t)}_{\mathrm{I},\mathbf{C}_\ell}$ denote the estimate of $P(\mathbf{T}(\mathrm{I}) = \mathbf{C}_\ell)$ after the tth iteration of the EM algorithm, and use $\widehat{\boldsymbol{\alpha}}^{(t)}$ to denote the collection of these estimates at that time. Furthermore, $\widehat{\boldsymbol{\Theta}}^{(t)} = \{\widehat{\theta}^{(t)}_{\ell,ijk}\}$ is the collection of probability parameter estimates in the classes after the tth iteration. The algorithm now proceeds by iterating over the following two update equations. First, we generate new estimates for $\alpha_{\mathrm{I},\mathbf{C}_\ell}$

$$\widehat{\alpha}^{(t)}_{\mathrm{I},\mathbf{C}_\ell} \leftarrow \frac{\sum_{\mathcal{C}:\,\mathcal{C}^{\downarrow\mathrm{I}}=\mathbf{C}_\ell} P(D|\mathbf{T}(\mathcal{I}) = \mathcal{C}, \widehat{\boldsymbol{\Theta}}^{(t-1)}) \cdot P(\mathbf{T}(\mathcal{I}) = \mathcal{C}|\boldsymbol{\alpha} = \widehat{\boldsymbol{\alpha}}^{(t-1)})}{\sum_{\mathcal{C}} P(D|\mathbf{T}(\mathcal{I}) = \mathcal{C}, \widehat{\boldsymbol{\Theta}}^{(t-1)}) \cdot P(\mathbf{T}(\mathcal{I}) = \mathcal{C}|\boldsymbol{\alpha} = \widehat{\boldsymbol{\alpha}}^{(t-1)})}. \tag{14}$$

The sum in the denominator is taken over all possible classifications $\mathbf{T}(\mathcal{I})$, whereas the sum in the numerator is restricted to classifications where I is classified to class $\mathbf{C}_\ell$. Note that $P(\mathbf{T}(\mathcal{I}) = \mathcal{C}|\boldsymbol{\alpha} = \widehat{\boldsymbol{\alpha}}^{(t-1)})$ is easy to calculate, since this probability is just the product of a subset of the elements in $\widehat{\boldsymbol{\alpha}}^{(t-1)}$.

Next, we update the estimates $\widehat{\boldsymbol{\Theta}}^{(t-1)}$. Let I be the instantiation containing X_i, i.e., $X_i \in \mathcal{Z}_{\mathrm{I}}$. Then $n^{(\mathrm{I},\mathbf{C}_\ell)}_{ijk}$ is the expected counts of the event $\{X_i = k,\ pa(X_i|\mathbf{C}_\ell) = j\}$ given that $\mathbf{T}(\mathrm{I}) = \mathbf{C}_\ell$. The distribution over the possible classification of the other

[5] To fit type uncertainty calculations into our OOBN framework, we will assume that for all $\mathbf{C}_\ell \in \mathcal{S}_{\mathrm{I}}$ we have that all nodes observed for I will be defined in $\mathcal{Z}_{\mathrm{I}}$ whenever $\mathbf{T}(\mathrm{I}) = \mathbf{C}_\ell$. Technically this is not necessary, but the implementation is simplified. Classes that do not meet this requirement cannot be candidate classes, and should therefore be removed.


Annals of Mathematics and Artificial Intelligence **32:** 221–243, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

Parameter learning in object-oriented Bayesian networks

Helge Langseth [a,b] and Olav Bangsø [b]

[a] *Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway*
E-mail: helgel@math.ntnu.no
[b] *Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Øst, Denmark*
E-mail: {hl, bangsy}@cs.auc.dk

This paper describes a method for parameter learning in Object-Oriented Bayesian Networks (OOBNs). We propose a methodology for learning parameters in OOBNs, and prove that maintaining the object orientation imposed by the prior model will increase the learning speed in object-oriented domains. We also propose a method to efficiently estimate the probability parameters in domains that are *not* strictly object oriented. Finally, we attack type uncertainty, a special case of model uncertainty typical to object-oriented domains.

Keywords: Bayesian networks, object orientation, learning

AMS subject classification: 68T05

1. Introduction

Bayesian Networks (BNs) [21,32] have established themselves as a powerful tool in many areas of artificial intelligence, including planning, vision, decision support systems and robotics. However, one of the main obstacles is to create and maintain very large domain models. To remedy this problem, object-oriented versions of the BN framework have been proposed in the literature [4,22]. Object-Oriented BNs (OOBNs) as defined in these papers offer an easy way of creating BNs, but the problem of assessing and maintaining the probability estimates still remain; conventional learning algorithms like [6] do not exploit that the domain is object oriented while learning.

In this paper we propose a learning method that is applied directly to the OOBN specification. It is proven that this learning method is superior to conventional learning methods in object oriented domains, and a method to efficiently estimate the probability parameters in domains that are *not* strictly object oriented is also proposed.

This paper is organized as follows: The rest of this section will create a starting point for our analysis by introducing OOBNs and the required notation and assumptions. In section 2 we outline the proposed learning method, and in section 3 we propose a framework for learning in domains that are only approximately object oriented. A special case of model uncertainty, typical to object-oriented domains, is handled in section 4, and we conclude in section 5.

instantiations, as well as conditional distributions over missing values, are replaced by expected values in the *E-step* of the EM algorithm. Similarly, $n_{ij}^{(\mathrm{I},\mathbf{C}_\ell)} = \sum_k n_{ijk}^{(\mathrm{I},\mathbf{C}_\ell)}$ is the expected counts of the event $\{pa(X_i|\mathbf{C}_\ell) = j\}$ under the assumption that $\mathbf{T}(\mathrm{I}) = \mathbf{C}_\ell$. The estimates for $\theta_{\ell,ijk}^{(t)}$ in class $\mathbf{C}_\ell$ are updated by

$$\widehat{\theta}_{\ell,ijk}^{(t)} \leftarrow \frac{\sum_{\mathrm{I}\in\mathcal{I}:\ X_i\in\mathcal{Z}_\mathrm{I}} \widehat{\alpha}_{\mathrm{I},\mathbf{C}_\ell}^{(t-1)} \cdot n_{ijk}^{(\mathrm{I},\mathbf{C}_\ell)}}{\sum_{\mathrm{I}\in\mathcal{I}:\ X_i\in\mathcal{Z}_\mathrm{I}} \widehat{\alpha}_{\mathrm{I},\mathbf{C}_\ell}^{(t-1)} \cdot n_{ij}^{(\mathrm{I},\mathbf{C}_\ell)}}. \tag{15}$$

Equation (15) is the natural extension of the update equation when the classification of all instantiations are known. In that case, all values of $\boldsymbol{\alpha}$ are fixed at either 0 or 1; the update rules are otherwise identical.

Iterating over the equations above will lead to a local maximum of the likelihood of the observed data. As a spin-off from the presented algorithm, equation (14) generates the posterior distribution over the possible classes of an instantiation. This task, which is known as *classification,* has a rich body of literature also within the BN community, see, e.g., [7,13]. The complexity of performing the parameter update steps is exponential in the number of instantiations the expert cannot classify with certainty. If the number of these unclassified instantiations is "large", it will be more efficient to implement a *Generalized EM algorithm*, in which the likelihood of the data is strictly increased in each iteration (but not necessarily maximized).

When we are only interested in classification (i.e., when the parameters are *known*), the type uncertainty task can be particularly easy computationally. First of all, we need

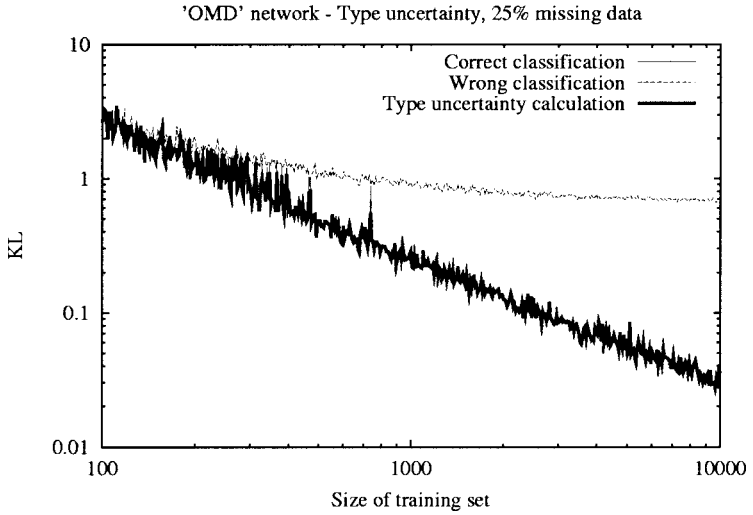


Figure 11. The empirical KL divergence versus the size of the database is displayed for object oriented learning with correct classification of COW1 (**Meat cow**), wrong classification of COW1 (**Milk cow**), and the results of the outlined method. The classification is fairly random for smaller data sizes, but as the data size gets larger the correct class is given a probability converging towards 1. The results of the correct classifier (thin line) are hidden underneath the results of the type uncertainty (thick line).

not perform the calculations in equation (15) since the parameters are known. Secondly, if the input and output sets of the classes in $\mathcal{S}_I$ do not contain missing values, the required likelihoods to classify I can be calculated locally (in the classes), and the larger model in which the instantiation is embedded will be of no interest for the type uncertainty calculations.

As an example, consider again OMD's stock. Assume he is uncertain about the class of COWl, whereas he is able to correctly classify the other three cows. His prior distribution for the class of COWl is that both classes are equally likely, and his data is reported with 25% missing values. In figure 11 the result of applying the proposed learning algorithms (equations (14) and (15)) are displayed, together with the results of a consistently *wrong* classifier (COWl assumed to be a **Milk cow**), an the consistently *correct* classifier (COWl assumed to be a **Meat cow**). The proposed method is capable of detecting the correct class after approximately 700 cases, and for larger data sizes the results of the proposed method are just as good as the consistently correct classifier.

5. Conclusions

In this paper we have proposed a learning method to learn parameters in OOBNs. It has been proven that this learning method is superior to conventional learning in object-oriented domains if the database is complete, and it is shown that as long as the OO assumption holds, the proposed learning algorithm will never be inferior to conventional learning. We have proposed to use Bayesian model averaging to estimate the probability parameters in domains that are *not* strictly object oriented, and showed by example that this methodology offers reasonable results. A method that enables us to handle situations where the object oriented model is not completely specified has also been described.

Acknowledgements

We would like to thank our colleagues in the Decision Support Systems group at Aalborg University for interesting discussions. In particular, Thomas D. Nielsen has provided constructive comments to an earlier version of this paper.

References

[1] N. Abe, M.K. Warmuth and J. Takeuchi, Polynomial learnability of probabilistic concepts with respect to the Kullback–Leibler divergence, in: *Proceedings of the 4th Annual Workshop on Computational Learning Theory (COLT 1991)* (Morgan Kaufmann, San Mateo, CA, 1991) pp. 277–289.

[2] O. Bangsø, H. Langseth and T.D. Nielsen, Structural learning in object oriented domains, in: *Proceedings of the 14th International Florida Artificial Intelligence Research Society Conference (FLAIRS-2001)* (AAAI Press, 2001) pp. 340–344.

[3] O. Bangsø and P.-H. 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 (2000).

[4] O. Bangsø and P.-H. Wuillemin, Top-down construction and repetitive structures representation in Bayesian networks, in: *Proceedings of the 13th International Florida Artificial Intelligence Research Society Conference*, eds. J. Etheredge and B. Manaris (AAAI Press, 2000) pp. 282–286.
[5] R. Bellazzi and A. Riva, Learning conditional probabilities with longitudinal data, in: *Working Notes of the IJCAI Workshop Building Probabilistic Networks: Where Do the Numbers Come from?* (AAAI Press, Montreal, 1995) pp. 7–15.
[6] J. Binder, D. Koller, S. Russell and K. Kanazawa, Adaptive probabilistic networks with hidden variables, Machine Learning 29 (1997) 213–244.
[7] J. Cheng and R. Greiner, Comparing Bayesian network classifiers, in: *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI'99*, eds. K.B. Laskey and H. Prade (Morgan Kaufmann, Stocholm, 1999) pp. 101–108.
[8] T.M. Cover and J.A. Thomas, *Elements of Information Theory* (Wiley, New York, 1991).
[9] R.G. Cowell, A.P. Dawid, S.L. Lauritzen and D.J. Spiegelhalter, *Probabilistic Networks and Expert Systems*, Statistics for Engineering and Information Sciences (Springer, New York, 1999).
[10] H. Cramér, *Mathematical Methods of Statistics* (Princeton University Press, Princeton, NJ, 1946).
[11] S. Dasgupta, The sample complexity of learning fixed-structure Bayesian networks, Machine Learning 29(2–3) (1997) 165–180.
[12] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B 39 (1977) 1–38.
[13] N. Friedman, D. Geiger and M. Goldszmidt, Bayesian network classifiers, Machine Learning 29 (1997) 131–163.
[14] N. Friedman and Z. Yakhini, On the sample complexity of learning Bayesian networks, in: *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)* (Morgan Kaufmann, San Francisco, CA, 1996) pp. 274–282.
[15] D. Geiger and D. Heckerman, Knowledge representation and inference in similarity networks and Bayesian multinets, Artificial Intelligence 82 (1996) 45–74.
[16] P.J. Green, On use of the EM algorithm for penalized likelihood estimation, Journal of the Royal Statistical Society 52(3) (1990) 443–452.
[17] D. Heckerman, A tutorial on learning with Bayesian networks, in: *Learning in Graphical Models*, ed. M.I. Jordan (MIT Press, Cambridge, MA, 1999).
[18] D. Heckerman, D. Geiger and D.M. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data, Machine Learning 20 (1995) 197–243. Also available as Microsoft Research Technical Report MSR-TR-94-09.
[19] D.F. Heitjan and S. Basu, Distinguishing "Missing At Random" and "Missing Completely At Random", The American Statistician 50(3) (1996) 207–213.
[20] J. Hoeting, D. Madigan, A. Raftery and C.T. Volinsky, Bayesian model averaging: A tutorial (with discussion), Statistical Science 14(4) (1999) 382–417. Corrected version at http://www.stat.washington.edu/www/research/online/hoetingl999.pdf.
[21] F.V. Jensen, *An Introduction to Bayesian Networks* (Taylor and Francis, London, UK, 1996).
[22] D. Koller and A. Pfeffer, Object-oriented Bayesian networks, in: *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, eds. D. Geiger and P.P. Shenoy (Morgan Kaufmann, San Francisco, 1997) pp. 302–313.
[23] W. Lam and F. Bacchus, Learning Bayesian belief networks: An approach based on the MDL principle, Computational Intelligence 10(4) (1994) 269–293.
[24] H. Langseth, Efficient parameter learning: Empiric comparison of large sample behaviour, Department of Computer Science, Aalborg University (2000). Available at http://www.cs.auc.dk/research/DSS/publications.
[25] K.B. Laskey and S.M. Mahoney, Network fragments: Representing knowledge for constructing probabilistic models, in: *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, eds. D. Geiger and P. Shenoy, (Morgan Kaufmann Publishers, San Francisco, CA, 1997) pp. 334–341.

[26] S.L. Lauritzen, The EM-algorithm for graphical association models with missing data, Computational Statistics and Data Analysis 19 (1995) 191–201.
[27] E.L. Lehmann, *Elements of Large-Sample Theory*, Springer Texts in Statistics (Springer, New York, 1999).
[28] R.J.A. Little and D.B. Rubin, *Statistical Analysis with Missing Data* (Wiley, New York, 1987).
[29] D. Madigan, J. Gavrin and A. Raftery, Eliciting prior information to enhance the predictive performance of Bayesian graphical models, Communication in Statistics – Theory and Methods 24 (1995) 2271–2292.
[30] D. Madigan and A. Raftery, Model selection and accounting for model uncertainty in grahical models using Occam's window, Journal of American Statistical Association 89 (1994) 1535–1546.
[31] L. Ortiz and L. Kaelbling, Accelerating EM: An empirical study, in: *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)* (Morgan Kaufmann, San Francisco, CA, 1999) pp. 512–521.
[32] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).
[33] A.J. Pfeffer, Probabilistic reasoning for complex systems, Ph.D. thesis, Stanford University (2000).
[34] M. Pradhan, G. Provan, B. Middleton and M. Henrion, Knowledge engineering for large belief networks, in: *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence* (Morgan Kaufmann, San Francisco, CA, 1994) pp. 484–490.
[35] G. Schwarz, Estimating the dimension of a model, Annals of Statistics 6 (1978) 461–464.
[36] D.J. Spiegelhalter and S.L. Lauritzen, Sequential updating of conditional probabilities on directed graphical structures, Networks 20 (1990) 579–605.
[37] S. Srinivas, A probabilistic approach to hierarchical model-based diagnosis, in: *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence* (Morgan Kaufmann, San Francisco, CA, 1994) pp. 538–545.
[38] B. Thiesson, Accelerating quantification of Bayesian networks with incomplete data, in: *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining* (AAAI Press, Menlo Park, CA, 1995) pp. 306–311.
[39] R.A. van Engelen, Approximating Bayesian belief networks by arc removal, IEEE Transactions on Pattern Analysis and Machine Intelligence 19(8) (1997) 916–920.
[40] J. Whittaker, *Graphical Models in Applied Multivariate Statistics* (Wiley, Chichester, 1990).
[41] Y. Xiang and F.V. Jensen, Inference in multiply sectioned Bayesian networks with extended Shafer-Shenoy and lazy propagation, in: *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI'99*, eds. K.B. Laskey and H. Prade (Morgan Kaufmann, Stocholm, 1999) pp. 680–687.
[42] Y. Xiang, D. Poole and M.P. Beddoes, Multiply sectioned Bayesian networks and junction forests for large knowledge-based systems, Computational Intelligence 9(2) (1993) 171–220.

V

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

To appear in Journal of Machine Learning Research

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 e.g. (Buntine, 1996, 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 e.g. 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.

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

©2002 Helge Langseth and Thomas D. Nielsen.

However, 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 e.g. (Mahoney and Laskey, 1996, Laskey and Mahoney, 1997, Koller and Pfeffer, 1997, Bangsø and Wuillemin, 2000b). Although these object oriented frameworks relieve some of the problems when modeling 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 modeling object oriented domains have been proposed in (Mahoney and Laskey, 1996, Laskey and Mahoney, 1997, Koller and Pfeffer, 1997, 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 (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 modeling 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), i.e., 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] (e.g. 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 (i.e., Instantiation-name.*Node-name*).

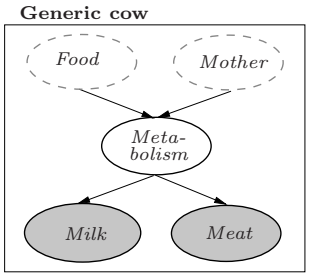
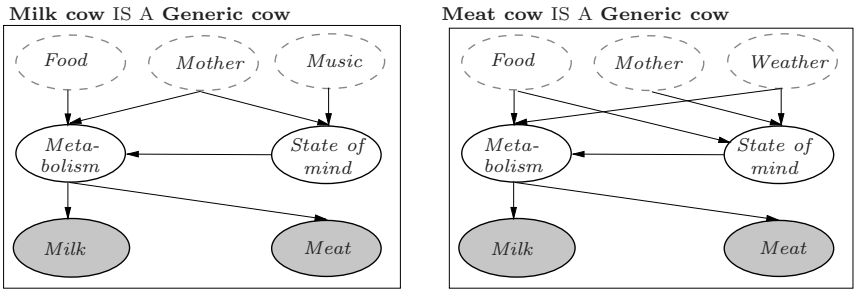


Figure 1: General properties common to all cows are described using the class **Generic cow**. The arrows are links as in normal BNs. The dashed ellipses are input nodes, and the shaded ellipses are output nodes.

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, e.g. 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 indicates that the leaf of

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



a) The **Milk cow** specification b) The **Meat cow** specification

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

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 (e.g., during inference).

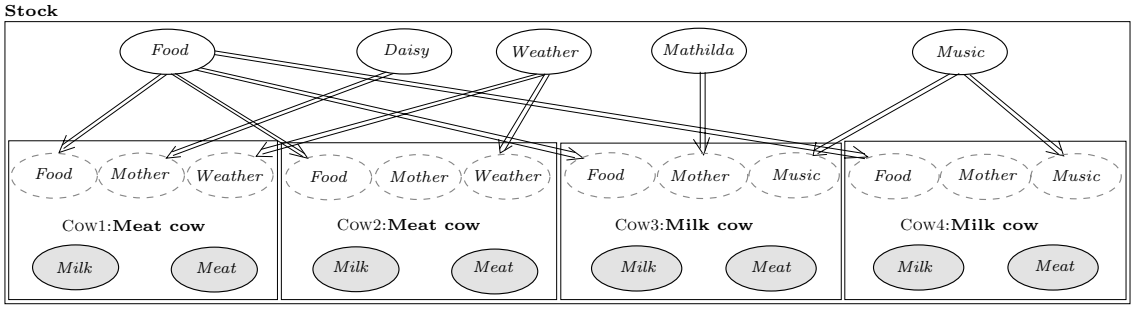


Figure 3: The **Stock** class with two instantiations of the **Milk cow** class and two instantiations of the **Meat cow** class. Note that some input nodes are not referencing any nodes.

If there is more than one instantiation of a class (e.g. Cow1 and Cow2), 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, which 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).

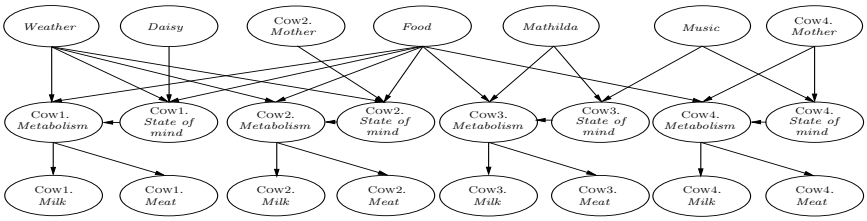


Figure 4: The underlying BN for the OOBN depicted in Figure 3.

Note that the 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 emphazising that an OOBN is just a compact representation of a (unique) BN which 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).

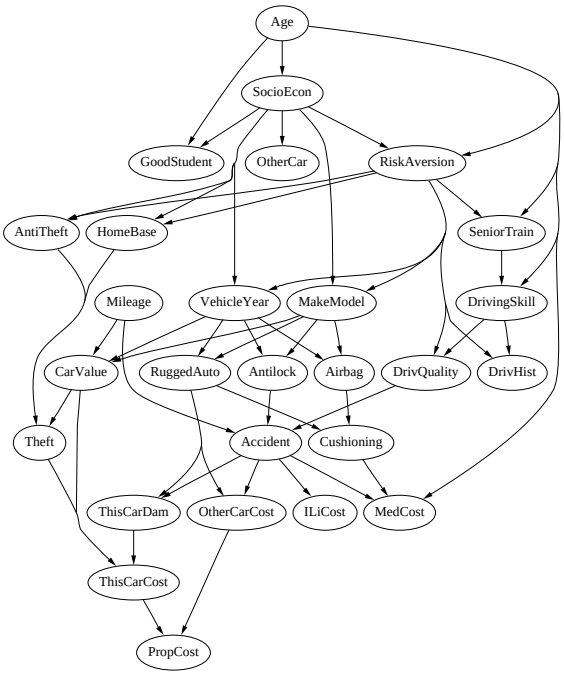


Figure 5: The insurance network, used for classifying car insurance applications.

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 s.t. they adhere to the design principle of high internal coupling and low external coupling, see e.g. (Mahoney and Laskey, 1996, 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 encap-

sulated 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 (i.e., the **Insurance** class). Note that only the interfaces of the encapsulated instantiations are shown.

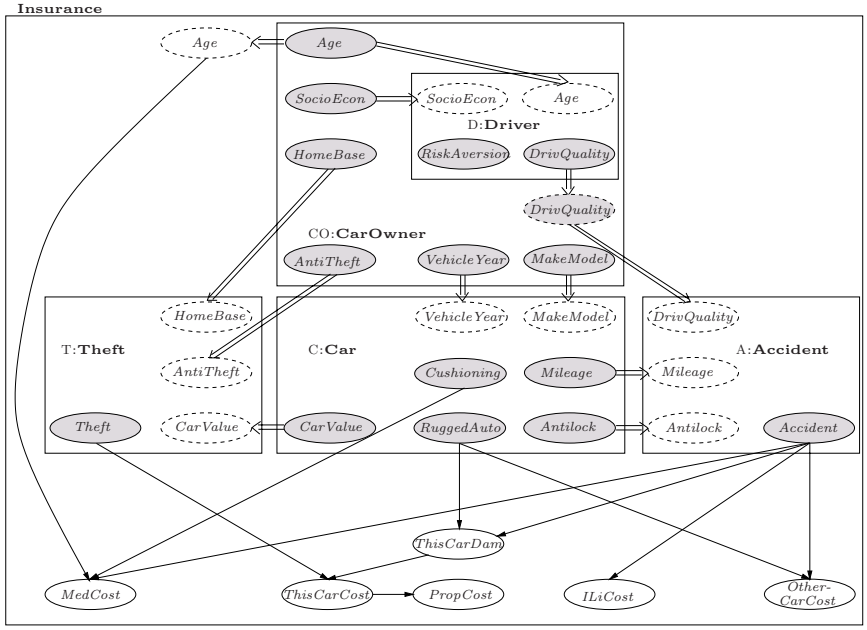


Figure 6: An OOBN representation of the insurance network. Notice that only the interfaces of the encapsulated instantiations are shown. Note also that we use a slightly non-standard graphical presentation for visualization purposes.

The **Insurance**-class is constructed s.t. 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 e.g. (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 modeling 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 modeling 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).

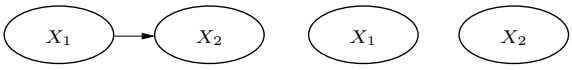
Consider a Bayesian network $BN = (B_S, \Theta_{B_S})$ over a set of discrete variables $\{X_1, X_2, \ldots, X_n\}$, where B_S is the graphical structure and Θ_{B_S} is the quantitative information. To describe B_S, the qualitative aspects of BN, we will use the following notation: r_i is the number of states for variable X_i, q_i is the number of configurations over the *parents* for X_i in B_S (denoted by Π_i), i.e., $q_i = \prod_{X_l \in \Pi_i} r_l$, 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} which 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 s.t. $(\theta_1, \theta_{21}, \theta_{22}) \in \Omega'$. For the empty graph in Figure 7b, the parameter space $\Omega'' \subset \Omega'$ corresponds to the parameter space Ω' where $\theta_{21} = \theta_{22}$, i.e., 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; some aspects of an OOBN (i.e., the OO-assumption) are not reflected in the underlying graphical structure, and 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}, i.e., 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, i.e., 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, i.e., $\Theta_{ij} \perp\!\!\!\perp \Theta_{ij'}$ for $j \neq j'$, and 5) the prior distribution of the parameters in every Bayesian network B_S has a Dirichlet distribution[5], i.e., there exist numbers (virtual counts) $N'_{ijk} > 0$ s.t.:

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

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, i.e., $N_{ijk} = \sum_{l=1}^{N} \gamma(X_i = k, \Pi_i = j : \boldsymbol{D}_l)$, 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 $\boldsymbol{D}_l$, and 0

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

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:

$$P(\mathcal{D}|B_S, \xi) = \int_{\Theta_{B_S}} \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} P(\Theta_{B_S}|B_S, \xi) d\Theta_{B_S}, \tag{4}$$

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:

$$P(\mathcal{D}, B_S|\xi) = P(B_S|\xi) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})}, \tag{5}$$

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) which 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 5) by having the user specify a *prior Bayesian network* B_p for $\boldsymbol{X}$ and an *equivalent sample size* N':

$$N'_{ijk} = P(X_i = k, \Pi_i = j|B_p, \xi) \cdot N'. \tag{6}$$

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 5 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 5 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

$$\text{score}(X_i, \Pi_i, \mathcal{D}) = \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})}. \tag{7}$$

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, i.e., 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 e.g. (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 (using e.g. the expectation-maximization (EM) algorithm (Dempster et al., 1977, Green, 1990)) before we can calculate the score of a model. I.e., 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:*

$$\begin{aligned} Q(B_S : B_S^n) &= \mathbb{E}_{\boldsymbol{h}}[\log P(\boldsymbol{h}, \boldsymbol{o}, B_S)|B_S^n, \boldsymbol{o}] \\ &= \sum_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{o}, B_S^n) \log P(\boldsymbol{h}, \boldsymbol{o}, B_S) \end{aligned}$$

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, i.e., 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):

$$\mathbb{E}_{\boldsymbol{h}}[\log P(\boldsymbol{h}, \boldsymbol{o}, B_S)|B_S^n, \boldsymbol{o}] = \sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{h}}[\log F_i(\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o}), B_S)|B_S^n, \boldsymbol{o}], \tag{8}$$

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, i.e., normalization of the prior distribution is not required. Friedman (1998) also examines an approximation for $\mathbb{E}_{\boldsymbol{h}}[\log F_i(\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o}), B_S)|B_S^n, \boldsymbol{o}]$:

$$\mathbb{E}_{\boldsymbol{h}}[\log F_i(\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o}), B_S)|B_S^n, \boldsymbol{o}] \approx \log F_i(\mathbb{E}_{\boldsymbol{h}}[\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h}, \boldsymbol{o})|B_S^n, \boldsymbol{o}], B_S). \tag{9}$$

7. An active research area within the learning community is the discovery of *hidden variables*, i.e. variables that 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 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 $\mathbb{E}_{\boldsymbol{h}}[\mathbf{N}_{i\cdot\cdot}(\boldsymbol{h},\boldsymbol{o})|B_S^n,\boldsymbol{o}]$ can be computed as:

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

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 in (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*, i.e., $P(\boldsymbol{X}[t+1]|\boldsymbol{X}[0],\ldots,\boldsymbol{X}[t]) = P(\boldsymbol{X}[t+1]|\boldsymbol{X}[t])$. 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 two 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 6.

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

$$\rho(X_i, \Pi_i) = \left(\begin{array}{c} n-1 \\ |\,\Pi_i\,| \end{array} \right)^{-1}$$

used in e.g. (Friedman and Koller, 2000). 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

$$\delta_i = \sum_{j \neq i} (\omega_{ij}^{+}\, \delta_{ij}^{+} + \omega_{ij}^{-}\, \delta_{ij}^{-}) \ , \tag{10}$$

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 $\omega_{ij}^{(\cdot)}$, 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, and we show how the underlying object oriented modeling 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 $X_i[t_\ell]$ $(t_\ell > 0)$, i.e., 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 (as done by e.g. 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 modeled 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 modeling 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 (i.e. a prior model). The a priori specification of an OOBN contains a list of class specifications and a grouping of the nodes into instantiations which 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*?"*

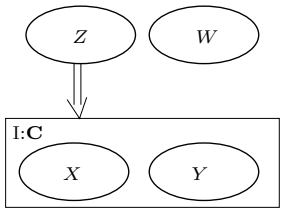


Figure 8: The figure depicts a possible way to describe knowledge about the structure of the domain; it shows an instantiation I and some of its encapsulating context (note that this is not strictly speaking an OOBN).

The use of such prior models is also supported by Equation 10, 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 which 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 10. 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, i.e., to work 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 utilizes 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, i.e., the task of *density estimation* (Silverman, 1986). Note that if focus had been on e.g. causal discovery (Heckerman, 1995a), classification (Friedman et al., 1997a), or generating a model that was able to predict well according to a predefined

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

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:

$$N_{ijk}^{\mathbf{C}_\ell} = \sum_{\mathrm{I}:\mathbf{T}(\mathrm{I})=\mathbf{C}_\ell} \sum_{t=1}^{N} \gamma(\mathrm{I}.X_i = k, \mathrm{I}.\Pi_i = j : \boldsymbol{D}_t). \tag{11}$$

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:

$$\text{O-score}(X_i, \Pi_i, \boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\mathcal{D}), \mathbf{C}_\ell) = \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij}^{\mathbf{C}_\ell} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk}^{\mathbf{C}_\ell} + N'_{ijk})}{\Gamma(N'_{ijk})}, \tag{12}$$

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:

$$P(\mathcal{D}, B_S \,|\, \xi) \propto \prod_{\mathbf{C}_\ell \in \mathcal{C}} \prod_{i:X_i \in \mathbf{C}_\ell} \rho(X_i, \Pi_i, \mathbf{C}_\ell) \cdot \text{O-score}(X_i, \Pi_i, \boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\mathcal{D}), \mathbf{C}_\ell) \ , \tag{13}$$

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

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

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 8 and Equation 13 we have (again we assume that the prior distribution is normalized):

$$\mathbb{E}_{\boldsymbol{h}}[\log P(\boldsymbol{o},\boldsymbol{h},B_S)\,|\,B_S^n,\boldsymbol{o}] = \sum_{\mathbf{C}_\ell\in\mathcal{C}}\sum_{i:X_i\in\mathbf{C}_\ell}\mathbb{E}_{\boldsymbol{h}}[\log F_{i,\mathbf{C}_\ell}(\boldsymbol{N}_{i\cdot\cdot}^{\mathbf{C}_\ell}(\boldsymbol{h},\boldsymbol{o}),B_S)|B_S^n,\boldsymbol{o}] \tag{14}$$

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 9 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 $P(\mathcal{D},B_S\,|\,\xi)$ factorizes 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 12) 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, e.g., 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* I*; each edge involved must have a node in both* I *and in the encapsulating context of* I *(directed into* 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 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 14).*

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.

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 11. 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.

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, 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, e.g. simulated annealing, for investigation 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, i.e., 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.

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* in (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 e.g. 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* $\mathbf{C}$ *of* I *s.t.* $P(\mathbf{T}(\mathrm{I}) = \mathbf{C}) > 0$*:*

A) Classify I *as an instantiation of class* $\mathbf{C}$*:* $\widehat{\mathcal{T}}'_{\mathrm{I}} \leftarrow \mathbf{C}$.

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*

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

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, which 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 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

$$D\left(f \,||\, \hat{f}_N\right) = \sum_{\boldsymbol{x}} f(\boldsymbol{x}|\Theta) \log\left[\frac{f(\boldsymbol{x}|\Theta)}{\hat{f}_N(\boldsymbol{x}|\hat{\Phi}_N)}\right]. \tag{15}$$

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):

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

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

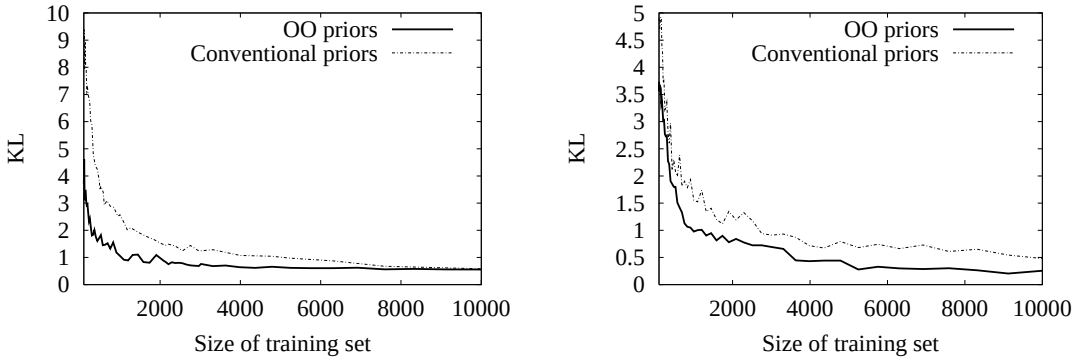
Observe that in the tests we do not consider the issue of running time. However, even though the proposed algorithms might seem more complex than the SEM algorithm (due to the nested iterations) the search space is in fact smaller and we therefore expect that the algorithm require fewer steps than the ordinary SEM algorithm, see also Section 5.

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, and 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,

14. The KL divergence is not a distance measure in the mathematical sense, as $D(f \,||\, g) = D(g \,||\, f)$ 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.

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" (i.e., 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. Note that learning of DBNs (see Section 3.4) is simply a special case of OOBN learning, since any DBN can be modeled by the usage of two OOBN classes (see Sections 2 and 4.2). Hence, the results in (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 in (Langseth and Bangsø, 2001) has been modified to make the differences between the classes sufficiently small for the problem to be challenging.[16]

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

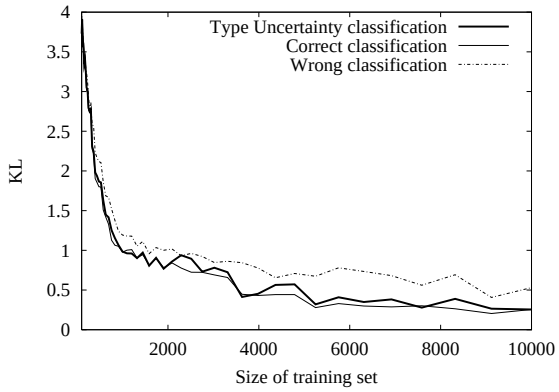


Figure 10: The KL divergence of the gold standard model vs. the generated models for the three cases "Type uncertainty classification" (Algorithm 4), the results of "Correct classification" and "Wrong classification".

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, i.e., the sum of the expected costs as indicated by the variables *ILiCist*, *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 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.[17] 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, which 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).

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.

17. 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.

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

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 Daphne Koller. Being Bayesian about network structure. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 201–210. Morgan Kaufmann Publishers, 2000. To appear in *Machine Learning*, 50(1–2), 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.

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 *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. 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.

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.

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, 1987. ISBN: 0-471-80254-9.

Suzanne M. Mahoney and Kathryn B. Laskey. Network engineering for complef 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*, 49(3):223–239 and 253–265, 1987.

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.