VTT INFORMATION

RESEARCH REPORT TTE1-2000-29

MODUS-Project
Case Study WasteWater

Method feasibility Study: Bayesian Networks

Version 1.1-1

16.10.2000

Mikko Hiirsalmi

Version history

Version	Date	Author(s)	Reviewer	Description
1.0-1	03.10.2000	Mikko Hiirsalmi		Delivered to steering group 5.10.2000
1.1-1	12.10.2000	Mikko Hiirsalmi	Ilkka Karanta, Esa Rinta-Runsala, Antoni Wolski	Corrected version. Still confidential.

Contact information

Mikko Hiirsalmi
VTT Information Technology
P.O. Box 1200, FIN-02044 VTT, Finland
Street Address: Tekniikantie 4 B, Espoo
Tel. +358 9 4561, fax +358 9 456 6027
Email: Mikko.Hiirsalmi@vtt.fi
Web: http://www.vtt.fi/tte

Last modified on 16 October, 2000
T:\Modus\Cases\MetsaSerla\Raportti\Prediction\rr2k29-c-ww-feasib2.doc

Abstract

Basic principles of Bayesian networks, inference with them and discovery of Bayesian network structures are briefly introduced. Then, the applicability of these methods to the analysis of process data is addressed. The case study problems involve mining of dependencies from training data and using the discovered dependency models for prediction of quality indicator values. Prediction results are presented as diagrams and commented. The predictions achieved are promising but it seems that with the current models the prediction accuracy is not good enough for the case problem.

With suitable training data, Bayesian dependency models may be discovered from the data and applied in many ways. The possibilities range from "What-If" -analysis of the effect of value changes to the probability distributions of the other variables to sequential decision making using influence diagrams. The generated models may be implemented as C programs similarly to the way tested in this case study.

Tiivistelmä

Aluksi esitetään lyhyt johdanto Bayes-verkkojen perusperiaatteisiin, niillä tapahtuvaan päättelyyn ja mallien automaattiseen oppimiseen esimerkkiaineistosta. Sitten tarkastellaan niiden soveltuvuutta prosessitiedon analysointiin. Kohdeongelmina tarkastellaan riippuvuuksien kaivuuta esimerkkiaineistosta ja muuttujien välisten riippuvuusmallien käyttöä laatuparametrien ennustamiseen. Ennustetulokset esitetään kaavioina. Tulokset ovat lupaavia, mutta ennustetarkkuus ei riitä asiakasongelmassa.

Bayes-verkkoja voidaan kaivaa opetusaineistosta, kunhan aineistossa esiintyy mielekkäitä riippuvuuksia. Opittuja malleja voidaan hyödyntää monin tavoin "Mitä Jos" –analyysistä, jossa tutkitaan muuttujien arvomuutoksien vaikutusta muiden muuttujien todennäköisyysjakaumiin, peräkkäisten päätösten tekemiseen vaikutuskaaviotekniikalla. Kehitetyt mallit voidaan viedä käytäntöön C-ohjelmina tässä testatulla tavalla.

Table of Contents

1 Introduction

Probabilistic representations, Bayesian networks in particular, may be used to represent and quantify dependencies between measured variables. Bayesian networks provide compact models for representing probabilistic dependencies between random variables. They allow one to represent the dependencies with a qualitative dependency diagram and the magnitude of the dependencies using conditional probabilities. Efficient algorithms exist for reasoning with Bayesian networks by propagating new evidence in the dependency network. Also methods for learning restricted types of Bayesian network have been created and are available as software tools.

We have applied Bayesian networks in an industrial case problem, called WasteWater. The case problem deals with data mining in ameasurement data base of an activated sludge waste water cleaning process. In the first phase, the task was to identify useful depedencies between the measurement entities. In the second phase, the aim was to identify predictive models from enriched training data. The accuracy of the generated Bayesian network models was then evaluated on validation data.

The results were found to be interesting while indicating that with the available data set size the predicted variables were probabilistically directly dependent on only a couple of variables. However, the produced prediction accuracy for a validation data set was not good enough for operational usage. However, these results agree with our experience with applying artificial neural network models for the same task [Hiir00]. Discretation of the variables into five states causes some of the available information to be lost. The Bayesian network techniques with discrete variables is also not most suitable for time series prediction tasks but is useful for better understood, statistically uncertain domains (e.g. fault diagnostics tasks).

In the report, the Bayesian network methodology is briefly introduced and then various methods for propagating evidence in these networks are reviewed. Then the process of learning the Bayesian networks is discussed. It consists of tasks of learning the dependency structure and the parameters (conditional probabilities) of the network from training data and prior information. Subsequently the applicability of Bayesian networks to our problem case is assessed and some experiences are described from applying Bayesian network learning to the task of mining a waste water measurement time series data. In the appendices, some existing and evaluated tools for learning Bayesian networks and for inferencing with Bayesian networks are outlined.

2 What are Bayesian networks?

Probability theory provides a model for representing and reasoning with uncertainty between dependent random variables. Uncertainty may be present as uncertain status of the variables or as nondeterministic dependence between the variables (same symptoms may indicate different faults). However, in systems based on pure probability theory, representation of the joint probability of the random variables is tedious as the number of parameters grows exponentially when new variables are introduced. Graphical models use

a combination of probability theory and graph theory to make reasoning with probabilities feasible. The idea is to indicate conditional independences between variables on a qualitative level using dependency graphs and by specifying the quantitative probabilistic model using local conditional or joint probabilities. Graphical models include Bayesian networks, Markov networks and chain graphs, see [Cast97] for further details. Bayesian networks are graphs with directed edges, Markov networks are graphs with undirected edges and chain graphs are either directed or nondirected. Their properties, representational power and solution methods vary. In this report we concentrate on the Bayesian networks.

A Bayesian network is a directed acyclic graph (DAG) with a probability table for each node. The nodes represent propositional variables that take values from given domains and the arcs connecting them represent the existence of direct causal influences between the linked variables. The strength of these influences is quantified with conditional probabilities. [Pear88]

A Bayesian network is a pair (G, P) where G is a DAG and P = { P_i } where P_i denotes probabilistic relationships (conditional dependency of x_i from its immediate parents) between x_i and its parents ({P (x_i | pa_i}. The Bayesian network represents a probability distribution over X having the product form:

$P(x_1,...,x_n) = ?_{i=1..n} P(x_i \mid pa_i)$.

In Figure 2-1, we can see an example of a simple Bayesian network which indicates that a random variable called Metastatic Cancer may affect variables called Increased Serum Calcium and BrainTumor. Variable Increased Serum Calcium affects the probability of Coma. Variable BrainTumor affects both Coma and Severe Headaches variables. The actual quantitative models for the dependencies are specified by the conditional probability tables.

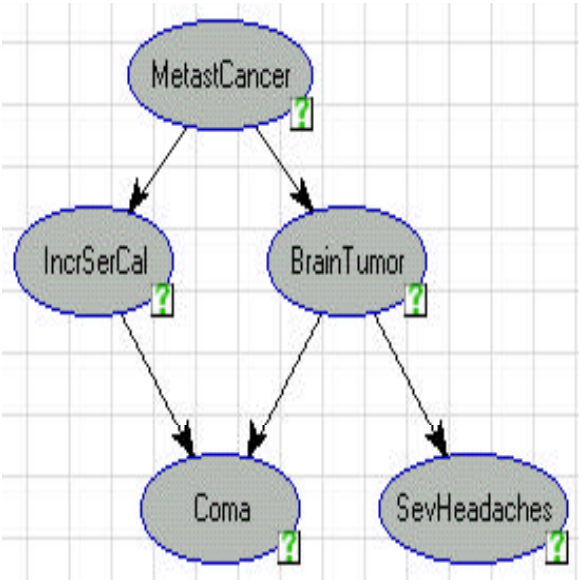

Figure 2-1 An example of a simple Bayesian network structure

The state domains of the variables are typically all discrete or all continuous. Some models also have combinations of discrete and continuous variables with some modelling restrictions, e.g. that a discrete variable may not be a child of a continuous one. For example, in the example above, variable Increased Serum Calcium could have two states (yes and no) indicating the existence of that clinical status.

Continuous models may be defined using parametric probability models like normal distribution, which is defined by giving the average and standard deviation parameters.

In this paper we shall concentrate on the discrete variable set as most of the available tools only support the discrete cases.

While modeling a problem with Bayesian networks, each variable needs to be represented with a node that has an exhaustive and mutually exclusive set of states. The state domains may be binary or multivalued. Then the uncertain dependencies between the variables need to be defined by drawing arcs between dependent variables. The non-existence of an arc between variables indicates conditional independence between the variables. Finally the prior and conditional probability dependencies need to be defined. The basic formalism just supports the definition of random variables as nodes and qualitative dependencies between them with arcs and quantitative dependencies as probability matrices. Other types of relationships, like temporal behaviour and taxonomic dependencies, need to be represented with other formalisms. Temporal behaviour may be represented by cloning the nodes for relevant time periods and specifying conditional dependencies between the successive time points. Dynamic Bayesian networks have been developed as a special formalism for representing temporal behaviour. They assume that during each time point same variables exist and have the same conditional dependencies. Dependencies between different time points are restricted to arcs between as few variables as possible and typically only dependencies between successive time points are modeled. This makes belief propagation more efficient.

Bayesian networks may be extended by including Decision and Value nodes into the models that are then called Influence Diagrams. Decision nodes represent decisions that the user may perform. Value nodes specify a utility function for the different states of the network. Influence diagrams are often used by operations researchers while modeling decision problems.

3 Reasoning with Bayesian networks

Once the Bayesian network has been specified one may draw conclusions from it by propagating evidence. Propagation algorithms have been developed to answer various types of queries. The following types of queries are possible over Bayesian networks:

1) Belief updating, which means computing the posterior probability of each proposition given a set of observations.

2) Maximising the expected utility of the problem (meu), which means finding a maximising assignment to a subset of decision variables when a utility function is available.

3) Finding the maximum posterior probability hypothesis (map), which means, given some evidence, finding an assignment to a subset of hypothesis variables maximising their probability.

4) Finding the most probable explanation (mpe), which means, given some evidence, finding a maximum probability assignment to the rest of the variables.

Exact, approximate and variational methods have been developed to solve these problems.

For general networks exact solution of the given problems is known to be NP-hard. However, polynomial propagation algorithms are available for singly-connected[1] networks. For relatively sparse networks, graph transformations may be used to produce singly-connected networks. Typical methods include conditioning and clustering. In conditioning, loop-cutset[2] nodes are clamped (instantiated, fixed) to different values, each solution is calculated and the solutions are averaged by weighting the different alternatives. In clustering, the graph is first converted to an undirected Markov network which is then triangulated to produce a tree of clusters which can be solved efficiently. Exact methods may be increasingly inefficient for certain types of network structures (tightly connected or ones with large loop-cutsets). Consult [Neap90] for details.

Monte Carlo methods [Jord99] provide a general approach to the design of approximate algorithms based on stochastic sampling. Rejection sampling, importance sampling and Markov chain sampling are examples of such algorithms. The basic idea is to generate a sample of size N from the joint probability distribution of the variables and then to use the generated sample to compute approximate values for the probabilities of certain events given the evidence. These approximate probabilities are the ratios of frequency of events in the sample to the sample size.

Variational methods [Jord99] are deterministic algorithms that provide upper and lower bounds to probabilities of interest. They have been used in a wide variety of settings covering statistics, statistical mechanics, quantum mechanics and finite element analysis. A complex problem is converted into a simpler problem. The general intuition is that a complex graph may be probabilistically simple due to various averaging phenomena. The traditional exact methods only consider the dependency structure of the graph and not the numerical strengths of the links. Variational methods have been successfully applied to various network architectures (huge two layer diagnostic networks QMR-DT, MLP neural networks, factorial hidden Markov networks, Bolzmann machines etc.) It is, however, not clear how to automatically define an appropriate variational transformation for a general architecture; some art is still involved. Potentially the best solutions could be achieved by

[1] A directed graph is singly-connected if for each pair of nodes in the graph there is at most one chain (a set of undirected arcs) connecting them.

[2] Loop-cutset nodes consist of a set of nodes that, when instantiated, break the communication pathways along the loops of the graph [Cast97].

combining the different approaches for each case so that variational methods could be used to simplify the graph in order to make exact solution methods feasible for the task.

In appendices 3, 4 and 5, we review the properties of three popular Bayesian belief network inference environments, Genie, NETICA and HUGIN. They support belief updating using exact propagation with join-tree clustering.

4 Learning Bayesian networks from data

Traditionally, Bayesian network models have been constructed manually by editing a network model. However, as training data is often available, automated learning methods have been developed for learning Bayesian network models from data. One may separate two main tasks: induction of the best matching qualitative dependency model from data and prior knowledge; and estimation of the local probabilities, both prior and conditional ones, from the data. Benefits of learning Bayesian network models include that integration of prior domain knowledge may be supported in various ways, the models are explainable on the qualitative level and may even be interpreted in a causal way. Consult any of [Cast97], [Frie98], [Heck96], [Bunt96] for further details.

Missing data (both missing values and nonobservable hidden or latent variables) complicates the learning process.

There are two general approaches to graphical probabilistic model learning from data: search & scoring methods and dependency analysis methods. In search & scoring methods, the problem is viewed as a search for a structure that fits the data best. One starts with a graph that has no edges and uses a search method to add edges to the graph one by one. After each addition a scoring method is used to evaluate the new structure in order to determine if it is better than the old one. If it is better, the addition is kept, otherwise it is removed. This process is continued until no new structure is better than the previous one. Different scoring criteria have been developed including Bayesian scoring method, entropy based method, minimum description length method and minimum message length method. Since learning Bayesian networks using search & scoring methods is NP-hard, many of the methods use heuristic search. Also, in order to reduce the search space, many of the algorithms require ancestral node ordering[3] as an input.

Chow-Liu Tree Construction algorithm (1968) serves as the basis of many of the search & scoring methods. It is an efficient algorithm for learning tree structured probabilistic networks from data. It has been extended to polytrees[4] [Pear88] but no polynomial algorithm exists for multiply connected networks. K2 algorithm (1992) is an algorithm capable of learning general Bayesian networks but it requires complete node ordering. [Chen98] describes many other approaches applying different scoring methods or removing the assumption of complete node ordering.

---

[3] In ancestral node ordering each of the nodes in the ordered list is only dependent on its predecessors.

[4] Polytrees are directed forests where there may be multiple root nodes but no loops exist.

Since a structure encodes many dependencies of the underlying model, dependency analysis methods try to discover the dependencies from the data and then use them to infer the structure. The dependency relationships are measured by using some kind of conditional independence (CI) tests.

[Chen98] describes many methods based on dependency analysis . The most efficient of them seems to be the PC algorithm by Spirtes and Glymour, 1991. It does not require node ordering and is able to automatically orient edges of the learned structure using CI tests.

Both of the general approaches have their pros and cons. In general, the dependency analysis approach is more efficient than the search & scoring approach for sparse networks. However, many of these algorithms require an exponential number of CI tests with many high order CI tests (CI tests with large condition sets), which may be unreliable unless the volume of data is enormous. The search & scoring based methods work with a wider range of probabilistic models although they may not find the best structure due to their heuristic methods.

Model averaging techniques have also been developed as sometimes the underlying model of a data set contains some uncertainty. They return several networks and use the average of these networks to perform belief propagation. All the previous algorithms assume that all the variables in the underlying models appear in the data sets (causal sufficiency). Some algorithms have been developed for situations with hidden or latent variables. Also some algorithms are capable of dealing with data sets containing missing values.

Various approaches and tools have been developed for learning Bayesian networks. Many of the inference tools available are able to estimate probability parameters from training data. Some tools exist for learning the dependency structure from complete data especially when some restrictions are given (like node ordering). Few tools are able to learn with missing data. Autoclass is a family of tools capable of learning classification models where the class labels are represented as hidden parent nodes. Naive Bayes models deal with discrete networks where a common cause affects a set of feature nodes that are conditionally independent of each other. Tree Augmented Naive Bayes (TAN) models [Frie98] extend Naive Bayes by allowing tree-structured dependencies between the feature nodes.

5 Applicability of Bayesian networks to knowledge discovery and data mining

We have tested the applicability of Bayesian networks in an industrial case study in the area of monitoring biological processes. The tasks considered involve the discovery of dependencies between measured process entities and creation of methods for predicting the future behaviour of key quality indicators based on data measured online. Below we introduce the chemical process studied. Then the process of discovering Bayesian networks from training data is introduced, and then the use of Bayesian network for forecasting is reported.

5.1 Description of the active sludge cleaning process

Our test case concerns two biological subprocesses of an active sludge waste water cleaning system being used at the Kirkniemi factory of Metsä-Serla company. Here we shall briefly describe the relevant aspects of the process studied.

Active sludge waste water cleaning is a complex process consisting of mechanical, biological and chemical subprocesses. Waste water is continuously fed to the process and then it flows through the process for days. The considered process consists of two partly connected cleaning lines both containing an aeration basin and a sedimentation basin. Before entering these lines, the waste water has been mechanically cleaned and its acidity has been neutralized and additional nutrients have been added to the system. The cleaning is based on allowing the bacteria, protozoans and other micro-organisms to eat organic waste.

In the aeration basins additional oxygen is dissolved into the water so as to keep the process aerobic and so to allow and accelerate the process. The micro-organisms and the organic waste form a biosludge, which is extracted from the water in the sedimentation phase. A selected part of the sedimented sludge is circulated through the process back to the aeration basin to be reactivated in order to reach a predefined sludge age (meaning the average amount of time that a particle is kept in the circulation). To maintain stability of the process and good quality of the sludge, excess sludge is periodically removed from the process and dried. Removing the sludge too soon may produce too much waste to be dried and to be transported to a dumping place. Also cycling the waste water too long in the system consumes excess cleaning capacity.

The circulation process is slow: a typical delay between beginning and end of the circulation process is 2-3 days. Furthermore, the biological aspects of the process have longer delays, typically 10-15 days. An overview of the process is presented in Figure 5-1.

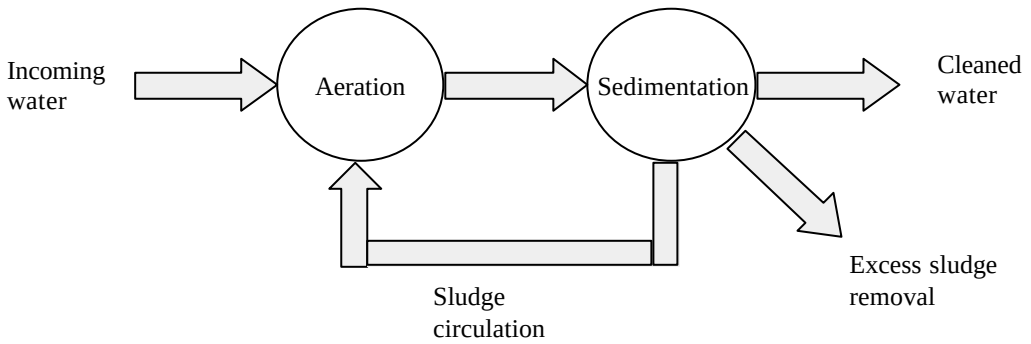

Figure 5-1 Biological waste water cleaning process.

The process operators control the process on the basis of following the process measurements, such as BOD[5] and COD[6] levels, and prior knowledge. In normal

[5] The biological demand for oxygen (BOD) is a measure of the amount of oxygen used by micro-organisms to decompose the organic matter in the wastewater.

situations, when quantity and quality of the incoming water is stable, control of the process is easy and well known. In abnormal situations, for example, when chemical concentrations of the input water change rapidly, the process can go out of balance. The biomass reacts slowly to changes in water quality and quantity, therefore out-of-balance situation in the waste water treatment plant occur several days after the actual cause has occurred. Furthermore, corrective actions are also slow and the results are usually observable only after quite a long delay period.

Process measurements are divided into two basic types: online measurements and laboratory analyses. Online measurements are obtained using sensors and the values from online measurements are available instantly, together with a timestamp. Sometimes online measurements may produce erroneous indications. This is due to fouling of the sensors and drifting of sensor calibration. Typically values from online measurements are available with a time resolution of several seconds, but in this case study averaged values of one hour are used.

Most process quality results are obtained using laboratory analyses. Analyses are performed by taking a sample which is then analyzed in laboratory by chemical experts. Results of the laboratory analyses are entered to the computer system manually with a timestamp characterizing the time when the sample was taken. Most of the laboratory analyses are performed in-house, but some analyses are performed in external neutral laboratories. Typically, results of in-house analyses are available within hours (by late afternoon) after the sample period has ended, but external laboratory results can have a delay of several weeks. The sample period may extend for several days.

5.2 On the process of using Bayesian networks for data analysis

The typical steps of the Bayesian networks based data mining process are represented in Figure 5.2.

In order to use Bayesian networks for data analysis the data has to be preprocessed (typically discretized and missing values replaced). In order to use time lagged variables in the tools used, one has to preprocess the data by adding the needed lagged values into each measurement row. (The tool could do this automatically, but typically does not.)

Once the data has been preprocessed, one starts the Bayesian network learning tool and starts the structure learning process. In BNPC2.0, it is possible to define thresholds on how strong the accepted dependencies must be. Also it is possible to integrate various forms of prior domain knowledge into the system by specifying a complete anchestral node ordering or partial orders for known indirect causes or temporal orders, causes and effects may be indicated and some dependencies may be forbidden. Also some nodes may be marked as root or leaf nodes.

---

[6] The chemical demand for oxygen (COD) is a measure of the amount of oxygen used to oxidize organic matter and to convert it to carbon dioxide and water.

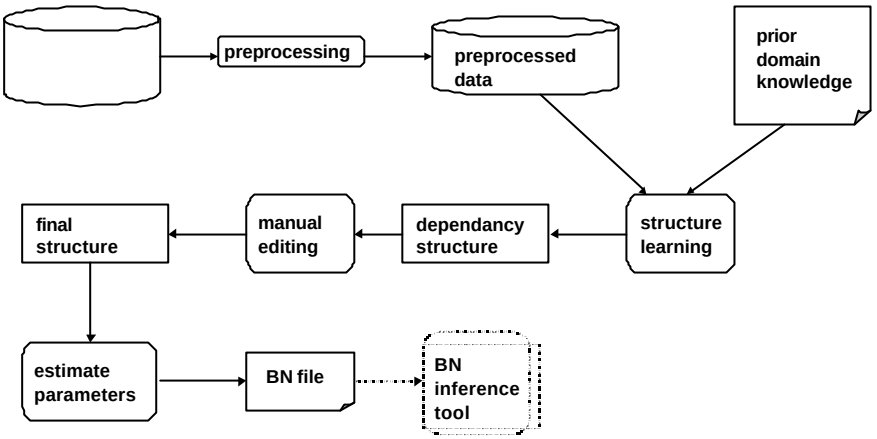

Figure 5-2 Typical steps of a data mining process based on Bayesian networks

The learning process produces a set of dependencies between the variables. Some of the dependencies may remain undirected as the appropriate direction cannot be determined from the given data. Such undirected dependencies must be manually directed using a graphical editor. It is also possible to edit the network by redirecting the links and by adding/removing links. Once all the links have been directed, one may save the network in the appropriate output file format for later manipulation by Bayesian network inference tools. During this process the probabilities of the links are estimated from the data.

One problem with BNPC2.0 is that the network is not automatically laid out in an aesthetic and user friendly manner. Therefore the user has to do some manual editing with a graphical editor in order to produce a clearer network when there are many links and nodes. It would also be useful if the lagged variables would be arranged in groups according to time slices or other criteria.

After learning, the network may be used with a Bayesian network inference tool (Hugin or Netica are supported). The structure and the probabilities can be studied by a graphical user interface. One may also perform what-if analysis using the tool by defining the likelihood of the states of a variable or by setting it to a certain value. The new probabilities are then calculated by propagating the evidence in the network.

The presentation of the qualitative dependencies between the variables as a network seems natural but with larger networks more structured presentations would be useful. While using the discrete variables, the conditional probability matrices tend to get very big, which makes their understanding less intuitive. For example, if a node has 5 parents with 2 states each, a conditional probability matrix with $2 * 2^5 = 64$ numbers is required. Therefore the current Bayesian networks are most applicable when the networks are not too dense.

5.3 Applicability of Bayesian networks to forecasting

Basic Bayesian networks do not support temporal dependencies between variables and their learning algorithms expect that the data items in each learning data record contain all the relevant data. Using preprocessing steps, training data records may be constructed so that older data items that are considered relevant for detecting dependencies are stored as new, "extra", artificial fields into the data record (like $O_2content_{[-2]}$ meaning the oxygen content two time steps before the current time). If no such preprocessing is done, dependencies may only be detected between variables measured at the same points of time.

Also temporally extended Bayesian networks have been developed such as Dynamic Bayesian networks (DBN) and Temporal Bayesian networks. DBNs are extensions of the successful Hidden Markov Models (HMM) that are often used in speech recognition and sequence data analysis tasks in biochemistry.

6 Analysis results produced by using Bayesian network learning tools

We tested Bayesian network tools in two phases on two separate data sets. In the first phase, we were examining a set of online measurements and laboratory analyses results organized as daily measurements covering data from the whole waste water cleaning plant. The original task was to identify dependencies between the sensor readings in order to detect useful information on the process dynamics. In the second phase, the aim was to build forecasting methods for predicting the quality indicator values in the future or alternatively to predict problematic conditions in time. We selected the aeration process as the most interesting subprocess of the plant and selected all available and relevant sensor readings and laboratory measurements that expected to have an effect on this task. A new set of data was formed.

In both cases, BNPC2.0 (see Appendix 1) was applied to learn Bayesian networks from the data. BNPC is a research prototype, which provides versatile file access formats (Excel, MS Access, etc.), is fast in detecting the network structure and provides output formats to three main input data formats of Bayesian network tools (Hugin, NETICA, the proposed standard for Bayesian network file interface). The tool also contains a built-in discretizer but it currently only works for data that is in MS Access format. Therefore, the data had to be first discretized, which was done with SPSS. Each of the variables was divided into 5 value domains so that each of them contained the same amount of

instances. Also BKD-tool (see Appendix 2) was considered but it was not tested with real data. It is a research prototype, which supports handling of missing values. The tool was not tested as it required a full ordering of the variables, which was not available.

The resulting networks were first investigated by using Hugin Bayesian network inference tool (their evaluation copy, see Appendix 3), which allows loading of networks with 200 states. Also Netica tool (see Appendix 4) was tried but its evaluation copy only allowed 15 node networks to be loaded. Therefore, it was not possible to test large models with this tool. Also the models could not be embedded in user programs as the interface libraries were not available in the evaluation versions.

In the second phase, a new Bayesian network reasoning tool called GENIE (see Appendix 5) was tested. The embedding of the generated models into user programs was also tested for forecasting future quality indicator values. Technically the embedding worked well.

Using Bayesian networks for prediction of future values requires discovery of a dependency model that relates together variables from successive time slices or in some other way embeds temporal features into the model. In Bayesian reasoning, the marginal probability distribution of any node may be updated upon acquiring evidence for other nodes. All the nodes are equal - there are no special input or output nodes. So the same model may be used for many purposes.

6.1 First phase: experiments with the original set of data records

In the diagrams of this phase, the prefixes of the variable names indicate the subprocess (PKAN3 is the waste water outlet of the newest factory, ESKI is a preliminary cleaning process, Ilmas3 is the newest aeration line, PALIE3 is the recycling of the waste water back to the ILMAS3 phase, YJLIE3 is the sludge that has been pumped from the system to be dried, KEMSY is the chemical cleaning phase of the process before releasing the cleaned water into the environment). The end of the names follows an internal database naming convention.

6.1.1 Dependencies between all measurement variables within the same time slice

At first, a static model was estimated for the whole data set in order to detect intervariable dependencies within one time slice. The following results were acquired for the qualitative dependencies between the variables while using the strictest pruning criteria. In Figure 6-1, one sees that some of the variables are termed unaffected by any others and on the other hand some nodes seem to have dependencies with many others (for example, ESKI_BOD is related to 10 nodes, it has 2 parents and 8 children). While domain experts were verifying functional dependencies identified from the same data set using association rules it was noticed that the dependencies detected by BNPC2.0 agreed very closely with the intuition of the experts. The number of irrelevant ones was much smaller than that suggested by functional dependencies. Some of the edge directions were noticed to be against temporal or causal order. However, if domain knowledge concerning the allowed partial orders between the nodes was introduced, the directions followed the intuitions much better.

The estimated probability matrices were not investigated.

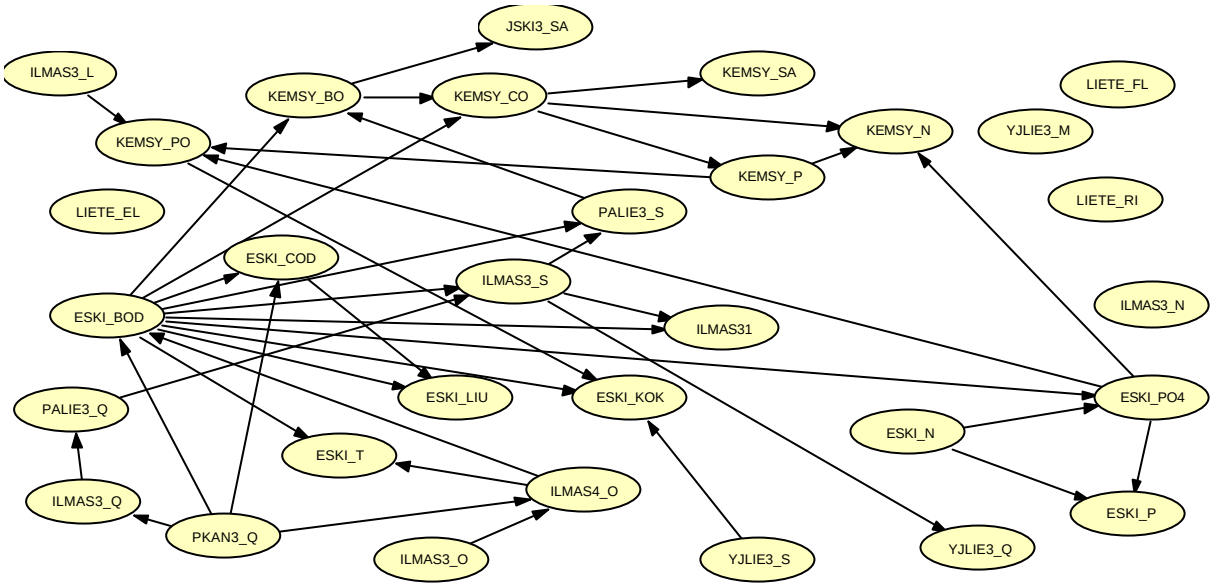

Figure 6-1 An example of a Bayesian network showing the detected dependencies between all variables at the same time slice.

6.1.2 Dependencies between selected variables within the same time slice

As viewing the whole dependency network at a time may be difficult to comprehend, a more compact static model (see Figure 6-2) was estimated for a limited set of variables. This was done in order to detect intervariable dependencies within one time slice and the following results were acquired for the qualitative dependencies between the variables. Note that some of the variables have been left alone without any dependencies because of a high threshold to accept dependencies. The network is easier to understand but seems to agree with the larger model as well.

Such networks could be used for forecasting when temporal aspects are added.

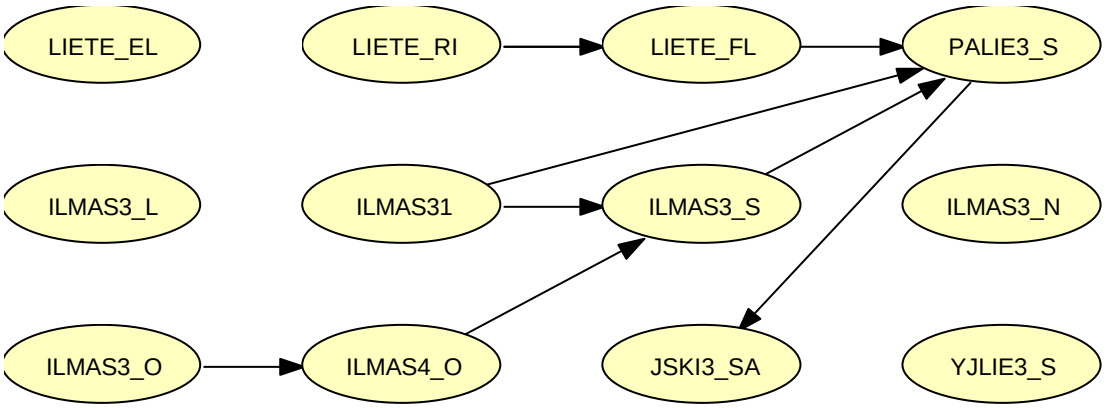

Figure 6-2 An example of a Bayesian network for the dependencies between a limited set of variables at the same time slice

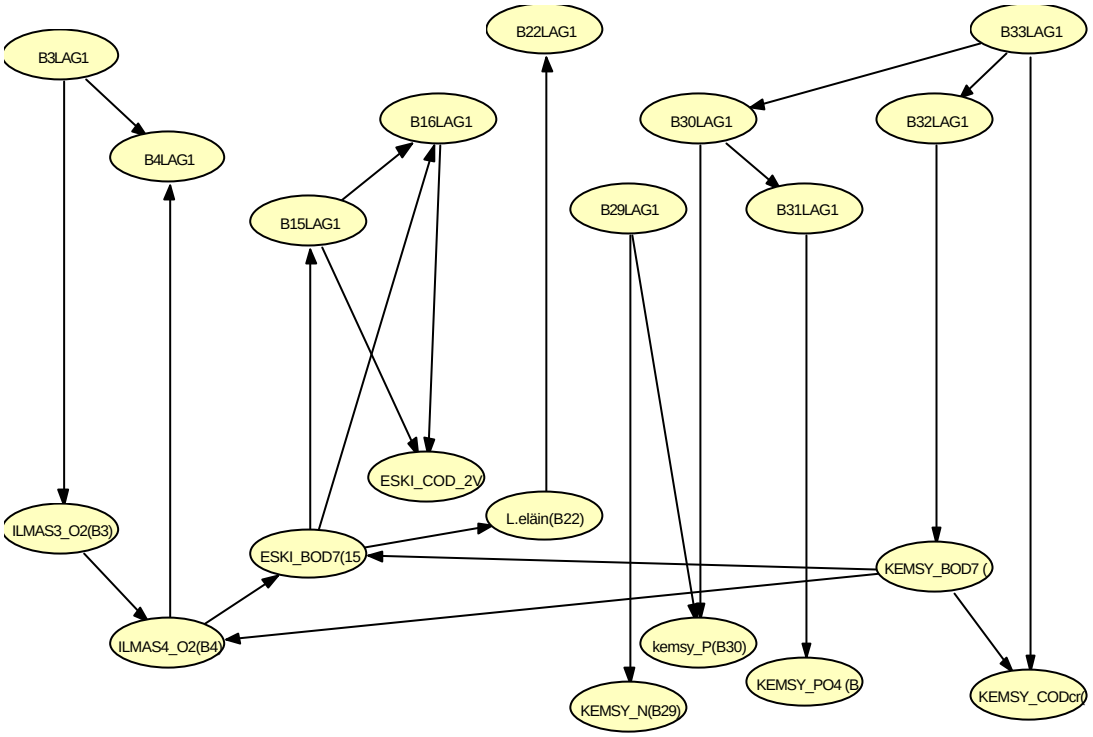

Figure 6-3 An example of a Bayesian network for the dependencies between a limited set of variables from two successive time slices (1 lagged time slice included)

6.1.3 Dependencies between selected variables covering successive time slices

Finally we investigated the dynamic dependencies within the system by analysing measurement data from two successive time slices for a limited set of variables. This type of model could be used for forecasting future values when new data is measured, especially when more than two slices are included into the model. More time slices could be added into the network but we have been using demo inference tools which do not allow one to load complex networks. In Figure 6-3, the lagged variables have been positioned to the top and the current variables are at the bottom. One may notice that some of the edges are directed counter intuitively from the current variables to the lagged ones. This could be corrected by inputting domain knowledge stating that lagged variables can not be caused by the later variable values (partial order).

6.2 Second phase: developing forecasting models to predict quality indicators

In the second phase, a Bayesian network model was estimated with preprocessed data set containing the measurements relevant for the aeration line 3 (ilmas3) and three levels of differences within those data sets (D1 stands for 1-day difference between values X_t - X_{t-1},

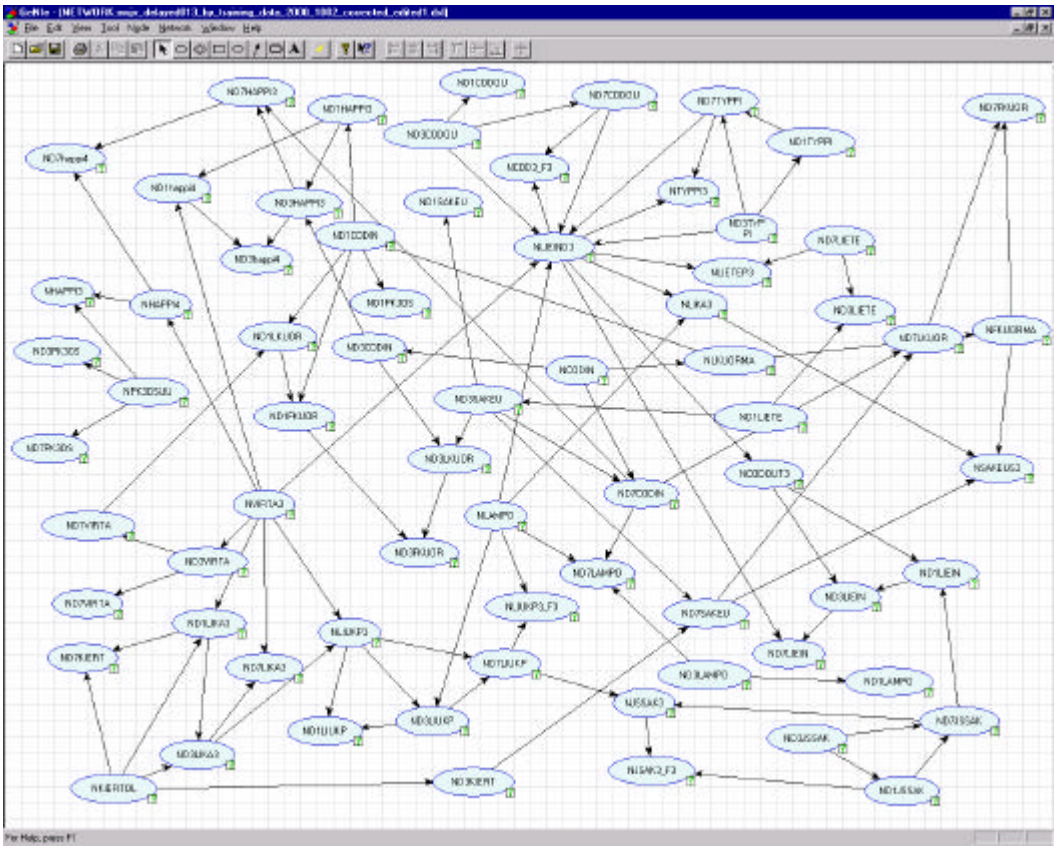

Figure 6-4 The Bayesian network discovered from the training data for the aeration line3

D3 stands for 3-day difference between values X_t - X_{t-3} and D7 stands for 7-day difference between values X_t - X_{t-7}). The models also contained future values (3-day forecasts) for selected quality indicators (NLiukP3_F3, NCOD3_F3 and NJSAK3_F3).

In Figure 6-4, we show the qualitative dependency structure of the discovered Bayesian network, which was learned from the training data (310 first records). One can see that the network is rather sparse but all the variables seem to be dependent on others in some way. Also there are no isolated subnetworks although the actual sensor readings and their temporal differences seem to be connected in most cases. While building the model some domain knowledge was included: the predicted variables were forced to be leaf nodes and clear external (environment) variables (NLAMPO, NVIRTA3 and NKIERTOL) were forced to be root nodes. One can see that especially the calculated variable (NLieind3) is strongly connected to other variables (6 parents and 6 children). From the dependencies we see that each of the target variables is only dependent on two variables: NLiukP3_F3 is only dependent on NLAMPO and on ND7LiukP (the seven day difference in observations of the target variable). NCOD3_F3 is dependent on NLIEIND3 and ND7CODOU. NJSAK3_F3 is dependent on NJSSAK3 and ND1JSSAK.

The prediction accuracy regarding the target variables of this Bayesian network model was tested by embedding the belief propagation into a C++ test program using the SMILE program library (see Appendix 5 for further information). The test consisted of loading the network from a model file. Then for a validation data set (70 last records available)

predictions were made for the target variables by inputting all the available evidence into the model and propagating the beliefs in the system and then by recording the results into a results file. The results were analyzed using MS/Excel in order to produce the following results.

In the first quality indicator, NLiukP3_F3, the states 3 and 4 indicate above normal level (>0.5) and the states 0-2 desired behaviour. In Figure 6-5, the 3-day predictions are shown for the validation set. They have been formed by weighting the state values according to the posterior belief of the state after propagating the evidence in the network. One can see that the predictions seem to follow the average behaviour of the time series. The predictions do not catch the low values that occur at the end of the validation period. The average absolute error of the predictions is 1.18 units (4 is the maximum possible error). The predictions had a 16.7% false positive rate[7] but a 93.8% false negative rate[8]. In Figure 6-6, one sees the predictions obtained by using the most probable (posterior probability) state as the forecasted value. One can easily see that the predicted values have more fluctuation than with the weighted average predictions. The low values at the end of the validation period are problematic also now. Therefore the match with reality is not good. The average absolute error of the predictions is now 1.43 units.

In Figure 6-7, one sees that for the whole data set the weighted predictions tend to follow the mean behaviour of the time series. For the training set the most probable state predictions seem to agree fairly well with the actual values (seeFigure 6-8).

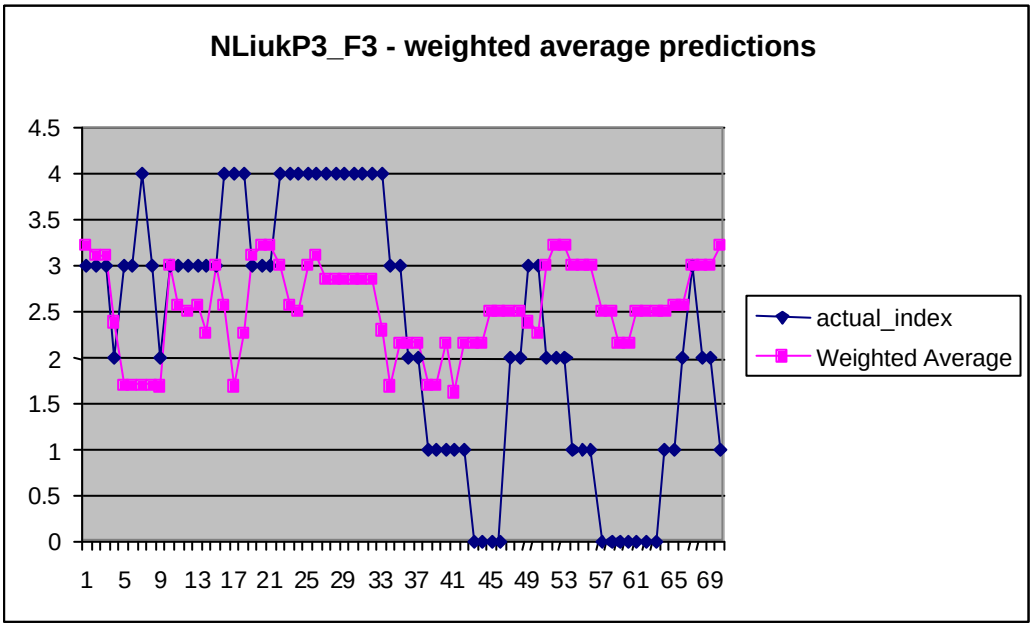

Figure 6-5 Three-day predictions for the categorized NLiukP3_F3 variable using average state predictions calculated by weighting the states with the posterior belief of the state

[7] Here false positive rate means the portion of the real low values that were erroneously predicted to be high ones

[8] Here false negative rate means the portion of the real high values that were erroneously predicted to be low ones

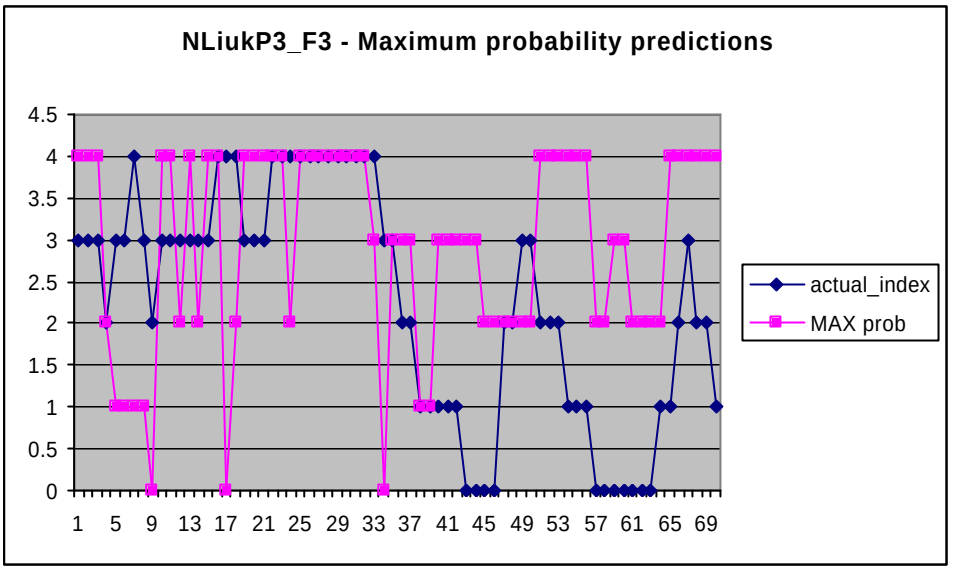

Figure 6-6 Three-day predictions for the categorized NLiuk_P3_F3 variable using the most probable (posterior probability) state as the prediction

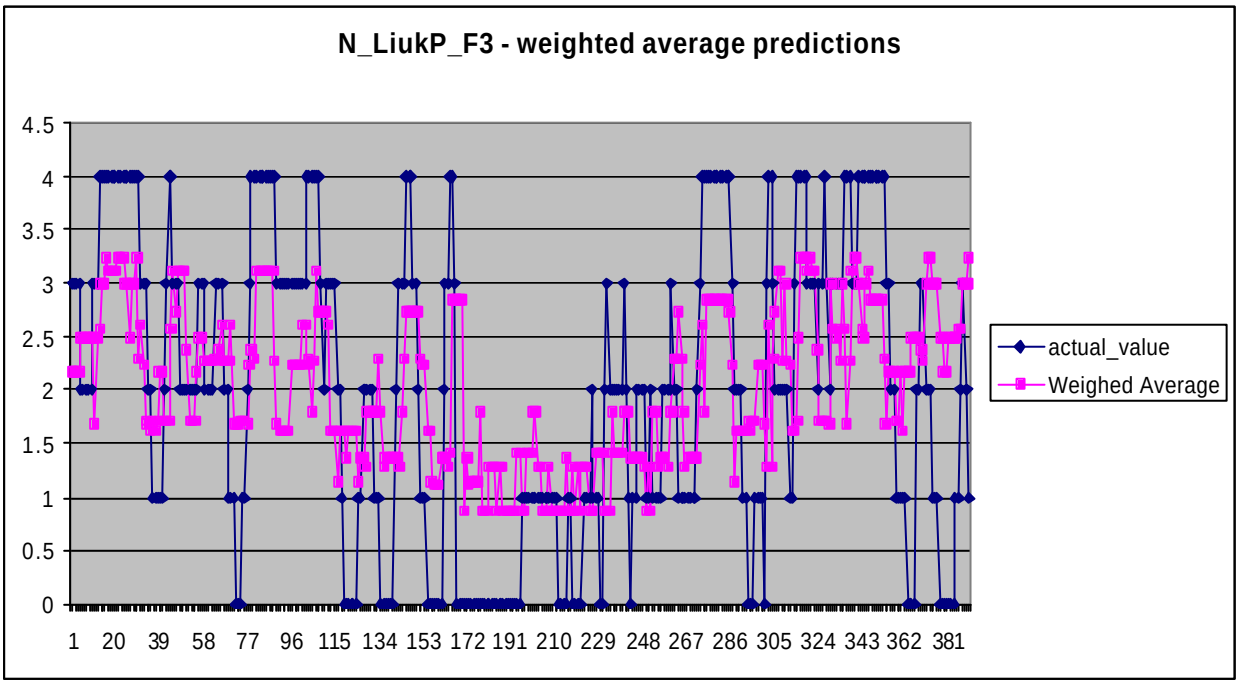

Figure 6-7 Three-day predictions (for all data) for the categorized NLiukP3_F3 variable using average state predictions calculated by weighting the states with the posterior belief of the state

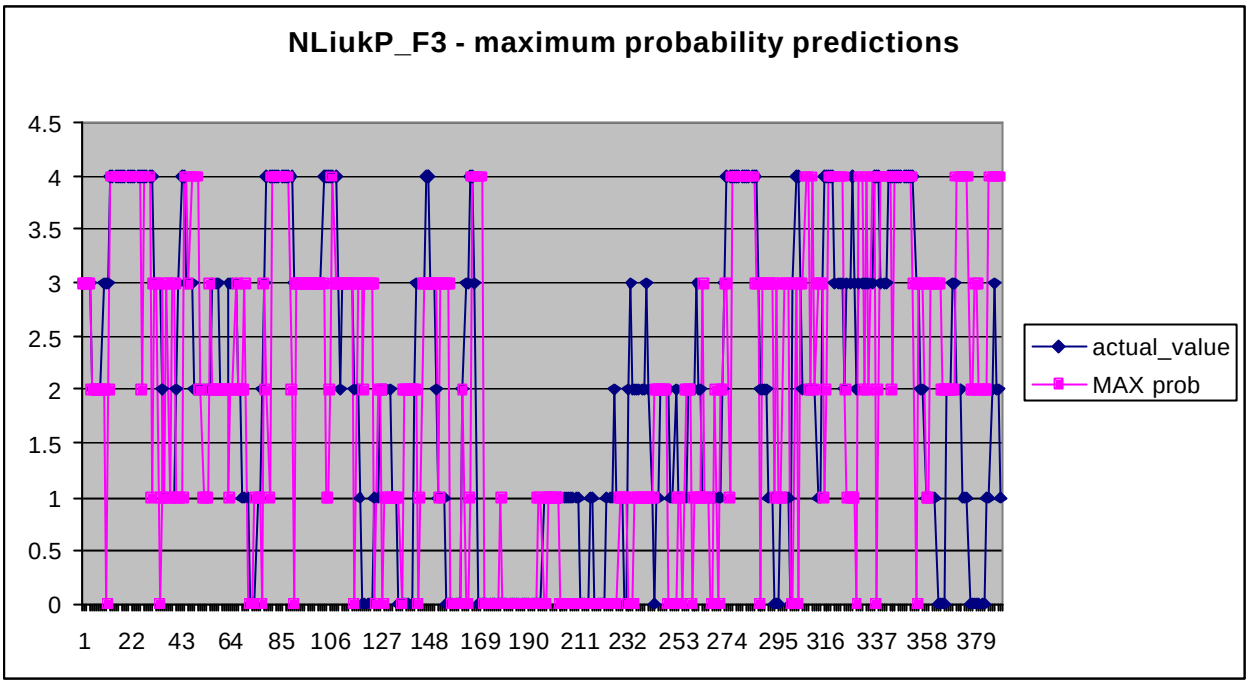

Figure 6-8 Three-day predictions(for all data) for the categorized NLiuk_P3_F3 variable using the most probable (posterior probability) state as the prediction

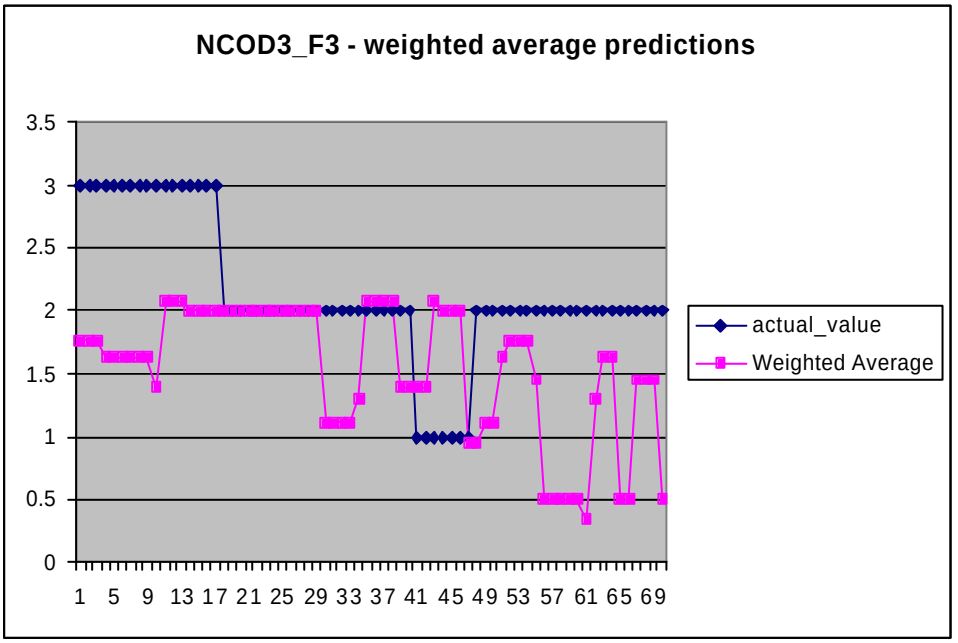

Figure 6-9 Three-day predictions for the categorized NCOD3_F3 variable using average state predictions calculated by weighting the states with the posterior belief of the state

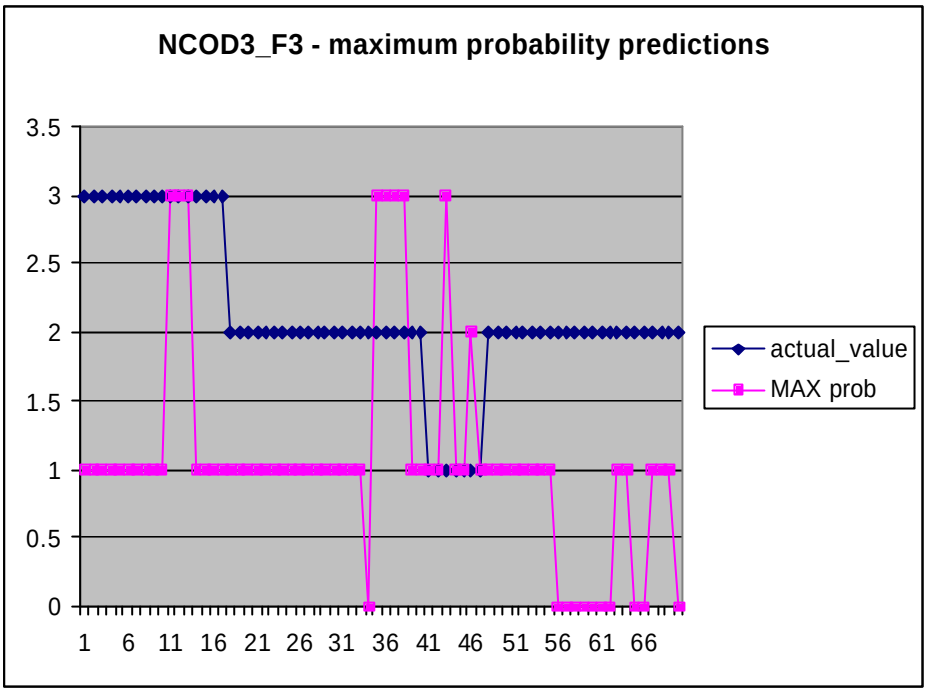

Figure 6-10 Three-day predictions for the categorized NCOD3_F3 variable using the most probable (posterior probability) state as the prediction

In the second quality indicator, NCOD3_F3, states 3 and 4 contain mostly above normal level values (>200) and the states 0-2 indicate desired behaviour. In Figure 6-9, the 3-day predictions are shown for the validation set. They have been formed by weighting the states according to the posterior belief of the states after propagating the evidence in the network. The match between the values is not very good. The average absolute error of the predictions is fare (0.75 units when 4 is the maximum possible error). The predictions had a 0% false positive rate but a 100% false negative rate. In Figure 6-10, one sees the predictions obtained by using the most probable (posterior probability) state. One can easily see that the predictions are even further from the actual values than with the weighted averages but the levels of the predictions change more rapidly. The average absolute error of the predictions is now 1.26 units.

In the third quality indicator, NJSAK3_F3, only state 4 contains some values with above normal level (>50) and the states 0-3 indicate desired behaviour. In Figure 6-11, the 3-day predictions are shown for the validation set. They have been formed by weighting the states according to the posterior belief of the states after propagating the evidence in the network. The predictions tend to be higher than the real values. The average absolute error of the predictions is not too large (0.85 units while 4 is the maximum possible error. In Figure 6-12, one sees the predictions obtained by assigning the prediction to the most probable (posterior probability) state. One can easily see that the predictions have some extreme errors and also the levels of the predictions change very rapidly. The average absolute error of the predictions is now 1.13 units.

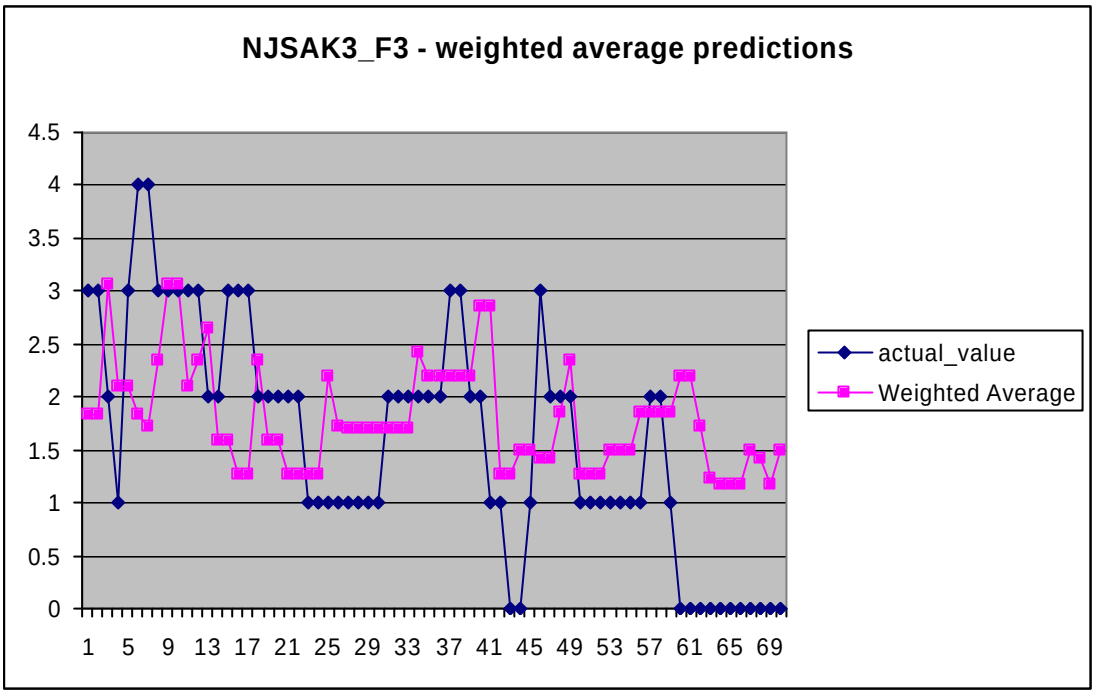

Figure 6-11 Three-day predictions for the categorized NJSAK3_F3 variable using average state predictions calculated by weighting the states with the posterior belief of the state

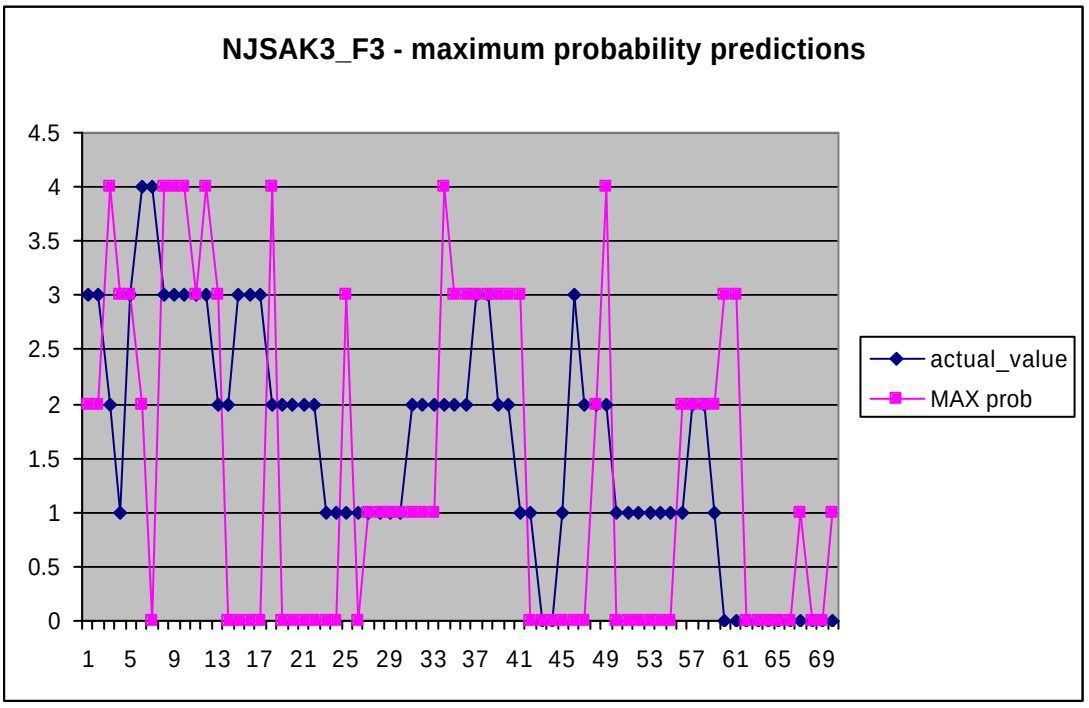

Figure 6-12 Three-day predictions for the categorized NJSAK3_F3 variable using the most probable (posterior probability) state as the prediction

7 Experiences with the discovery of Bayesian networks

BNPC2.0 was used to infer the structure of a Bayesian network from categorized data. The tool allowed one to select the desired variables from the available ones. Also one could specify parameters and various sorts of domain knowledge. One problem with the specification of the domain knowledge is that one was not able to save previously defined domain knowledge and later adapt it – it had to be defined over and over again even when slight changes were needed. This made it difficult to specify all the available knowledge. Typically the domain information requires the definition of hundreds of partial orders for either temporal (lags) or other dependencies (like structural model dependencies). This could be easily corrected in the tool by allowing the user to store the previously defined domain knowledge into a database or file and allowing him to integrate that data when needed. Also the embedding of historical variable values to the same record could be supported directly by the tool. With minor domain knowledge, the tool was able to direct all the dependencies. The edges remaining undirected have to be directed manually using a network editor.

When all the edges have been directed the model may be saved in a file format and the probabilities are estimated from the training data at the same time. We used the Netica-format as it was the only common format with the Bayesian network reasoning tool, Genie, we were using. Unfortunately, Genie expected the model names to be shorter than specified by BNPC and the variable dependencies had to be defined so that all the predecessors of the node had to be defined earlier in the file (BNPC defined the dependencies in an alphabetical order). Therefore, the model file had to be manually edited before it could be imported into Genie. In order to produce pleasing network diagrams, one had to manually arrange the nodes using the graph editor – which is fairly tedious with big diagrams.

Once the required modifications had been done it was possible to reason with the model by embedding it into user programs using the SMILE library.

In phase one, the produced qualitative network diagrams were found to agree with the intuition of the domain experts concerning the dependencies. However, many times the suggested dependency direction was counter intuitive but this could be easily corrected in the model discovery phase by manually redirecting the dependencies.

8 Summary

The Bayesian network formalism and some of the available inference and learning tools have been briefly introduced. Our industrial case problem, the activated sludge waste water cleaning process, is then introduced and the process of discovering Bayesian dependency networks from measurement databases in then outlined. Next we report experiences with using Bayesian networks in the analysis. In the first phase, the task was to identify useful depedencies between the measurement entities. In the second phase, the aim was to identify predictive models from enriched training data. The accuracy of the generated Bayesian network models was then evaluated on validation data.

The results were found to be interesting while indicating that with the available data set size the predicted variables were probabilistically directly dependent on only a couple of variables. This suggests that that the values of the future variables may be predicted upon entering the current values for those variables only. However, the produced prediction accuracy for a validation data set was not good enough for operational usage. These results agree with our experience with applying artificial neural network models for the same task [Hiir00].

Discretation of the variables into five states causes some of the available information to be lost. The Bayesian network techniques with discrete variables is also not most suitable for time series prediction tasks but is useful for better understood, statistically uncertain domains (e.g. fault diagnostics tasks).

Acknowledgements

The embedded prediction models developed in phase 2 and described in this paper were created using the GeNIe modeling environment developed by the Decision Systems Laboratory of the University of Pittsburgh (http://www.sis.pitt.edu/~dsl).

References

[Bunt96] Wray Buntine, Graphical Models for Discovering Knowledge. In Advances in Knowledge Discovery and Data Mining. 1996, pp.59-82.

[Cast97] Enrique Castillo, Jose Manuel Gutierrez, Ali S. Hadi, Learning Bayesian Networks, Chapter 11 in Expert Systems and Probabilistic Network Models (Monographs in Computer Science), Springer-Verlag, 1997.

[Chen98] Cheng, J., Bell, DA, Liu, W., Learning Bayesian Networks from Data: an Efficient Approach Based on Information Theory. An unpublished technical report, http://www.cs.ualberta.ca/~jcheng/Doc/report98.pdf, 41 pages, 1998.

[Che97a] Cheng, J., Bell, DA, Liu, W., Learning belief networks from data: an information theory based approach. In Proceedings of the Sixth ACM International Conference on Information and Knowledge Management (CIKM'97), 1997.

[Che97b] Cheng, J., Bell, DA, Liu, W., An algorithm for Bayesian network construction from data. In Proceedings of the 6th International Workshop on Artificial Intelligence and Statistics (AI&STAT'97), 1997.

[Frie98] Nir Friedman, Moises Goldszmidt, AAAI-98 Tutorial: Learning Bayesian Networks from Data, 1998, http://robotics.Stanford.EDU/~moises/tutorial/index.htm .

[Heck96] Heckerman, D., Bayesian Networks for Knowledge Discovery. In Advances in Knowledge Discovery and Data Mining. 1996, pp.273-305.

[Hiir00] Hiirsalmi, M., Prediction of quality indicators based on measurement history, Research Report TTE1-2000-28 of the VTT Information Technology. 2000.

[Jord99] Michael Jordan, Zoubin Ghahramani, Tommi Jaakkola, Lawrence Saul, An Introduction to Variational Methods for Graphical Models, In Jordan, Learning in Graphical Models, 1999.

[Neap90] Richard E. Neapolitan, Probabilistic Reasoning in Expert Systems : Theory and Algorithms, 1990.

[Pear88] Judea Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, 1988.

[Ramo98] M. Ramoni and P. Sebastiani, Bayesian Methods for Intelligent Data Analysis, KMi Technical Report KMI-TR-67, Knowledge Media Institute, The Open University, June 1998.

[Ram97a] M. Ramoni and P. Sebastiani, Discovering Bayesian Networks in Incomplete Databases, Technical Report KMI-TR-46, Knowledge Media Institute, The Open University, March 1997.

[Ram97b] M. Ramoni and P. Sebastiani, *Learning Bayesian Networks from Incomplete Databases*, Technical Report KMI-TR-43, Knowledge Media Institute, The Open University, February 1997.

[Ram97c] M. Ramoni and P. Sebastiani, Efficient Parameter Learning in Bayesian Networks from Incomplete Databases, Technical Report KMI-TR-41, Knowledge Media Institute, The Open University, January 1997.

Appendix 1: Belief Network PowerConstructor learning software

BN PowerConstructor [Chen98] is a knowledge discovery tool designed to extract Bayesian networks from complete databases. The aim is to provide efficient methods for learning the best matching Bayesian network dependency structure from data and to implement the algorithms in an embeddable platform.

This algorithm belongs to the group of conditional independence (CI) test based algorithms. It does not require complete node ordering but can use such information as constraints. Mutual information calculation is used as the quantitative CI test to avoid the exponential complexity on CI tests. The algorithm extends Chow and Liu's tree construction algorithm to belief network construction by using a three phase mechanism: drafting, thickening and thinning. In the drafting phase the mutual information of each pair of nodes is calculated as a measure of closeness and a draft is created based on this information. In the thickening phase, edges are added when the pair of nodes cannot be graph-separated (d-separated in the case with complete node ordering, otherwise a much more complex search is required) and the nodes are conditionally dependent based on a CI test. In the thinning phase, each edge of the graph is examined using CI tests and will be removed if the two nodes of the edge can be graph-separated. If the complete node ordering is not known, an edge orientation procedure is required at the end.

The algorithm has been tested on common data sets and has been found to operate efficiently. Empirical results show that the running time is roughly linear to the number of training cases and $O(N^2)$ to the number of attributes when the complete node ordering has been given and $O(N^4)$ when the node ordering is not available [Cheng98].

The tool has been tested on artificially generated training cases like the ALARM (A Logical Alarm Reduction Mechanism) Bayesian network with 37 nodes and 46 arcs, and the Hailfinder network (used for forecasting severe summer hail storms in Colorado) having 56 nodes and 66 arcs. It was empirically noticed that with 10000 training cases of ALARM and no node ordering Algorithm B was able to construct an almost correct network in 16 minutes on a Pentium 133MHz PC and in about 27 minutes for the Hailfinder network with 10000 cases. It was noticed that the running time is closely related to the number of CI tests required and that most of the time is spent in accessing the database. With such sparse networks, the running time with Algorithm B is not much slower than with Algorithm A. The system has also been successfully applied to analyze real world data.

The user may provide domain knowledge to the system in various ways by either specifying a complete ancestral node ordering or partial orders for known indirect causes or temporal orders, causes and effects may be indicated and some dependencies may be forbidden. Also some nodes may be marked as root or leaf nodes.

The algorithms apply to discrete-valued variables. A built-in discretizer is available for detecting continuous variables and for discretising them. However, it was not yet available for data stored in Excel file format.

The binary version of the tool is available for free download for 32-bit Windows systems (http://www.cs.ualberta.ca/~jcheng/download.htm). It is easily embeddable to data mining and knowledge based systems as it has been implemented as an ActiveX DLL.

BN PowerConstructor has a wizard-like user interface where the user first selects the data source and then the fields to include, then he specifies the possibly available domain knowledge and then learns the structure of the Bayesian network dependencies. The structure may be edited using a simple network editor. Possibly remaining undirected arcs have to be directed before the model parameters may be estimated and the model may be exported into a file format. Many common Bayesian network file formats are available for: Hugin, Netica and the Bayesian network standard format proposal.

One of the shortcomings of BN PowerConstructor is that sometimes the tool does not know how to orient some of the nodes and the orientation has to be done manually using the BN editor. However, the undirected arcs are difficult to find in the editor.

Support for continuous variables and for hidden or latent variables is being planned.

Appendix 2: Bayesian Knowledge Discoverer (BKD) learning software

Bayesian Knowledge Discoverer (BKD) [Ramo98], [Ram97a], [Ram97b], [Ram97c] is a knowledge discovery tool designed to extract Bayesian Belief Networks (BBN) from possibly incomplete databases. The aim is to provide sound and accountable statistical methods for extracting BBNs from databases. The tool provides facilities to propagate evidence in BBNs, to select an applicable model and to estimate the parameters for a selected BBN dependency structure. Belief propagation is performed using goal-oriented propagation with incremental absorption of evidence (introduced by Schachter).

For model selection, the user has to provide a node order for the variables (specifying the ancestral ordering of the variables). The available search methods include greedy search, arc inversion and exhaustive search. In greedy search, an arc is added from a possible parent if this increases the log likelihood of the dependency. In arc inversion a greedy search is performed where one attempts to add an arc from any possible node as long as no cycles are introduced. In exhaustive search, all possible combinations of nodes are explored and the one with the highest log likelihood is selected unless a cycle is introduced. The model selection extends Cooper&Herskovitz' approach to incomplete databases by introducing a new estimation method called Bound&Collapse (BC). BC computes the extreme probability distributions consistent with the available data. In the Bound step, the set of estimates consistent with the available information are calculated and in the Collapse step, the resulting interval is collapsed to a point estimate via a convex combination of the extreme points with weights depending on the assumed pattern of missing data.

The user may provide domain knowledge to the system by inputing prior probabilities and his confidence on these numbers by giving an imaginary sample size or by editing the network structure.

The algorithms apply to discrete variables but also unsupervised discretization of continuous variables is supported. The discretization method divides the variable into four states defined by intervals.

The tool executable version is publicly available (http://kmi.open.ac.uk/projects/bkd/) for Windows/NT and has been implemented with Common Lisp and CLOS but the source code is not publicly available.

BKD has a command line interface and also a graphical user interface acting as a visual front end command line interface. Typically, the tool is used by first selecting the desired data source for learning data and then loading the variable names from the data file field names. If needed, the data must be discretized. A new model structure may be searched by using Generate Model command, which finds the most likely network structure or by manually editing the dependency arcs. Then the conditional and prior probabilities may be adjusted with the command Quantify from Data. It uses possible prior probabilities as priors when estimating the probabilities from data. The priors may be initialized by the command Initialize. Model dependencies may be edited and new probabilities may be estimated from data. However, there is no quality indicator showing how well the model fits the data.

Currently there is no API for connecting the program to other modules. The tool operates as a standalone program using files as its external interface.

Currently the order of the nodes can not be adjusted but the nodes have to be introduced so that parents are specified before their children (in the ancestral order).

Also the number of variables has to match the number of fields in the database. There can be no external or latent variables in the estimated models. Anyhow, they can be introduced later into the model using other tools or manually.

Appendix 3: HUGIN Bayesian network tool

The HUGIN system is a tool for constructing Bayesian network based inference modules for decision support systems. These modules are able to represent uncertainty in the status of the variables and in the probabilistic dependencies between the variables. Also influence diagram representations are supported.

The HUGIN system provides both an application programming interface (HUGIN API) and a graphical environment and development facilities for interactively defining Bayesian network structures and associated probability matrices. Discrete variable types and marginal continuous variables (ones that may be conditionally dependent on discrete and continuous nodes) are currently supported. A discrete node may not contain continuous nodes as parents. The tool allows the user to define the conditional probabilities as full matrix tables or by using composite expressions allowing generalized interfaces (like logical functions, noisy-OR formalisms and discrete and continuous probability distributions). Currently it is still not possible to represent local structures capturing the independences between the parent node states [Nir99].

In the API, support is also provided for approximating the networks and for compressing sparse probability tables. Also detection of conflicts in the evidence is supported. The models may be adjusted by estimating new prior probabilities based on gathered case databases of known cases and combining them with the previous existing priors (parameter learning).

Once the models have been defined they may be used in an inference engine, which allows one to compute the marginal (in equilibrium or in consistent state) probability of each node given the available evidence. Also one may calculate the most probable combination of variable states given the evidence. While using the influence diagrams, one may calculate the most cost effective decision.

The inference module uses an exact propagation algorithm which is based on transforming the dependency network into an undirected Markov network by clustering the nodes in order to form a junction tree where each node contains as few states as possible. This algorithm provides good solutions for networks that are relatively sparse. Such networks are typical for real world decision support systems at least after suppressing the weak dependencies from the models by pruning.

Hugin is distributed in various product packages: HUGIN Lite is a demonstration version of HUGIN Explorer, which is limited to networks containing at most 200 states. It is distributed free of charge in http://www.hugin.dk/ . HUGIN Explorer is a graphical expert system shell for construction and execution of Bayesian belief networks. Hugin Professional contains the graphical expert system and the HUGIN API either as a C library or as an ActiveX server. Hugin Professional+ contains HUGIN API in both alternative formats.

Hugin has licenses for commercial and for educational usage.

HUGIN prices	Commercial		Educational	
2.11.1999				
(In Euro)				
	1 user	site	1 user	site
HUGIN Explorer	3000	10000	1000	3000
HUGIN Professional	7500	20000	2500	5000
HUGIN Professional+	10000	25000	3500	6000

It is clear that for integrating HUGIN applications with embedded systems would require HUGIN API and therefore at least HUGIN Professional is needed.

HUGIN has been licensed by many organizations researching and applying Bayesian networks. It is being used in decision support, diagnosis and health monitoring, trouble-shooting tasks and also risk management and safety assessment tasks.

Major development projects participated by the HUGIN Expert company include EU ESPRIT projects Advocate and Serene. In Advocate, HUGIN is used for situation assessment and diagnosis in order to increase performance of Unmanned Underwater Vehicles. In Serene, Hugin is used as a tool to create a quantitative safety assessment of the evaluated system by combining many different types of evidence regarding system safety. In this project, object oriented belief network (OOBN) formalism has been created and also some tools and methods for acquiring the knowledge have been created. OOBN formalism is planned to be integrated into the HUGIN framework for version 0. Hugin Expert is also developing automated decision support systems for Customer Support at Hewlett-Packard. HP acquired 40% of the company in 1998 and close co-operation is now taking place.

Appendix 4: Netica Bayesian network tool

Netica is a program implementing Bayesian networks and influence diagrams. It is licensed by Norsys Software Corp. from Vancouver, Canada, http://www.norsys.com .

The program provides a graphical development environment to edit Bayesian networks or influence diagrams and to perform inferences with them. The networks are solved using the junction tree algorithm (like Hugin). Interactions between variables may be defined using conditional probability tables or using equations. The probabilities may also be learned form training cases. The system supports delayed links between variables. Such models are automatically transformed into static models.

It is possible to reverse individual links of the network (the tool updates the probabilistic dependencies automatically) and also to remove nodes (the system updates the probability of the other nodes as appropriate).

Additionally, an application programmer's library (API) C programmer's interface is available for integrating the system to other programs.

A demo version (does not save models consisting of more than 15 variables and does not load models larger than 50 variables) is available from Norsys' web pages.

The graphical user interface is supported in MS/Windows and Macintosh platforms. The API is supported in MS/Windows, PC/Linux and SunOS.

Prices for the graphical interface (NETICA):

	Regular Pricing	Introductory Offer
Commercial	$1185 US	$585 US
Educational / Personal	$585 US	$285 US
Volume or Site license	Contact Norsys	

Prices for the C API interface (NETICA API):

Regular Pricing	Introductory Offer	
Commercial	$1385 US	$685 US
Educational / Personal	$585 US	$285 US
Volume or Site license	Contact Norsys	

Additionally, if the API is used in a commercial product, then a per copy license fee applies, and this must be arranged with Norsys. Depending on the volume of the product, the price of the product, etc., but is generally quite affordable ($10 - $100).

Appendix 5: Genie/Smile Bayesian network tool

Genie is a program providing support for inferencing with Bayesian networks and influence diagrams. It has been developed by the Decision Systems Laboratory, University of Pittsburgh and is available for research purposes (http://www.sis.pitt.edu/~dsl).

Genie provides a graphical development environment for editing Bayesian networks and influence diagrams and to perform inference with them. The networks are solved using the junction tree algorithm (like Hugin). Interactions between variables may be defined using conditional probability tables. The models are saved in various file formats. Genie has a propritary format but supports also many others, like the Netica format and the standard proposal format.

Additionally, an application programmer's library (API) C programmer's interface, called SMILE, has been made available for integrating the system to other programs. The library may be used for research purposes. The source code is not available.

The graphical user interface is supported in MS/Windows and Linux platforms. The API is supported in MS/Windows and Linux.

While testing, the tool seemed to be very stable and easy to use. It serves useful purpose while making it possible to test Bayesian network formalisms in an easy way.