Dynamic Data Feed to Bayesian Network Model and SMILE Web Application

Nipat Jongsawat[1], Pittaya Poompuang[2], Assoc.Prof.Dr.Wichian Premchaiswadi[3]
Graduate School of Information Technology in Business, Siam University, Thailand
nipatj@yahoo.com[1], pitaya_pp@yahoo.co.th[2], wichian@siam.edu[3]

Abstract

A complete diagnostic Bayesian network model cannot be achieved and the result of the constructed model cannot be guaranteed unless correct and reliable data are provided to the model. In this paper, we propose a technique to dynamically feed data into a diagnostic Bayesian network model in the first part of this paper. In the second part of the paper, a case study of several factors that have an impact on students for making a decision in enrollment is transformed into a Bayesian network model. The last part of the paper discusses a web user interface for the model in terms of its design and diagnosis. The user is allowed to perform a diagnosis of the model through the SMILE web application interface.

1. Introduction

Constructing Bayesian network models [8] is a complex and time consuming task. It is difficult to obtain complete and consistent models but to get the correct and reliable probability data for the designed models is much more difficult. Normally, there are two methods to enter the probability values into the chance node of a Bayesian network model. The first method is to consult an expert for the probability values and enter them into the models. The second method is to obtain probability values from statistical or learned data [6]. Both methods use static data, not dynamic data. The second method acts like dynamic data but it is not. The statistical data from a database need to be loaded and processed each time to get the probability values. This works similar to batch processing. Finally, users still need to enter probability values into the model by manual feeding the data by hand. It is not possible to have real-time processing. The probability values are fed to every node of the model and the joint probability distribution is computed at the final stage when the model is performing Bayesian updates. The disadvantage of using manually fed data or static data is that it cannot be performed in using real-time processing, monitoring, and updating.

In this paper, we propose a technique for feeding data into the Bayesian network model dynamically. Users can perform Bayesian inference in real-time. They can compute the impact of observing values of a subset of the model variables on the probability distribution over the remaining variables. Using the software tool called "GeNIe", presented in section 3, for constructing a Bayesian network model there are some limitations such as dependent platform and is unusable on a global basis. This paper proposes the SMILE web application to overcome these limitations. The students' attitude on several factors in an enrollment decision is the basis of our case study in this paper and is also transformed into a Bayesian network model in the SMILE web application.

2. Fundamentals

This section is intended to describe the fundamentals and techniques for implementing a Bayesian network model in general. They are the followings:

2.1. Bayesian network

Bayesian networks (also called belief networks, Bayesian belief networks, causal probabilistic networks, or causal networks) (Pearl 1988) [10] are acyclic directed graphs in which nodes represent random variables and arcs represent direct probabilistic dependencies among them. The structure of a Bayesian network is a graphical, qualitative illustration of the interactions among the set of variables that it models. The structure of the directed graph can mimic the causal structure of the modeled domain, although this is not necessary. When the structure is causal, it gives a useful, modular insight into the interactions among the variables and allows for prediction of the effects of external manipulation.

Nodes of a Bayesian network are usually drawn as circles or ovals. The following simple Bayesian network, shown in Figure 1, represents two variables, Curriculum and Enrollment, and expresses the fact that they are directly dependent on each other.

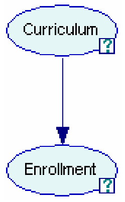
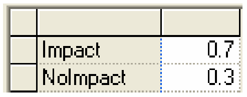


Figure 1. An example of bayesian network.

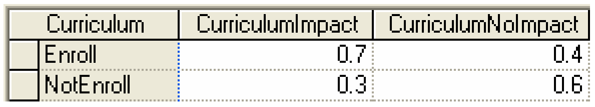
A Bayesian network also represents the quantitative relationships among the modeled variables. Numerically, it represents the joint probability distribution among them. This distribution is described efficiently by exploring the probabilistic independence among the modeled variables. Each node is described by a probability distribution conditional on its direct predecessors. Nodes with no predecessors are described by prior probability distributions. For example, the node Curriculum shown in Figure1 will be described by a prior probability distribution over its two outcomes: Impact and NoImpact. See Figure 2 below.



Impact	0.7
NoImpact	0.3

Figure 2. Prior probability distribution for a curriculum node.

The enrollment node will be described by a probability distribution over its outcomes (Enroll, NotEnroll) conditional on the outcomes of its predecessor (node Curriculum outcomes, Impact and NoImpact). See Figure 3 below.



Curriculum	CurriculumImpact	CurriculumNoImpact
Enroll	0.7	0.4
NotEnroll	0.3	0.6

Figure 3. Conditional probability values for an enrollment node.

Both the structure and the numerical parameters of a Bayesian network can be elicited from an expert. They can also be derived from data, as the structure of a Bayesian network is simply a representation of independencies in the data and the numbers are a representation of the joint probability distributions that can be inferred from the data. Finally, both the structure and the numerical probabilities can be a mixture of expert knowledge, measurements and objective frequency data.

2.2. Bayesian updating

Bayesian updating, also referred to as belief updating, or somewhat less precisely as probabilistic inference [7], [9] is based on the numerical parameters captured in the model. The structure of the model which is an explicit statement of the independencies in the domain helps in making the algorithms for Bayesian updating [9] more efficient. All algorithms for Bayesian updating are based on a theorem proposed by Rev. Thomas Bayes (1702-1761) and is known as Bayes Theorem.

Belief updating in Bayesian networks is computationally complex. In the worst case, belief updating algorithms are NP-hard (Cooper 1990) [7]. There exist several efficient algorithms, however, that make belief updating in graphs consisting of tens or hundreds of variables tractable. Pearl (1986) developed a message-passing scheme that updates the probability distributions for each node in a Bayesian network in response to observations of one or more variables. Lauritzen and Spiegelhalter (1988), Jensen et al. (1990), and Dawid (1992) proposed an efficient algorithm that first transforms a Bayesian network into a tree where each node in the tree corresponds to a subset of variables in the original graph. The algorithm then exploits several mathematical properties of this tree to perform probabilistic inference.

Several approximate algorithms based on stochastic sampling have been developed. Of these, best known are probabilistic logic sampling (Henrion 1998), likelihood sampling (Shachter & Peot 1990, Fung & Chang 1990), and backward sampling (Fung & del Favero 1994), Adaptive Importance Sampling (AIS-BN) (Cheng & Druzdzel 2000), and Approximate Posterior Importance Sampling (APIS-BN) (Yuan & Druzdzel 2003). Approximate belief updating in Bayesian networks has been also shown to be worst-case NP-hard (Dagum & Luby 1993).

2.3. SMILE

SMILE (Structural Modeling, Inference, and Learning Engine) [3] is a fully platform independent library of functions implementing graphical probabilistic and decision-theoretic models [5], such as Bayesian networks, influence diagrams, and structural equation models. Its individual functions, defined in the SMILE Application Programmer Interface (API), allow creating, editing, saving, and loading graphical models, and using them for probabilistic reasoning and decision making under uncertainty.

SMILE can be embedded in programs that use graphical probabilistic models as their reasoning engines. Models developed in SMILE can be equipped with a user interface that best suits the user of the resulting application. SMILE is written in C++ in a platform-independent fashion and is fully portable. Model building and the reasoning process are under full control of the application program as the SMILE library serves merely as a set of tools and structures that facilitates them. The sample source code below is the main function of SMILE that contains the core functions of the implemented model SMILE.

```
int main()
{
    CreateNetwork();
    InfereceWithBayesNet();
    UpgradeToInfluenceDiagram();
    InferenceWithInfluenceDiagram();
    ComputeValueOfInformation();
    return(DSL_OKAY);
};
```

2.4. GeNIe

The GeNIe's name [1] and its uncommon capitalization originate from the name Graphical Network Interface, given to the original simple interface to SMILE, the library of functions for graphical probabilistic and decision-theoretic models. GeNIe is an outer shell of SMILE. It is a development environment for building graphical decision-theoretic models. It is implemented in Visual C++ and draws heavily on MFC (Microsoft Foundation Classes). It allows for building models of any size and complexity, limited only by the capacity of the available memory of the computer.

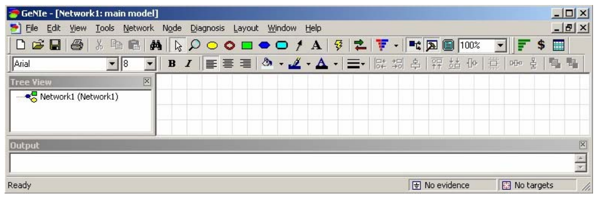

GeNIe - [Network1: main model]
File Edit View Tools Network Node Diagnosis Layout Window Help
Tree View
Network1 (Network1)
Output
Ready
No evidence
No targets

Figure 4. The main GeNIe interface.

3. Graphical bayesian network model

3.1. Bayesian network model in GeNIe

This section describes the graphical Bayesian network model [2] in GeNIe as shown in Figure 5. The students' attitude on several factors in an enrollment decision has been proposed as a case study for the model. This model contains ten variables or nodes. There are nine parent nodes thus there are no predecessor nodes and one child or predecessor node. The outcomes of each parent node are identical. It consists of impact and no impact values. There are also two outcomes for the child node (the enrollment node), enroll and not enroll values. The probability values for each parent node and the values for each state combination with an enrollment node are further defined by an expert.

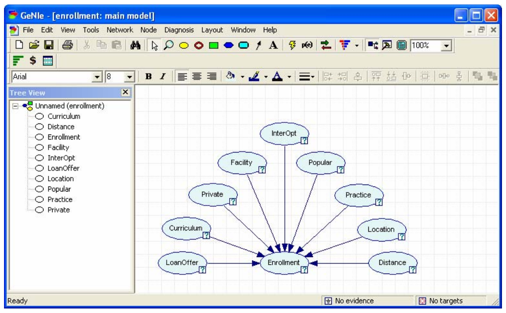


Figure 5. Graphical bayesian network model in GeNIe.

When the specified outcome of each node and their probability values are defined, the belief updating is ready. The belief update allows for performing Bayesian inference [3]. It is used to compute the impact of observing values of a subset of the model variables on the probability distribution over the remaining variables. Working with this model and performing Bayesian inference, we can answer simple questions. For example, the question: "What is the chance for impact for every parent node if the expert judges the prospects for impact to be enroll?" The evidence for the enrollment variable is set at the value of "enroll" as shown in Figure 6. We have observed a value of the enrollment variable and ask it to update its probability distribution over all parent variables. The result is shown in Figure 7.

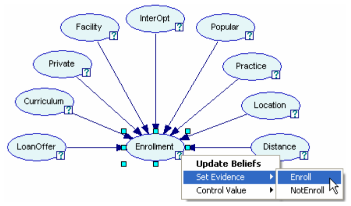
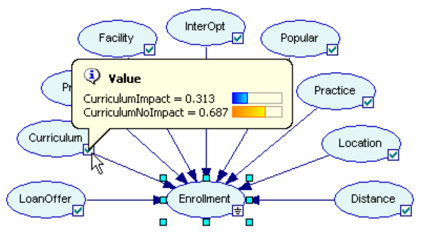


Figure 6. Setting evidence at enroll outcome for an enrollment node.



Value
CurriculumImpact = 0.313
CurriculumNoImpact = 0.687

Figure 7. The posterior probability distribution over a curriculum node.

Constructing a Bayesian network model in GeNIe is simply done. There are a lot of tools provided in GeNIe for working and implementing a model but GeNIe has some limitations. Firstly, GeNIe only runs under the Windows operating systems. GeNIe is implemented in Visual C++ and draws heavily on the MFC (Microsoft Foundation Classes), which runs only on a Windows platform. It does not support cross- platform, web or an Internet-based application environment so that there are some limitations for its use on a worldwide basis. Secondly, the probability value of each variable node must be entered manually. This means that the probability determination method must be done before using GeNIe. The probability values can be obtained by asking the experts, statistical methods, or learned data from a database. However, the probability values are still put into the model by hand because GeNIe itself cannot support real-time or dynamic data. Thirdly, a graphical presentation such as pie chart or bar chart in GeNIe is intentionally designed for displaying an individual node. It does not present an overview or comparison for similar outcomes of all nodes. Lastly, the model in GeNIe is static, not dynamic. The model needs to be loaded, have some values changed, and observe the results after updating beliefs one at a time.

3.2. Client/server architecture for SMILE web application

To overcome these limitations of GeNIe mentioned in 3.1. We designed the SMILE web application that works similar to GeNIe. GeNIe is the interface to SMILE for a windows platform. The SMILE web application is the interface of SMILE on the web or an Internet-based platform. It means that the SMILE web application can support real-time data processing that GeNIe cannot. It also supports a dynamic data feed into the model. See Client/Server Architecture of the SMILE web application in Figure 8.

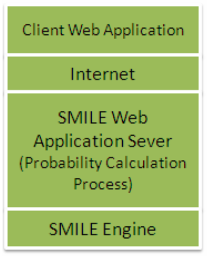

Client Web Application
Internet
SMILE Web Application Sever (Probability Calculation Process)
SMILE Engine

Figure 8. Client/server architecture of SMILE web.

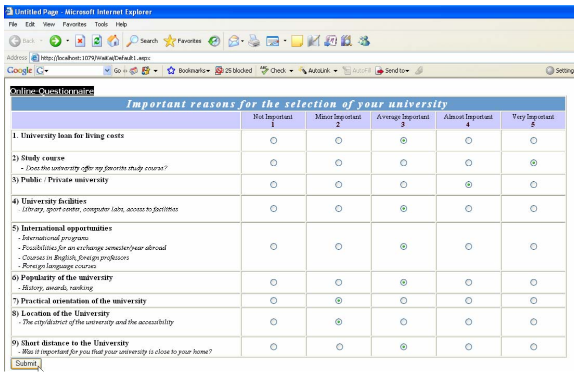
In the client/ server architecture of the SMILE web application, the client web application is designed in order to collect data from students through an online questionnaire. The data from the client is sent over the Internet to the server. The server web application or SMILE web is designed to handle incoming data, calculate probability values and put them into each chance node, construct the Bayesian network model in .xdsl file format [4], feed the calculated probability values into the model, call the core functions of SMILE, read and update probability values for each node in database, send all parameters to SMILE, receive values from SMILE and visualize the results. Both the client and server web application are implemented in the ".NET" environment. Web pages are created by ASP.NET and the code behind is developed in visual C#.net. The code behind the web server application contains the core functions of SMILE such as CreateNetwork, InfereceWith BayesNet, and ComputeValueOf Information. A CreateNetwork function is mainly used for creating the Bayesian network model. This function creates chance nodes, adds arcs from one node to other nodes, and fills in the conditional probability distribution for all nodes in the model. An Inferece WithBayesNet function is used to read the .xdsl file or model, specify the clustering algorithm, update the network or update beliefs, set an evidence for each node and obtain the returned result values. The clustering algorithm in the second function works in two phases: (1) compilation of a directed graph into a junction tree, and (2) probability updating in the junction tree. It has been a common practice to compile a network and then perform all operations in the compiled version. The clustering algorithm [9] is the fastest known exact algorithm for belief updating in Bayesian networks. The clustering algorithm is the SMILE web default algorithm and should be sufficient for most applications. When networks become very large and complex, the clustering algorithm may not be fast enough. In that case, it is suggested that the user choose an approximate algorithm, such as one of the stochastic sampling algorithms. The

"ComputeValueOf Information" function is used to compute an expected value of information for the model.

4. Implementation

According to the Client/Server Architecture of SMILE Web mentioned in section 3, SMILE web is designed to work in a more flexible manner for analyzing and diagnosing reasoning. It is designed for worldwide users, who can access the Internet for diagnosing the model. It overcomes platform dependent, limitations on graphical presentation, and the manual data entry for a Bayesian network model found in GeNIe. To implement SMILE web, there are four main components according to the client/server architecture as follows: 1) Client Web Application, 2) SMILE Server Web Application, 3) Probability Calculation Process, and 4) SMILE Engine.

The first part, client web application, is an online questionnaire designed for prospective students. They are asked to fill out the questionnaire before downloading an application form from the university website. See Figure 9.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:1079/WaiKai/Default1.aspx
Google Go Bookmarks 25 blocked Check AutoLink AutoFill Send to Setting

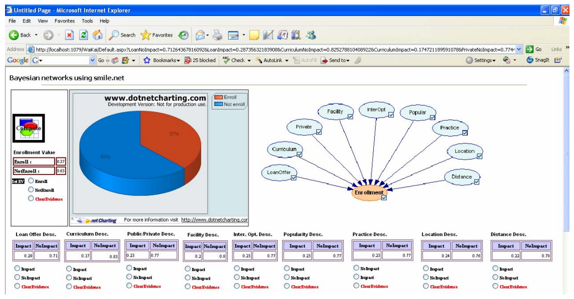
Online-Questionnaire

Important reasons for the selection of your university	Not Important 1	Minor Important 2	Average Important 3	Almost Important 4	Very Important 5
1. University loan for living costs	○	○	●	○	○
2) Study course - *Does the university offer my favorite study course?*	○	○	○	○	●
3) Public / Private university	○	○	○	●	○
4) University facilities - *Library, sport center, computer labs, access to facilities*	○	○	●	○	○
5) International opportunities - *International programs* - *Possibilities for an exchange semester/year abroad* - *Courses in English, foreign professors* - *Foreign language courses*	○	○	●	○	○
6) Popularity of the university - *History, awards, ranking*	○	○	●	○	○
7) Practical orientation of the university	○	●	○	○	○
8) Location of the University - *The city/district of the university and the accessibility*	○	●	○	○	○
9) Short distance to the University - *Was it important for you that your university is close to your home?*	○	○	●	○	○

Submit

Figure 9. Online questionnaire for prospected students.

The second part, SMILE Server Web Application, is designed for the reasoning aspect of the web user interface for SMILE. Users can update beliefs and perform diagnosis through the SMILE web application as GeNIe did, See Figure 10 and 11. The third part, Probability Calculation Process, is actually a probability calculation function in the SMILE web application. It receives the data from client web application (online questionnaire) and processes the probability values in real-time. Moreover, it is responsible for feeding the probability values into the model dynamically. The advantage of this function is that we can get real-time data and probability values for the model that GeNIe could not do. The last part, the SMILE Engine, receives data from the SMILE web application. SMILE's functions such as Create Network (), InfereceWithBayesNet (), and Compute ValueOfInformation () are called to perform according to its operation. The resulting values are sent back to the SMILE web application. The SMILE engine is written in C++ in a platform-independent fashion and is fully portable. The web application's interface is defined in terms of a collection of C++ classes that form the "body" of the library and can be used from within an application program. These classes allow building graphical models, editing, saving and loading them, and using them for probabilistic reasoning and decision making under uncertainty.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:1079/WaiKai/Default.aspx?LoanNoImpact=0.7126436781609 28LoanImpact=0.28735632183908&CurriculumNoImpact=0.825278810408922&CurriculumImpact=0.174721189591078&PrivateNoImpact=0.774
Go Links
Google Go Bookmarks 25 blocked Check AutoLink AutoFill Send to Settings SnagIt
Bayesian networks using smile.net
Compute
www.dotnetcharting.com
Development Version: Not for production use.
Enroll
Not enroll
37%
63%
.net Charting For more information visit http://www.dotnetcharting.com
Facility
InterOpt
Popular
Private
Practice
Curriculum
Location
LoanOffer
Distance
Enrollment

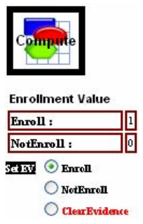
Enrollment Value

Enroll :	0.37
NotEnroll :	0.63

Set EV ○ Enroll ○ NotEnroll ○ ClearEvidence

	Impact	NoImpact
Loan Offer Desc.	0.29	0.71
Curriculum Desc.	0.17	0.83
Public/Private Desc.	0.23	0.77
Facility Desc.	0.2	0.8
Inter. Opt. Desc.	0.23	0.77
Popularity Desc.	0.23	0.77
Practice Desc.	0.23	0.77
Location Desc.	0.24	0.76
Distance Desc.	0.22	0.78

Figure 10. SMILE web application.



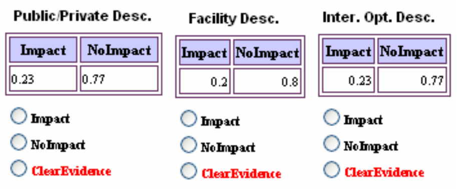
Enrollment Value

Enroll :	1
NotEnroll :	0

Set EV [✓ Enroll] ○ NotEnroll ○ ClearEvidence

Figure 11. Setting evidence at enroll outcome for an enrollment node.

Users are allowed to perform diagnosis by setting evidence at one variable or node and exploring the probabilistic independencies among the modeled variables. See the sample variables, Public/Private University, Facilities, and International Opportunity, in Figure 12.



	Impact	NoImpact
Public/Private Desc.	0.23	0.77
Facility Desc.	0.2	0.8
Inter. Opt. Desc.	0.23	0.77

○ Impact ○ NoImpact ○ ClearEvidence

Figure 12. Three sample nodes for observing values.

The Clear Evidence option is also provided for canceling the diagnosis and going back to use the original values in the calculation. Users can set and clear the evidence at every node in the model in order to perform diagnosis. The graphical representation of SMILE web is shown in Figure 13, 14, and 15.

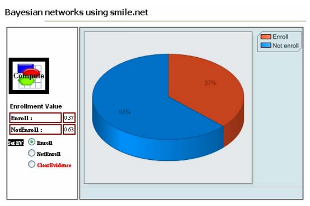


Figure 13. Pie chart for enrollment node.

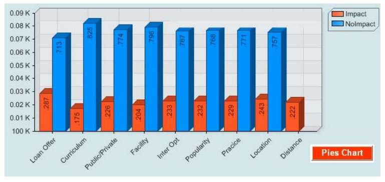


Figure 14. Bar chart for parent nodes.

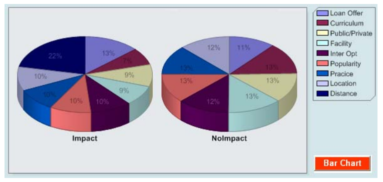


Figure 15. Pie chart for parent nodes.

5. Conclusions and Future Work

GeNIe, Graphical Network Interface, is designed for a windows environment. It works well on a windows platform. It cannot be run on a web or Internet-based platform. That is why there is some limitation for its use on a worldwide basis. Another thing is that it does not support is real-time data processing. To overcome the limitations of GeNIe, the SMILE web application was designed and implemented on a client/server architecture mentioned in section 3. GeNIe is an outer shell of SMILE. SMILE web is also the outer shell of SMILE. The difference is that the SMILE web application is basically constructed in a web-based environment. SMILE web calls and submits parameters to the core functions of SMILE directly. After processing, SMILE returns all computed values back to SMILE web. SMILE web represents the Bayesian network model on a website. It is the model that users, who access the Internet, can utilize to perform diagnosis. They can update the probability distributions for each variable in a Bayesian networks in response to observations of one or more variables. SMILE web also provides a function to handle dynamic data, compute probability values in real-time, and enter them into the model. This paper presents the first step for developing SMILE web. The next step is to enhance the efficiency of SMILE web by improving the SMILE web interface, including more functions, and increasing the flexibility for model creation. The final phase for SMILE web development will be to enable it to handle influence diagrams and structural equation models. Users can use SMILE web for choosing a decision alternative that has the highest expected gain or utility. SMILE web can become a part of their life for helping users in decision making.

References

[1] http://genie.sis.pitt.edu

[2] http://genie.sis.pitt.edu/wiki/Probabilistic_Decision_Support_System:_Bayesian_Networks

[3] http://genie.sis.pitt.edu/wiki/SMILE:_Probabilistic_Inference_in_Bayesian_Networks

[4] http://genie.sis.pitt.edu/wiki/Appendices:_XDSL_File_Format_-_XML_Schema_Definitions

[5] Marek J. Druzdzel and Roger R. Flynn, "Decision Support Systems", Encyclopedia of Library and Information Science, Second Edition, 2002.

[6] Agniezka Onisko, Marek J. Druzdzel, Hanna Wasyluk, and Warsaw, "Learning Bayesian Network parameters from Small Data Sets", International Journal of approximate Reasoning, 27(2):165-182, 2001.

[7] Gregory F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks", Artificial Intelligence, 42(2–3): 393–405, March 1990.

[8] Pieter C. Kraaijeveld, Marek J. Druzdzel, "GeNIeRate: An Interactive Generator of Diagnostic Bayesian Network Models", Air Force Office of Scientific Research under grant F49620-03-1-0187 and by Intel Research.

[9] Paul Dagum and Michael Luby, An optimal approximation algorithm for Bayesian inference, Artificial Intelligence, 93:1–27, 1997.

[10] M. Henrion. Some practical issues in constructing belief networks. In L. N. Kanal, T. S. Levitt, and J. F. Lemmer, editors, Uncertainty in Artificial Intelli-gence 3, pages 161–173. North-Holland, Amsterdam, 1989.