GeNIeRate: An Interactive Generator of Diagnostic Bayesian Network Models

Pieter C. Kraaijeveld
Man Machine Interaction Group
Delft University of Technology
Mekelweg 4, 2628 CD Delft, the Netherlands
`p.c.kraaijeveld@ewi.tudelft.nl`

Marek J. Druzdzel
Decision Systems Laboratory
University of Pittsburgh
Pittsburgh, PA, 15260, USA
`marek@sis.pitt.edu`

Abstract

Constructing diagnostic Bayesian network models is a complex and time consuming task. In this paper, we propose a methodology to simplify and speed up the design of very large models. The models are based on two simplifying assumptions: (1) the structure of the model has three levels of variables and (2) the interaction among the variables can be modeled by Noisy-MAX gates. The methodology is implemented in an application named: GeNIeRate, which aims at supporting construction of diagnostic Bayesian network models consisting of hundreds or even thousands of variables. Preliminary qualitative evaluation of this application shows great promise. We are planning to conduct a systematic study to compare GeNIeRate to traditional techniques for building Bayesian network models and we hope to be able to present the results at the workshop.

1 Introduction

Bayesian Networks (BN) [Pearl, 1988] are acyclic directed graphs with each node representing a variable and each arc representing typically a causal relation among two variables. Although exact and approximate inference in Bayesian networks are both worst-case NP-hard [Cooper, 1990; Dagum and Luby, 1997], they still perform well, in our experience, for practical diagnostic models consisting of several hundred or even thousand nodes.

A BN consists of a qualitative and a quantitative part. The qualitative part is an acyclic directed graph reflecting the causal structure of the domain, the quantitative part represents the joint probability distribution over its variables. Every variable has a conditional probability table (CPT) representing the probabilities of each state given the state of the parent variable. If a variable does not have any parent variables in the graph, the CPT represents the prior probability distribution of the variable. A BN is able to calculate the posterior probability of an uncertain variable given some evidence obtained from related variables. This property and the intuitive way BN model complex relationships among uncertain variables makes it a very suitable technique for building diagnostic models. Diagnosis is quite likely the most successful practical application of BN.

While the existing diagnostic BN models perform very well, the technique is still not widely used and accepted. One of the main reasons for this is that building a BN model is a laborious and time consuming task. During the model building process, both the qualitative and quantitative parts of the BN have to be designed. This can be done in three different ways: (1) both structure and parameters can be learned from data without human interaction, (2) consulting a domain expert to design the structure and the parameters, or (3) combine learning from data and consulting a domain expert. In order to learn successful diagnostic BN models from data one would need a very large data set, which is almost never available for diagnostic models. So building a diagnostic BN model will most of the time come down to an interaction of a knowledge engineer and a domain expert. They will consult technical manuals, test procedures, and repair databases to define the variables in that domain, determine the interactions among them, and to elicit the parameters. To give an idea of the time used to design a diagnostic BN model: the construction of the HEPAR-II model [Oniśko *et al.*, 2001] used to diagnose liver disorders took approximately 300+ hours of which roughly 50 hours were spent with domain experts. The model consisted of 73 variables and its numerical parameters were learned from a data set of patient cases.

One of the first large diagnostic systems that used Bayesian networks was the QMR-DT system [Shwe *et al.*, 1991]. This system was a probabilistic reformulation of the Quick Medical Reference (QMR), a rule-based system based on the INTERNIST-1 knowledge base developed at the University of Pittsburgh [Miller *et al.*, 1982]. The QMR-DT system contains approximately 5000 variables. The authors made several simplifying assumptions to deal with the complexity of constructing this network. The system was a BN with only two levels of variables representing two different types of variables: *diseases* and *findings*. The structure was such that the *disease* variables influence the outcome of the *findings*. While this structure was simple, its performance was close to the original QMR. But since the users of the system knew the independence assumptions the inconsistencies of QMR-DT could easily be explained.

Inspired by the QMR-DT system, we propose in this paper a methodology to make diagnostic BN models in an intuitive and fast way for users who are not necessarily experts in Bayesian networks. Using our methodology, a domain expert will be able to build a model without any interaction with a knowledge engineer. Skaanning [2000] addresses the same problem applied in an application called *BATS Author*. The BATS Author hides the causality of the models from the user and also has the assumption that only one single fault can occur. Our methodology supports multiple faults and shows the causal structure of the graph. We believe that this will support the user in understanding what she is doing while building a diagnostic BN model.

To keep the model building process simple for the user, the methodology is based on some simplifying assumptions. The first assumption is in the qualitative part of the model. The structure is such that it can only consist of three levels of variables. The top level represents *context* variables, the middle level represents the *fault* variables and the bottom level represents *evidence* which are the effects of when a *fault* occurs. Relations are only allowed in a top-down way: from *contexts* to *faults* and from *faults* to *evidences*. The second simplification is in the quantitative part. All the variables are modeled using Noisy-MAX gates. A Noisy-MAX gate is a generalization of the Noisy-OR gate for multi-valued variables. The goal of using the Noisy-MAX gate is to reduce the size of the CPT's of the variables. It implies that the number of parameters that have to be elicited by the user decreases from exponential to linear in the number of parents of that variable. Research shows that many interactions in practical models can be approximated by the Noisy-MAX gates [Zagorecki and Druzdzel, 2004]. With these simplifications we sacrifice some theoretical modeling power and precision but we believe, and we plan to test this in practice, that the resulting models will not be significantly less accurate while being much easier to build.

Our proposed methodology is implemented in an application named: GeNIeRate. This program also contains different features to speed up the model building process.

The remainder of this paper is structured as follows. Section 2 gives an introduction about Bayesian networks. Section 3 discusses our proposed three level structure of diagnostic BN models. Section 4 will explain the Noisy-MAX gate. Section 5 gives a complete description of GeNIeRate. Finally, Section 6 discusses a description of a pilot experiment testing GeNIeRate.

2 Bayesian Networks

Bayesian networks are acyclic directed graphs in which nodes represent random variables and arcs represent direct probabilistic dependencies among them. A Bayesian network encodes the joint probability distribution over a set of variables $\{X_1, \ldots, X_n\}$, where n is finite, and decomposes it into a product of conditional probability distributions over each variable given its parents in the graph. In case of nodes with no parents, prior probability is used. The joint probability distribution over $\{X_1, \ldots, X_n\}$ can be obtained by taking the product of all of these prior and conditional probability distributions:

$$\Pr(x_1, \ldots, x_n) = \prod_{i=1}^{n} \Pr(x_i | Pa(x_i)) \ . \qquad (1)$$

Figure 1 shows a highly simplified example Bayesian network modeling causes of a car engine failing to start. The variables in this model are: *Age* of the car (A), dead *Battery* (B), dirty *Connectors* (C), *Engine* does not start (E) and *Red* warning lights on your dashboard (R). For the sake of simplicity, we assumed that each of these variables is binary. For example, R has two outcomes, denoted r and $\overline{r}$, representing "Red warning lights are present" and "Red warning lights are absent," respectively.

A directed arc between B and E denotes the fact that whether or not the battery is dead will impact the likelihood of the engine failing to start. Similarly, an arc from A to B denotes that the age of the car influences the likelihood of having a dead battery.

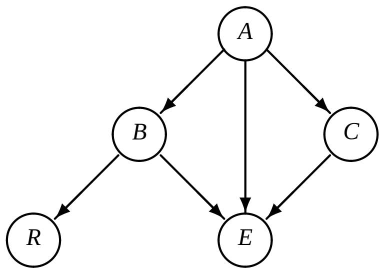


Figure 1: An example Bayesian network for engine problem

Lack of directed arcs is also a way of expressing knowledge, notably assertions of (conditional) independence. For instance, lack of a directed arc between A and R encodes the knowledge that the age of the car does not influence the chance of the car having red lights at the dashboard, only indirectly through the variable dead battery B. These causal assertions can be translated into statements of conditional independence: R is independent of A given B. In mathematical notation,

$$\Pr(r|b) = \Pr(r|b, a) \ .$$

Similarly, the absence of arc $B \rightarrow C$ means that whether or not the car has a dead battery will not influence the chance of having dirty connectors.

These independence properties imply that:

$$\Pr(a, b, c, r, e) = \Pr(a) \ \Pr(b|a) \ \Pr(c|a) \ \Pr(r|b) \ \Pr(e|a, b, c) \ ,$$

i.e., that the joint probability distribution over the graph nodes can be factored into the product of the conditional probabilities of each node given its parents in the graph. Please note that this expression is just an instance of Equation 1.

The assignment of values to observed variables is usually called *evidence*. The most important type of reasoning in a probabilistic system based on Bayesian networks is known as *belief updating* or *evidence propagation*, which amounts to computing the probability distribution over the variables of interest given the evidence. This evidence propagation makes Bayesian networks very suitable for diagnosis. For example, in the model of Figure 1, the variable of interest for diagnosis could be B and the focus of computation could be the posterior probability distribution over B given the observed values of A, R, and E, i.e., $\Pr(b|a,r,e)$. Bayesian network software can be applied to calculate this posterior probability. Today's software is capable of very fast belief updating in models consisting of hundreds or even thousands of variables. After the belief updating the software can make a decision or support the user in making a decision what actions to perform given that probability.

3 The BN3M model

We believe that there are three fundamental types of variables in diagnostic models. The first type (1) are variables representing the failures of the device or a specific part of the device. We will call these failure variables: *Faults*. The second type (2) are variables which have an observable effect if the device is in some faulty state. Sometimes you cannot observe the effect of the given *fault* clearly, then you can perform a test which will give you information about the state of the device. These observation or test variables will be called: *Evidence* variables. The last type of variables (3) are variables which indicate context properties of the device that may influence the risk of causing a *fault*. We will call these variables: *Context* variables. Example *context* variables are the age of the device or the history of failures of the device.

In the QMR-DT model, mentioned in the introduction, the authors distinguish two types of binary variables: *diseases* and *findings*. These variables are graphically structured in two levels in which the *disease* variables influence the *findings*. This structure is based on some independence assumptions. The first of these is *marginal independence of the diseases*, which amounts to no arcs among the disease variables. The second is *conditional independence of the findings*, which amounts to no arcs among *finding* variables. Figure 2 shows an example of the graphical structure of the QMR-DT model.

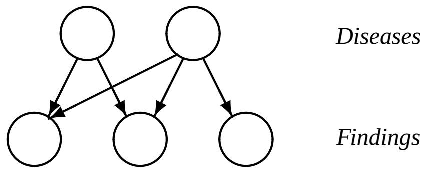


Figure 2: The two-level QMR-DT model

The structure of the QMR-DT model was simple, yet the performance of the model was close to that of the original rule-based QMR system. This motivated us to design an extension of their structure to connect our three types of variables in a diagnostic BN model. The authors of the QMR-DT system already mention the fact that assuming their two level structure is not always very accurate. They state that it would be more accurate to model some of their *findings* representing, for example, historical findings as parent variables of the *diseases*. We decided to cover this inaccuracy in our model in making the *context* variables the parents of the *fault variables*. We also decided to support multi-valued variables instead of binary variables used in QMR-DT. This means that our structure becomes a three level structure with the *context* variables on top, *fault* variables in the middle and the *evidence* variables at the bottom. We call this structure: 3-level Bayesian network using Noisy-MAX gates (BN3M). An example of this structure is showed in Figure 3. We will discuss the Noisy-MAX gate in Section 4.

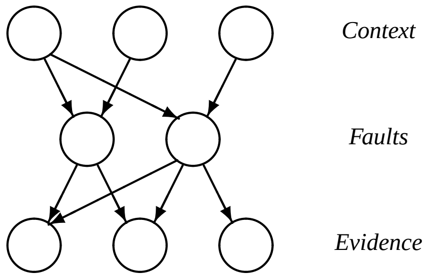


Figure 3: The BN3M model

As mentioned in the introduction, this structure sacrifices some modeling power and precision. For example, the arc between variables A and E of the example given in Figure 1 cannot be created in our methodology. Whether this theoretical imprecision has a big impact on the practical performance of a large model remains a question which we are planning to test with a systematic study. A model that is theoretically very precise may turn out to be inferior in practice because elicitation of a huge number of parameters from a human expert may decrease their quality.

4 The Leaky Noisy-MAX gate

In order to gain speed in designing the quantitative part of the model, the interaction among the variables in our structure is approximated by canonical interaction models that require fewer parameters. One type of canonical interaction, widely used in Bayesian networks, is known as the Noisy-OR gate. This gate was first introduced outside the BN domain by [Good, 1961]. Later it was applied in the context of BN's [Pearl, 1986], and it became very popular among BN model builders because it reduces the growth of a CPT of a variable within a BN from exponential to linear in the number of parents. An extension of the Noisy-OR gate for multi-valued variables is the Noisy-MAX gate [Henrion, 1989; Díez, 1993]. For the sake of simplicity, we will discuss the Noisy-OR gate in depth and give a short description of the extended Noisy-MAX gate later in this section.

The Noisy-OR gate models a non-deterministic interaction among n binary parent cause variables X and a binary effect variable Y. Every variable has two states: a distinguished state, which represents that the variable is in its normal working state. Commonly this is *absent* or *false* and a non-distinguished state: *truth* or *present*. The effect variable

Y works as a deterministic OR gate. This means that if all the parent variables are *absent*, the child variable is also *absent*. However if a parent variable X_i is *present* and all other parent variables are *absent*, it has a probability p_i of causing the effect y. These probabilities p_i address the noisy property of the gate and have to be elicited by the model builder or can be learned from data. The probabilities are fairly easy to understand since they can be represented by questions like: *What is the probability that the effect y will occur, given that only one cause X_i is present and all other causes are absent?* In other words,

$$p_i = Pr(y|\overline{x}_1, \overline{x}_2, \ldots, x_i, \ldots, \overline{x}_{n-1}, \overline{x}_n) \ . \tag{2}$$

The probability of an effect y to occur given a subset X of causes which are *present* is now formally given by:

$$p(y|X) = 1 - \prod_{i=1}^{n}(1 - p_i) \ . \tag{3}$$

This formula is sufficient to derive the complete CPT of Y conditional on its predecessors $X_1, X_2, \ldots, X_n$.

Henrion [1989] proposed a direct extension of the Noisy-OR gate which models that an effect y can also occur if all the causes are *absent* and called it: *leaky Noisy-OR* gate. This can be modeled by introducing an additional parameter p_0, which is called the *leak probability*, formally given by:

$$p_0 = Pr(y|\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) \ . \tag{4}$$

The leak probability represents the phenomenon that an effect occurs spontaneously, i.e., in absence of any of the causes that are modeled explicitly.

In the leaky Noisy-OR gate, p_i ($i \neq 0$) no longer represents the probability that X_i causes y given that all other parent variables are absent, but rather the probability that Y is present when X_i is present and every other explicit parent causes (all the X_j's such that $j \neq i$) are absent.

An alternative way of eliciting the parameters of a leaky Noisy-OR gate is given in [Díez, 1993]. Díez defined p'_i as the probability that Y will be true if X_i is present and every other parent of Y including unmodeled causes (the leak) are absent:

$$1 - p'_i = \frac{1 - p_i}{1 - p_0} \ . \tag{5}$$

His method amounts essentially to asking the expert for these parameters p'_i.

Converting the parameters p'_i to p_i is straightforward:

$$p_i = p'_i + (1 - p'_i)p_0 \ . \tag{6}$$

It follows that the probability of Y given a subset of parent variables X is given in the leaky Noisy-OR gate by the following formula:

$$p(y|X) = (1 - (1 - p_0)) \prod_{i=1}^{n} \frac{1 - p_i}{1 - p_0} \ . \tag{7}$$

The difference between the two proposals has to do with the leak variable. While Henrion's parameters p_i assume that the expert's answer includes a combined influence of the parent cause in question and the leak, Díez's parameters p'_i explicitly refer to the mechanism between the parent cause in question and the effect with the leak absent. Conversion between the two parameters is straightforward using Equation 6. If the Noisy-OR parameters have to be elicited by an domain expert, Díez's definition is more convenient, since the question to be answered is more intuitive for the expert: *What is the probability that the effect y will occur when you know that all modeled and unmodeled causes X are absent?* Henrion's definition will be more convenient if the parameters are learned from data.

The Noisy-OR gate can be generalized if it is used for multi-valued variables. These variables are allowed to have more than two states. However, these variables still contain a distinguished state. The difference is that the CPT of the effect variable Y is the deterministic MAX function instead of the deterministic OR function. Therefore it is called Noisy-MAX gate or, if the leak probability is included, leaky Noisy-MAX gate. In our proposed methodology we set the leak probability distribution of an effect variable $p_0(Y)$ by default to 0 for the non-distinguished states $y_i, \ldots, y_{n-1}$ and 1 for the distinguished state $\overline{y}$:

$$p_0(Y) = \begin{cases} 1 & \text{if } Y = \overline{y} \\ 0 & \text{otherwise} \end{cases} \ . \tag{8}$$

In this way the interaction among variables is by default Noisy-MAX, it becomes leaky Noisy-MAX if an user defines a specific leak probability distribution for the effect variable.

To give an example of the reduction of the number of parameters that have to be elicited by the user we define the number of states of a parent variable X_i as n_{X_i} and the number of states of an effect variable y as n_y. The total number of parameters N that have to be elicited by the model builder using leaky Noisy-MAX becomes: $N = \sum_{i=1}^{n}(n_{X_i} - 1)(n_y - 1) + 1$, compared to the exponential number of parameters using CPT: $N = n_y \prod_{i=1}^{n} n_{X_i}$.

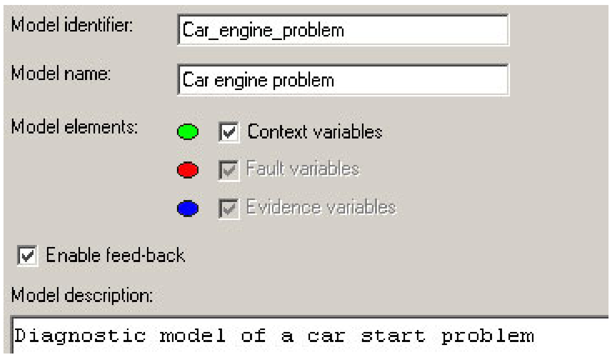
Using Noisy-OR and Noisy-MAX gates for some (not all interactions could be approximated by these gates) of the conditional distributions in the HEPAR-II model [Oniśko *et al.*, 2001] not only reduced the number of parameters that had to be elicited from 3714 to 1488 but also improved the diagnostic performance of the model at that time.

5 GeNIeRate

We embedded the ideas presented in Sections 3 and 4 in an interactive environment for generation of diagnostic BN models that we call GeNIeRate. GeNIeRate is a member of the family of software developed at the Decision Systems Laboratory (DSL) of the University of Pittsburgh. This software is developed for the purpose of probabilistic modeling with special extensions for diagnostic inference, such as rank ordering, tests, and case management. The main part of this family is SMILE (Structural Modeling Reasoning, and Learning Engine) which is a library of C++ classes implementing graphical probabilistic and decision-theoretic models. In order to use SMILE in other programming languages some wrappers are developed: jSMILE for Java, SMILE.NET for a Microsoft .NET environment and pocketSMILE for the Pocket PC. Next to the libraries, there are some applications with a GUI using these libraries. GeNIe is the main application, it is a development environment for building graphical decision models. Other applications are ImaGeNIe which is a general purpose model building interface and GeNIeRate which is discussed in this paper. GeNIeRate is developed in Java therefore it uses jSMILE as its modeling library. All the software can be downloaded at: `http://www.sis.pitt.edu/~genie`.

The main goal of GeNIeRate is to support a user in building the simplified diagnostic BN3M models in a fast and intuitive way. In order to keep the model building process simple GeNIeRate contains three different screens, representing three steps for building a BN3M model: (1) add General information, (2) add the Variables, and (3) add the Relations and the probabilities (G-V-R). We will discuss these three steps in this section and support our description with different screenshots of GeNIeRate.

In the first tab, the user can define general model information: name, domain, and description of the model. Since not all designed models will contain *context* variables, these variables can be switched on or off. Furthermore, feedback can be enabled which amounts to feedback given by GeNIeRate for each action of the user. If the user is not a BN expert, this feedback will help with natural language dialogs to get familiar with this technique and give the user some insight about the causality of the graphical structure of his model. Figure 4 shows a part of this first tabbed screen.



Model identifier: Car_engine_problem

Model name: Car engine problem

Model elements:
- 🟢 ☑ Context variables
- 🔴 ☑ Fault variables
- 🔵 ☑ Evidence variables

☑ Enable feed-back

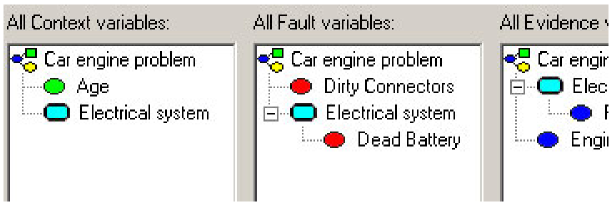
Model description:

```
Diagnostic model of a car start problem
```

Figure 4: Part of the model info panel

In the second tab, the variables of the model can be added, edited or deleted. For each type of variable (*context*, *fault* and *evidence*) there is a tree structure representing the variables of that type added to the model. Figure 5 shows a part of this screen containing the different tree structures.

GeNIeRate uses the concept of a *system* to divide large diagnostic models into manageable parts. A system is typically a module of a device that, while interconnected with other modules, can be thought of in separation from the rest of the device. Typically devices, whether natural or artificial, are composed hierarchically and have clearly identifiable modules, for example electrical system, power train, break, or steering system in an automobile. Systems are the largest entities through which the user can interact with a model, i.e., the users can view a system's variables after selecting that system and add variables to it or remove variables from it. This supports working on a specific part of the model, ignoring the rest of the model and, since humans can only focus on a small number of things at a time, limiting the amount of information presented to the user.



All Context variables:
- Car engine problem
  - Age
  - Electrical system

All Fault variables:
- Car engine problem
  - Dirty Connectors
  - Electrical system
    - Dead Battery

All Evidence
- Car engir
  - Elec
    - F
  - Engi

Figure 5: Part of the model variables panel

Furthermore, documentation can be added to the variables and the states of the variables. This documentation can be a treatment for each state of a *fault* variable or a question for each *context* or *evidence* variable. The documented questions and treatments can later be used when the model is applied in diagnostic software. Answers to these questions can be used as evidence for the belief updating (Section 2) to calculate the most probable *fault(s)*. After this calculation, the documented treatment of that fault can be presented to help the user make a decision how to fix his problem.

The last screen allows for creation of relations among the variables. When, for example, a *fault* variable is selected, *context* variables can be added as parents or *evidence* variables as children of that *fault*. After relations are created, the graph at the right side of the screen shows the relevant part of the graph (see Figure 6).

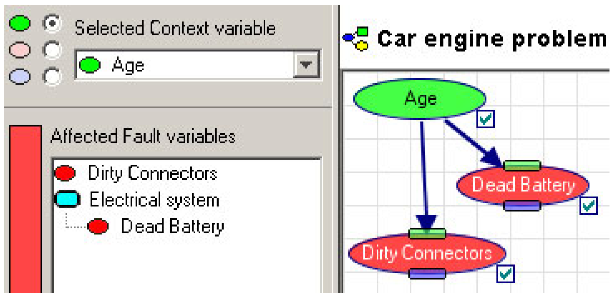


Figure 6: Part of the model relations panel

A *system* can be selected to show only the relations of the selected variable with the variables within that system. Navigating the graph can be done by clicking on the top or the bottom of a node to show the parents or children of that node respectively.

All the parameters can be defined by double-clicking on a node or an arc in the graph. If a node has parents, the leak-probability distribution for that variable can be defined, otherwise the prior probability distribution can be defined. If an arc in the graph is selected the Noisy-MAX parameters for the relation represented by that arc can be defined.

After the diagnostic BN model has been created, it can be saved and used for applying diagnosis. This can be done after loading the network into GeNIe, which contains a tool for

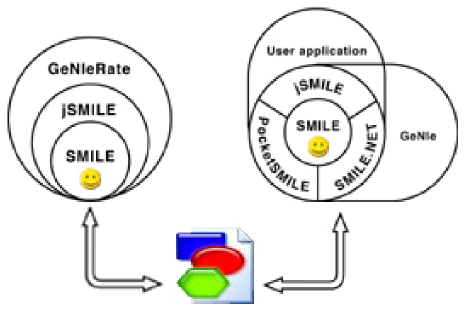


Figure 7: GeNIeRate architecture and interaction with other DSL software

diagnosis. The network could also be loaded in user applications using (j)SMILE or other BN building software (Figure 7).

6 Experiment

A qualitative evaluation of our methodology and GeNIeRate was performed by Mr. M. D. Campbell, a professional consultant with 5 years of practical experience in building diagnostic BN models. GeNIeRate was received warmly and Mr. Campbell estimated that this methodology will be of great help especially in building large diagnostic models. He mentioned that a domain expert usually spends 90% of the time in analyzing and understanding his diagnostic problem as a probabilistic model. However, his estimate was that our approach will reduce the model building time for an inexperienced Bayesian network model builder.

We are planning to conduct an empirical study for the performance of our methodology with the HEPAR-II model described in the introduction. HEPAR-II is a Bayesian network consisting of 73 variables applied for diagnosing liver disorders. The construction of this model took about 300 hours of which 50 hours were spent with domain experts. HEPAR-II was built in collaboration between DSL, the Bialystok University of Technology and the Medical Center for Postgraduate Education in Warsaw, Poland. We are planning to rebuild the HEPAR-II model with our methodology and compare the performance of this new HEPAR-II-BN3M to the original HEPAR-II. We will also look at the time it took to construct the new network. We hope to be able to present the results of this study at the workshop.

7 Concluding remarks

This paper has presented a methodology for building large diagnostic Bayesian networks. An implementation of the methodology was received well by a professional consultant in building diagnostic Bayesian networks. His estimate is that using this methodology for building large diagnostic BN models will reduce the model building time especially for an inexperienced BN model builder.

Acknowledgments

This research was supported by the Air Force Office of Scientific Research under grant F49620-03-1-0187, and by Intel Research. We would like to thank Mr. M. D. Campbell whose suggestions have improved GeNIeRate enormously.

References

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

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

[Díez, 1993] F. J. Díez. Parameter adjustement in Bayes networks. The generalized noisy OR–gate. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 99–105, Washington D.C., 1993. Morgan Kaufmann, San Mateo, CA.

[Good, 1961] I. Good. A causal calculus (I). *British Journal of Philosophy of Science*, 11:305–318, 1961.

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

[Miller *et al.*, 1982] Randolph A. Miller, Harry E. Pople, Jr., and Jack D. Myers. Internist–1, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307(8):468–476, August 1982.

[Oniśko *et al.*, 2001] Agnieszka Oniśko, Marek J. Druzdzel, and Hanna Wasyluk. Learning bayesian network parameters from small data sets: Application of noisy-or gates. *International Journal of Approximate Reasoning*, 27(2):165–182, 2001.

[Pearl, 1986] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

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

[Shwe *et al.*, 1991] M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST–1/QMR knowledge base: I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30(4):241–255, 1991.

[Skaanning, 2000] Claus Skaanning. A knowledge acquisition tool for Bayesian-network troubleshooters. In *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)*, pages 549–557, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

[Zagorecki and Druzdzel, 2004] Adam Zagorecki and Marek Druzdzel. Knowledge engineering for building bayesian networks: How common are noisy-max distributions in practice?, 2004.