Production Systems and Information Engineering
Volume 4 (2006), pp. 81–93

A MATLAB GRAPHICAL TOOL TO SUPPORT KNOWLEDGE ENGINEERING

GÁBOR VÁMOS
Budapest University of Technology and Economics, Hungary
Department of Control Engineering and Information Technology
vamos@iit.bme.hu

[Received May 2005 and Accepted May 2006]

Abstract. Bayesian networks give an efficient probabilistic model representation scheme to encode both expert knowledge and information extracted from database. Although the probabilistic model representation and the storage space problem can be solved using Bayesian networks, the model evaluation (inference) operation become complicated. The practical solution of this problem is an optimum between the accuracy of model and the complexity of the model evaluation task. This trade-off is more easily found if the knowledge engineers could receive an immediate feedback during the developing phase of the network about the computational resources required to evaluate the probabilistic model. This paper presents a Bayesian network modeling software tool focusing on its graphical user interfaces and data structures.

Keywords: Bayesian Networks, Knowledge Engineering

1. Introduction

Probabilistic models [8, 6, 7] of real world phenomena are increasingly used in commercial applications, especially in decision support systems and in different type of fault or fraud detection tasks. Such applications have two essential, usually independently considered components: the knowledge base (abstract description of real world phenomena, i.e. a model) and the model evaluation engine.

In the case of an uncertain problem domain, the model is often given as a joint probability distribution function (PDF) over a finite set of variables. For discrete variables, the joint probability function may be stored in a table. This may result simple inference engines at the price of huge required storing capacity which grows exponentially with the number of variables and their

possible values. In fact, the required storage capacity impedes the direct use of joint probability tables in real applications.

Bayesian networks offer an efficient representation structure using conditional probability tables obtained from the factorization of the joint probability function. This factorization is implied by a set of independence relations which is represented by a directed acyclic graph (DAG). This graphical structure helps to integrate two vital knowledge sources when creating the probabilistic model: expert knowledge and information extracted from databases. However, the evaluation of these models is more complex. From a practical point of view, the main problem is to find an optimum between the model accuracy and the complexity of its evaluation. This trade-off should be more easily found if the knowledge engineers could receive an immediate feedback about the computational resources required to evaluate his or her actual probabilistic model.

Our research has a twofold objective. First, we wish to revisit some existing algorithms used to evaluate Bayesian networks, and second, to develop a user friendly environment to design such networks providing information about the computational complexity of the model during its creation. This paper focuses on our second objective and presents a prototype software tool designed using MATLAB.

The remaining part of the paper is organized as follows. The next section (2) gives an overview on the mathematical background of Bayesian networks and on the related probability inference problem. Section 3 summarizes the model creation methodology, the steps of model creation are described in Section 4. Section deals with the implemented Bayesian modeling software tool. The first part of the section is devoted to the main role of the user interface. This is followed by a short overview about the main concepts of system architecture. Section 6 ends with the description of the implemented functions. Section 7 specifies the data structures used to maintain the DAG property during the editing process. Conclusions and some remarks about the future works close the paper.

2. Bayesian networks

The objective of the modeling is to find an abstract representation of the information coded in the database and that of the human knowledge accumulated as experience.

In probabilistic models all variables are aleatory variables and each record in the database is a realization of these variables. Then the model is given by the joint probability distribution function (PDF) over all variables as

$$P \quad V_1 \times V_2 \times \quad \times V_n \rightarrow [0,1] \tag{2.1}$$

Bayesian networks [7] are widely used to represent efficiently a joint PDF, so it become one of the most popular uncertainty knowledge representations and reasoning technique in AI. The structure is able to code and to integrate expert knowledge and formerly measured values of problem domain variables (stored in databases).

Definition 1 (Bayesian networks). *A Bayesian network over a set of variables* $\mathbf{U} = \{V_1, \ldots, V_n\}$ *consists of a graphical and a quantifying component:*

1. *Graphical component. Directed acyclic graph:* $\mathcal{G}$. *Each node in the graph represents a variable in* $\mathbf{U}$. *The set of parents of a variable* V *is denoted by* Π_V.
2. *Quantifying component. Each variable* V *in* $\mathbf{U}$ *(i.e. each node in* $\mathcal{G}$*) is quantified with a conditional probability distribution (CPT) denoted by* $P(V|\Pi_V)$.

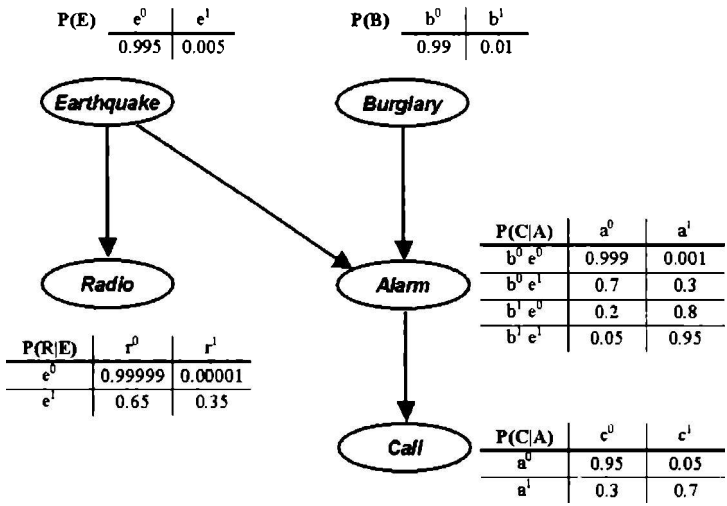


Figure 1. The classical [7] Alarm Bayesian network with CPTs

These random variables can be either discrete or continuous depending on the nature of the problem domain. We will consider discrete variables in the sequel. The Bayesian network corresponds to a joint PDF over $\mathbf{U}$:

$$P(\mathbf{U}) = \prod_{i=1}^{n} P_i(V_i|\Pi_{V_i}). \tag{2.2}$$

The query of a variable V_q in a Bayesian network based on a set of evidences $\mathbf{E} \subset \mathbf{U}$ (variables with known values or distributions) corresponds to an inference procedure. The inference in a Bayesian network consists of fixing the values of a subset of the variables and to marginalize (2.2) w.r.t. V_q:

$$P_{V_q}(V_q) = \sum_{\mathbf{U}\setminus\{V_q,\mathbf{E}\}} P(\mathbf{U}). \tag{2.3}$$

This results a probability distribution over V_q which combines the knowledge encoded in the network (i.e. in $P(\mathbf{U})$) and in the collected evidence. The direct marginalization according to (2.3) is a computationally expensive operation because it needs numerous evaluations of the joint PDF coded by the network. Therefore the marginalization takes place usually in a different secondary structure called tree of clusters using the so called Probability Propagation in Tree of Clusters (PPTC) algorithm [4]. Roughly speaking, cluster trees are undirected, acyclic graphs whose nodes are clusters containing sets of the nodes from the original network. The rules of transformations leading from the original Bayesian network to a cluster tree ensure that both entities correspond to the same joint PDF

There are several exact and approximate inference algorithms, whose computational complexity is NP-hard [1, 2]. The most popular exact inference algorithm is the so called clique-tree propagation algorithm [4]. Our research places the emphasis on this approach [3].

3. A probabilistic modeling methodology using Bayesian networks

The applied model creation methodology is meshing to the degree of observability of problem domain [5]. We pressume the full observability over many human experts and large datebases. According to our approach the model creation process consists of several overlapping steps, some of them being repeated in an iterative way. This section summarizes the main working phases of this process.

Since Bayesian networks yield an effective way to represent factorizable joint PDFs, our methodology is especially customized to this approach. The following human collaborators intervene during the model creation process [9]:

- Human Expert: he or she holds possession of practical and/or theoretical knowledge about the problem domain.
- Statistician: skilled in the classical statistical methods of data analysis.
- Knowledge Engineer: proficient in the creation of knowledge based probabilistic model.

The model creation process consists of the following steps.

- Surveying knowledge sources: discovery of the appropriate knowledge sources. The probabilistic model is generally a theoretical (mathematical) abstraction, which depicts efficiently the behavior of the circumscribed real world domain. In our approach, the model integrates the knowledge hiding in measured data patterns (stored in databases) and the knowledge of Human Expert (collected using questionnaires and interviews). This phase is undertaken by the Knowledge Engineer in cooperation with the Human Expert.
- Raw data preprocessing and determining the relevant probabilistic variables. This phase processes the raw data gathered about the problem domain. In the possession of reports made during the preceding phase with the Human Expert, the Knowledge Engineer determines a potential collection of probabilistic variables (including future evidence and query variables) in line with the Statistician, who analyzes the quality and relevance of raw databases w.r.t. the set of probabilistic variables.
- Stipulating the structure of probabilistic network. The Statistician and the Knowledge Engineer divide the probabilistic network into partitions. Some of them will be trained using data from database, the rest will be appraised by the Human Experts.
- Putting the pieces together. The Knowledge Engineer assembles the trained and appraised parts of model using again the Human Expert's knowledge. This results the first raw but complete probabilistic model.
- Finalizing the qualitative and quantitative composition. The Knowledge Engineer verifies the Bayesian network using test data.

Recall that at each phase one may need to step back and reiterate previous phases if the actual results are not satisfying.

4. Model creation steps

The probabilistic model representation and the storage space problem of a joint PDF can be solved using Bayesian networks, but the model evaluation (inference) operation become complicated.

Recall that the first objective aims to obtain quantitative information to create the graphical component and the second one helps to determine qualitatively the conditional PDFs associated to the nodes of the Bayesian network. This modeling procedure results an iteration of steps as illustrated in Figure 2.

The transformation between the cluster tree and the Bayesian network (already mentioned in Section 2) is not a one-to-one relation, since several cluster tree may correspond to the same Bayesian network. A cluster tree is said to be

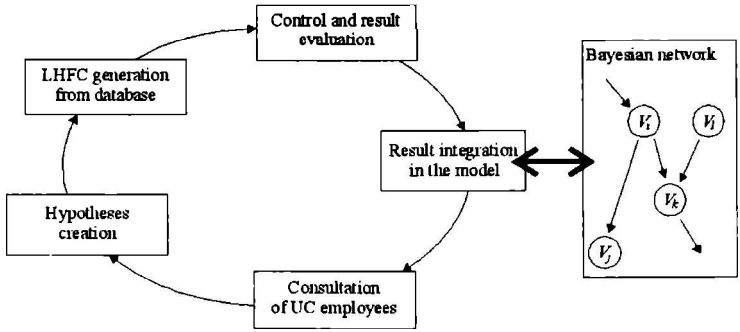


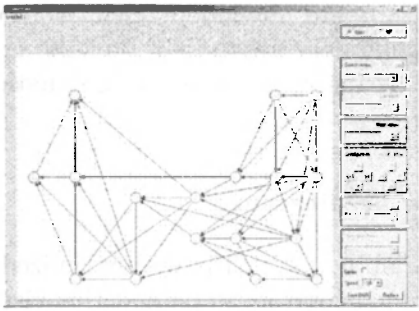
Figure 2. Iterative modeling with Bayesian network

optimal if it is of minimal width. Roughly speaking, the width of the cluster tree is directly related to the computational cost of a query evaluation. The search for the optimal cluster tree is a complex task impeding the possibility to give the Knowledge Expert an exact complexity measure of the designed network at each step of the above methodology.

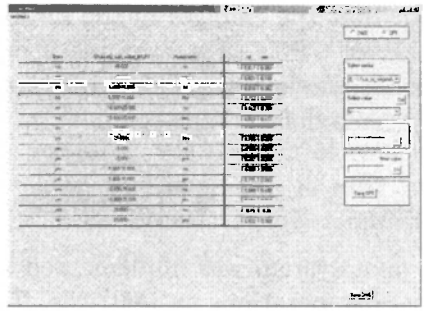
5. The modeling software tool

It is clear from the methodology described in the previous section that the Knowledge Engineer has the occasion to trade-off between the accuracy of the probabilistic model and the complexity of the inference task during the creation of each direct connection between a pair of nodes. There are a lot of commercial Bayesian software tools integrated with ergonomic graphical interfaces to support this graph manipulation process. From the well distributed software packages we emphasize Netica, GeNIe, Hugin. [10, 11, 12]. Some of them have high level abstraction interfaces, which can be accessed for development using up-to-date programming languages (C++, Java etc.).

However, these environments are too closed for the optimization at low level of PPTC, so we have constructed a new framework using MATLAB, including a simple graphical but a more complex developer interface. In this environment several optimization approaches (testing different triangulation and complexity information feedback heuristics) become realizable and examinable. In this section our framework is presented that supports the Knowledge Engineer to find the compromise between accuracy and computational complexity.



(a) Surface to edit the DAG component



(b) Surface to edit the CPT component

Figure 3. Graphical user interfaces

5.1. GUI functionality and structure

The GUI has two distinct windows for supporting the modifications of the DAG (directed acyclic graph) structure and the numerical data of CPT (conditional probability distribution). Figure 3 shows screenshots of both windows.

There are two frames in each window, the right ones (larger) visualize the DAG or the CPT, the left ones (smaller) displays the modification commands.

In the DAG manipulation view (Figure 3) some operations are just modifying the vista of the graph (scroll the graph or a node), the others are changing the structure of the DAG (add/remove edges). The main commands for the DAG view are the following:

- Scrolling figure: the displayed graph is movable in the window.
- Adding or removing a node.
- Manipulating a node. After the selection of a node (the current node is marked using different color) the most important operations become executable:
  - modification of the position of the current node in the picture,
  - changing the name of the current node,
  - assigning new parent to the current node (add a new edge), removing a parent from the current node (delete an edge).

It is important to pay more attention to operation where a new edge is added to the graph. We present this problem in detail in Section 6.

The role of the left frame in CPT manipulation view (Figure 3) is the same as in DAG view, but the bigger right pane becomes two sided, because the tasks of displaying and manipulating probability values are executed in the same pane.

Recall that the changes in the DAG structure are not independent from the context of CPTs. For example if one edge is deleted, the CPT of the child node becomes simpler. This software tool is capable to handle such situations using some default strategies.

6. Document/View architecture

The pilot system was implemented in Matlab 6.1, which is well optimized to carry out operations on matrices. The fundamental ideas behind this visualization interface follows the well known and the widely used Document/View architecture. There are distinctly constructed matrices for the calculation-storing process and the visualization proposition. These differences are demonstrated with the two variants of data representation form of Bayesian Networks.

Tables 1 and 2 enumerate the main components of data structure in which Bayesian networks are stored and used during the evaluation (inference) process.

Table 1. BNet data structure for storing Bayesian networks.

Name	Type	Description of use
adjMatrix	sparse	adjacency matrice
nodeSize	int. vector	each nodes values
nodes	structure	representing the nodes
edgeConstraints	sparse	training strategy
timeConstraints	struct	training strategy

Table 2. Node of BNet data structure for storing Bayesian networks

Name	Type	Description of use
varName	string	current node name
varStateNames	string vector	name of the values
selfVar	double	identification the nodes
selfVarSize	integer	values of current node
CPD	matrice	conditional probability table

The three types of data are the following:

- Basic data: These components correspond to the mathematical definition of Bayesian networks.

- Sub data: It makes executable and balance able such procedures as training the network using database
- Redundant data: Some pieces of information in the Basic data are replicated in other structures in order to make efficient some operations during the inference algorithms. For example, the number of values of nodes (i.e. variables) are stored implicitly and explicitly in each node structure and this data also appears in a collection of BNet nodeSize (which is in turn useful in the triangulation process).

Tables 3 and 4 contain the graph visualization structures.

Table 3. "drawingnode" matrix for the visualization

Name	Type	Description of use
index	integer	numerical identification
name	strings	verbal identification
posx	double	horizontal location
posy	double	vertical location
markerhandler	double	accessing the graphical object
texthandler	double	accessing the graphical object

Table 4. "drawingedge" matrix for the visualization

Name	Type	Description of use
parIndex	integer	start point identification
chIndex	integer	end point identification
parposx	double	horizontal location of parent node
parposy	double	vertical location of parent node
chposx	double	horizontal location of child node
chposy	double	vertical location of child node
markerhandler	double	to access the graphical object
arrowhandler	double	to access the graphical object

The main data types in these tables are the following:

- Basic data: the most important data from the Bnet basic data, like the node and adjacency information.
- Localization data: horizontal and vertical positions of graphical components.
- Data to the graphical objects: these handlers ensure the modifications of graphical components accessing to a complex data structure.

The basic data of matrix drawingedge and drawingnode render a permeable way between the document and the view structure. Besides to the identical content, the adjacency matrix is mapped to the column of parIndex and chIndex.

7. Incremental graph expansion

Referring to Subsection 5.1, the problem of graph structure modification will be demonstrated focusing on the graph extension. There are two ways to modify the structure of a connected graph: adding or removing the directed connection between nodes. The edge removing operation is simpler: The Knowledge Engineer selects the current node, the GUI loads the name of the parents of the current node into the Remove parent menu. The user selects one parent node (the corresponding edge become marked) from this menu, and then using the Ok button the operation is executed. After then the corresponding field of adjacency matrix become zero and the CPT of current node will be changed.

In contrast of the edge removing operation, the edge adding function more complex, because during the operation one needs to guarantee that the extended graph is still acyclic. The adding edge procedure is not too complex for the first sight, the operation is decomposed to the following steps. First a start node has to be assigned. Then the GUI loads to the Add child menu the names of potential children to the current node. The user selects one node from this list. After the approvement of actual operation, the adjacency matrix and the CPT of child node will be modified.

The key momentum is the selection of potential children to the current node. In this step the GUI selects the nodes which may hurt the DAG property if considered as children. To support this operation, a special structure is maintained to represent the reachability of each node from the other nodes. This information is stored in the so called `reachable_from` matrix. This matrix has two columns, the first contains the identifier of a node, the second contains the list of nodes, from which the identified node is reachable along directed edges. We define the size of this matrix as the total number of all elements in the list of the second column.

The algorithm operates according to an elimination scheme. Every node without children node is processed in the `while` sequence, which is executed n times. The currently processed node is selected first. Then the reachability list of the selected node is actualized with the parent nodes using the union operation. Going further, one has to check the lists of the other nodes. If the

selected node is contained in some list of other node, this list must be actualized with the list of selected node (taking the union of the two list). Then the current node is eliminated.

Figure 4 shows a chain graph with 6 nodes. The `reachable_from` matrices for this graph reads

1.
2. 1
3. 1, 2
4. 1, 2, 3
5. 1, 2, 3, 4
6. 1, 2, 3, 4, 5

This example represents the worst case size of the `reachable_from` matrix. For n nodes, it is not greater then: $\frac{n \cdot (n-1)}{2}$ The proof of this statement uses the following three lemmas. Recall that the $\mathcal{G}$ graph is ordered, if there is a bijection α such that: $\alpha \quad \mathbf{V} \rightarrow (1, 2, \ldots \ n)$

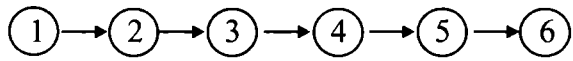


Figure 4. Chain DAG with 6 nodes

Lemma 1. *There is an ordering of nodes for every $\mathcal{G}(\mathbf{V}, \mathbf{E})$ DAG, such that every edge directs from a lower numbered node to a higher numbered node ($i < j$ for every pair of $(i, j) \in \mathbf{E}$).*

Proof. The proof consists of giving the algorithm resulting the ordering.

Input: $\mathcal{G}(\mathbf{V}, \mathbf{E})$ DAG.

Output: node ordering.

Temporal variables:

- $\mathcal{G}'$: graph is the actual states of elimination sequence
- $\mathbf{R}$: contains the set of not eliminated nodes whose have not descendant node, but not parent node in $\mathcal{G}'$
- V: the actually processed (eliminated node)
- T: temporal node
- $\mathbf{P}$: set of nodes, which appears as the first element of $(V_i, V_j) \in E$ edge pairs
- $\mathbf{C}$: set of nodes, which appears as the second element of $(V_i, V_j) \in E$ edge pairs

Initialize: creation of $\mathbf{C}$ and $\mathbf{P}$, $\mathbf{R} = \mathbf{C} \backslash \mathbf{P}$, $\mathcal{G}' = \mathcal{G}$, $n = |\mathbf{V}|$, $i = 1$

```
while i ≠ n
    V ∈ R
    α(V) = i //numbering the
             //selected node
    i = i + 1
    P = P\V //eliminating the
             //selected node
    C = C\V //eliminating the
             //selected node
    R = C\P //G' = G'(V\V ,E\(V,.))
endwhile
```

Lemma 2. *A given graph $\mathcal{G}$ is acyclic if there is an ordering α such that for every edge (V_i, V_j) we have $\alpha(V_i) < \alpha(V_j)$.*

Proof. The lemma is shown by indirection. Consider a graph $\mathcal{G}$, which have an ordering and a hypothesized cycle. Let $g \in V$ be the node with the lowest ordering of the assumed cycle. But, by the construction of the ordering, it is impossible to direct an edge into g from any other node of cycle, hence g cannot be a node of the hypothetical cycle. Therefore the Lemma follows.

Lemma 3. *Let $\mathcal{G}(\mathbf{V}, \mathbf{E})$ be a DAG. The $\mathcal{G}$ contains a maximal number of edges, if $\mathbf{E}$ contains every branches (V_i, V_j) such that $\alpha(V_i) < \alpha(V_j)$, i 1 . n, $j = 1 \ldots n$ for a given ordering α of $\mathbf{V}$*

Proof. From Lemmas 1 and 2, it follows that such a graph exists. It follows also from the construction that there is an edge between any two nodes. Hence any new edge one can put in the graph is such that the ordering of its starting node is grater that the ordering of its ending node. But such an edge creates a cycle with the already existing edge between the same nodes, hence the graph is maximal.

Proposition 1. *The maximal size of* `reachable_from` *matrices is $n(n-1)/2$.*

Proof. By construction, the size of the maximal DAG as defined in Lemma 3 is $n(n-1)/2$. Since the set of edges of all DAGs is a subset of the edges of the maximal DAG (choosing the right ordering) the proposition follows.

This proposition gives in fact the worst case for the number of operations needed to check weather the introduction of a new node hurts the DAG property.

8. Conclusion

In this paper Bayesian network based modeling techniques, and a network editor tool and its GUI have been presented. The Matlab environment was suitable for implementing incremental model generation methods. The open environment makes it possible to extend and improve the application; this way several optimization approaches (testing different triangulation and complexity information feedback heuristics) become realizable and examinable.

REFERENCES

[1] G.F. Cooper, *The computational complexity of probabilistic inference using bayesian belief networks*, Artificial Intelligence (1990), no. 42, 393–405.

[2] P. Dagum and M. Luby, *Approximating probabilistic inference in bayesian belief networks is np-hard*, Artificial Intelligence **1** (1993), no. 60, 141–153.

[3] C. Huang and A. Darwiche, *Inference in belief networks: a procedural guide*, Intl. J. Approximate Reasoning **3** (1996), no. 15, 225–263.

[4] S.L. Lauritzen and D.J. Spiegelhalter, *Local computations with probabilities on graphical structures and their applications to expert systems*, Proc. of the Royal Statistical Society (1988), no. 50, 154–227.

[5] K. Murphy, *A Brief Introduction to Graphical Models and Bayesian Networks*, Department of Computer Science at U. C. Berkeley, 2001

[6] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, San Mateo, Calif., 1988.

[7] P. Norvig and S.J. Russel, *Mesterses intelligencia modern megkelben (artificial intelligence. a modern approach.)*, Panem-Prentice Hall, Budapest, 2000.

[8] D.E. Heckerman and M. Henrion E.J. Horowitz H.P. Lehmann G.F. Cooper M.A. Shwe, B. Middletown, *Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base*, Meth. Inform. Med. (1991), no. 30, 241–255.

[9] G. Vámos and Á. Nagy and B. Kiss, *Bayesian network based modelling for flaw detection in metallic fusion welds using x-ray images*, Proc. of 4th Workshop on European Scientific and Industrial Collaboration Promoting Advanced Technologies in Manufacturing, Wesic, Miskolc (2003), no. 4, 181–186.

[10] Netica Application, http://www.norsys.com/netica.html

[11] GeNIe Development Environment, http://genie.sis.pitt.edu/downloads.html

[12] Hugin Expert Systems, http://www.hugin.com