Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 515–521
ISBN 978-83-60810-22-4
ISSN 1896-7094

Passive Construction of Diagnostic Decision Models: An Empirical Evaluation

Parot Ratnapinda
Decision Systems Laboratory
School of Information Science
University of Pittsburgh
Pittsburgh, PA 15260
Email: par34@pitt.edu

Marek J. Druzdzel
Decision Systems Laboratory, School of Information Sciences
and Intelligent Systems Program,
University of Pittsburgh, Pittsburgh, PA 15260, USA
Faculty of Computer Science, Białystok Technical University,
Wiejska 45A, 15-351 Białystok, Poland

***Abstract*—Bayesian networks have proven their value in solving complex diagnostic problems. The main bottleneck in applying Bayesian networks to diagnosis is model construction. In our earlier work [1], we proposed passive construction of diagnostic models based on observation of diagnosticians solving diagnostic cases. This idea has never been tested in practice. In this paper, we describe an experiment that tests an interactive prototype system called MARILYN on implementation of a system based on passive construction of diagnostic model, by inputting four hundred help desk cases collected at the University of Pittsburgh campus computing lab. We show that while the system's diagnostic accuracy continues to increase with the number of cases, it reaches very reasonable levels after merely tens of cases.**

I. INTRODUCTION

BAYESIAN networks (BN) are convenient tools for building models in various domains, including diagnosis. Even though the existing BN models have proven useful in diagnosing complex problems [2], they have not yet been widely adopted in practice. One of the main reasons is that constructing BN models is a complex and time consuming task. Building a BN for a domain of application involves three tasks: (1) identifying important variables, (2) identifying relationships among these variables and (3) obtaining probabilities [3]. If diagnostic models are complex such as the diagnosis of liver disorders *HEPAR* [4], it is not uncommon for this task to take hundreds of hours per model.

We built a prototype of a system that we called MARILYN [5] that builds diagnostic models by observing user diagnostic work-flow data. This idea of such a system originates from our collaborators at Intel Research, John Mark Agosta and Tom Gardos, who are working on an industrial version of the system [1]. MARILYN was tested informally but never before in a real diagnostic environment. In this paper, we present an evaluation of MARILYN diagnostic model construction by means of four hundred help desk cases at a University of Pittsburgh campus computing lab.

The remainder of this paper is structured as follows. Section II is an introduction to Bayesian networks. Section III describes MARILYN. Finally, Section IV reports the results of an empirical evaluation of MARILYN.

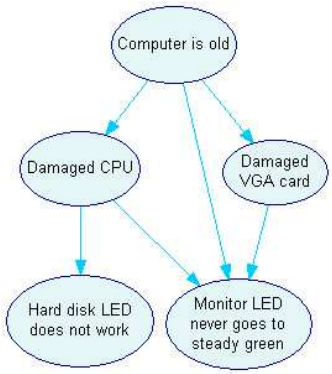

Fig. 1. An example Bayesian network modeling computer hardware problems

II. BAYESIAN NETWORKS

Bayesian Networks [6] are probabilistic models which consist of a qualitative and a quantitative part. The qualitative part are acyclic directed graphs in which nodes represent variables and arcs represent probabilistic relations among them. The quantitative part represents the joint probability distributions over its variables. Every variable has a conditional probability table (CPT) representing the probabilities of each state given the states of its parent variables. If a variable does not have any parents in the graph, the CPT represents the prior probability distribution over the variable. The joint probability distribution over a set of variables $\{x_1, \ldots, x_n\}$ 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 simple Bayesian network modeling two computer hardware problems. The variables in this model are: *Computer is old*, *Damaged CPU*, *Damaged VGA card*, *Hard disk LED does not work* and *Monitor LED never goes to steady green*. For simplicity, we took each of these variables to be binary. For example, the *Monitor LED never goes to steady green* node has two outcomes *true* and *false*.

TABLE I
AN EXAMPLE OF CONDITIONAL PROBABILITY TABLE OF THE *Monitor LED never goes to steady green* NODE

Damaged CPU	True				False			
Damaged VGA card	True		False		True		False	
Computer is old	True	False	True	False	True	False	True	False
True	0.7696	0.712	0.424	0.28	0.712	0.64	0.28	0.1
False	0.2304	0.288	0.576	0.72	0.288	0.36	0.72	0.9

A directed arc between *Damaged CPU* and *Hard disk LED does not work* indicates that *Damaged CPU* will affect the likelihood of *Hard disk LED does not work*. Similarly, an arc from *Computer is old* to *Damaged VGA card* indicates that computer age influences the likelihood of a damaged VGA card.

The most important type of reasoning in Bayesian networks is known as belief updating and amounts to computing the probability distribution over variables of interest given the evidence. This makes Bayesian networks suitable for diagnosis. For example, in the model of Figure 1, the variable of interest could be the *Damaged CPU* node and the Bayesian networks could compute the posterior probability distribution over this node given the observed values of *Computer is old*, *Hard disk LED does not work*, and *Monitor LED never goes to steady green*. Once the network updates the probability values, these can be used to make a diagnostic decision.

Bayesian networks have one fundamental problem: The size of their CPTs grows exponentially with the number of parents (and so, it is 2^n for n binary parents). Table I shows the CPT for the node *Monitor LED never goes to steady green*. The node has three parents and the size of the CPT is $2^3 = 8$.

III. MARILYN

Passive model building is based on the idea that past cases solved by diagnosticians can provide considerable information about the domain. Every case, passively observed by the system can add information to the model and, in the long run, construct a usable model.

MARILYN is a web-based application that implements this idea. It is written in C# and ASP.NET, using the Microsoft SQL database to store data and utilizes a Bayesian reasoning engine based on SMILE running under the Microsoft Windows Vista Server. MARILYN is available at the Decision Systems Laboratory's web site at the following location: `http://barcelona.exp.sis.pitt.edu/`.

A. Model Structure

MARILYN is based on the BN3M model [7], which distinguishes three fundamental types of variables in diagnostic models.

- *Fault* variables representing the causes of the problem that we try to diagnose, which can be, for example, device malfunctions.
- *Observation* variables, which will be observable if the device is in a faulty state (this includes results of diagnostic tests).

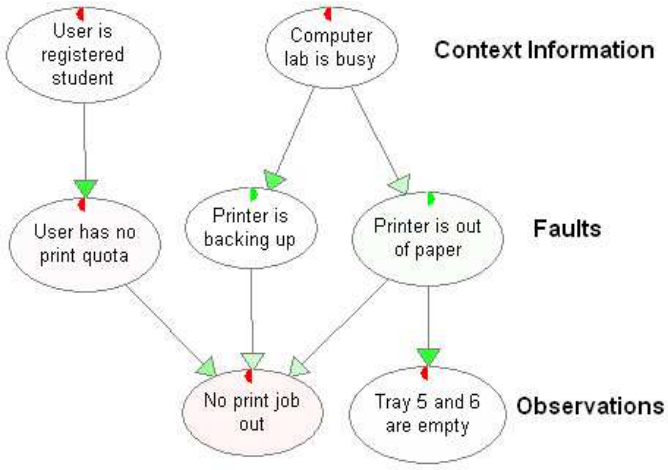

Fig. 2. An example of three-level Bayesian Network diagnostic model structure

- *Context information* variables are the background, history, or other information known by the technician performing the diagnosis that may influence the risk of a *fault* and, therefore, are relevant to the diagnosis.

The model structure consists of three levels, with the *context information* variables on the top, the *fault* variables in the middle, and the *observation* variables at the bottom. We call this structure: 3-level Bayesian network using DEMORGAN gate (BN3D). Figure 2 shows an example of this structure. We have two context variables: *User is registered student*, which influences *User has no print quota*, and *Computer lab is busy*, which influences the faults *Printer is backing up* and *Printer is out of paper*. If the user told us that he or she has not received his/her job yet, we can set the observation of the node *No print job out*. We can observe *Tray 5 and 6 are empty* by checking the printer paper tray.

B. DEMORGAN Gate

Independence of Causal Influences (ICI) models [8] provide a solution to the exponential growth of the CPT parameter problem, mentioned in Section II, by assuming that parent variables cause the effect independently of each other. The advantage of this assumption is that the number of parameters becomes linear, rather than exponential, in the number of parents.

MARILYN is based on the ICI model called DEMORGAN gate [9]. DEMORGAN gate combines Noisy-OR and Noisy-AND models and is capable of modeling opposing influences.

The DEMORGAN gate models four fundamental types of cause-effect interactions between an individual parent X and a child node Y.

- Cause: X is a causal factor for Y.
- Barrier: X is a negated causal factor for Y.
- Requirement: X is required for Y to be present.
- Inhibitor: X inhibits Y, i.e., its presence prevents Y.

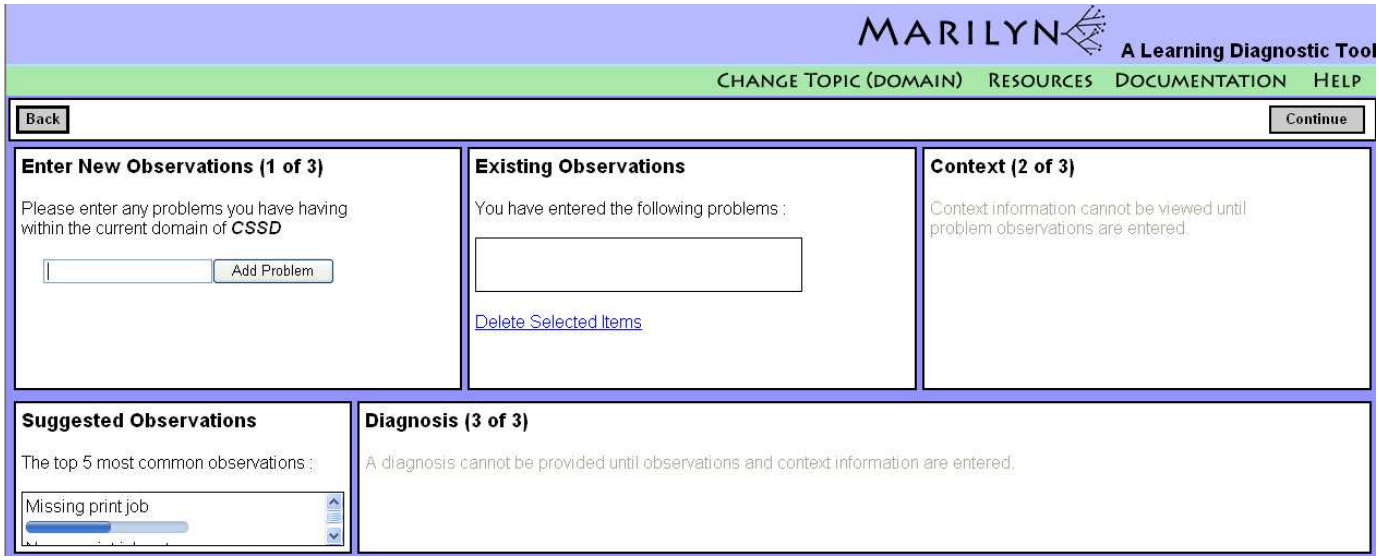

Fig. 3. MARILYN web interface for adding observation

a) Causes and barriers: are modeled by the Noisy-OR gate [10], which is a noisy version of the following Boolean formula

$$Y = X_1 \vee X_2 \vee \ldots \vee X_m \ , \tag{2}$$

where Xs stand for causes or barriers.

b) Requirements and inhibitors: The presence of an inhibitor U_i is sufficient to cancel the child effect. We can express the effect that a set of inhibiting influences have on Y by the following Boolean function

$$\overline{Y} \equiv U_1 \vee U_2 \vee \ldots \vee U_n \ , \tag{3}$$

where Us stand for requirements or inhibitors.

MARILYN is currently based only on promoting influences. Although, we plan to add other models of interaction in the future.

C. Entering Data

MARILYN follows a diagnostician work-flow. There are three phases of a diagnostic session: observation, context information, and diagnosis. These correspond to different screens (1) *add observation*, (2) *add context information*, and (3) *add a fault (problem)*. Figure 3 shows the *add Observation* screen. The *Suggested* area, located in the left bottom corner, gives a list of suggested information related to the current screen ranked from the most to the least probable. The user will use the *Back* and the *Continue* buttons to move through these three steps. MARILYN allows for working with other domains by using the *Change the domain* icon bar.

Suppose, a lab user has told a lab consultant that she has not received her print job. This user is a registered student, who tried to print a PDF document from the University *CourseWeb* web site.

In the first step, the lab consultant, who will work with a new observation window, in which she can input some related observations to the lab user problems which is *No print job out* in the text box in the *Enter New Observations (1 of 3)* panel shown in Figure 4. The *Existing Observations* panel shows the input so far and allows the lab consultant also to delete observations. Once she has finished entering observations, the lab consultant can click the *continue* button on the top right corner shown in Figure 3 to proceed onto the context information screen.

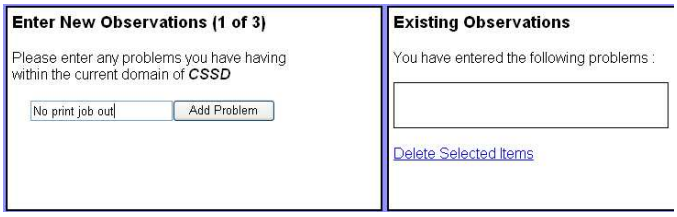

Fig. 4. Add observation panels

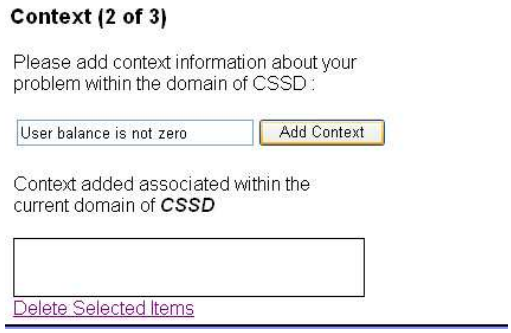

Fig. 5. Add context information panels

In the second step, the lab consultant may input history or information of the problems. She may check the lab user printing balance to see if the lab user has enough balance to print. With printing problems, the lab consultant will browse the print queue server called *Pharos* to check the history of

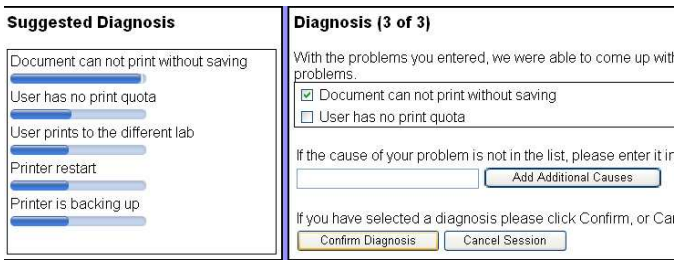

Fig. 6. Add diagnosis or fault panels

the lab user print jobs. After gathering all related information, she will input three variables: *User balance is not zero*, *No print job in Pharos system* and *Print PDF from CouseWeb* in any order in the text box in the *Context (2 of 3)* panel shown in Figure 5. Once entering context information is done, the lab consultant can click *continue* in the top right corner to continue onto selecting possible causes screen.

In the third step, the user enters the final diagnosis for the case at hand. She can select the diagnoses which were suggested by the MARILYN model or enter a new diagnostic. MARILYN diagnostic window suggests a list of possible diagnoses ranked by their posterior probabilities, as implied by the underlying model (Figure 6). In this example, the lab consultant worked with the network that is based on twenty five computing lab help desk cases, and the system was able to give a correct suggestion to the lab consultant. The first suggestion was *Document can not print without saving*. This refers to a specific requirement of the computing lab that the lab users can not print some documents directly from the web site without saving them first to local space on the disk. Next, the lab consultant selects the causes by clicking on check box in the *Diagnosis (3 of 3) panel* and confirms the session by clicking the *Confirm Diagnosis* button in order to save the session in the database. The lab consultant can cancel the session by clicking on the *Cancel Session* button if she does not want to record the diagnosis session. However, if the causes do not appear on the *causes* list, the lab consultant can type in a text field and then click the *Add Additional Causes* button, the new causes will be added to the *causes* list. After submitting, the lab consultant can export a built model in .xdsl format file which can be downloaded from a link on the last page. The .xdsl file needs to be opened in a program called QGENIE developed by the Decision System Laboratory, University of Pittsburgh and available at `http://genie.sis.pitt.edu/`.

D. Storing and Loading Bayesian Networks

The database consists of four tables: *arcs*, *diagnosis*, *domains* and *nodes*, shown in Figure 7. The *arcs* table with seven fields (*id*, *diagnosisid*, *parentid*, *childid*, *type*, *weight*, and *domainid*) stores the information about casual interactions among variables. The *diagnosis* table with three fields (*id*, *date*, and *domainid*) stores the number of diagnosis session that have been entered in the system. The *domains* table with two fields (*id*, and *name*) stores the diagnostic domains.

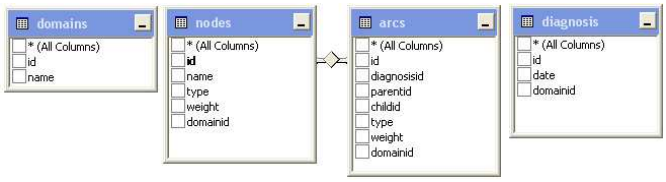

Fig. 7. Database storing diagnostic information in MARILYN

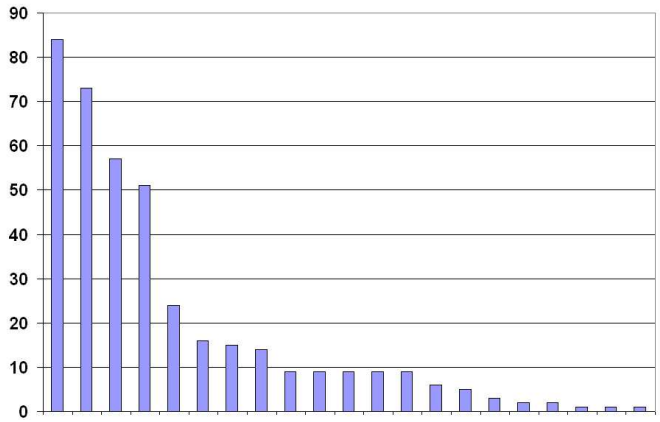

Fig. 8. Problem distribution of four hundred help desk cases

Finally, the *nodes* table with five fields (*id*, *name*, *type*, *weight*, and *domainid*) stores every variable that have been entered in any step.

In our example, we worked with the default domain called *CSSD* which has an id value of 1. The lab consultant entered one observation, three context information, and one fault. These were stored in the *nodes* table. Each of them will have an unique id and name. There are three types of nodes indicated by three values: 1 is an *observation* node, 2 is a *context information* node, and 3 is a *fault* node. Weight values for each node depend on the number of diagnostic cases that involve that particular node. Every time we enter MARILYN, the diagnosis id will be added to the *diagnosis* table. If we enter nothing or cancel the session, the diagnosis id will still be added. However, this will have no effect on the weight calculation, because without confirming diagnosis, no information will be saved in the database. Once we confirm the diagnosis, MARILYN records and writes all information to the database.

When a user starts MARILYN, the system constructs a Bayesian network from the existing database. First, the algorithm finds all nodes that contain at least one parent. Second, for each child, the algorithm finds all the arcs to be created according to the number of parents that refer to the child node. Finally, the weight and probabilities are calculated based on the path, the value of that path, and the total number of diagnostic sessions in which the path occurred.

IV. EMPIRICAL EVALUATION

A. Experiment Data

The first author has been working as a campus computing lab consultant between fifteen and twenty hours per week

TABLE II
AN EXAMPLE OF CAMPUS COMPUTING LAB LOG

Lab	Observation	Observation2	Context Information	Context Information2	Problem
CL	Printer LED is blinking red	Tray 2 is empty			Printer is out of banner
CL	Missing print job		Computer lab is busy		Printer is backing up
CL	Printer LED is blinking red	Tray 5 and 6 are empty			Printer is out of paper
CL	No any print job out		User is not register student	User balance is zero	User has no print quota
PH	Printer LED is blinking red	Paper jam message			Printer jam
SH	No any print job out		Print job appears in Pharos system	Print job appears in other lab	User prints to the different lab
SH	Printer LED is blinking red	Replace toner message			Printer is out of toner
CL	Missing print job		No print job in Pharos system	Print PDF from couseweb	Document can not print without saving
CL	Print job has missing pages				Printer restart

in the fall of 2008 and the spring of 2009. This has offered an excellent opportunity to collect real help desk data. Each user query resulted in a diagnostic session that we recorded thoroughly. University of Pittsburgh has seven computing labs that operate twenty four hours a day, seven days a week. Campus computing lab users consist of staff, faculty, alumni, visiting scholars, and students. The campus computing lab help desk problems include, but are not limited to, printing job problems, printer troubleshooting, and other diagnostic problems. In each case, the data is collected in three steps. First, the lab consultant collects the observation of the problem by asking or receiving data from a lab user. Second, the lab consultant records context information. This includes the lab user status findings from the print job *Pharos* server log, and additional questions asked the lab user. Third, the lab consultant investigates the fault according to the observation and context. Once all data had been received, then we recorded in the computing lab log notebook and later organized in the Microsoft Office Excel work book shown in Table II. The data consist of three types of variables: observation data, context information, and final diagnosis. In this campus computing lab help desk domain, the collected data indicate that there must be at least one observation and only one problem; however the context information may be not available for all cases. Therefore, data range from one to two observations, context information range from zero to three and only one problem occurs. Among the four hundred cases, there are a total of 17 different observations, 12 different context information, and 21 different problems. We show the distribution of frequency for the four-hundred cases in Figure 8. This figure shows that the distribution of problems is skewed with the top five problems covering more than 70% of the cases. We suspect that skewness of the problem set is quite typical of most diagnostic domains. Essentially, some problems are typical and occur ofter with others occurring only sporadically.

B. Experimental Design

We entered four-hundred cases, following the order of the computing lab log into MARILYN and recorded the diagnostic windows results. We sorted the results from the most to the lease likely. The default size of the diagnostic windows is five suggestions. However, we only choose the first three suggestions by MARILYN to be used for comparing with the computing lab log. Figure 9 shows the number of different

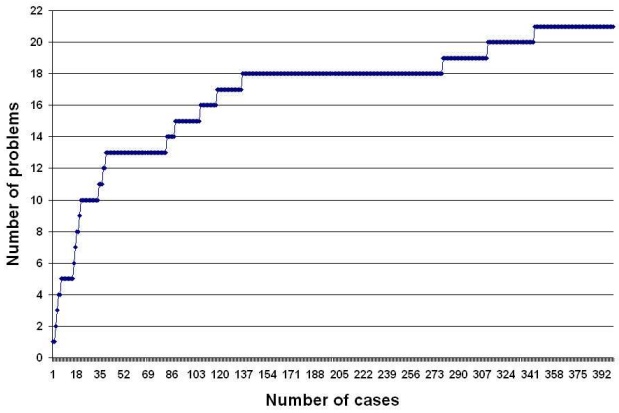

Fig. 9. The accumulate of the computing lab help desk problems entering to the MARILYN.

problems covered as a function of the number of cases growth. In a total of 21 problems in our domain, 13, 15, and 18 problems are covered in the first 50, 100, and 150 cases respectively. This number assumes that the MARILYN model should be able to give suggestions for most of the cases after seeing the first one hundred and fifty cases.

C. Results

Figure 10 shows the Bayesian networks created by MARILYN after four-hundred cases. Twelve *context information* nodes are located at the top layer. Twenty one *Fault* nodes are located in the middle layer. Sixteen *observation* nodes are located at the bottom layer. The number of parents per child ranges from one to eight. If the child node has only one parent node, it indicates that the parent has a strong influence on the child.

If a child contains many parents, MARILYN distributes a different weight for each arc. The color of the arcs represents the strength of influence of the parent on the child node. The darker the color is, the stronger the influence of the parent. For example, the observation node *Only banner sheet printed* has three fault parent nodes. The model assigns weight for each arc according to the number of cases that have been input to MARILYN. Figure 11 shows that from left to right, the arc weights are 80%, 44% and 5% for the nodes *Paper size not available A4*, *Paper size not available custom paper*, and *Printer restart*, respectively.

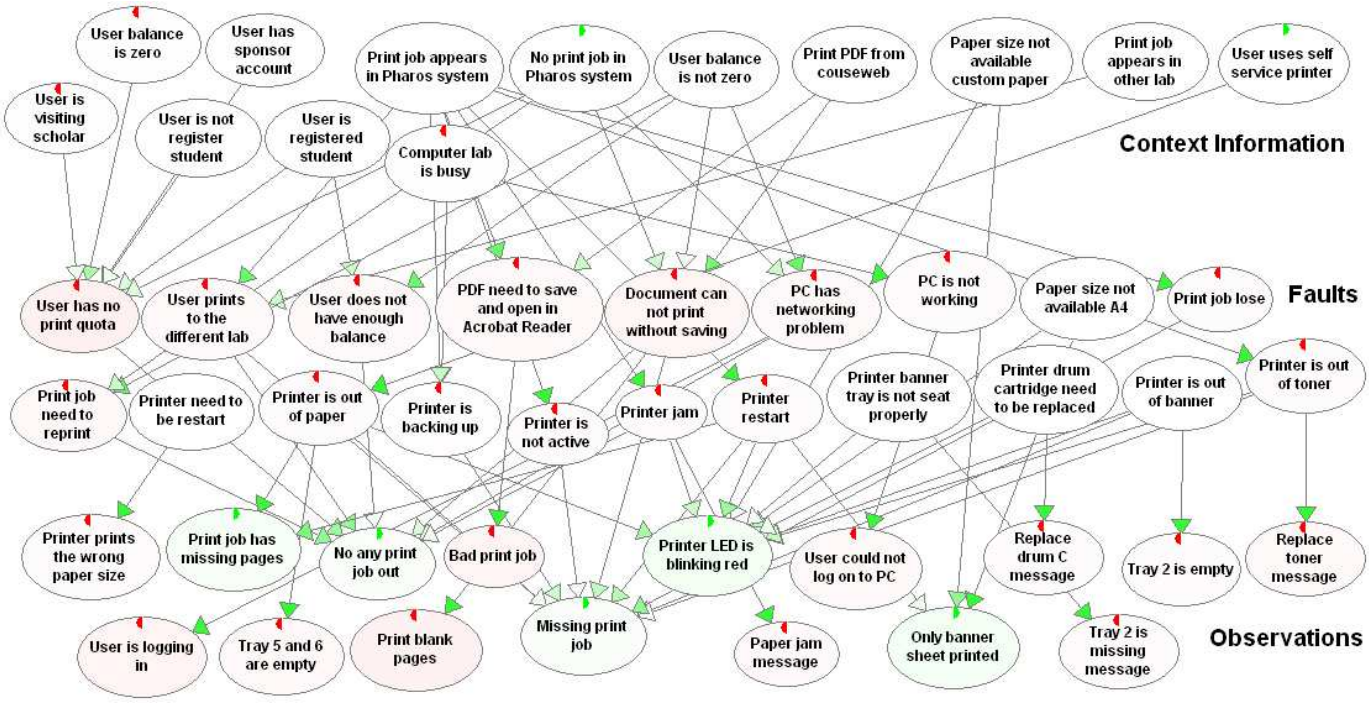

Fig. 10. The four hundred cases BN3D model built by Marilyn.

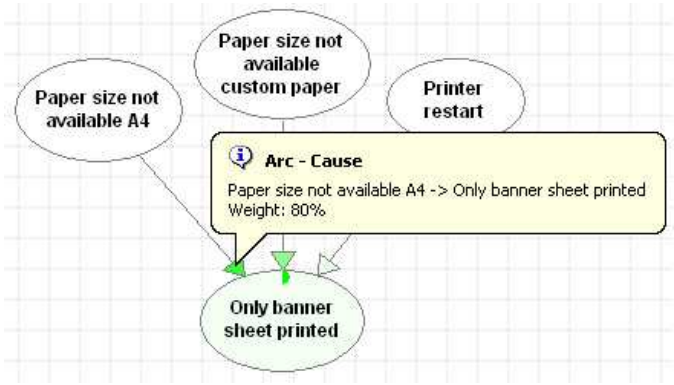

Fig. 11. An example of weight assigned of a child node with three-parent nodes

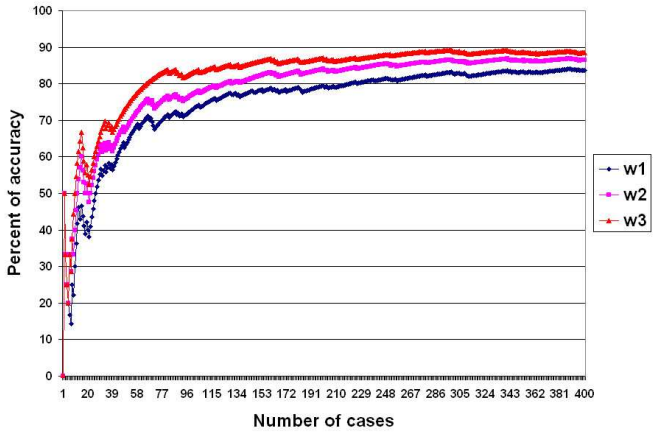

Fig. 12. The graph shows percent of accuracy as a function of the number of cases

We use the same measure of performance as [11]. We are interested in MARILYN suggestions that the list of possible diagnoses contains the correct diagnosis for a small set of values. We chose a *window* of W=1, 2, and 3.

Figure 12 shows the performance of MARILYN (in terms of percentage accuracy) as a function of the number of cases that have been entered into the system. With four-hundred cases, MARILYN reached the accuracy of 83%, 86% and 88% for W1, W2, and W3 respectively. The result shows that for the first fifty cases, MARILYN was not too good and gave an accuracy around 60% for W1, because the system need to collect enough diagnostic information to be able to perform well. However, after roughly, 70 cases, MARILYN's performance become very good.

Figure 13 shows MARILYN's accuracy as a function of the number of problems. It looks like there is a direct linear dependence between the number of problems observed by the system and its accuracy.

D. Model management

There are two critical practical issues related to MARILYN model management: (1) dealing with the free text input of the variables, and (2) model growth [1].

MARILYN allows its users to enter free text, which the program interprets as the name of a variable. In our experiment, we control users input by classifying the information, recorded

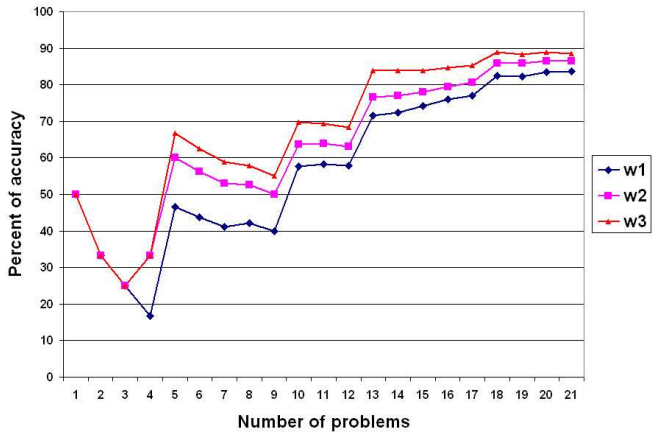

Fig. 13. The graph shows percent of accuracy as a function of the number of problems

in our computing lab log. Therefore, we do not deal with the problem of redundant nodes or wrong data. However, if a user enters redundant nodes or wrong data, it is possible to fix the model directly in the database in what we call "expert mode." For our domains, the number of parameters for each session was not large. To prevent redundancy and common typos in the future, we can provide additional drop down lists designed by domain experts to help a user to understand the parameters of the domain better.

The second important issue with MARILYN is how the system handles the growth of the model structure. Our reasoning engine SMILE is capable of handling large networks, and performs Bayesian inferences in a fraction of a second. The challenge for the algorithm is how to handle the growth of the number of parents for a single node. We suspect that MARILYN will perform slower as the number of parent nodes per node increases. In our experiment, even after four hundred cases have been entered in the system, this problem was not noticeable. There were only two nodes that contained the maximum number of parents per node, which is eight. This number may not be large enough to degrade the performance of MARILYN.

V. CONCLUSIONS

MARILYN is a passive diagnostic model construction tool which is able to give suggestions based on information entered by diagnosticians. We conducted an experiment to evaluate MARILYN's accuracy, testing the model by means of four-hundred cases of computing lab help desk data. This is a fairly realistic and popular application, similar to the problems faced by most call centers. The results of our experiment showed that MARILYN was capable of giving a very reasonable suggestion once the system has acquired enough information. This occurred after roughly 70 cases. Even though our experiment offers just one data point that addresses the problem and this type of systems need to be tested more in practice. We believe that the result is very promising, and passive model building approach will be useful in the future for various diagnosis domains.

We plan to improve and refine the MARILYN model building algorithm and probability calculation for better accuracy. With this data set, we can compare this result with other Bayesian network learning algorithms such as Naive Bayes. We also plan to adjust the web user interface, which is the crucial part of the model, and compare the performance of the learned model against the performance of a real domain user.

ACKNOWLEDGMENT

This work has been supported by the Government of the Kingdom of Thailand, the Air Force Office of Scientific Research grant FA9550–06–1–0243, and by Intel Research. Implementation of Marilyn is based on SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://genie.sis.pitt.edu/`. We would like to thank Erik Pols for his initial implementation of MARILYN. Christopher Faraglia and his classmates at the University of Pittsburgh improve this interface significantly. We thank anonymous reviewers for the PITA'09 Conference for valuable suggestion that improved the paper.

REFERENCES

[1] J. M. Agosta, T. R. Gardos, and M. J. Druzdzel, "Query-based diagnostics," in *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM–08)*, M. Jaeger and T. D. Nielsen, Eds., Aalborg, Denmark, 2008, pp. 1–8.

[2] D. Heckerman and J. S. Breese, "Decision-theoretic troubleshooting," *IEEE Transactions on Systems, Man & Cybernetics, Part A (Systems & Humans)*, vol. 26, no. 6, pp. 838–842, 1999.

[3] M. J. Druzdzel and L. C. van der Gaag, "Building probabilistic networks: "Where do the numbers come from?" guest editors' introduction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 4, pp. 481–486, July–August 2000.

[4] A. Onisko, "Probabilistic Causal Models in Medicine: Application to Diagnosis of Liver Disorders." Ph.D. dissertation, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, Warsaw, March 2003.

[5] E. Pols, "Marilyn a guided maintenance system that represents direct probabilistic influences among diagnostic knowledge," School of Information Sciences, University of Pittsburgh, Pittsburgh, PA, Technical Report, Apr. 2007.

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

[7] P. Kraaijeveld and M. J. Druzdzel, "GeNIeRate: An interactive generator of diagnostic Bayesian network models."

[8] F. J. Díez and M. J. Druzdzel, "Canonical probabilistic models for knowledge engineering," Unpublished manuscript, 2008.

[9] P. P. Maaskant and M. J. Druzdzel, "An ICI model for opposing influences," in *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM–08)*, M. Jaeger and T. D. Nielsen, Eds., Aalborg, Denmark, 2008, pp. 185–192.

[10] M. Henrion, "Some practical issues in constructing belief networks," in *Uncertainty in Artificial Intelligence 3*, L. Kanal, T. Levitt, and J. Lemmer, Eds. New York, N. Y.: Elsevier Science Publishing Company, Inc., 1989, pp. 161–173.

[11] A. Oniśko, M. J. Druzdzel, and H. Wasyluk, "Learning Bayesian network parameters from small data sets: Application of Noisy-OR gates," *International Journal of Approximate Reasoning*, vol. 27, no. 2, pp. 165–182, 2001.