Evaluation Results for a Query-Based Diagnostics Application

John Mark Agosta, john.m.agosta@intel.com
Intel Labs, Santa Clara, CA, USA

Omar Zia Khan and Pascal Poupart {ozkhan,ppoupart}@cs.uwaterloo.ca
University of Waterloo, Waterloo, Ontario, Canada

Abstract

Query-Based Diagnostics refers to the simultaneous building and use of Bayes network diagnostic models, removing the distinction between elicitation and inference phases. In this paper we describe a successful field trial of such a system in manufacturing. The detailed session logs that are collected during use of the system reveal how the model evolved in use, and pose challenging questions on how the models can be adapted to session outcomes.

1 Introduction

Diagnostic models for optimizing fault isolation represented as Bayes networks is a mature field that has seen numerous practical applications. These models have been shown to perform well given one has a comprehension model of the domain to which they apply. In many cases the challenge is to come up with an adequate model: This is the so-called "knowledge elicitation bottleneck." As described in a previous paper (Agosta et al., 2008) we proposed an approach to ease model creation, by combining the elicitation and model-building phase with the model use phase so that the model is improved as it is used. This combination accords well with the expert's view of the process: In the midst of actual problem-solving is the time when the expert is most aware of her mental model, and is best able to express it. We've called this approach *query-based diagnostics,* (QBD) to highlight the dependence of the model on the user's inquiries as they use the model. We report here the field trial results that we had proposed to undertake then. The reader is referred to that paper for the motivation and details of the web-based application which is described therein.

A diagnostic troubleshooting QBD application employing these ideas has been successfully fielded and evaluated by Intel Manufacturing to validate its use in practice. We will refer to it here as "The QBD Application." This paper describes the design and scope of the evaluation trial and the factors that led to its success.

At the start of our involvement with our client, we attempted to implement knowledge-based Bayes network diagnostics for corrective maintenance of factory equipment. Our client was critical of the need to pre-build models by conventional elicitation methods, because engineers and technicians could not be spared to be taken out of their daily activities for model-building, which in their opinion properly belonged under their control. Also repair knowledge is dynamic and would quickly get out of date. The alternative of learning models by statistical methods is not practical because failures are relatively rare by their nature, making available data too sparse. Our response to this quandary was to integrate model creation within the routine work-flow by which they were trained, by this aforementioned approach of mingling model creation and use.

Corrective maintenance refers to unanticipated failure of equipment that takes it out of production. The transition from operating to not operating is the *breakdown*; the transition back is the *repair*. When a factory operation is capacity limited, as opposed to demand limited, equipment breakdowns that constrain

capacity incur significant revenue opportunity losses. The purpose of the evaluation trial was to validate that such losses could be substantially avoided by adoption of the QBD Application.

The QBD Application contributes to the Lean Manufacturing philosophy of the firm, specifically by standardizing problem-solving for diagnosis and repair. "Lean" refers to the methods made popular by Toyota for continuous efficiency improvement (Womack, 1990). The justification for the Application as a way to promote knowledge-sharing and the criteria on which it was evaluated both have their roots in "Lean."

The trial was evaluated by the QBD Application's estimated contribution to the overall profitability of the factory. By virtue of running the software in the field we were also able to collect detailed session logs capturing user actions that revealed how the Application performed. After a brief look at the relevant literature in Section 2, we detail the analysis and findings of the evaluation, and the case they make for QBD in Section 3.

The success of the trial leads to several new challenges. One raised in our previous work is the adaptation of the model as sessions generate verified cases. In the course of the trial it also became apparent that user feedback, both active and passive in the sense of expert users following or not following the model's recommended steps can be used to improve the model. We discuss this in Section 4 of the paper.

2 Background

For the developments in Bayes networks for building normative troubleshooting models and the flurry of research done in the 80's and 90's on this topic we refer the reader to (Jensen, 2001), particularly in the bibliography included in the Preface. In a conventional application the model guides the user during a diagnostic session with a ranking of causes and tests. Causes are ranked by their marginal posteriors, conditioned on the evidence offered by the user, and tests are ranked by diagnostic value, often approximated as mutual information, or decrease in entropy of the causes' marginals. The model is run repeatedly, creating a dialog with the user of test suggestions alternating with test execution and the entering of new evidence into the model.

As described in the previous paper, the QBD application extends the concept of a diagnostic session by with editing functionality for active input of causal relations between variables, and simple inferences of causal relations based on passive observation of model building steps all captured by detailed logging of diagnostic sessions. Thus at any point in the session, the actions available to the user are

1. Enter a test result as evidence,
2. Create a new cause or new test node,
3. Add or remove a dependency arc between a cause and test node,
4. Choose a cause on which to perform a repair, ending the session.

In this way we were able to create models without making the Bayes network explicit. The goal is to have the software bootstrap the modeling task, so that, in principle the software could be fielded before any model-building commenced. Not surprisingly the model complexity is limited. The models are relatively sparse bipartite graphs with probabilities set qualitatively; see Figure 2 for an example. Despite their simplicity they suit the situation well, with the right blend of dynamicism, ease of use and comprehensive knowledge-capture.

There is a small but growing literature on applications of diagnostics Bayes networks and their evaluation in use (Pourret et al., 2008). This literature is largely concerned with their accuracy (Przytula et al., 2003). Published cases where the net benefit of an application in use are unknown to the authors, especially of the type we offer here.

3 Evaluation Trial

This section describes the purpose of the trial, the evaluation design, and what the results were.

The trial ran for seven weeks in one factory, with the team responsible for four types of equipment used in semiconductor process fabrication. The evaluation effort was carried out by the team that owns the Lean program.

3.1 Purpose of evaluation

An "ROI" study is required by the firm before it will adopt new software. Although the factory has implemented comprehensive data collection and analysis in all aspects of process and equipment monitoring, there is no system for the capture and use of "soft" user knowledge. In particular, support for maintenance activities extends only to on-line display of reference manuals. The QBD Application was the first trial in Manufacturing where the collection of soft data has been justified by its effect on improving key measures of performance.

3.2 ROI analysis

With the caveat that confidentiality considerations prevent revealing in this paper actual financial cost numbers, we can say that the trial showed substantial improvements in key measures of performance by which the success of the factory is evaluated. Expressed in dollar amounts, the net value of the trial would more than justify the annual budget for the entire lab where the Application was developed!

In the ROI analysis design, the value of the Application was expressed solely by its effect on equipment utilization. Capital utilization and resultant improvement in revenue dwarfed all other cost and benefit terms. Direct labor and material cost and savings, for instance were immaterial in comparison.

3.2.1 Trial and Model Assumptions

Despite the use of the term "ROI", the value analysis consisted of computing change in revenue net of cost, extrapolated to a year.

Assumptions: Three assumptions were made: 1) The equipment to which the model is applied constrains current production capacity. 2) The model would self-populate during the trial. 3) Technicians of various abilities would be involved, some contributing knowledge to the Application, others benefiting from the knowledge that had been entered.

Improvement in Means over Baseline	
MTBF	14%
MTTR	24%

Table 1: Mean values for performance improvements for weeks 14-20 compared to weeks 1-13.

Benefit: Management systems in the factory track Time To Repair (TTR) and Time Between Failure (TBR) by recording the times each machine goes down and comes back up. The change in equipment availability is a function of Mean TTR and Mean TBR. Improvement in factory throughput is estimated from availability of the capacity-limiting equipment step in the production process. Then, knowing weekly production throughput and financial contribution of each unit of product, additional revenue can be estimated. In short, starting with uptime and downtime statistics to estimate improvement in availability, one can estimate

$$\begin{aligned} &increased\_revenue \\ &= availability\_improvement(MTTR, MTBF) \\ &\quad \times production\_volume \\ &\quad \times contribution/unit \end{aligned} \quad (1)$$

Cost: The only direct cost was one additional half day of the planned Lean Manufacturing training. There were no additional R&D or IT infrastructure costs. One might argue that the time spent in sessions with the Application should be counted: From the session logs, we generously estimated that a total of less than two weeks of labor over the entire trial.

3.2.2 Results

We present here quantitative results for one of the four types of equipment involved. The conclusions are similar for each. MTTR and MTBR improved for the pilot period compared to the baseline period. Boxplots for both periods are shown in Figure 1. **The total repair time decreased by 41% while Mean Time to Repair decreased by 24%, as shown in**

Variance	Weeks		Decrease	P value
	1-13	14-20		
Repair Variance	1443.	148.8	0.103	0.0054
Availability Variance	50.51	6.90	0.137	0.011

Table 2: Variance reduction for the trial period compared to the previous period.

Table 1. More impressive is the improvement due to the variance of availability, which decreased by more than a factor of 10, as shown in Table 2, a decrease that passed a conventional test for statistical significance.

Since the QBD Application was an integral part of a new Lean program, it is not possible to tease apart the contribution of the Lean philosophy and the introduction of the Application; the software would not have been used had it not been part of the program, and the Lean workflow would not have been enforced without the software.

Clearly the decrease in repair variance has substantial value for manufacturing performance, on a par with the value of improvements in MTBF and MTTR. Ironically there were no financial analysis techniques available to value this decrease, even though its contribution may be more important than the value attributed to the increase in availability.

3.3 Model-building During Sessions

A benefit of the detailed session logging is the record of when and how model building occurred during diagnostic sessions. In this section we consider the sessions for the three machines for one type of equipment. The session logs captured the entire sequence of user actions: the symptom indicating the breakdown, each cause or test selected, what values tests were set to, and most importantly when causes, tests or dependencies were added to the models.

If we also had had detailed TTR and TBF records for each machine we could have associated sessions with repairs to obtain a more detailed analysis of the relationship between them, and a session-level analysis of effectiveness.

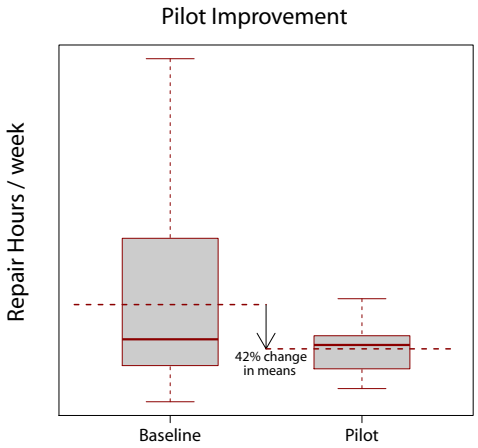

Figure 1: Compared to the previous period, the mean repair hours per week decreased substantially, relieving a production bottleneck and leading to an increase in overall factory availability.

We validated our presumption in the previous paper, that **within a few months models would mature in size, incorporating a useful level of coverage.** Model building occurred continually and extensively during the trial, validating our supposition that intermingling model-building with diagnostic sessions can replace conventional means of model elicitation. During the trial's seven weeks, users ran 157 *sessions,* 54 of which terminated with a repair action. Also we counted the number of diagnostic and model building *actions* in each session; the cumulative counts over all sessions, shown in Figure 3, show the rate of model-building. Our surrogate for diagnostic steps were the actions of setting the value of a test variable. Model building steps (e.g. adding a cause, test or dependency) out-paced diagnostic steps, except for a few short intervals near the end of the trial. We interpret this to imply that the models had not yet reached "saturation" by the trial's end. We would expect that we'd see the rate of testing surpass the rate of additions to the model as models continued to mature.

The arcs shown in the created network, Figure 2, were inferred from the sequence of user

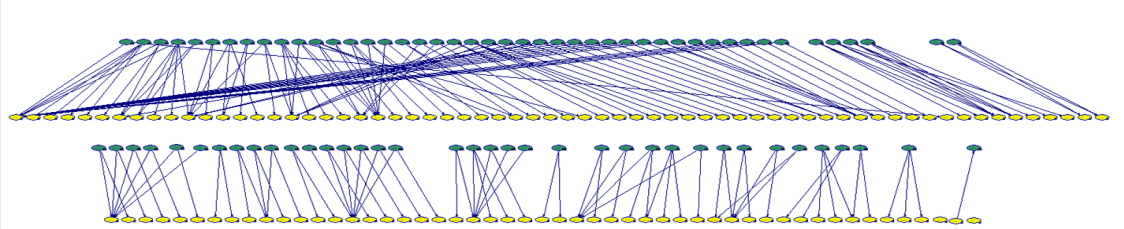

Figure 2: The bipartite network created in use during the pilot contains 115 test and 82 root cause variables, connected by 188 dependencies. Causes appear in the top row, and tests in the bottom.

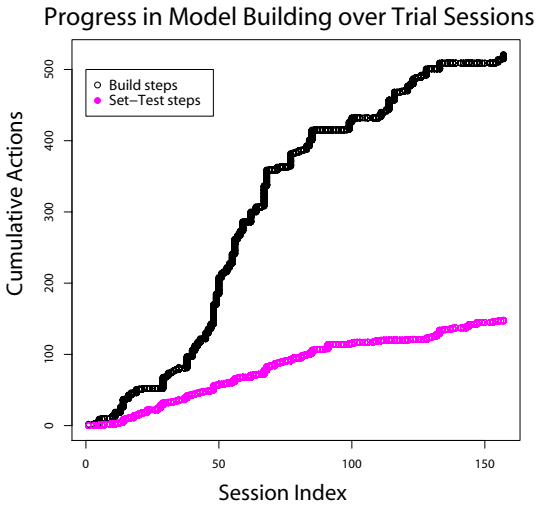

Figure 3: Model building occurred consistently over the course of the 157 sessions during the trial, with a slight decline during the end, and out-pacing the rate of diagnostic actions, except occasionally towards the end.

actions. The result is to generate a singly connected, network of test and cause pairs with all causes linked to one common symptom. All arcs that were not generated in this automatic fashion were added explicitly, using a cause-to-test linking feature in the interface.

4 Adaptation

In this section we present a method where we can make improvements to models by incorporating the results of explicit user feedback from sessions. There are two types of feedback we consider, first the user's ranking of causes, secondly of tests. Both types are represented as cases, serving as constraints to which the model should conform, as explained in the previous paper, where this consistency condition was presented:

Definition 1 (Case Cause-Consistency[1]). A model M^* is *cause-consistent* with a case j, to level k, if the list of ordered fault marginals given the evidence $\mathbf{e}^{(j)}$ agrees with the case: $\mathsf{P}\left(C^{(1)} \mid \mathbf{e}^{(j)}, M^*\right) \geq \ldots \geq \mathsf{P}\left(C^{(k)} \mid \mathbf{e}^{(j)}, M^*\right)$.

A case that raised cause consistency would occur if, after completing the diagnosis test sequence, the diagnostician discovers in the course of repair that C^* is the cause of the breakdown, not the C recommended by the model.

Extending the last paper, we consider another consistency condition that follows from the user's behavior in selecting tests in a different order than the diagnostic ranking provided by the model.

Definition 2 (Case Test-Consistency). A model M^* is *test-consistent* with a case j, to level k, if the list of ordered diagnostic test values $\mathsf{MI}\left(T^{(i)} \mid \mathbf{e}^{(j)}\right)$ for tests $T^{(i)}$ given the evidence $\mathbf{e}^{(j)}$ agrees with the case: $\mathsf{MI}\left(T^{(1)} \mid \mathbf{e}^{(j)}, M^*\right) \geq \ldots \geq \mathsf{MI}\left(T^{(k)} \mid \mathbf{e}^{(j)}, M^*\right)$.

For example, if, after observing the vector of evidence $\mathbf{e}$, the diagnostician picks test T^* rather than a test T that is ranked with higher diagnostic value, then the model is inconsistent with the diagnostician's choice. In the QBD Application, diagnosticians cannot indicate an inconsistency in test selections directly, but inconsistencies appear when the highest ranked test computed by the model with the evidence at that point in the session sequence is not the one chosen by the diagnostician. This is can be determined by running the model and comparing

[1] This is the definition from (Agosta et al., 2008)

its recommendations to test actions in the session log.

4.1 Modifying the network

If we believe the cases offered by experts that are inconsistent with the model are correct (and thus the model is giving the wrong recommendation), the model can be improved by modifying it to be consistent with the cases.

Consider the simple case of a four node network, as shown in Figure 4, but with the arc from C_1 to T_2 missing, such that the network is singly connected. The upper layer nodes labeled with C_x represent possible causes. The lower layer nodes labeled T_y represent possible diagnostic tests. Assume the case where the diagnostician ranks T_1 higher than T_2, in contrast to the model's ranking, which is the reverse. Consistency could be restored by either modifying the CPT of T_1, since it arbitrates between the two causes, or, assuming T_1 is a Noisy-OR, symmetric in both causes, exploiting its "Independence of Causal Influences" property using T_2 to change the balance between causes. The first approach is more direct, and seems most natural given its role in relating the two causes.

In the opposite case where the diagnostician ranks T_2 higher than T_1, in contrast to the model's ranking, the same options apply, to change the CPT of either T_1 or T_2 to restore consistency. However it may be most natural to add the arc from C_1 to T_2, so that T_2 is informative of both causes.

Once the network is multiply connected, such as the fully connected version in Figure 4, then intuitions are not so simple about how to modify the network to achieve consistency, and we propose a method that resorts to formulating the problem as a constrained optimization problem. We show in the next section how this might be done, and why the constraints being non-convex make this problem challenging.

4.2 Model Refinement as Optimization

We consider how a test-consistency constraint may be applied to an existing model by revising the parameters of the model such that the model becomes consistent with the case. To elaborate our approach for model refinement, we use the simple diagnostic Bayesian network shown in Figure 4.

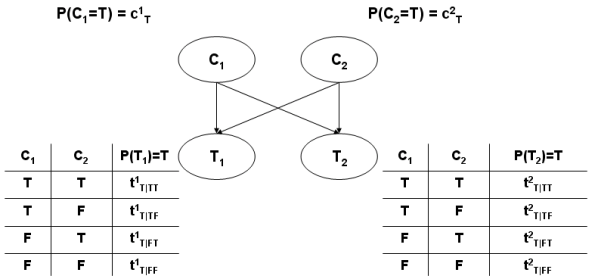

Figure 4: A bipartite diagnostic Bayes network with two causes and two tests.

Assume that both causes and tests are binary variables. The variable $c_i^{(x)}$ indicates the value for $\mathsf{P}(C_x = i)$ whereas the variable $t^y_{m|c_1c_2}$ indicates the value for $\mathsf{P}\left(T_y = m \,|\, C_1 = c_1, C_2 = c_2\right)$. The tables in Figure 4 only show half of the CPTs for these variables for the case where the value of the variable is true.

If a model is inconsistent with a case, it means that the parameters of this Bayesian network *i.e.*, the Conditional Probability Tables (CPTs) are not accurate due to which the ranking of tests is incorrect. Thus, our goal is to refine these parameters (or CPTs). If the model is consistent with a case, we may not need to refine the parameters, however it is still necessary to impose a constraint so that future refinements due to another case do not result in a model that is inconsistent to a previous case.

Consider the case where an expert chooses test T_1 initially without any evidence. We express the diagnostic value as mutual information. Thus the mutual information $\mathsf{MI}(C_1, C_2 \,|\, T_1)$ of the causes C_1, C_2 and test T_1 will be higher than that of the causes C_1, C_2 and test T_2. We can express this information as the following inequality constraint.

$$\mathsf{MI}(C_1, C_2 \,|\, T_1) \geq \mathsf{MI}(C_1, C_2 \,|\, T_2) \qquad (2)$$

Writing out the mutual information:

$$\mathsf{MI}(C_1 \ldots C_n \,|\, T_y) = \mathsf{H}(C_1 \ldots C_n) - \sum_{m \in \{T,F\}} t^y_m \mathsf{H}(C_1 \ldots C_n | T_y = m), \qquad (3)$$

where $\mathsf{H}(C_1 \ldots C_n)$ refers to the joint entropy of the network's causes, computed as follows:

$$\mathsf{H}(C_1 \ldots C_n) = - \sum_{c_1, c_2 \in \{T,F\}} \mathsf{P}(c_1, c_2) \log_2 (c_1, c_2). \tag{4}$$

Using Equations 3 and 4, we can rewrite the constraint in Equation 2 as follows.

$$\begin{aligned} 0 \geq & - \sum_{t_1, c_1, c_2 \in \{T,F\}} \mathsf{P}(t_1) \mathsf{P}\left(c_1, c_2 \,|\, t_1\right) \log_2 \mathsf{P}\left(c_1, c_2 \,|\, t_1\right) \\ & + \sum_{t_2, c_1, c_2 \in \{T,F\}} \mathsf{P}(t_2) \mathsf{P}\left(c_1, c_2 \,|\, t_2\right) \log_2 \mathsf{P}\left(c_1, c_2 \,|\, t_2\right) \end{aligned} \tag{5}$$

The constraint shown in Equation 5 is non-linear. This is evident if we use Bayes' theorem and rewrite the constraint only using the parameters of the Bayes network as shown here:

$$\begin{aligned} 0 \geq & \sum_{i,j \in \{T,F\}} c_i^{(1)} c_j^{(2)} \sum_{y=1}^{2} \sum_{m \in \{T,F\}} t_{m|i,j}^{(y)} \times \\ & \left[\log_2 t_{m|i,j}^{(y)} + \log_2 c_i^{(1)} + \log_2 c_j^{(2)} - \log_2 k_m^{(y)} \right] \end{aligned} \tag{6}$$

In the above equation, $k_m^{(y)}$ are normalization constants. They can be evaluated as follows.

$$k_m^{(y)} = \sum_{i,j \in \{T,F\}} t_{m|i,j}^{(y)} c_i^{(1)} c_j^{(2)}$$

The CPTs for the causes are the priors over those causes. This information can be learned using historical data as it only relates to the frequency with which a cause occurs. Thus, we can exclude this from the set of the parameters to be learned or refined and restrict ourselves to the CPTs of the tests, which are of the form $t_{m|i,j}^{(y)}$ in Equation 6.

The above constraint is the simplest possible version of a network, where there are only two causes, two nodes and no evidence available. For more complex networks, the constraints will be more complicated.

Most significantly, this constraint is not convex. It is well-known that entropy is concave, while mutual information is neither concave nor convex (Cover and Thomas, 1991). Since our constraint is a difference of mutual information it follows that it is a difference of two non-convex functions. There has been previous work on using EM algorithm with constraints to learn the parameters of a Bayes network (Niculescu et al., 2006). However, the constraints considered in those settings are linear, and thus convex.

This problem is also different as unlike traditional parameter learning problems, we do not have a lot of data to learn from. Thus, the traditional objective function of maximizing the data log likelihood may also not be useful as it could lead to over-fitting. Finally, we are already provided an existing model which needs to be refined rather than a new model learned from scratch. An alternate approach is to minimize distance from the original model using a measure such as an L_p-norm. However, this may not lead to a robust approach since when the model is not consistent, the distance measure will force the refined model to be at a corner point of the new feasible region. Instead, we are interested in an objective function that forces the parameters of the refined model to lie in the interior of the new feasible region, as far as possible from all the constraint surfaces. In this manner, the refined model is more likely to be consistent with any future constraints. Currently we are investigating the use of KL-divergence as a possible objective function that can help us achieve this goal.

5 Conclusion

The results presented in this paper show overwhelming benefits from employing an implementation of a QBD Application; benefits that outweighed costs by orders of magnitude, and made a material difference in the factory's estimated financial performance. One may wonder what this is attributable to, given the simplicity of the models and their lack of maturity, being created in the course of the trial. The answer is that the success of the trial relied on success in all three areas, financial, organizational and human interface. The lessons learned from this trial are

several:

1. Diagnostic applications have a natural application in improving equipment efficiency. In capital-intensive companies, improving capital efficiency justifies such applications since these savings dwarf those in other areas such as direct costs of labor and materials.

2. Lean manufacturing principles pave the way for introducing technology to improve efficiency, by making it clear to the organization where the technology fits. Management in the organization views the application as a way to capture and share "tribal" knowledge for Lean problem solving.

3. A new technology has to make the job easier for the person who uses it, or else it will not be adopted. Here the users comments were varied, ranging from "no tool (i.e. computer application) has as comfortable a user interface as QBD"; to critical: "System design still needs improvement to simplify usage" and "(It) Would be best if system was integrated into existing tools."

It is well known that working in a domain is facilitated by agreeing on a common, familiar vocabulary with the domain expert. Analogously, adopting a standard work-flow from the way that the client believes the diagnostic task should be performed, as the application workflow is also key to the application's acceptance. Knowledge elicitation methods recognize the importance of process and task analysis (Schreiber et al., 2000). Ours is a stronger statement, that the process steps incorporated in the application adopt the normative model that the organization uses.

In addition to the validation of the application, the trial identified areas for future work. The plan for the current software is to improve the interface and integration into existing software systems, and prove these changes in additional trials. Also the trial revealed other places where user actions may be used as input for the automated learning methods we are developing to improve the model, for instance Test-Consistency. Furthermore we are also considering the applicability of QBD to other diagnostic problem solving tasks, and to extend the method with test and repair costs and with replacement and repair actions, in a dynamic model.

Acknowledgments

It is our pleasure to give credit to Marek Druzdzel, Thomas Gardos, Matthias Giessler, Dan Peters, and Bryan Pollard, who in different ways contributed to and made this project possible. The QBD Application was built using the SMILE library.

References

John Mark Agosta, Tom R. Gardos, and Marek J. Druzdzel. 2008. Query-based diagnostics. In *Probabilistic Graphical Models*, September.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory.* John Wiley & Sons, Inc.

Finn V. Jensen. 2001. *Bayesian Networks and Decision Graphs.* Springer Verlag.

Radu Stefan Niculescu, Tom M. Mitchell, and R. Bharat Rao. 2006. Bayesian network learning with parameter constraints. *Journal of Machine Learning Research*, 7:1357–1383.

Olivier Pourret, Patrick Na'im, and Bruce Marcot, editors. 2008. *Bayesian Networks: A Practical Guide to Applications.* John Wiley & Sons, Inc.

K. Woytek Przytula, Denver Dash, and Don Thompson. 2003. Evaluation of Bayesian networks used for diagnostics. In *IEEE Proceedings Aerospace Conference.*

Guus Schreiber, Hans Schreiber, Anjo Akkermans, Robert de Anjewierden, Nigel Shadbolt Hoog, Walter Van de Velde, and Bob Wielinga. 2000. *Knowledge Engineering and Management: The CommonKADS Methodology.* MIT Press.

James P. Womack. 1990. *The Machine That Changed the World.* Simon & Schuster.