Degradation Modeling and Algorithm for On-line System Health Management using Dynamic Hybrid Bayesian Network

Chonlagarn Iamsumang, Ali Mosleh, Mohammad Modarres
The Center for Risk and Reliability
University of Maryland College Park, Maryland, USA

Abstract: This paper presents a new modeling method and computational algorithm for reliability inference with dynamic hybrid Bayesian network. It features a component-based algorithm and structure to represent complex engineering systems characterized by discrete functional states (including degraded states), and models of underlying physics of failure, with continuous variables. The methodology is designed to be flexible and intuitive, and scalable from small localized functionality to large complex dynamic systems. In System Health Management applications, this method introduces a well-defined interface between continuous system component status and discrete system functionality within the network model. Markov Chain Monte Carlo (MCMC) inference is optimized using pre-computation and dynamic programming for real-time monitoring of system health. The scope of this research includes new modeling approach, computation algorithm, and an example application for on-line System Health Management.
Keywords: On-line System Health Management, Dynamic Hybrid Bayesian Network

1. INTRODUCTION

With increasing complexity of today's engineering systems that contain various component dependencies and degradation behaviors, there has been increasing interest in real-time System Health Management (SHM) capability to continuously monitor sensors, software, and hardware components for detection and diagnostic of safety-critical systems. The modeling framework should be flexible to accommodate the complexity of component dependencies and failure behaviors, such as sequence-dependent failures, functional dependencies, etc.

Bayesian Networks (BN) [1][2] and their extension for time-series modeling known as Dynamic Bayesian Network (DBN) [3][4] have been shown by recent studies to be capable of providing a unified framework for system health diagnosis and prognosis [5][6][7]. Bayesian Network has many modeling features, such as multi-state variables, noisy gates, dependent failures, and general posterior analysis [8][9][10]. It also allows a compact representation of the temporal and functional dependencies among system components [11][12].

The main advantage of using BN in system reliability is its simplicity to represent systems and the efficiency for obtaining component associations. Another important benefit of BNs is that they enable us to integrate information from different sources, including experimental data, historical data, and prior expert opinion. This feature is particularly useful for the reliability assessment of fault tolerant systems, where failure data from tests and field operations are sparse and obtained from diverse source of information. Bayesian networks are particularly well suited to modeling systems that we need to monitor, diagnose, and make predictions about, all under the presence of uncertainty.

However, one of the barriers to applying BN to real-world problems is to be able to adequately handle the "hybrid models", which contain both discrete and continuous variables with general static and time-dependent failure distributions. Despite the advances in BN researches, the previous applications of BNs as mainstream technology for SHM problems remain modest. To date, the BN framework has only partially addressed these limitations [13][14][15][16]. The vast majority of BNs used in real world applications are either purely discrete or purely continuous.

For hybrid BNs containing mixtures of discrete and continuous nodes with non-Gaussian distributions, exact inference becomes computationally intractable [17]. The common approach to handling (non-

Gaussian) continuous nodes is to discretize them using some pre-defined range and intervals [18]. This is cumbersome, error prone and usually inaccurate.

Even though a universal framework for hybrid BN is currently impracticable, a special case algorithm can be effective in SHM where a relatively small subset of possible values covers a large proportion of all possible values typically encountered. This paper presents a hybrid BN-based methodology for component degradation model and efficient algorithms to apply them in on-line health monitoring of complex systems.

The focus of this research is to enable probabilistic diagnosis and prognosis of system in real-time by optimizing SHM modeling and Markov Chain Monte Carlo inference with pre-computation and dynamic programming to reduce the computation time and number of inferences required. Efficient computation allows on-line system monitoring and provides on-demand system health inquiry for operators to make maintenance decision and to prioritize which part of the system to investigate to avoid an accident.

2. PROPOSED METHODOLOGY AND ALGORITHM

2.1. Hybrid Bayesian Network

For SHM modeling, it is advantageous and intuitive to consider a hybrid system, typically with the continuous variables being modeled as continuous and the system's functionality probability being discrete.

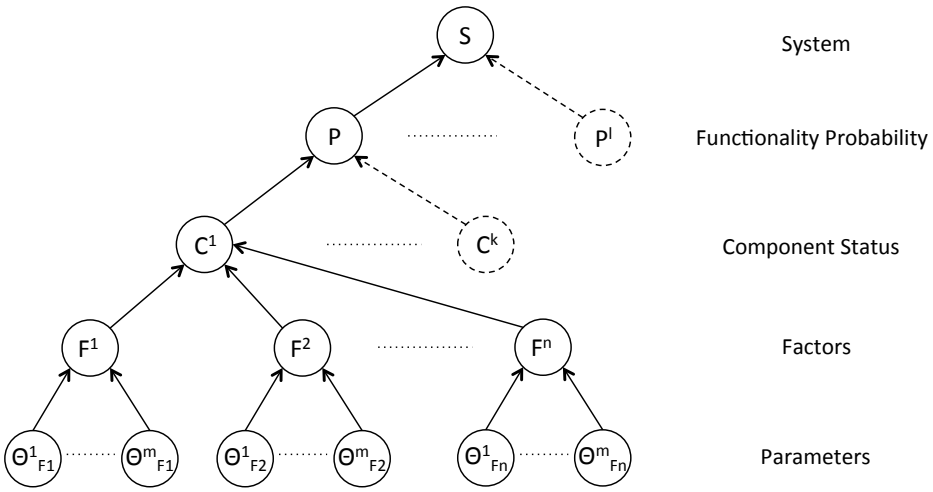

Figure 1: Overview of different levels in SHM Bayesian Network

The proposed complex system hybrid BN can be separated into 5 levels as shown in Figure 1, according to the typical characteristics of the nodes. The BN combines high-level functionality nodes with low-level physical of failure nodes. Here are the descriptions of each level:

1. System node: this is the highest level of the BN nodes (it has no children), it represents the state of the whole system and usually indicates whether or not the system is working as intended.
2. Functionality probability nodes: these nodes are designed to be abstract discrete nodes that represent various functionalities, which are required for the system to operate. The node can be requirement for operation of a single component, or multiple components.
3. Component status nodes: these are continuous nodes representing states of physical components susceptible to specific failure mechanisms in the system. These values should be measurable directly or indirectly, and they are expected to degrade over the lives of the components.

4. Factor nodes: these nodes contribute to the degradation of the components. They can be component internal factors related to material properties or physical characters, or they can be external factors such as environmental stresses or temperature.
5. Parameter nodes: these nodes are hyper-parameters that describe probability distributions of the factors.

It is to be noted that each level of the HBN could by itself be represented as a complex BN model. It can contain a combination of different layers of nodes that have the same type.

Reliability concerns arise when some critically important materials or devices degrade with time. Let *C* represent a critically important material/device parameter. This parameter degrades over the life of the component. The value itself can either increase (threshold voltage of a semiconductor device, increase in leakage of a capacitor, increase in resistance of a conductor) or decrease (decrease of pressure in a vessel, decrease of spacing between mechanical components, decrease in lubricating properties of a fluid). Figure 2 presents the SHM BN at a specific time, *t*. The shaded areas show continuous nodes that are related to each component.

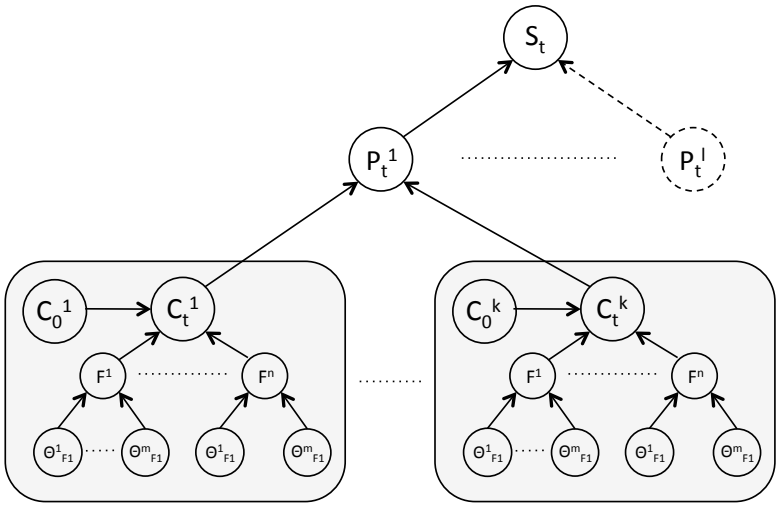

Figure 2: SHM Bayesian network at specific time *t*.

A Taylor expansion about t=0 produces the Maclaurin Series, assuming that *C* changes monotonically and relatively slowly over the lifetime of the material/device:

$$C(t) = C_{t=0} + \left(\frac{\partial C}{\partial t}\right)_{t=0} t + \frac{1}{2}\left(\frac{\partial^2 C}{\partial t^2}\right)_{t=0} t^2 + \cdots \quad (1)$$

By assuming that the higher order terms in the expansion can be approximated by simply modeling degradation of component/device parameter *C* with a power-law equation:

$$C = C_0[1 \pm A_0(F_1, \ldots, F_n)t^m] \quad (2)$$

where, C_0 is the value of C at $t = 0$. Summation (+) is used when the parameter *C* increases with time, while subtraction (-) is used when the parameter *C* decreases with time. Parameter A_0 is generally material/microstructure dependent. It is not only a function of material variations, but also a function of other factors, such electrical, thermal, mechanical and chemical environments to which the device is exposed. The parameter m and other parameters are considered to be constant for the component/device. Considering a BN at a time slice of a given system, t indicates the current life of the component/device.

For a component/device to fail, the amount of degradation must reach a critical value, C_{crit}. Therefore, the time to failure, $T_{failure}$, is then:

$$T_{failure} = \left[\frac{1}{\pm A_0(F_1, \dots, F_n)}\left(\frac{C_{crit} - C_0}{C_0}\right)\right]^{1/m} \tag{3}$$

Since the component status and their parents are continuous nodes, and the functionality probability nodes are discrete, the interface between these different types of nodes becomes critical. In general hybrid BN when continuous nodes have discrete parents, there are simple conditional inference techniques such as in conditional linear Gaussian (CLG) model. Difficulty arises when discrete nodes have continuous parents, which is the case for our SHM network. However in this case, even though discrete functionality probability nodes have continuous component status nodes, they are related by degradation thresholds.

Discrete functionality nodes can contain more than 2 states with thresholds between the transitions of one state to the other. Let the threshold value between functionality state i and j be $C_{th,i/j}$. The most common case would be state i denotes the component function, and state j denotes the component does not function. Let P_i be the probability of functionality being in state i. The probability P_i is then the probability that the component status C is lower than the threshold value $C_{th,i/j}$. Figure 3 shows a typical component exponential degradation function and the overlap of probability distributions of C and $C_{th,i/j}$.

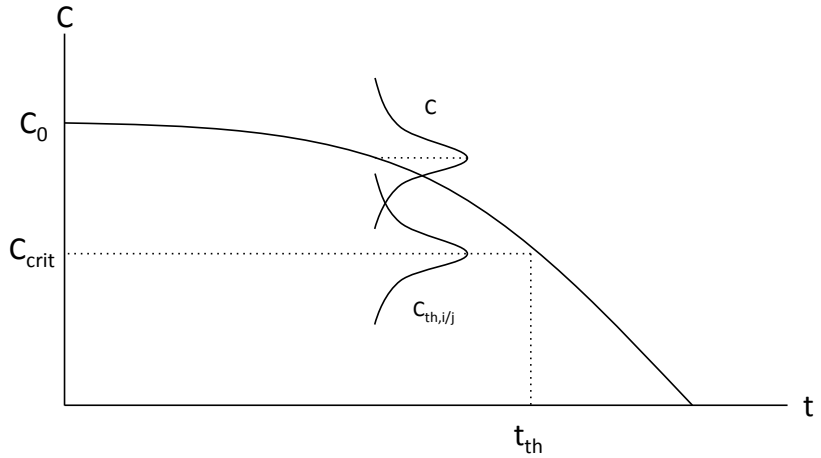

Figure 3: Overlap of probability distribution of component status and its threshold.

Let a functionality node has n states, the probabilities of being in the states are $P_1, \dots, P_n$. Assume the state of the functionality node changes monotonically according to the component degradation status:

$$C_{th,i-1/i} < C_{th,i/i+1} \text{ for } i = 2, \dots, n-1 \tag{4}$$

Therefore,

$$P_i = prob\left(C_{th,i-1/i} < C < C_{th,i/i+1}\right) \tag{5}$$

Analytically, P_i can be calculated from the following convolution equation:

$$P_i = \int_{-\infty}^{C_{th,i/i+1}} \int_{C_{th,i-1/i}}^{\infty} \int_{C_{th,i-1/i}}^{C_{th,i/i+1}} p\left(C_{th,i-1/i}\right) \cdot p(C) \cdot p\left(C_{th,i/i+1}\right) dC dC_{th,i/i+1} dC_{th,i-1/i} \tag{6}$$

If there are k component status parameters contribute to this functionality then the state of the functionality node conditionally depends on comparison between the status of each component and its threshold values.

$$P_{i^1 \dots i^n} = prob\left(C^1{}_{th,i^1-1/i^1} < C^1 < C^1{}_{th,i^1/i^1+1}, \dots, C^n{}_{th,i^n-1/i^n} < C^n < C^n{}_{th,i^n/i^n+1}\right) \quad (7)$$

2.2. Dynamic Bayesian Network

Dynamic Bayesian Network (DBN) is a Bayesian network that includes a temporal dimension. This new dimension is managed by time-indexed random value t to indicate time stage of the nodes. A set of nodes at certain stage contains random variables relative to time slice t. An arc that links two variables belonging to different time slices represents a temporal probabilistic dependence between these variables. Variables can be modeled to have impact on the future distribution of the other variables. These impacts are defined as transition probabilities between the stats of variables at time step t and $t + \Delta t$.

A DBN describes the joint distribution of a set of variables **θ.** This is a complex distribution, but may be simplified by using the Markov assumption. The Markov assumption requires only the present state of the variables $\boldsymbol{\theta}_t$ to estimate $\boldsymbol{\theta}_{t+1}$, i.e. $p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_0,\dots,\boldsymbol{\theta}_t) = p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$ where p indicates a probability density function and bold letters indicate a vector quantity. Additionally, the process is assumed to be stationary, meaning that $p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$ is independent of t.

For SHM Bayesian network, the main variables that change between time slices are component parameters. Components degrades over time, therefore, the status of components at a certain time slice depend on their status at the previous time slice and the factors affecting the degradation processes during that transition.

$$p(C_t) = p(C|C_{t-\Delta t}, \{F_t^1, \dots, F_t^n\}) \quad (8)$$

Given that F_t^i is the average value of factor i between time slice $t - \Delta t$ and t. Δt should be set according to the system under interest and how often the parameters can be observed, such as frequency of sensor signals. The benefit of continuous monitoring and inferences of variable that cannot be observed directly is to detect anomaly in the system before failure actually occur. Figure 4 shows a two-time-slice representation of a dynamic SHM Bayesian network.

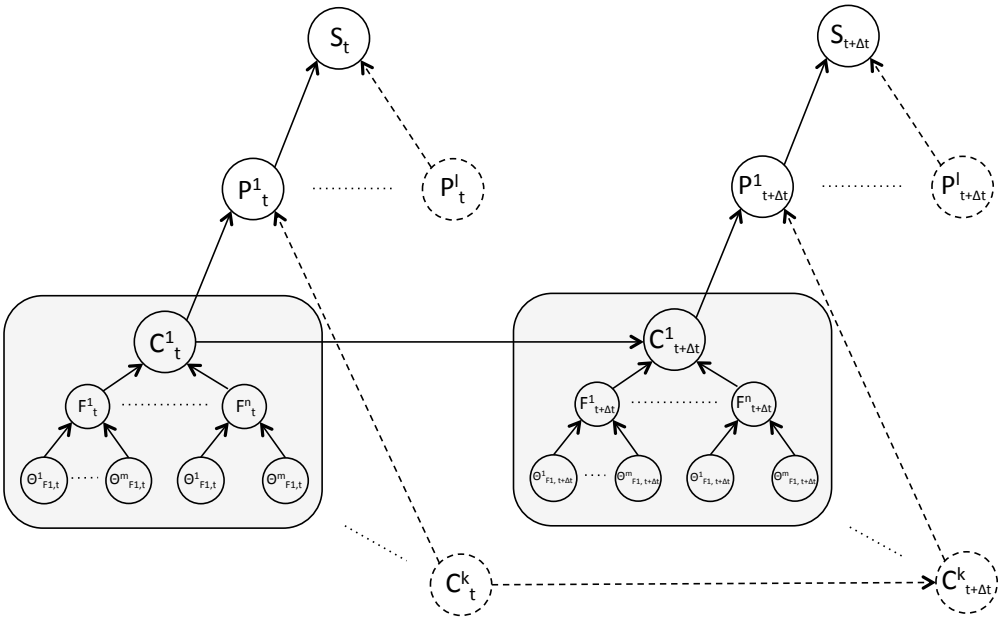

Figure 4: Two-time-slice representation of a dynamic SHM Bayesian network

At any point in time during system operation, any value of variables in the system can be derived by probabilistic inference to compare with its expected value to see if the probability is still in the acceptable range and the system as a whole is working as intended. With continuous monitoring, the trajectory of the degradation processes can be estimated form our knowledge of the health of the system. We can then use this information to estimate remaining useful life (RUL) of components and plan maintenance accordingly.

2.3. Inference

Bayesian network is a complete model for the variables and their relationships. Therefore, it can be used to answer probabilistic queries about them. The main application is to use BN to realize updated knowledge of the state of a subset of variables when other variables (the evidence variables) are observed.

Bayes' rule with continuous variables:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int d\theta\, p(D|\theta)p(\theta)} \tag{9}$$

Let θ be a parameter value and D is data value of the evidence, $p(\theta|D)$ is then the posterior probability of getting parameter value θ when data value D is presented.

In real world SHM applications, there are various types of parameter distributions, which make it difficult to calculate full marginal distributions analytically. Therefore, sampling techniques can be used to approximate the distributions instead. Expected values of a distribution can be estimated as follow:

$$E[p(\theta|D)] \approx \frac{1}{N}\sum_{n=1}^{N} p\left(\theta^{(n)}|D\right) \tag{10}$$

Where $\theta^{(1)}, \dots, \theta^{(n)}$ are the sample values of parameter θ.

There are many ways to sample these values, the key idea is to let θ values be points in state space and find a way to walk around so that the likelihood of visiting any point θ is proportional to $p(\theta)$. Therefore, the sampler will spend more time sampling from the distribution where the probability is high, and spending less time sampling from where the probability is low. This can be achieved by using Markov chain Monte Carlo (MCMC) algorithm [19][20].

The procedure for updating the belief about the system state as new information becomes available is called Bayesian recursive filtering.

$$p(\theta_t|D_{1:t}) = \frac{p(D_t|\theta_t)p(\theta_t|D_{1:t-1})}{\int d\theta\, p(D_t|\theta_t)p(\theta_t|D_{1:t-1})} \tag{11}$$

Under certain assumptions, such as when the system is linear Gaussian, the belief state will be of a known parametric form and computationally efficient solutions to the filtering problem (e.g. Kalman filter, extended Kalman filter, unscented Kalman filter) are available. Outside such assumptions, a computationally feasible method for inference in the DBN is particle filtering, a form of sequential Monte Carlo based on Bayesian recursive filtering. Common particle filtering methods are based on sequential importance sampling (SIS) [21].

2.4. Computational Algorithm for On-line SHM

In highly complex systems, MCMC algorithm requires large amount of computational time for inference in hybrid DBN. The computation time grows exponentially with each additional layer of network and becomes infeasible with large number nodes. The computation time makes it impossible for on-line health monitoring of complex systems. To solve this problem, special case algorithm for SHM is introduced to reduce the number of computations and the amount of time required for each computation.

One of the main characteristics of SHM in contrast of other applications is that during a normal operation, the environmental factors that affect component degradation process are expected to be roughly the same and predictable. Therefore, instead of performing Bayesian updating at a specific time interval, it only needs to be done when a factor value changes outside of expected range.

$$|f_t - f_{t-1}| > \epsilon_f \tag{12}$$

Where ϵ_f depends on the sensitivity of component status due to the change in value of that factor. Please note that this is possible because component status is a function of time. Therefore, the degradation of a component between time period t_i to t_j where the change in factor value is less than ϵ_f will take a normal distribution $\mathcal{N}(\mu_f, \sigma_f)$ for $\Delta t = t_j - t_i$.

Since the values are predicted to be in certain ranges, it is possible to perform pre-computation for all combinations of possible values in the ranges before the system is in operation. The results are then stored in a database, such that they can be pulled quickly to approximate the inferences in real-time. More computation should be conducted and more results should be added to the database as the health of the system is being monitored such that the database will cover all the possible computations that may be needed in the future.

With continuous range of parameter values, it is impossible to pre-compute every possible outcome. The goal of pre-computation is to cover enough values of observable parameters, so that the values of unobservable parameters can be accurately interpolated from the results. There are two factors in considering the selection of possible values.

First is the range of observable parameters after a time period Δt. The selections should cover full range of possible values. There should be at least one selected value at lower bound and one selected value at upper bound. The common range is from 5th percentile to 95th percentile, or more accurately 0.5th percentile to 99.5th percentile.

Second is the number of selections within the bound: the higher the number of selections, the more accurate results from interpolation will be. The density of selections should be proportional to the probabilistic density of the observable parameters. For example, if 19 values should be selected, then they should be the values at 5^{th}, 10^{th}, … , 90^{th}, 95^{th} percentiles. Therefore, for a given measurement interval Δt, we can estimate the set of possible values and use those values to pre-computed possible outcomes.

There are two different types of observable parameters. The first one is the parameters that change over time. This is usually the case for component status parameters. For pre-computation to be feasible, the changes must be predictable. For a component status parameter, the change in value can be computed from its degradation equation for a given Δt. Figure 5 shows example expected value, 5^{th} percentile, and 95^{th} percentile values.

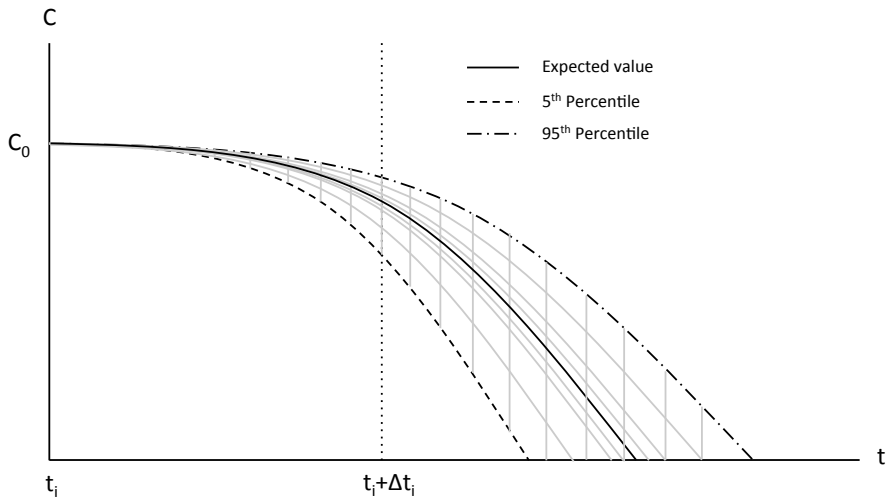

Figure 5: Example component status degradation with 5th percentile, and 95th percentile values.

For this case, the range of possible values grows over time. Therefore, the number of selection should increase proportionally with the range to keep the interval between selected values the same, thus, keep the accuracy of interpolation constant. For example, if there is N number of selections per variable, the selections are:

$$C_s = \left\{C^{p_{low}th}, C^{p_{low}+\delta th}, C^{p_{low}+2\delta th}, \dots C^{p_{high}th}\right\} \tag{13}$$

$$\delta = \frac{p_{high} - p_{low}}{N_{selections} - 1} \tag{14}$$

The other type of observable parameters is constant parameters. These parameters are usually Gaussian distributed. For this case, the range always stay constant, therefore, the selections remain the same throughout the life of the component.

If the observed values are always in the predicted range, the accuracy of the results depends upon the number of selections for pre-computation. The number of selections is the number of selections at each time-slice multiplies be the number of measurement intervals. The number of pre-computations is then the number selections for each observable times the number of observables parameters.

$$N_{pre-computation} = \sum_{j=0}^{T_c/\Delta t} \left(\prod_{i=1}^{n} N_{selections,i,t+(j\Delta t)} \right) \tag{15}$$

Where $N_{selections,i,t}$ is the number of selections of observable parameter i at time t. n is the number of observable parameters. T_c is the component life. The total computation time then can be estimated.

$$T_{pre-computation} = N_{pre-computation} \cdot T_{average-per-computation} \tag{16}$$

For MCMC computation, the average computation time is proportional to the number iterations. The higher the number of iterations, the higher accuracy of the result will be. Therefore, there is a trade-off between computation time and accuracy. For pre-computation, the decision between higher number of value selections or higher number of iteration per computation must be made.

One advantage of the isolation among component sub-tree is that time intervals do not have to be uniform for all components. Measurement/inspection intervals can be based on the rate of component degradation and possible change to component parameters. They can also be dynamically changed during the life a component depending on its status. For example there can be less frequency of measurements during the early life of a component due to less probability of failure. Then increase the

frequency when the component approaches the end of life. The time interval between measurements, Δt, should then be inverse proportional to the amount of change of the parameter C, $\Delta t \propto 1/\Delta C$. Thus, the sampling rate around a certain evidence value will be proportional to the probability that the evidence value could happen and how much different in values to the possible values around it at certain period of time.

Dynamic programming is a method for solving complex problems by breaking them down into simpler subproblems. It is applicable to problems exhibiting the properties of overlapping subproblems and optimal substructure. When applicable, the method takes far less time than naive methods that don't take advantage of the subproblem overlap. In general, to solve a given problem, we need to solve different parts of the problem (subproblems), then combine the solutions of the subproblems to reach an overall solution. Often when using a more naive method, many of the subproblems are generated and solved many times. The dynamic programming approach seeks to solve each subproblem only once, thus reducing the number of computations: once the solution to a given subproblem has been computed, it is stored the next time the same solution is needed, it is simply looked up. This approach is especially useful when the number of repeating subproblems grows exponentially as a function of the size of the input.

Using dynamic programming can reduce the precomputation time for Bayesian Network inference drastically. Instead of computing full inferences for each set of evidence values, dynamic programming algorithm retain marginal results that can be reused with similar set of evidence values.. There are three steps for the algorithm:

1. Use logic-sampling algorithm and degradation model to generate all possible evidence values according to its probability of occurring. Not all evidence nodes have to be instantiated for each case, only the evidence nodes that are required for observing nodes are instantiated.
2. Check and construct a cache by comparing each generated case to those already in the cache. If the case is found to be new, this algorithm determines, the joint probability of the case's evidence using the algorithm in the third step.
3. The marginal posterior-probability distributions over the diagnosis nods are determine, then the values of the evidence nodes, the joint probability of the evidence set, and the marginal posterior-probability distributions for the diagnosis node are stored in the cache.

Figure 6 shows two example cases where dynamic programming can reduce the number of computation. The first case is when nodes have the same set of parent nodes, thus the same sets of possible marginal probability distributions for discrete nodes. The second case is when continuous parameters have several trajectories that can reach the same values after some period of time. In addition, if more computations are needed during an operation in the event where evidence values reaches the bound of expected values, dynamic programming provide a set of marginal results that can be used for possible faster inference of values outside the pre-computed cache.

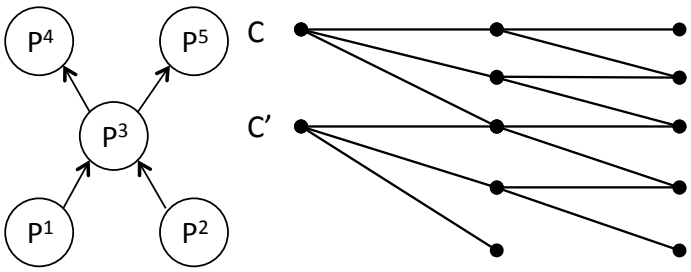

Figure 6: Example cases where dynamic programming reduces number of computations

Since both deterministic and approximate inference were found to be NP-hard [22][23], the computation complexity for both discrete functionality and continuous component degradation model are exponential in the network's treewidth. Figure 7 shows a plot presenting differences between pre-computation time with and without dynamic programming.

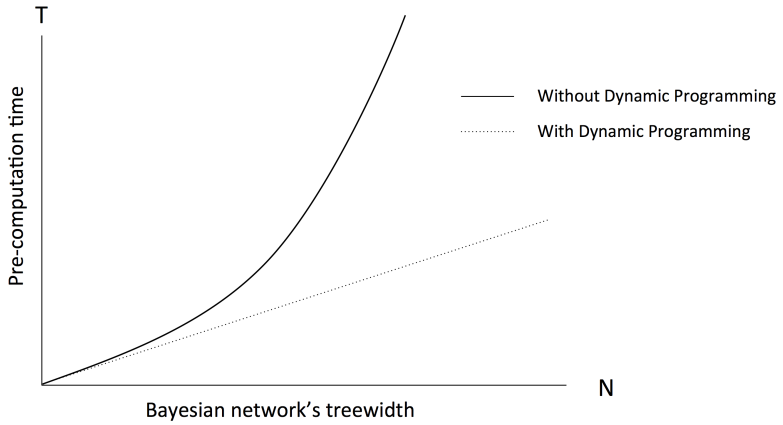

Figure 7: Inference pre-computation time with and without dynamic programming.

3. EXAMPLE APPLICATION

Electromigration (EM) in advanced integrated circuits (ICs) was considered to demonstrate the proposed methodology. For both Al-based and Cu-based metallization, EM has historically been a significant reliability concern.

The model generally used to describe EM time-to-failure takes the form:

$$TF = B_0 \left(J^{(e)} - J_{crit}^{(e)} \right)^{-n} \exp\left(\frac{Q}{K_B T}\right), \tag{17}$$

where: TF is the component time to failure. B_0 is a process/material-dependent coefficient. $J^{(e)}$ is the electron current density. $J_{crit}^{(e)}$ is a critical (threshold) current density which must be exceeded before significant EM is expected. n is the current density exponent. Q is the activation energy.

Using degradation model of component/device parameter C with the power-law equation:

$$C = C_0\left[1 - A_0\left(J^{(e)}, T\right)t^m\right] \tag{18}$$

We can derive at the following relationship:

$$C = C_0\left[1 - A_0 \cdot \left(J^{(e)} - J_{crit}^{(e)}\right)^r \cdot \exp\left(\frac{-Q}{K_B T}\right) t^m\right] \tag{19}$$

Since both current density $J^{(e)}$ and temperature T are expected to be normally distributed between time *t-1* to *t*,

$$\begin{aligned} J^{(e)} &= \mathcal{N}\left(\mu_J, \sigma_J\right) \\ T &= \mathcal{N}\left(\mu_T, \sigma_T\right) \end{aligned} \tag{20}$$

In the context of simple health monitoring in this example, $A_0, Q, r,$ and m are considered to be constant parameters representing material/device internal factors. These parameters can also be modeled with probabilistic distributions. The BN model of a component affected by EM is shown in Figure 8.

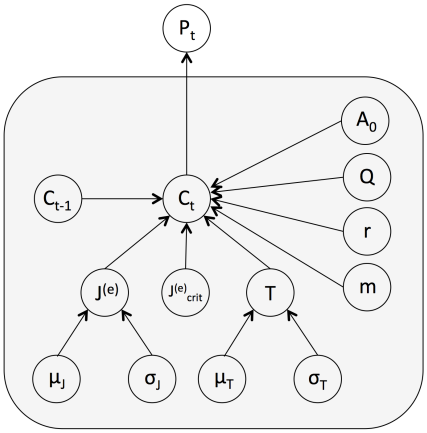

Figure 8: BN of a component affected by EM.

Consider an Al-alloy under high temperature operation, with current density J = 2×10^6 A/cm^2 and at a metal temperature T = 200 °C. Assuming an activation energy of Q = 0.8 eV and the current density exponent of n = 2. Using conservative design approach, assume J_{crit} = 0. Consider the data set shown in Figure 9 of $J^{(e)}$ and T during an operation.

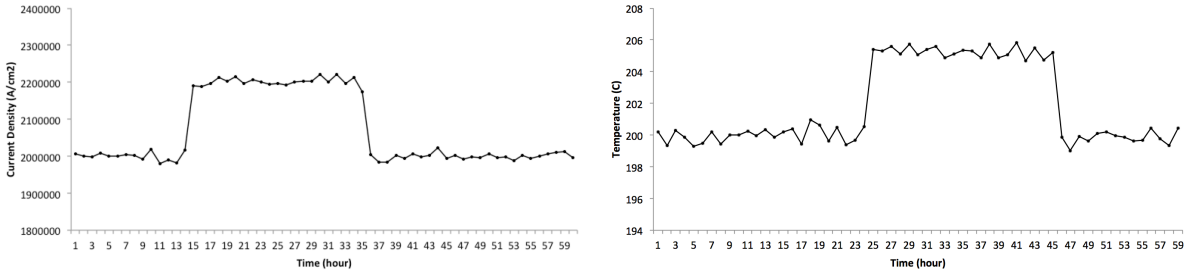

Figure 9: Current density and temperature data set

The data is retrieved once per minute during one hour of operation. In traditional Bayesian updating method, a calculation is required at each time step, which means 60 inferences have to be performed. With the proposed algorithm, only 4 inferences are needed when the values of current density and temperature changes out of ϵ_f range. Approximate inference of component status is available almost instantly with pre-computation of C_t at t = 1,…,60, with the range of J between 1.8×10^6 A/cm^2 to 2.2×10^6 A/cm^2, and T between 90°C to 120°C. Figure 10 shows an example plot of component status degradation under electromigration vs. time at different current density and temperature, including from the data set.

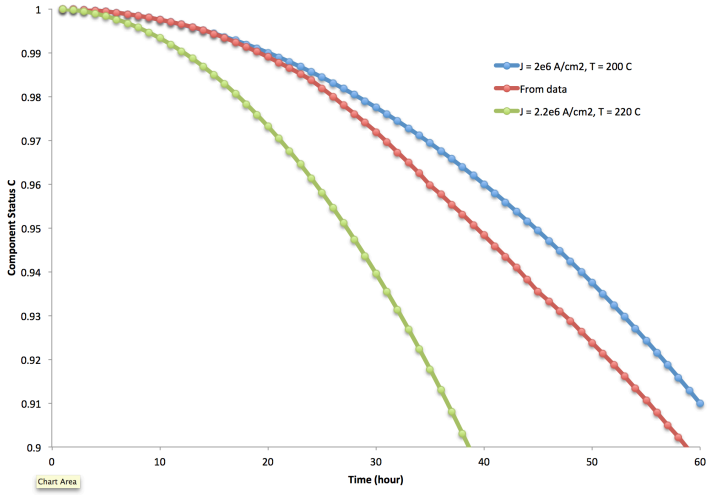

Figure 10: Plot of component status under electromigration vs. time at different current density and temperature, including from the data set

4. CONCLUSION

This research presents new modeling approach, computational algorithms, and an example application for on-line System Health Management. New method of using dynamic hybrid Bayesian Network were introduced with component-based algorithm and structure to represent complex engineering systems in a way that it allows accurate representation of underlying physics of failure by using empirical degradation model with continuous variables.

With dynamic hybrid Bayesian Network model requiring Markov Chain Monte Carlo for probabilistic inference, this paper develops computational algorithms that enables monitoring and diagnosing complex systems in real-time. The algorithms use the characteristics of System Health Management applications to allow reduction of number of inference required and reduce the calculation time by the means of pre-computation and dynamic programming.

References

[1] Pearl, J. (1986). Fusion, Propagation and Structuring in Belief Networks. *Artificial Intelligent , 29*, 241-288.

[2] Jensen, F. (2001). *Bayesian Networks and Decision Graph* . Springer.

[3] Friedman, N. (1998). The Bayesian structural EM algorithm. In G. F. Cooper (Ed.), *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)* (pp. 129-138). Morgan Kaufmann.

[4] Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, UC Berkeley , Dept. Computer Science .

[5] Ferreiro, S., Arnaiz, A., Sierra, B., & Irigoien, I. (2011). A Bayesian network model integrated in a prognostics and health management system for aircraft line maintenance. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering , 225* (8), 886-901.

[6] Tobon-Mejia, D. A., Medjaher, K., Zerhouni, N., & Tripot, G. (2012). A Data-Driven Failure Prognostics Method Based on Mixture of Gaussians Hidden Markov Models. *Reliability, IEEE Transactions on , 61* (2), 491-503.

[7] Schumann, J., Rozier, K. Y., Reinbacher, T., Mengshoel, O. J., Mbaya, T., & Ippolito, C. (2013). Towards Real-time, On-board, Hardware-supported Sensor and Software Health Management for Unmanned Aerial Systems. *Annual Conference of the Prognostics and Health Management Society.*

[8] Wilson, A. G., & Huzurbazar, A. V. (2007). Bayesian networks for multilevel system reliability. *Reliability Engineering & System Safety , 92* (10), 1413-1420.

[9] Langseth, H., & Portinale, L. (2007). Bayesian networks in reliability. *Reliability Engineering & System Safety , 92* (1), 92-108.

[10] Doguc, O., & Ramirez-Marquez, J. E. (2009). A generic method for estimating system reliability using Bayesian networks. *Reliability Engineering & System Safety , 94* (2), 542-550.

[11] Boudali, H., & Dugan, J. B. (2006). A continuous-time Bayesian network reliability modeling, and analysis framework. *Reliability, IEEE Transactions on , 55* (1), 86-97.

[12] Weber, P., & Jouffe, L. (2006). Complex system reliability modelling with dynamic object oriented bayesian networks (DOOBN). *Reliability Engineering & System Safety , 91* (2), 149-162.

[13] Lauritzen, S. L., & Jensen, F. (2001). Stable Local Computation with Conditional Gaussian Distributions. *Stat. & Comp. , 11*, 191-203.

[14] Moral, S., Rumi, R., & Salmeron, A. (2001). Mixtures of Truncated Exponentials in Hybrid Bayesian Networks. In *ECSQARU 2001. LNCS (LNAI)* (Vol. 2143, pp. 156-167). Springer, Heidelberg.

[15] Lerner, U. N. (2002). *Hybrid Bayesian Networks for Reasoning About Complex Systems.* Standford University, Dep. of Comp. Sci. Stanford.

[16] Shenoy, P. P. (2006). Inference in Hybrid Bayesian Networks Using Mixtures of Gaussians. In *Uncertainty in Artificial Intelligence* (pp. 428-436). AUAI Press, Corvallis.

[17] Boyen, X., & Koller, D. (1998). Tractable inference for complex stochastic processes. *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* .

[18] Neil, M., Tailor, M., Marquez, D., Fenton, N., & Hear. (2007). Inference in Bayesian Networks using dynamic discretisation. *Statistics and Computing , 17* (3), 219-233.

[19] Cousins, S. B., Chena, W., & Frisse, M. E. (1993). A tutorial introduction to stochastic simulation algorithms for belief networks. *Artificial Intelligence in Medicine , 5* (4), 315-340.

[20] Dagum, P., & Horvitz, E. (1993). A Bayesian analysis of simulation algorithms for inference in belief networks. Networks. *23* (5), 499-516.

[21] Chen, Z. (2003). Bayesian filtering: From Kalman filters to particle filters, and beyond. *Statistics* , 1-69.

[22] Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence , 42* (2-3), 393-405.

[23] Dagum, P., & Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard . *Artificial Intelligence , 60* (1), 141-154.