Efficient Dependency Computation for Dynamic Hybrid Bayesian Network in On-line System Health Management Applications

Chonlagarn Iamsumang, Ali Mosleh, Mohammad Modarres

The Center for Risk and Reliability
University of Maryland College Park, Maryland, USA
kci@umd.edu, mosleh@umd.edu, modarres@umd.edu

ABSTRACT

This paper presents a new dependency 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. 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.

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 Network (BN) (Pearl, 1986) (Jensen, 2001) and their extension for time-series modeling known as Dynamic Bayesian Network (DBN) (Friedman, 1998) (Murphy, 2002) have been shown by recent studies to be capable of providing a unified framework for system health diagnosis and prognosis (Ferreiro, Arnaiz, Sierra, & Irigoien, 2011) (Tobon-Mejia, Medjaher, Zerhouni, & Tripot, 2012) (Schumann, Rozier, Reinbacher, Mengshoel, Mbaya, & Ippolito, 2013). Bayesian Network has many modeling features, such as multi-state variables, noisy gates, dependent failures, and general posterior analysis (Wilson & Huzurbazar, 2007) (Langseth & Portinale, 2007) (Doguc & Ramirez-Marquez, 2009). It also allows a compact representation of the temporal and functional dependencies among system components (Boudali & Dugan, 2006) (Weber & Jouffe, 2006).

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 (Lauritzen & Jensen, 2001) (Moral, Rumi, & Salmeron, 2001) (Lerner, 2002) (Shenoy, 2006). 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 (Boyen & Koller, 1998). The common approach to handling (non-

Gaussian) continuous nodes is to discretize them using some pre-defined range and intervals (Neil, Tailor, Marquez, Fenton, & Hear, 2007). 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 online health monitoring of complex systems.

The focus of this research is to enable probabilistic diagnosis and prognosis of system in real-time by optimizing 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

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.

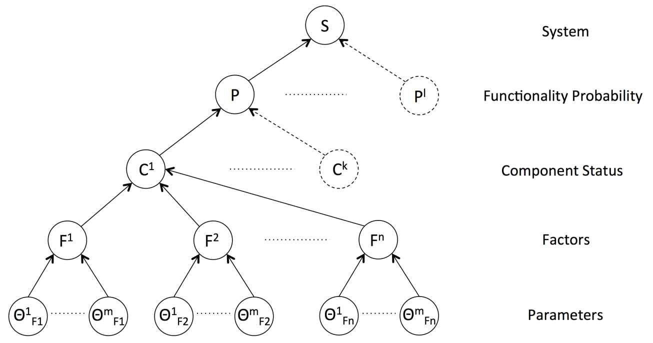

Figure 1: Overview of different levels in SHM Bayesian Network

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 nodes with 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.
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.
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 stress 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 does not have to be only one layer as shown in Figure 1, it can be 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.

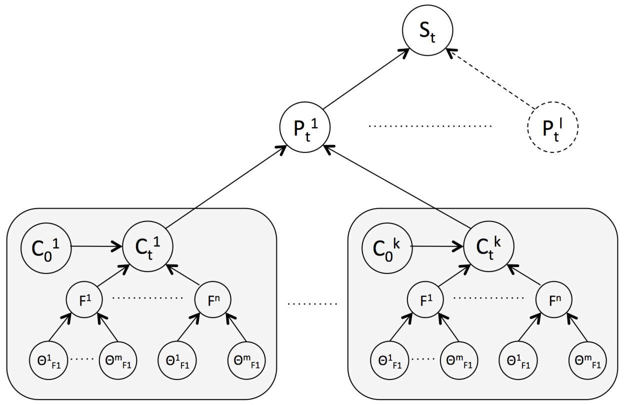

Figure 2: SHM Bayesian network at specific time t.

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:

$$C = C_0[1 \pm A_0 t^m] \tag{2}$$

Where C_0 is the value of C at $t = 0$, A_0 is material/device-dependent coefficient, and m is the power-law exponent. Both A_0 and m are parameters that can be learned from component/device degradation data. Summation (+) is used when the parameter C increases with time, while subtraction (-) is used when the parameter C decreases with time.

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.

$$A_0 = A_0(F_1, \dots, F_n) \tag{3}$$

Therefore, we have:

$$C = C_0[1 \pm A_0(F_1, \dots, F_n)t^m] \tag{4}$$

m and other parameters are considered to be constant for the component/device. Considering a Bayesian network at a time slice of a given system, t is then constant and 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:

$$T_{failure} = \left[\frac{1}{\pm A_0(F_1, \dots, F_n)}\left(\frac{C_{crit} - C_0}{C_0}\right)\right]^{1/m} \tag{5}$$

Since the component parameter 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 BNs, 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 $C_{th,i/j}$.

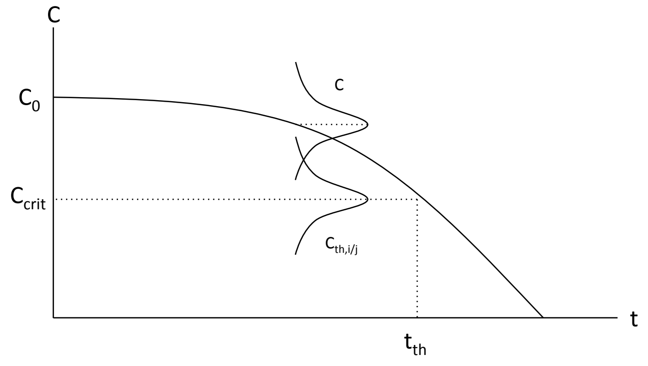

Figure 3: Overlap of probability distribution of component status and its threshold.

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:

$$C_{th,i-1/i} < C_{th,i/i+1} \quad \text{for } i = 2, \dots, n-1 \tag{6}$$

Therefore,

$$P_i = prob\left(C_{th,i-1/i} < C < C_{th,i/i+1}\right) \tag{7}$$

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

$$\begin{aligned} P_i &= \int_{-\infty}^{C_{th,i/i+1}} \int_{C_{th,i-1/i}}^{\infty} \int_{C_{th,i-1/i}}^{C_{th,i/i+1}} p\left(C_{th,i-1/i}\right) \cdot p(C) \\ &\cdot p\left(C_{th,i/i+1}\right) dC\, dC_{th,i/i+1}\, dC_{th,i-1/i} \end{aligned} \tag{8}$$

If there are many component critical 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.

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 $\boldsymbol{\theta}$. 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. $p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_0,...,\boldsymbol{\theta}_t) = p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$ 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.

$$p(C_t) = p(C|C_{t-\Delta t}, \{F_t^1, \dots, F_t^n\}) \tag{9}$$

Given that F_t^i is the average value of factor i between time slice $t - \Delta t$ and t.

Figure 4 shows a two-time-slice representation of a dynamic SHM Bayesian network. Δt should be set according to the system under interest and how often the parameters can be observed, such as frequency of sensor signals.

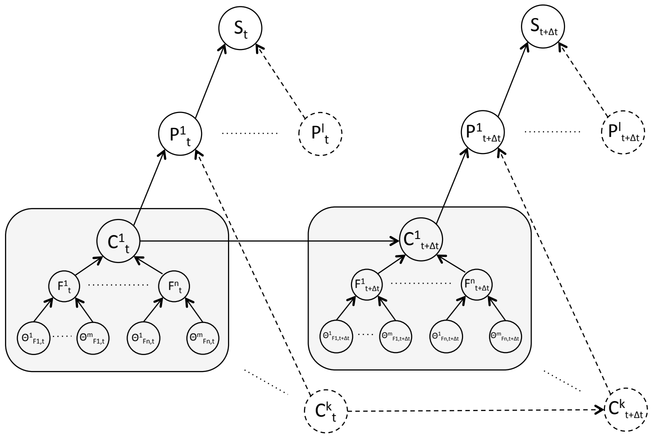

Figure 4: Two-time-slice representation of a dynamic SHM Bayesian network

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 states of a subset of variables, when the other variables (the evidence variables) are observed.

Bayes' rule with continuous variables:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int d\theta\, p(D|\theta)p(\theta)} \tag{10}$$

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:

$$E[p(\theta|D)] \approx \frac{1}{N}\sum_{n=1}^{N} p(\theta^{(n)}|D) \tag{11}$$

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 (Cousins, Chena, & Frisse, 1993) (Dagum & Horvitz, 1993).

MCMC algorithms produce random walks over a probability distribution. By taking a sufficient number of steps in this random walk, the MCMC simulation algorithm visits various regions of the parameter space in proportion to their posterior probabilities. We can, for inferential purposes, summarize the iterates obtained in these random walks much as we would summarize an independent sample from the posterior distribution.

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

$$p(\theta_t|D_{1:t}) = \frac{p(D_t|\theta_t)p(\theta_t|D_{1:t-1})}{\int d\theta\, p(D_t|\theta_t)p(\theta_t|D_{1:t-1})} \tag{12}$$

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) (Chen, 2003).

3. COMPUTATIONAL ALGORITHM

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.

$$|f_t - f_{t-1}| > \epsilon_f \tag{13}$$

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 $\Delta t = t_j - t_i$.

3.1. Pre-computation

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 there is N number of selections per variable, the selections are:

$$C_s = \left\{C^{p_{low}th}, C^{p_{low}+\delta th}, C^{p_{low}+2\delta th}, \dots C^{p_{high}th}\right\} \tag{14}$$

$$\delta = \frac{p_{high} - p_{low}}{N_{selections} - 1} \tag{15}$$

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, 5th percentile, and 95th percentile values.

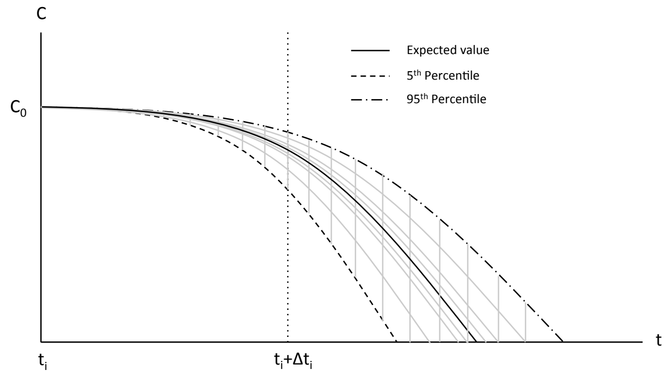

Figure 5: Example component status degradation with 5th percentile, and 95th percentile values.

For this case, the range of possible values grows over time. Therefore, the number of selections should increase proportionally with the range to keep the interval between selected values the same, thus, keep the accuracy of interpolation constant.

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.

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.

$$\Delta t \propto \frac{1}{\Delta C} \tag{16}$$

The time interval between measurements, Δt, should then be inverse proportional to the amount of change of the parameter C. Therefore, 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.

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.

$$N_{pre-computation} = \sum_{j=0}^{T_c/\Delta t} \left(\prod_{i=1}^{n} N_{selections,i,t+(j\Delta t)} \right) \tag{17}$$

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.

$$T_{pre-comp} = N_{pre-comp} \cdot T_{average-per-comp} \tag{18}$$

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 tradeoff 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.

3.2. Dynamic Programming

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 pre-computation 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. First, 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.

Second, 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.

Third, the marginal posterior-probability distributions over the diagnosis nods are determined, 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.

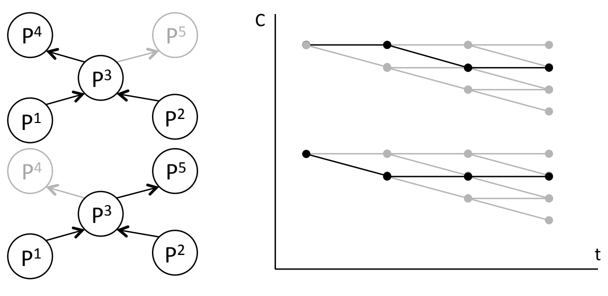

Figure 6: Example cases where dynamic programming reduces number of computations

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.

Since both deterministic and approximate inference were found to be NP-hard (Cooper, 1990) (Dagum & Luby, 1993), 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. Without storing marginal probability distribution results for further computations, all approximate inference computations are required for pre-computation, thus increases computation time exponentially with network's treewidth.

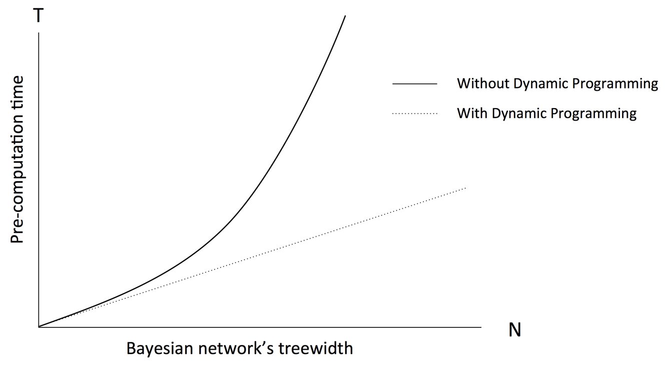

Figure 7: Inference pre-computation time with and without dynamic programming.

3.3. Efficient Dependency Algorithm

In the case that components in the system are dependent to each other because there have common factors, an efficient algorithm is required to maintain the proposed modular component model.

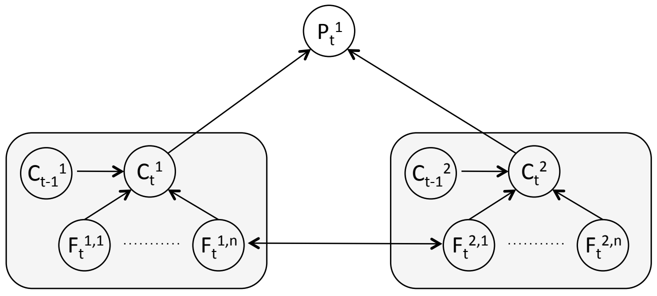

Figure 8: BN of components with common factor

Figure 8 shows an example of 2 components system where both component shares the same factor ($F_t^{1,n}$ and $F_t^{2,1}$). The common approach is to combine the nodes, however, this method is not ideal due to the following reasons:

First, even though the components share the same factor, there is very likely a spatial different between the two components. By combining the nodes, the possibility of decoupling them is eliminated from future analysis. For example, two components are directly in contact of each other and they are assumed to always have the same temperature. There is a chance that in some scenarios, the two components are separated due to an external event or unexpected degradation, the model should be flexible enough to handle this situation.

Second, combining the nodes makes the model no longer modular. Continuous variables inference cannot be done within the component sub-system. This leads to huge increase in complexity and computation time.

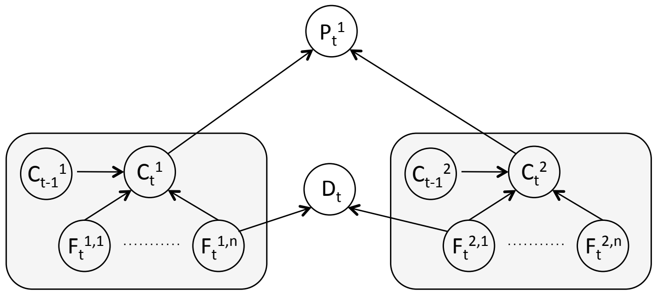

Figure 9: Proposed BN with observable common factor

Let D_t be a node representing the value observed from a detector/sensor used to measure a common factor. If the common factor is observable, factor $F_t^{1,n}$ and $F_t^{2,1}$ can be directly derived from the measurement value of D_t. Therefore, inference calculations for each component stay modular. Figure 9 shows the proposed BN when the common factor is observable.

$$p(C_t) = p(C|C_{t-\Delta t}, \{F_t^1, \dots, F_t^n\}) \tag{19}$$

$$p\left(F_t^{1,n}\right) = p\left(F_t^{1,n}|D_t\right) \tag{20}$$

If the common factor is unobservable, the inference calculation can be done by placing a hidden node D_t as an imaginary measurement node between $F_t^{1,n}$ and $F_t^{2,1}$, shown in Figure 10.

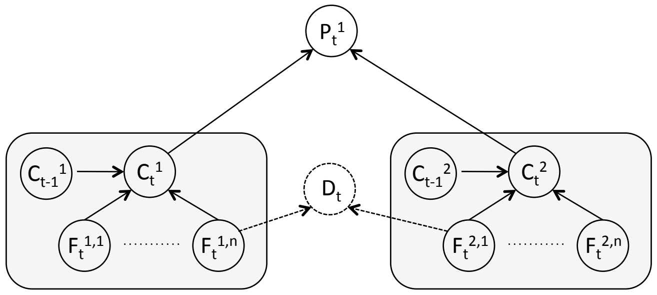

Figure 10: Proposed BN with unobservable common factor

Since we know that $F_t^{1,n}$ and $F_t^{2,1}$ are more likely to have the same value, $p(F_t^{1,n},F_t^{2,1})$ is expected to have a distribution similar to Figure 11.

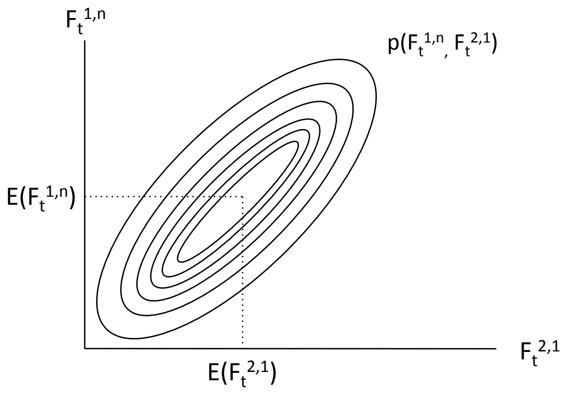

Figure 11: Probability distribution of the common factor.

One method to reduce computation complexity and keep the inference calculation modular is to incorporate pre-computation approximation. Pre-computation generates possible subsets of values of variables according to their probability distribution. For this case:

$$for\ p(P_t^1|C_t^1, C_t^2) \sim \forall i,j\ p\{P_t^1|(C_t^1)_i,(C_t^2)_j\} \tag{21}$$

Therefore, the combination of $\{(C_t^1)_i,(C_t^2)_j\}$ that have higher probability are the ones that the values of $F_t^{1,n}$ and $F_t^{2,1}$ are similar.

$$p(\{(C_t^1)_i,(C_t^2)_j\}|F_t^{1,n} \approx F_t^{2,1}) > p(\{(C_t^1)_i,(C_t^2)_j\}|F_t^{1,n} \not\approx F_t^{2,1}) \tag{22}$$

Using this method, the most probable explanation (MPE) can be derived in real-time from the pre-computation cache.

In summary, this section presented new comprehensive computational algorithms that support the proposed SHM model with dependency between components. The combination of pre-computation and dynamic programming techniques is shown to be a feasible method for real-time system-wide inferences in complex hybrid BN.

4. EXAMPLE APPLICATION

Consider integrated circuits (ICs) with both electromigration (EM) and stress migration (SM) degradations. Let $C^{(1)}$ and $C^{(2)}$ be component status degrading under EM and SM respectively.

$$C^{EM} = C_0{}^{EM}\left[1 - A_0{}^{EM}\left(J^{(e)} - J^{(e)}_{crit}\right)^{r^{EM}} \exp\left(\frac{-Q^{EM}}{K_B T}\right) t^{m^{EM}}\right] \tag{23}$$

$$C^{SM} = C_0{}^{SM}\left[1 - A_0{}^{SM}(L)^{r^{SM}} \exp\left(\frac{-Q^{SM}}{K_B T}\right) t^{m^{SM}}\right] \tag{24}$$

$J^{(e)}$ is the electron current density. $J^{(e)}_{crit}$ is a critical (threshold) current density which must be exceeded before significant EM is expected. L is the tensile stress in the metal for a constant strain.

The BN model of a component affected by EM and SM is shown in Figure 12.

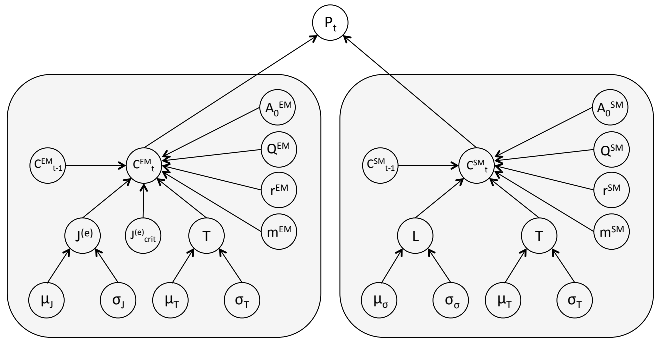

Figure 12: BN of a component with EM and SM degradations

Assume $J^{(e)}$, L, and T are expected to be normally distributed between time *t-1* to *t*,

$$J^{(e)} = \mathcal{N}(\mu_J, \sigma_J), L = (\mu_L, \sigma_L), T = \mathcal{N}(\mu_T, \sigma_T) \tag{25}$$

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.

Consider an Al-alloy under high temperature operation, with current density J = 2×10⁶ A/cm² and at a metal temperature T = 200 °C. Assume an activation energy of Q = 0.8 eV for electromigration and 0.6eV for grain boundary diffusion.

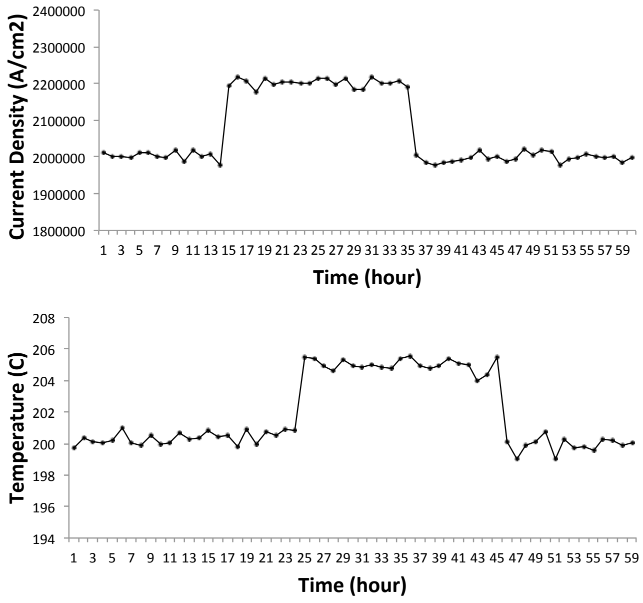

Figure 13: Current density and temperature data set

The current density exponent of n = 2 and stress migration exponent of n=2. Using conservative design approach, assume $J_{crit} = 0$.

The data of current density and temperature, show in Figure 13, is retrieved once per hour during 60 hours of operation. Figure 14 shows an example plot of component degradation under electromigration vs. time at different current density and temperature, including from the data set. Approximate inference of component parameter 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.

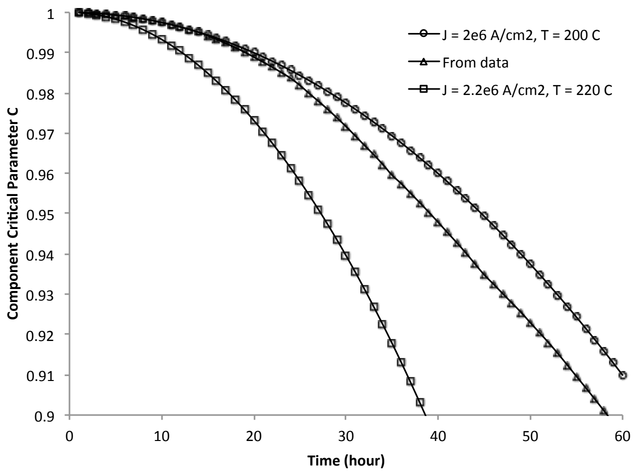

Figure 14: Plot of component parameter C^{EM} vs. time at different *J* and *T*, including from the data set

With traditional BN modeling, both failure modes have temperature as a common factor. Therefore, the component parameters, CEM and CSM, have the same parent node, T. In this case, any approximate inference will require full marginal distribution of both failure mode variables. The amount of time for sampling and computation increases exponentially with the number of variables in the inference calculation. With the proposed technique, the failure modes stay modular and approximate inferences can be achieved at much lower cost because of lower number of variables in the calculation. For this example, approximate inference calculation will only involve parameters of failure mode EM and failure mode SM, but not both of them combined.

This also allows real-time inquiry of the states of degradations of all components in the system without computing full inference of all nodes every time there is new information. In real applications, a sensor on tensile stress may collect data every second, while another sensor on current density collect data every a tenth of a second. The health information of the system can be updated every tenth of a second, without having to performing approximate inference for EM failure mode as often.

Consider a more complex example where the system consists of 50 electrical components that have 2 failure modes. Figure 15 shows a plot of amount of time required as a function of number of failure modes that have the same dependent factor. Assume it takes 1 second to calculate 1,000 iteration of an average marginal distribution computation and an approximate inference requires 10,000 iterations to reach reasonably accurate result. Using the proposed technique, the computation stays roughly the same, while traditional computation time increases exponentially with the number of dependent failure modes.

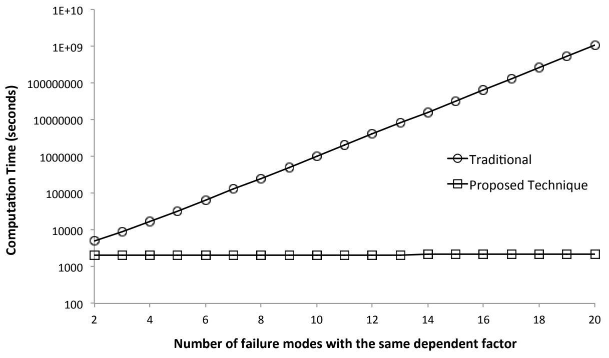

Figure 15: Plot of computation time vs. number of failure modes with the same dependent factor

As mentioned in the previous section, with pre-computation method, the accuracy of inference computation depends mainly on the number of selections of possible values of the variables. Using EM failure mode in the earlier example, Figure 16 shows the average percentage of accuracy against the number of selections of *J* and *T* values.

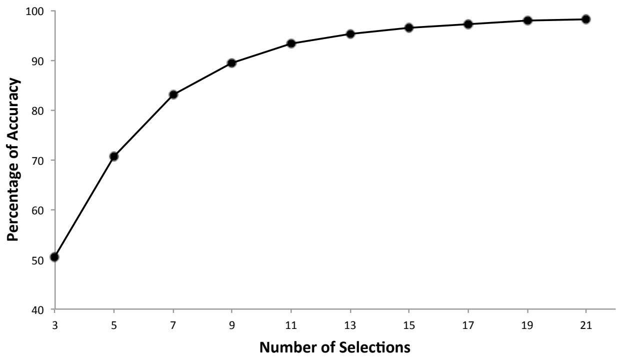

Figure 16: Plot of percentage of accuracy vs. number of selections of *J* and *T* values

The optimal number of selections depends on the accuracy required by the particular application and how much pre-computation time is available. In more complex system, the number of selections should also be varied depending on the sensitivity of each variable.

5. CONCLUSION

This research presents new modeling approach, computational algorithms, and an example application for efficient dependency calculation in on-line System Health Management. Hybrid dynamic Bayesian network modeling were introduced with component-based structure and algorithm 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

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

Boyen, X., & Koller, D. (1998). Tractable inference for complex stochastic processes. *Proceedings of the fourteenth conference on uncertainty in artificial intelligence*.

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

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

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.

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

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

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.

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.

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.

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

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

Lauritzen, S. L., & Jensen, F. (2001). Stable local computation with conditional Gaussian distributions. *Stat. & Comp.,* 11, 191-203.

Lerner, U. N. (2002). *Hybrid Bayesian networks for reasoning about complex systems.* Standford University, Dep. of Comp. Sci. Stanford.

Moral, S., Rumi, R., & Salmeron, A. (2001). Mixtures of truncated exponentials in hybrid Bayesian networks. *ECSQARU 2001. LNCS (LNAI)* (Vol. 2143, pp. 156-167). Springer, Heidelberg.

Murphy, K. (2002). *Dynamic Bayesian networks: representation, inference and learning.* PhD thesis, UC Berkeley, Dept. Computer Science.

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

Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligent,* 29, 241-288.

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 managements society.*

Shenoy, P. P. (2006). Inference in hybrid Bayesian networks using mixtures of Gaussians. *Uncertainty in Artificial Intelligence* (pp. 428-436). AUAI Press, Corvallis.

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.

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

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

BIOGRAPHIES

Chonlagarn Iamsumang is a Ph.D. candidate in Reliability Engineering at A.J. Clark School of Engineering at the University of Maryland - College Park. He received his M.S. in Nuclear Science and Engineering from MIT in 2010 and B.S. in Physics and Engineering from Brown University in 2006. His current research focuses on applying dynamic hybrid Bayesian Network for engineering System Health Management.

Ali Mosleh is Distinguished Professor and holder of the Evelyn Knight Chair in Engineering at the University of California in Los Angeles. Prior to that he was the Nicole J. Kim Eminent Professor Chair in Engineering and the Director of the Center for Risk and Reliability at the University of Maryland. He was elected to the US National Academy of Engineering in 2010, and is a Fellow of the Society for Risk Analysis, and the American Nuclear Society, recipient of several scientific achievement awards, and consultant and technical advisor to numerous national and international organizations, including appointment by President George W. Bush to the U.S. Nuclear Waste Technical Review Board, a position in which he continued to serve in the administration of President Obama. He conducts research on methods for probabilistic risk analysis and reliability of complex systems and has made many contributions in diverse fields of theory and application.

Mohammad Modarres is Minta Martin Professor of Engineering and Director of Reliability Engineering at A.J. Clark School of Engineering at the University of Maryland - College Park. His research areas are probabilistic risk assessment and management, uncertainty analysis and physics of failure degradation modeling. He has over 300 papers in archival journals and proceedings of conferences, four books, one handbook, four edited books and five book chapters in various areas of risk and reliability engineering. He is a University of Maryland Distinguished Scholar. He received his PhD in Nuclear Engineering from MIT in 1980 and M.S. in Mechanical Engineering also from MIT.