Diagnosing Intermittent and Persistent Faults using Static Bayesian Networks

Brian W. Ricks[1], Ole J. Mengshoel[2]

[1] *Carnegie Mellon University, NASA Ames Research Center, Moffett Field, CA 80523 USA*
bwr031000@utdallas.edu

[2] *Carnegie Mellon University, NASA Ames Research Center, Moffett Field, CA 80523 USA*
Ole.J.Mengshoel@nasa.gov

ABSTRACT

Both intermittent and persistent faults may occur in a wide range of systems. We present in this paper the introduction of intermittent fault handling techniques into ProDiagnose, an algorithm that previously only handled persistent faults. We discuss novel algorithmic techniques as well as how our static Bayesian networks help diagnose, in an integrated manner, a range of intermittent and persistent faults. Through experiments with data from the ADAPT electrical power system test bed, generated as part of the Second International Diagnostic Competition (DXC-10), we show that this novel variant of ProDiagnose diagnoses intermittent faults accurately and quickly, while maintaining strong performance on persistent faults.[1]

1 INTRODUCTION

Quick and accurate diagnosis of faults is important in a number of applications, and may have positive implications for safety, performance, economy, and ecology. Clearly, there is a wide range of fault types. For the purpose of this paper, we distinguish between *persistent* and *intermittent* faults. Persistent faults, once they appear, do not disappear, while intermittent faults do. Intermittent faults pose difficulties to diagnostic algorithms, for several reasons including the following. First, if diagnosis is not performed continuously, it might be that the intermittent faults are not present when diagnosis is active. Second, when some intermittent faults are present, they may look like persistent faults. (In fact, intermittent faults often evolve into more serious persistent faults over time.) It is then up to a diagnostic algorithm to catch the intermittent behavior, and also characterize it as such.

Diagnosis of intermittent faults has been considered in several application areas, such as digital circuits and systems (Breuer, 1973; Varshney, 1979), sequential circuits (Savir, 1980), and wireless sensor networks (Khilar & Mahapatra, 2007). Different mathematical methods for handling intermittent fault have been used, including discrete-time Markov chains (Breuer, 1973), continuous-time Markov chains (Su et al., 1978), Hidden Markov models (Ying et al., 2000; Daidone et al. 2006), and Bayesian methods (Abreu et al., 2009) .

In this paper, we investigate the use of multivariate probabilistic techniques, in particular Bayesian networks (Pearl, 1988). With an emphasis on the intermittent case, we present how intermittent and persistent faults are represented in Bayesian networks and used for fault diagnosis. We discuss the *ProDiagnose* diagnostic algorithm (DA), which originally handled persistent faults only (Ricks & Mengshoel, 2009), including how it was expanded to handle intermittent faults. In particular, we combine a static Bayesian network with algorithms using various thresholds that in effect make the distinction between intermittent and permanent faults. Our discussion includes experiments with data from ADAPT (Poll et al., 2007), a real-world electrical power system (EPS) located at the NASA Ames Research Center. These experiments illustrate that our ProDiagnose algorithm, when augmented with intermittent fault handling, performs very well on both intermittent and persistent fault scenarios.

A limitation of some previous research is that it often makes a single-fault assumption (Savir, 1980; Ying et al., 2000) or assumes that fault magnitude is fixed, reflecting a preponderance of previous work on

[1] This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

digital circuits and systems, where Boolean logic {0,1} is used and faults (such as stuck-at-0 or stuck-at-1) have a fixed magnitude. In contrast, ProDiagnose handles situations in which faults have varying fault magnitudes. Due to our use of Bayesian networks, which can decompose large systems containing multiple components that might fail (while maintaining computational feasibility), we are also not restricted to single faults. Instead, we handle multiple faults, which might interact in non-trivial ways, and where some faults are intermittent and other faults might be permanent. In this respect, our research is similar to other work taking a model-based (de Kleer, 2007) or Bayesian approach (Abreu et al., 2009).

The rest of this paper is organized as follows. In Section 2 we discuss Bayesian networks and their application in diagnosis. Section 3 briefly presents intermittent faults, while in Section 4 we discuss diagnosis of intermittent and persistent faults in the ProDiagnose algorithm. In Section 5 we present results from experiments, and Section 6 concludes and sketches areas of future work.

2 PRELIMINARIES

2.1 Bayesian networks

We use a probabilistic approach to diagnosis, based on Bayesian networks (BNs) (Lauritzen & Spiegelhalter 1988; Pearl 1988). A BN is a directed acyclic graph (DAG) in which each node represents a random variable and has a conditional probability distribution associated with it. The directed edges typically represent the causal dependencies between nodes. In this paper, we consider discrete random variables where distributions are represented as conditional probability tables (CPTs). A CPT's size is dependent on the number of states of this node – its cardinality – as well as the number of parent nodes and the cardinalities of these nodes. By *clamping*, or providing evidence $\boldsymbol{e}$ to, a subset $\boldsymbol{E}$ of a BN's nodes $\boldsymbol{X}$, the answer to various interesting probabilistic queries can be computed. Formally, we have $\boldsymbol{e} = \{(E_1, e_1), (E_2, e_2), (E_3, e_3), \ldots\}$ where $\boldsymbol{E} = \{E_1, E_2, E_3, \ldots\}$. These queries include the marginal posterior distribution over one node X, denoted BEL(X, $\boldsymbol{e}$), a set of nodes $\boldsymbol{X}$, denoted BEL($\boldsymbol{X}$, $\boldsymbol{e}$), or most probable explanations over nodes $\boldsymbol{X} - \boldsymbol{E}$, denoted MPE($\boldsymbol{e}$). The answers to these queries can then be used to diagnose the system itself.

While *dynamic* BNs (DBNs) have previously been used for diagnosis, we focus on *static* BNs in this paper. Static BNs are not time-sliced as DBNs are, so any processing that requires knowledge of time must be handled externally to the BN, and then reflected in the evidence $\boldsymbol{e}$. Even though this external processing is a complicating factor, there are also several benefits associated with it. Typically, using a static BN is computationally faster than using a DBN. Also, static BNs are perhaps easier develop and maintain, since they typically are significantly smaller and contain fewer parameters. Finally, when creating a BN-based application one often starts with a static BN, and indeed the BNs discussed in this paper were derived from static BNs that already handled many of the diagnostic tasks required of them (Ricks & Mengshoel, 2009).

2.2 Fault Diagnosis Using Bayesian Networks

Fault diagnosis using BNs first requires the creation of the BN itself, modeled after the physical system that we are interested in diagnosing. For the purpose of this paper, we will use electrical power systems (EPSs) as an example. A typical EPS consists of power sources, a distribution network, sensors, and various loads.

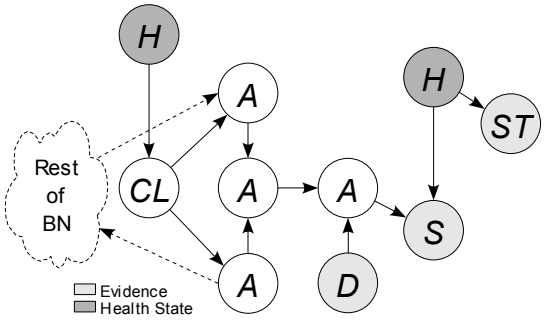

Figure 1: The Bayesian network representation of a load, such as a fan, pump or light bulb, with a sensor.

Figure 1 shows a BN fragment that represents a typical load with a sensor; the sensor is used for monitoring. A DA will clamp, in the BN, evidence $\boldsymbol{e}$ in the form of raw and discretized sensor values. Discretization may be needed for continuous-valued sensors, such as current, voltage temperature, and light sensors, since our BN contains discrete nodes only. In Figure 1, clamping takes place for the sensor node *S*. Evidence is also derived from sensor values, giving clamping of the delta *D* and stuck *ST* nodes.

The health nodes $\boldsymbol{H}$ provide the health state of each sensor and component represented in the BN. A DA, such as ProDiagnose, will use the posterior distribution over $\boldsymbol{H}$, as reflected in answers to BEL($\boldsymbol{H}$, $\boldsymbol{e}$) or MPE($\boldsymbol{e}$) queries, to determine which sensors and components within the EPS are healthy and which are faulty at a specific point in time. We will denote the states for any BN node X by $\Omega(X) = \{x_1, \ldots x_k\}$. For a health node H specifically, we use $\Omega(H) = \{h_1, \ldots h_k\}$. A state $h_i \in \Omega(H)$ is considered the most probable state for H if h_i has the greatest posterior probability of all states in $\Omega(H)$ in BEL(H, $\boldsymbol{e}$) or if h_i is part of a most probable explanation as computed by MPE($\boldsymbol{e}$).

Informally, h_i represents the health state of the sensor or component represented by H in the BN.

In this paper, we often consider only three states, h_1, h_2, $h_3 \in \Omega(H)$: nominal $h_1 = \nu$, persistent fault $h_2 = \pi$, and intermittent fault $h_3 = \iota$. If we have a fault state h_i but do not care whether it is persistent, $h_i = \pi$, or intermittent, $h_i = \iota$, we say $h_i = \varphi$. A nominal state ν represents a healthy sensor or component. A persistent fault state π represents a condition in which a sensor or component has been diagnosed with a fault that does not go away. An intermittent fault state ι describes a condition where a fault comes and goes. Intermittent faults are discussed next.

3 THE NATURE OF INTERMITTENT FAULTS

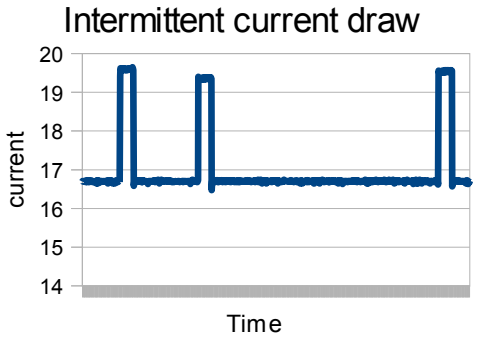

Figure 2: Real-world intermittent fault involving a faulty load drawing too much current from a source intermittently.

An example of intermittent fault behavior is shown in Figure 2. A current of 16-17 A is nominal, so the figure shows three time intervals with faulty behavior. However, we note that fault amplitude varies, time between faulty behavior varies, and time of faulty behavior may vary as well. In addition, there is noise both during the nominal and faulty conditions, and one does not want to classify noise as an intermittent fault. This example reflects how intermittent fault behavior may not follow a "nice" pattern. Consequently, it becomes necessary to develop diagnostic algorithm that can be adjusted to varying real-world conditions to accurately diagnose intermittent faults.

Generally, we consider intermittent faults to follow a square wave pattern. For example, the high parts of the wave can represent points in time with a fault condition, and the low parts can represent a healthy condition. Each high-low pattern represents one cycle in the wave. Still, there can be ambiguity between intermittent faults and other fault types. Specifically, there is often ambiguity when a fault first occurs; it may be unclear whether it is intermittent or persistent.

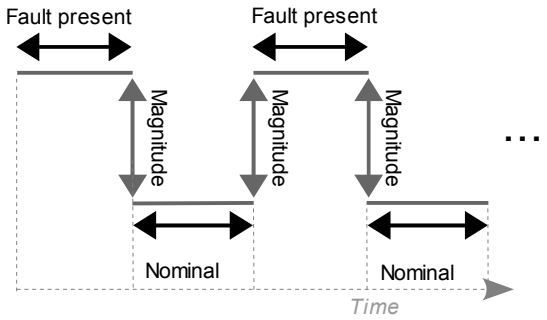

Figure 3: Square-wave pattern of an intermittent fault

4 DIAGNOSING INTERMITTENT FAULTS USING BAYESIAN NETWORKS

Reflecting the difference between intermittent and persistent faults, somewhat different diagnostic technique are needed. In this section we discuss the handling of both fault types in ProDiagnose, but emphasize intermittent faults.

4.1 The ProDiagnose Algorithm

The fundamental structure of ProDiagnose's on-line diagnosis process remains (Ricks & Mengshoel, 2009). Samples and events are input to ProDiagnose, and a set of diagnosis candidates is output. Each diagnosis candidate contains zero or more fault pairs $\{(H_1, \varphi_1), (H_2, \varphi_2), \ldots\}$. Each pair (H_i, φ_i) consists of a health node H_i and a fault state φ_i. The fault state represents a persistent or an intermittent fault.

ProDiagnose employs a probabilistic reasoning engine to compute the candidate set from input evidence $\boldsymbol{E} = \{(E_1, e_1), (E_2, e_2), \ldots\}$. In particular, nodes $\boldsymbol{I} \subseteq \boldsymbol{E}$ play a key role in distinguishing between intermittent and persistent faults. Using the algorithm Count (see Section 4.3), we attempt to distinguish intermittent from persistent faults taking place in $H \in \boldsymbol{H}$ by tracking patterns that are typical of intermittent fault behavior, and clamp the node $I \in \boldsymbol{I}$ accordingly, where $H \rightarrow I$. There are different ways to form $\boldsymbol{I} \subseteq \boldsymbol{E}$ such that evidence for or against intermittent behavior is communicated to the BN, and doing this computation based on the samples and events input to ProDiagnose is a major topic of this paper. We discuss two different ways to integrate $\boldsymbol{I}$ with the remaining evidence $\boldsymbol{E} - \boldsymbol{I}$ (see Section 4.4 and Section 4.5 respectively). In addition to being an intermittent node I, an evidence node $E \in \boldsymbol{E}$ may be a sensor node S, a change node CH, a stuck node ST, or a delta node D (Ricks & Mengshoel, 2009).

Rather than operating directly on a BN, and in order to decrease computation time and make it more predictable, this reasoning engine uses an arithmetic circuit. An arithmetic circuit, which can support both

computation of marginals and most probable explanations, is compiled from a Bayesian network (Darwiche 2003; Chavira & Darwiche 2007). The benefits that arithmetic circuits bring in the areas of speed and predictability are very important in real-time systems such as those on-board aircraft or spacecraft.

4.2 Handling Intermittent Faults

We now discuss ProDiagnose's external processing, which handles intermittent faults and takes place outside the BN, but provides evidence that is input to the BN.

If an intermittent fault is present, ProDiagnose will initially make the more conservative assumption that the fault is persistent Π. As the fault starts to exhibit a square-wave pattern, ProDiagnose's intermittent fault handling attempts to detect it. Once this pattern has been established, a diagnosed fault type may change from persistent, $H = \Pi$ to intermittent, $H = I$.

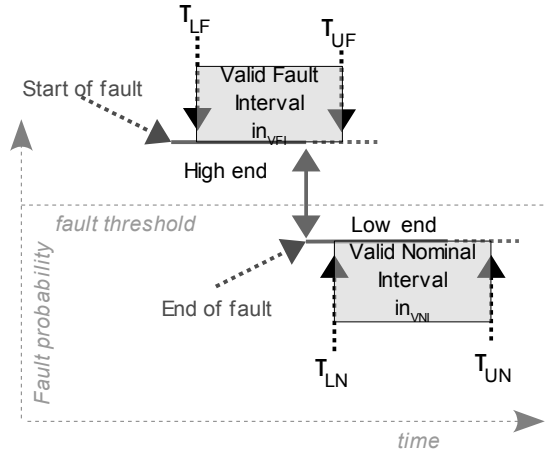

Figure 4: Diagram of the thresholds used to determine tolerances of the wave shape.

Figure 4 illustrates different parameters used in ProDiagnose's tracking of the square-wave patterns seen for intermittent faults (see Figure 3). The tracking of a potentially intermittent fault will only begin if a fault $H = \Pi$ is diagnosed, and if this fault has possible intermittent behavior, $H = I$, associated with it. If the fault is indeed intermittent, then each transition from high end to low end will need to occur within the *valid fault interval*, in_{VFI}, and each transition from low end to high end will need to occur within the *valid nominal interval*, in_{VNI}. These two time intervals represent the valid range in which the upper and lower part of the square wave is considered to fit a square-wave pattern. Each valid interval is defined by a lower threshold, τ_{LF} and τ_{LN} for in_{VFI} and in_{VNI} respectively, and an upper threshold, τ_{UF} and τ_{UN} for in_{VFI} and in_{VNI} respectively.

Our square-wave pattern-matching algorithm, Count, is presented below. The main idea of the algorithm, activated for each $H \in \boldsymbol{H}$, is to count the number of times an intermittent cycle (part of a square wave) is consecutively seen, and flag it to ProDiagnose if this so-called cycle count exceeds a certain threshold.

Count's input parameters, not already discussed, are as follows: cycle_count is the number of (sequential) square-wave pattern cycles seen; cycle_threshold trips to set intermittent state of I once cycle_count ≥ cycle_threshold; is_intermittent = true if intermittent pattern is found, false otherwise; fault_counter counts the number of sequential faults for each cycle; nominal_counter counts the number of sequential nominal states for each cycle; persistent_faults is a list of faults that have an intermittent counterpart.

```
1  Algorithm Count(cycle_count,
   new_cycle, in_VFI, in_VNI, cycle_threshold,
   is_intermittent, fault_counter,
   nominal_counter, persistent_faults, τ_LF,
   τ_UF,  τ_LN, τ_UN)
2      wait until fault F ∈ persistent_faults
       is present
3        if F was present during last call
4          if new_cycle = false AND
              in_VFI = true AND
              in_VNI = true
5             cycle_count ← cycle_count + 1
6           if cycle_count = cycle_threshold
7             is_intermittent ← true
8             return cycle_threshold
9          if new_cycle = false
10           Reset(new_cycle, in_VFI, in_VNI,
             fault_counter, nominal_counter)
11         fault_counter ← fault_counter + 1
12         if τ_LF <= fault_counter <= τ_UF
13           in_VFI ← true
14         else if fault_counter <= τ_UF
15          in_VFI ← false
16         else
17           Reset(new_cycle, in_VFI, in_VNI,
               fault_counter, nominal_counter)
18       else
19         if in_VFI = false
20           Reset(new_cycle, in_VFI, in_VNI,
             fault_counter, nominal_counter)
21           return cycle_count
22       new_cycle ← false
23       nominal_counter ← nominal_counter + 1
24         if  τ_LN <= nominal_counter <= τ_UN
25           in_VNI ← true
26         else if nominal_counter <= τ_UN
27           in_VNI ← false
28         else
29           Reset(new_cycle, in_VFI, in_VNI,
             fault_counter, nominal_counter)
30   return cycle_count

1  Algorithm Reset(new_cycle, in_VFI, in_VNI,
   fault_counter, nominal_counter)
2    new_cycle ← true
3    in_VFI ← false
4    in_VNI ← false
5    fault_counter ← 0
6    nominal_counter ← 0
```

The Count algorithm is only invoked by ProDiagnose when an intermittent fault has not already been established for a health node $H \in \mathbf{H}$, and when we have a (persistent) fault $H = \Pi$, and thus are in the high end of the square wave, above the fault threshold (see Figure 4). There are two cases. The first case is that the fault diagnosis persists. Then, at some point fault_counter > τ_{LF}, and we enter the Valid Fault Interval. The second case is that the fault diagnosis disappears before fault_counter > $\tau_{LF.}$ Now, Count resets, since the fault duration is too short to be considered a valid pattern.

If Count continues past τ_{LF}, the pattern is inside a valid fault interval, reflected by $in_{VFI} \leftarrow$ true. If, while in_{VFI} = true, the fault $H = \Pi$ is no longer diagnosed, then the high end of this cycle is valid and remembered by Count. If, however, Count hits the *upper fault threshold*, τ_{UF}, then the current high end of the square wave is discarded and $in_{VFI} \leftarrow$ false, since a diagnosis that continues beyond τ_{UF}, is considered to be truly persistent. If this happens, Count simply resets and tries to establish a lower fault bound τ_{LF} again.

If a valid high end is found for the current cycle, then ProDiagnose tracks the nominal range (low end), using the *lower nominal threshold*, τ_{LN}, and the *upper nominal threshold*, τ_{UN}. This works in a similar way to tracking the high end. If the same fault is diagnosed either before the lower nominal threshold is reached (while in_{VNI} = false) or if the upper nominal threshold is hit, then Count resets. However, if this same fault is diagnosed while in the valid nominal interval (which in turn implies a valid high end interval), then we have a valid cycle, and Count increments the cycle count.

The overall pattern matching follows this procedure for each cycle sequentially, and if the *cycle threshold* is reached, then ProDiagnose will change the state of the *I* node to intermittent. If an encountered cycle is not valid, then Count will reset with a zero cycle count.

The thresholds mentioned above are currently set manually, based on fault scenarios that ProDiagnose is likely to encounter. As Section 5 will show, these thresholds can be set quite loosely without a significant drop in diagnostic accuracy.

ProDiagnose uses two techniques to input the output of the Count algorithm to the Bayesian network model. These techniques are discussed next.

4.3 Bayesian Network External Diagnosis

External diagnosis involves complete detection of an intermittent fault, outside the BN, and clamping of evidence *I* for this fault in the BN. All of this processing is accomplished through ProDiagnose.

The BN used with the external diagnosis technique handles clamping of the intermittent node *I* as determined by ProDiagnose. The *I* node has two states, nominal and faulty. Once ProDiagnose decides that a fault is intermittent, it will clamp the corresponding *I* to its faulty state, otherwise, it is clamped to its nominal state. The CPT for the intermittent node gives a 100% probability of the health state (parent) being intermittent, and this forces the health node to be intermittent, $H = \mathbf{I}$.

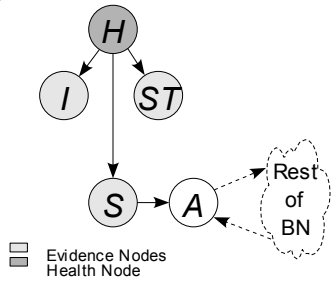

Figure 5: External diagnosis BN model of a typical sensor using intermittent diagnosis via the intermittent node *I*.

Figure 5 shows the BN configuration for a sensor. The *I* node is a direct child of the health node *H*. This allows the faulty state to be clamped with a corresponding probability of an intermittent health state of 100%.

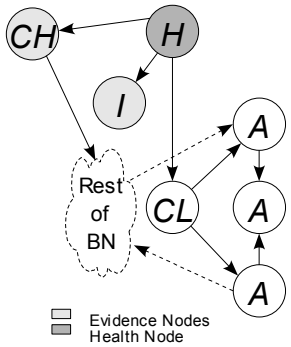

Figure 6: External diagnosis BN model of a typical load (component) using intermittent diagnosis via the intermittent node *I*.

Figure 6 shows the BN configuration for a typical load. The change node *CH* provides extra evidence to the load to determine its health state. Since we do not directly receive information about the state of loads, this is derived from the evidence clamped to the sensor node *S*. Change nodes take the cumulative sum (CUSUM) of a source sensor's values over time and are useful for measuring small changes in a sensor's readings (Ricks & Mengshoel, 2009). For CUSUM, it is preferable to use a sensor that most directly reflects the behavior of a load, for instance a current sensor directly upstream of the load. For many loads, this extra evidence can help detect small changes in the state of

the load itself, which would otherwise be lost due to discretization for the sensors that affect this load's state (and this also saves us from having to add more states to all the sensors in the BN).

4.4 Bayesian Network Internal Diagnosis

Internal diagnosis consists of combining the external processing integrated into ProDiagnose and Bayesian reasoning to determine the intermittent state for any sensor or component. The primary difference, compared to external diagnosis, is that the intermittent state probability is determined by the BN, as opposed to this state being forced from ProDiagnose.

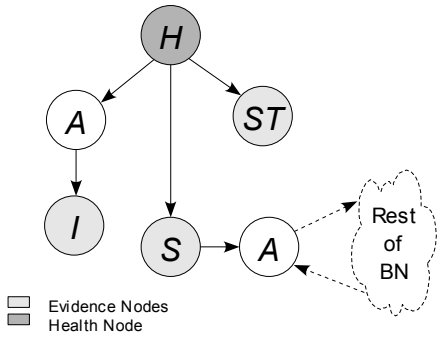

Figure 7: Internal diagnosis BN model of a typical sensor, using intermittent diagnosis via the intermittent node *I*.

With internal diagnosis, ProDiagnose's external processing will still use Count to compute the cycle count. However, unlike the approach discussed in Section 4.3, each cycle count may be represented by a different state within *I*, and consequently this count is used by the BN computation to determine the probability of an intermittent fault state $H = i$. Since ProDiagnose is aware of the number of discrete states for the cycle count in the BN, once the last threshold, corresponding to the final state of *I*, has been tripped, then this state will be clamped for all future inferences to prevent the DA from backtracking due to noisy or bad sensor data. However, this will not force $H = i$; *H*'s state will also depend on other evidence.

State	nominal	faulty
one	0.6	0.05
two	0.39	0.05
twoGreater	0.01	0.9

Table 1: The conditional probability P(*I* | *H*) for an intermittent *I* node in an internal diagnosis BN.

Figure 7 depicts a sensor configuration using a Bayesian network for the internal diagnosis technique. In an internal diagnosis BN, the intermittent *I* node contains states that represents the number of cycles, detected by Count, for an intermittent state. As the number of cycles computed by Count increases, different states of the *I* node are clamped accordingly. The probability of an intermittent state for the health *H* node increases as the cycle count increases (Table 1).

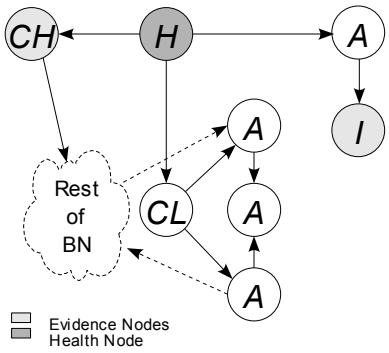

Figure 8: Internal diagnosis BN model of a typical load (component), using intermittent diagnosis via the intermittent node *I*.

Loads in internal diagnosis BNs are modeled in a similar way as in external diagnosis BNs, except with the *I* node structure presented in Figure 8.

5 EXPERIMENTS

ProDiagnose was tested against a set of scenarios taken from the ADAPT testbed, more specifically the DXC-10 DP1 training set. These scenarios consisted of nominal runs, in which no faults were injected into the EPS, faulty runs involving persistent faults, and faulty runs involving intermittent faults. In judging the correctness of the diagnosis by ProDiagnose, two metrics were used: fault detection and fault isolation. Fault detection involves the time that *a* fault from the correct faulty sensor or component is initially diagnosed, and fault isolation involves the time in which the *correct* fault from the correct faulty sensor or component is initially diagnosed. For intermittent fault runs, when the persistent fault (or what looks like one) is initially diagnosed, this will constitute our fault detection. Once ProDiagnose has tracked enough consecutive cycles to trip the *cycle threshold*, then the diagnosis changes to to the intermittent fault, and the moment that this happens constitutes our fault isolation. We also give the total number of misclassified faults and the total number of intermittent classification errors. Each mis-diagnosis of a fault will usually result in 2 classification errors: one for missing the correct

fault, and one for (incorrectly) diagnosing another fault instead.

5.1 Methodology

Multiple configurations of ProDiagnose were used in the experiments. The internal diagnosis technique, which never clamps with 100% intermittent probability, is potentially more powerful in more ambiguous scenarios, since other evidence input to the BN can override evidence that is highly suggestive of an intermittent fault. Because none of the intermittent scenarios available to us showcased this behavior, the results from the internal diagnosis BN were in preliminary experiments almost indistinguishable from those of the external diagnosis BN. Consequently, we opted to report, in this section, on experimental results for the internal diagnosis BN only. Here, our goal is to find a ProDiagnose configuration that minimized both false positives and false negatives in terms of intermittent faults.

5.2 ProDiagnose Configurations

The baseline ProDiagnose refers to the diagnostic algorithm used in the DXC-09 rematch competition. This ProDiagnose revision does not support intermittent fault detection inherently, and so it is to be expected that this baseline will perform poorly on intermittent fault diagnosis. We will refer to the baseline ProDiagnose in the results as ProDiagnose *I*.

ProDiagnose configurations II-V are designed to work with the DXC-10 evaluator, which means that the DA itself is aware of the two different fault types (persistent and intermittent) involved and will diagnose the appropriate fault based on the input data given. This is in stark contrast to the baseline ProDiagnose which is unaware of intermittent fault behavior in general. For DXC-10, the DA itself must make the distinction and diagnose a fault as incorrect for the diagnosis to be considered correct.

ProDiagnose *II* represents a safe implementation of the intermittent pattern matching, with very tight lower/upper fault and nominal thresholds. Ideally ProDiagnose *II* should have a very low false positive rate. This configuration also uses a high cycle thresholds to further decrease the chances of false positive intermittent faults.

ProDiagnose *III* and ProDiagnose *IV* represent configurations of the intermittent pattern matching that use trade-offs between lower/upper fault/nominal thresholds and the cycle threshold. ProDiagnose *III* uses a higher cycle count whereas ProDiagnose *IV* uses looser upper and lower fault/nominal thresholds.

ProDiagnose *V* uses an aggressive configuration for intermittent fault detection. The upper and lower fault and nominal thresholds are set loosely (not quite as loose as configuration IV), and the cycle threshold is set low to maximize the chances of catching an intermittent fault.

5.3 Experimental Results

ProDiagnose Configuration	Classification errors	Intermittent Classification Errors	Detection time (ms)	Isolation time (ms)
I	65	**18**	17969	85444
II	57	**8**	17969	88008
III	55	**6**	17970	83190
IV	53	**4**	17969	72266
V	53	**4**	17970	72266

Table 2: Results from the five different ProDiagnose configurations (external diagnosis)

The results, reported in Table 2, matched what we expected. ProDiagnose I did not correctly diagnose any of the intermittent fault scenarios at all, which resulted in 18 classification errors and the second highest isolation times from the 5 configurations. Configurations IV and V showed the quickest isolation times and had the fewest intermittent classification errors, which is a direct result of the loose intermittent tracking thresholds and low cycle threshold. Configuration IV may be more prone to misdiagnosing persistent faults as intermittent, though, especially in scenarios where the persistent fault disappears momentarily due to sensor noise or other anomalies. Note, however, that none of the configurations produced intermittent false positives. Configuration V, which tightens the intermittent tracking thresholds relative to configuration IV, would be a safe alternative in this regard. The detection times were nearly identical for all the configurations due to the initial fault diagnosis being outside the scope of our intermittent extensions to ProDiagnose.

6 CONCLUSION AND FUTURE WORK

Intermittent faults present a new challenge to diagnostic algorithms that were originally designed with persistent faults in mind. In this paper we have presented the integration of techniques that handle intermittent faults into ProDiagnose. Using data from the ADAPT electrical power system, we have shown experimentally that ProDiagnose now handles both persistent and intermittent faults with high accuracy, and can also differentiate between these two fault types.

Future work includes research into dynamic Bayesian networks (DBNs), which likely would be able to perform intermittent diagnosis with lesser demands

on external ProDiagnose algorithms. Much of this research will be focused on implementing DBN models from current static BNs, as well as possible computational challenges associated with DBNs.

ACKNOWLEDGMENT

This work was, in part, supported by NASA grant NNA08205346R as well as NSF grants CCF-0937044 and ECCS-093197. We would also like to thank Scott Poll, David Garcia, David Nishikawa, Craig Harrison and numerous others at the NASA Ames Research Center for generating the ADAPT data for our experiments, and for helping in many other ways.

REFERENCES

(Abreu et al., 2009) R. Abreu, P. Zoeteweij, and A. J. C. Van Gemund. A new Bayesian approach to multiple intermittent fault diagnosis. In *Proc. of IJCAI-09*, pp. 653-658, Pasadena, CA, 2009.

(Breuer, 1973) M. A. Breuer. Testing for Intermittent Faults in Digital Circuits. *IEEE Trans. Comput*. Vol C-22, No. 3, pp. 241-246, 1973.

(Chavira & Darwiche, 2007) M. Chavira and A. Darwiche. Compiling Bayesian Networks using Variable Elimination. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07*), (Hyderabad, India), pp. 2443-2449, 2007.

(Daidone et al., 2006) A. Daidone, F. Di Giandomenico, A. Bondavalli, and S. Chiaradonna. Hidden Markov Models as a Support for Diagnosis: Formalization of the Problem and Synthesis of the Solution. In *Proc. IEEE Symposium on Reliable Distributed Systems*, pp. 245-256, 2006,

(Darwiche, 2003) A. Darwiche. A Differential Approach to Inference in Bayesian Networks. *Journal of the ACM*, vol. 50, no. 3, pp. 280-305, 2003.

(de Kleer, 2007) J. de Kleer. Diagnosing Intermittent Fault. In *Proc. of 18th International Workshop on Principles of Diagnosis (DX-07)*, 2007.

(Khilar & Mahapatra, 2007) P. M. Khilar and S. Mahapatra. Intermittent Fault Diagnosis in Wireless Sensor Networks. In *Proc. International Conference on Information Technology*, pp. 145-147, 2007.

(Lauritzen & Spiegelhalter, 1988) S. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion), *Journal of the Royal Statistical Society series B*, vol. 50, no. 2, pp. 157-224.

(Pearl, 1988) J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann.

(Poll et al., 2007) S. Poll, A. Patterson-Hine, J. Camisa, D. Garcia, D. Hall, C. Lee, O. J. Mengshoel, C. Neukom, D. Nishikawa, J. Ossenfort, A. Sweet, S. Yentus, I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos. Advanced Diagnostics and Prognostics Testbed. In *Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07)*, (Nashville, TN), pp. 178-185, 2007.

(Ricks & Mengshoel, 2009) B. Ricks and O. J. Mengshoel. The diagnostic challenge competition: Probabilistic Techniques for Fault Diagnosis in Electrical Power Systems. In *Proceedings of 20th International Workshop on Principles of Diagnosis (DX-09)*, (Stockholm, SE), pp. 415–422, 2009.

(Savir, 1980) J. Savir. Detection of Single Intermittent Faults in Sequential Circuits. *IEEE Transactions on Computers*, Volume 29, pp. 673-678,1980,

(Su et al., 1978), S.Y.H. Su, I. Koren, and Y.K. Malaiya. A Continuous-Parameter Markov Model and Detection Procedures for Intermittent Faults. *IEEE Transactions on Computers*, Volume C-27, Number 6, pp. 567 -570, 1978.

(Ying et al., 2000) J. Ying, T. Kirubarajan, K. R. Pattipati, and A. Patterson-Hine. A Hidden Markov Model-Based Algorithm for Fault Diagnosis with Partial and Imperfect Tests. *IEEE Trans. on SMC*, Part C, Vol. 30, No. 4, pp. 463-473, 2000.

(Varshney, 1979) P. K. Varshney. On Analytical Modeling of Intermittent Faults in Digital Systems. IEEE Transactions on Computers. Volume C-28, Number 10, pp. 786 -791, 1979.