Integrating Probabilistic Reasoning and Statistical Quality Control Techniques for Fault Diagnosis in Hybrid Domains

Brian Ricks[1], Craig Harrison[2], Ole J. Mengshoel[3]

[1] *University of Texas at Dallas, Richardson, TX, 75080, USA*

bwr031000@utdallas.edu

[2] *University of Maine, Orono, ME, 04469, USA*

craig.harrison@umit.maine.edu

[3] *Carnegie Mellon University, NASA Ames Research Center, Moffett Field, CA, 80523, USA*

ole.mengshoel@sv.cmu.edu

ABSTRACT

Bayesian networks, which may be compiled to arithmetic circuits in the interest of speed and predictability, provide a probabilistic method for system fault diagnosis. Currently, there is a limitation in arithmetic circuits in that they can only represent discrete random variables, while important fault types such as drift and offset faults are continuous and induce continuous sensor data. In this paper, we investigate how to handle continuous behavior while using discrete random variables with a small number of states. Central in our approach is the novel integration of a method from statistical quality control, known as cumulative sum (CUSUM), with probabilistic reasoning using static arithmetic circuits compiled from static Bayesian networks. Experimentally, an arithmetic circuit model of the ADAPT Electrical Power System (EPS), a real-world EPS located at the NASA Ames Research Center, is considered. We report on the validation of this approach using PRODIAGNOSE, which had the best performance in three of four industrial track competitions at the International Workshop on Principles of Diagnosis in 2009 and 2010 (DXC-09 and DXC-10). We demonstrate that PRODIAGNOSE, augmented with the CUSUM technique, is successful in diagnosing faults that are small in magnitude (offset faults) or drift linearly from a nominal state (drift faults). In one of these experiments, detection accuracy dramatically improved when CUSUM was used, jumping from 46.15% (CUSUM disabled) to 92.31% (CUSUM enabled).

1. INTRODUCTION

Arithmetic circuits (ACs) (Darwiche, 2003; Chavira & Darwiche, 2007), which may be compiled from Bayesian networks (BNs) (Lauritzen & Spiegelhalter, 1988; Pearl, 1988) to achieve speed and predictability, provide a powerful probabilistic method for system fault diagnosis. While arithmetic circuits represent a significant advance in many ways, they can currently only represent discrete random variables (Darwiche, 2003; Chavira & Darwiche, 2007; Darwiche, 2009). At the same time, systems that one wants to diagnose are often hybrid–both discrete and continuous (Lerner, Parr, Koller, & Biswas, 2000; Poll et al., 2007; Langseth, Nielsen, Rumí, & Salmeron, 2009). For example, important fault types such as drift and offset faults are continuous and also induce continuous evidence (Poll et al., 2007; Kurtoglu et al., 2010).

The literature describes two main approaches to handling hybrid behavior in a discrete probabilistic setting: discretization (Kozlov & Koller, 1997; Langseth et al., 2009) and uncertain evidence, in particular soft evidence (Pearl, 1988; Chan & Darwiche, 2005; Darwiche, 2009). Naive discretization performed off-line leads to an excessive number of states, which is problematic both from the point of view of BN construction and fast on-line computation (Langseth et al., 2009). Discretization can be performed dynamically on-line (Kozlov & Koller, 1997; Langseth et al., 2009), however this is inconsistent with this paper's focus on off-line compilation into arithmetic circuits and fast on-line inference. Uncertain evidence in the form of soft (or virtual) evidence (Pearl, 1988) can be used to handle, in a limited way, continuous random variables (Darwiche, 2009). Typically, soft evidence is limited to continuous children of discrete random variables with two discrete states (0/1, low/high, etc.). In addition, the soft evidence approach requires changing the probability parameters on-line, in the AC or BN, and is thus more complicated from a systems engineering or verification and validation (V&V) point of view.

In this paper, we describe an approach to handle continuous behavior using discrete random variables with a small number of states. We integrate a method from statistical quality control, known as cumulative sum (CUSUM) (Page, 1954), with probabilistic reasoning using arithmetic circuits (Darwiche, 2003; Chavira & Darwiche, 2007). We carefully and formally define our approach, and demonstrate that it can diagnose faults that are small in magnitude (continuous offset faults) or drift linearly from a nominal state (continuous drift faults).

Experimentally, we show the power of integrating CUSUM calculations into our diagnostic algorithm ProDiagnose (Ricks & Mengshoel, 2009a, 2009b), which uses arithmetic circuits. We consider the challenge of diagnosing a broad range of faults in electrical power systems (EPSs), focusing on our development of an arithmetic circuit model of the Advanced Diagnostics and Prognostics Testbed (ADAPT), a real-world EPS located at the NASA Ames Research Center (Poll et al., 2007). The experimental data are mainly from the 2010 diagnostic competition (DXC-10) (Kurtoglu et al., 2010). In addition to the challenge of hybrid behavior, this data is sampled at varying sampling frequency, may contain multiple faults, and may contain sensor noise and other behavioral characteristics that are considered nominal behavior of ADAPT. Using the DXC data, we perform a comparison between experiments (i) with CUSUM and (ii) without CUSUM, and find significant improvements in diagnostic performance when CUSUM is used. In fact, ProDiagnose was the best performer in the two competitions making up the industrial track of DXC-09, and a winner of one of the competitions in the industrial industrial track of DXC-10.[1]

In this paper, we extend previous research on the use of CUSUM with BNs and ACs (Ricks & Mengshoel, 2009b) in several ways. Specifically, we use CUSUM for drift faults (previously it was used for offset faults only); provide a more detailed discussion and analysis; and report on experimental results for a new dataset (DXC-10) that contains a broader range of faults, including abrupt faults, intermittent faults, and drift faults. Our CUSUM approach is crucial for handling two types of continuous faults, namely offset faults and drift faults.

The rest of this paper is structured as follows. In Section 2. we introduce concepts related to Bayesian networks, arithmetic circuits, CUSUM, and the fault types we investigate. Section 3. presents integration of CUSUM into our fault diagnosis approach, the ProDiagnose algorithm, and discuss both the modeling and diagnostic perspectives. We present strong experimental results for electrical power system data in Section 4., and conclude in Section 5..

[1]Please see `http://www.dx-competition.org/` for details.

2. Preliminaries

2.1 Bayesian networks and Arithmetic Circuits

Diagnostic problems can be solved using *Bayesian networks* (BNs) (Lauritzen & Spiegelhalter, 1988; Pearl, 1988; Darwiche, 2009; Choi, Darwiche, Zheng, & Mengshoel, 2011). A Bayesian network is a directed acyclic graph (DAG) where each *node* in the BN represents a *discrete* random variable,[2] and each edge typically represents a causal dependency between nodes. Distributions for each node are represented as *conditional probability tables* (CPTs). Let $\mathbf{X}$ represent the set of all nodes in a BN, $\Omega(X) = \{x_1, \ldots, x_m\}$ the states of a node $X \in \mathbf{X}$, and $|X| = |\Omega(X)| = m$ the cardinality (number of states). The size of a node's CPT is dependent on its cardinality and the cardinality of each parent node. By taking a subset $\mathbf{E} \subseteq \mathbf{X}$, denoted the evidence nodes, and *clamping* each of these nodes to a specific state, the answers to various probabilistic queries can be computed. Formally, we are providing *evidence* $\mathbf{e}$ to all nodes $E \in \mathbf{E}$, in which $\mathbf{E} = \{E_1, E_2, \ldots, E_n\}$, $\mathbf{e} = \{(E_1, e_1), (E_2, e_2), \ldots, (E_n, e_n)\}$, and $e_i \in \Omega(E_i)$ for $1 \leq i \leq n$ and $n \leq m$. Probabilistic queries for BNs include the marginal posterior distribution over *one* node $X \in \mathbf{X}$, denoted BEL$(X, \mathbf{e})$, over a *set* of nodes $\mathbf{X}$, denoted BEL$(\mathbf{X}, \mathbf{e})$, and most probable explanations over nodes $\mathbf{X} - \mathbf{E}$, denoted MPE$(\mathbf{e})$.

While Bayesian networks can be used directly for inference, we compile them to *arithmetic circuits* (ACs) (Chavira & Darwiche, 2007; Darwiche, 2003), which are then used to answer BEL and MPE queries. Key advantages to using ACs are speed and predictability, which are important for resource-bounded real-time computing systems including those found in aircraft and spacecraft (Mengshoel, 2007; Ricks & Mengshoel, 2009b, 2010; Mengshoel et al., 2010). The benefits of using arithmetic circuits are derived from the fact that BEL computations, for example, amount to simple addition and multiplication operations over numbers structured in a DAG. Compared to alternative approaches to computation using BNs, for example join tree clustering (Lauritzen & Spiegelhalter, 1988) and variable elimination (Dechter, 1999), AC computation has substantial advantages in terms of speed and predictability, even when implemented in software as done in this paper and previously (Chavira & Darwiche, 2007; Darwiche, 2003; Ricks & Mengshoel, 2009b; Mengshoel et al., 2010). The fundamental limitation of ACs is that they may, in the general case, grow to the point where memory is exhausted. In particular, this is a problem in highly connected BNs. The BNs investigated in this paper, as well as in similar fault diagnosis applications, are typically sparse (Mengshoel, Poll, & Kurtoglu, 2009; Ricks & Mengshoel, 2009a, 2009b; Mengshoel et al., 2010; Ricks & Mengshoel, 2010), and memory consumption turns out not to be a problem.

[2]Continuous random variables can also be represented in BNs, however in this article we focus on the discrete case.

2.2 Cumulative sum (CUSUM)

Cumulative sum (*CUSUM*) is a sequential analysis technique used in the field of statistical quality control (Page, 1954). CUSUMs can be used for monitoring changes in a continuous process' samples, such as a sensor's readings, over time. Let $\delta_p(t)$ represent the CUSUM of process p at time t. Then, taking $s_p(t)$ to be the unweighted sample (or sensor reading) from process p at time t, we formally define CUSUM as:

$$\delta_p(t) = [s_p(t) - s_p(t-1)] + \delta_p(t-1). \quad (1)$$

If $\delta_p(t)$ crosses a *threshold*, denoted as $v(i)$, a change in process p's samples can be recorded. These thresholds represent points at which an interval change occurs in p—a transition from one interval to another (indicating a change from nominal to faulty in a closely related health node, for example). In other words, thresholds provide discretization points for our continuous CUSUM values. It is assumed that a process p starts out with $\delta_p(t)$ such that $v(i-1) \leq \delta_p(t) < v(i)$ for some i. Formally, an interval change for process p occurs when, at any time t, $\delta_p(t) < v(i-1)$ or $\delta_p(t) \geq v(i)$, in which $v(i-1)$ and $v(i)$ are a pair of thresholds at levels $i-1$ and i, respectively. Thresholds themselves are independent of time, in that they can be crossed at any time t. Values of thresholds are configurable, and obtained through experimentation.

Often, a set of thresholds for a sensor will only contain two thresholds. For these cases, we refer to $v(0)$ as the *lower threshold* and $v(1)$ as the *upper threshold*. This implies that the interval set of p has a cardinality of 3. The initial CUSUM value ($\delta_p(0)$) with respect to $v(0)$ and $v(1)$ will be $v(0) \leq \delta_p(0) < v(1)$. A lower and upper threshold are thus used to trigger an interval change if $\delta_p(t)$ ventures outside a nominal range bounded by the interval $[v(0), v(1))$ of the real line $\Re$.

2.3 Continuous Offset and Drift Faults

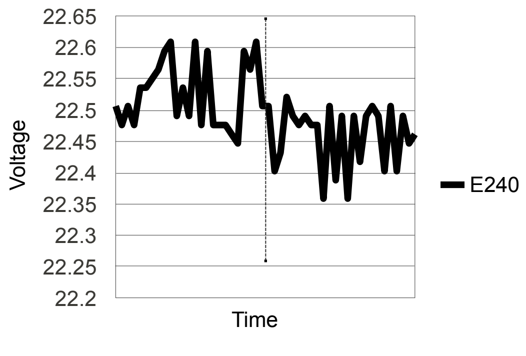

Figure 1. Graph illustrating an abrupt battery degradation over a span of 10 seconds, manifested as a voltage drop for sensor E240. The vertical dotted line on the graph indicates when the fault occurred.

We consider a system consisting of parts.[3] For example, a system might be an electrical power system, and parts might be a battery, a wire, a voltage sensor, a temperature sensor, etc. Let $p(t)$ denote a measureable property of a part at time t. We now consider how a persistent fault, which is the emphasis of this paper, takes place.[4] Let $p_n(t)$ denote the value of the property *before* fault injection, and $p_f(t)$ denote the value of the property *after* fault injection. More formally, let t^* be the time of fault injection. We then have:

$$p(t) = \begin{cases} p_n(t) \text{ for } t < t^* \\ p_f(t) \text{ for } t \geq t^* \end{cases}.$$

We can now formally introduce continuous offset and drift faults. A simple model for a *continuous (abrupt) offset fault* at time t is defined as follows (Kurtoglu et al., 2010):

$$p_f(t) = p_n(t) + \Delta p, \quad (2)$$

where Δp is an arbitrary positive or negative constant representing the offset magnitude. In other words, we do not know the values of Δp or t^* ahead of time, however once Δp is injected at time t^*, it does not change.

A key challenge is that Δp will be small for small-magnitude offset faults. For example, degradation of a battery (Figure 1), is often very small in magnitude (low voltage drop). When discussing the change in a sensor reading of a property, the notation Δs_p (sensed offset) rather than Δp (actual offset) is used. Sensor noise, while not reflected in (2), can mask the fault almost completely. In Figure 1, the magnitude of sensor noise would make diagnosis very difficult without first filtering data. An orthogonal issue is the large number of states needed in a discrete random variable, if a naive approach is used by representing a large number of offset intervals directly, which we would like to avoid.

A simple model for a *continuous drift fault* for a process p at time t is defined as follows (Kurtoglu et al., 2010):

$$p_f(t) = p_n(t) + m(t - t^*), \quad (3)$$

in which m is the slope. For example, a drifting sensor could output values that start gradually increasing at a roughly linear rate (Figure 2). As seen in Figure 2, drift faults may not be so obvious at first, due to sensor and other noise not reflected in (3). In a static Bayesian environment, the lack of time-measurement may make these faults appear as continuous offset faults. Not only would that diagnosis be incorrect, but the time of diagnosis may be quite a while after the initial fault, depending on the time t from t^* the drifting value crossed a threshold $v(i)$.

A major goal of our research is to correctly and quickly diagnose continuous offset and drift faults while minimizing the number of discretization levels of random variables.

[3]A "part" is either a "component" or a "sensor" according to the terminology used in this paper. In the DXC-10 documentation (Kurtoglu et al., 2010), the term "component" rather than "part" is used.

[4]The case of intermittent faults has been discussed previously (Ricks & Mengshoel, 2010).

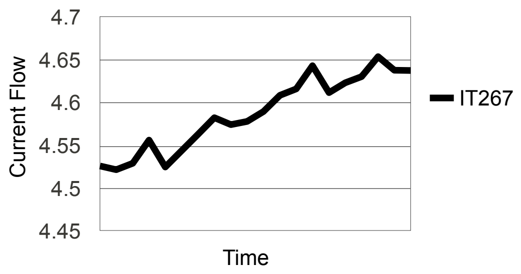

Figure 2. Graph of continuous drift fault behavior for a current sensor IT267 during a 10 second time span.

3. Probabilistic Diagnosis Algorithm

The ProDiagnose diagnostic algorithm takes input from the environment, translates it into evidence, and computes a posterior distribution. The posterior distribution is then used to compute a diagnosis (Ricks & Mengshoel, 2009a, 2009b; Mengshoel et al., 2010; Ricks & Mengshoel, 2010). In this section we first summarize how ProDiagnose computes diagnoses (Section 3.1) and the types of BN nodes it uses (Section 3.2). We then discuss how sensor readings, CUSUM, and Bayesian discretization fit together (see Section 3.3), and finally how ProDiagnose handles continuous offset and drift faults by means of CUSUM techniques (Section 3.4 and Section 3.5).

3.1 The ProDiagnose Diagnostic Algorithm

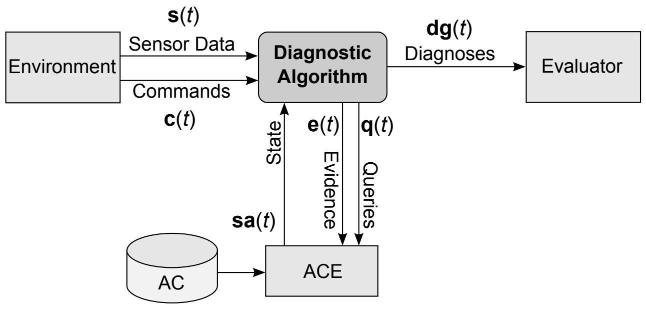

Figure 3. The ProDiagnose DA architecture.

The ProDiagnose diagnostic algorithm (DA) integrates the processing of environmental data (Figure 3) with an inference engine and post-processing of query results. At the highest level, the goal of ProDiagnose is to compute a diagnosis $\mathbf{dg}(t)$ from sensor data $\mathbf{s}(t)$ and commands $\mathbf{c}(t)$. Input from the environment takes the form of sensor data, $\mathbf{s}(t)$, and commands, $\mathbf{c}(t)$. ProDiagnose performs diagnosis when it receives sets or subsets of sensor data, which is expected at regular sample rate(s). It uses $\mathbf{s}(t)$ and $\mathbf{c}(t)$ to generate evidence, $\mathbf{e}(t)$, reflecting the state of the system, which is then passed to the AC inference engine (ACE). The generation of $\mathbf{e}(t)$ from $\mathbf{s}(t)$ and $\mathbf{c}(t)$ is non-trivial, and includes the use of CUSUM as discussed below. The health state of the system, $\mathbf{sa}(t)$, is returned in response to a probabilistic query $\mathbf{q}(t)$, which is either an MPE or BEL query, MPE($\mathbf{e}(t)$) or BEL($\mathbf{H}, \mathbf{e}(t)$) respectively. ProDiagnose then uses $\mathbf{sa}(t)$ to generate a diagnosis of the system, $\mathbf{dg}(t)$, by extracting the faulty states (if any) of the BN nodes $\mathbf{H}$ that represent the health of the system being diagnosed. The algorithms used to compute CUSUM and perform offset and drift diagnosis (see Equation 5 and Algorithm 1) are called from within ProDiagnose (Figure 3) as further discussed in the following subsections.

One strength of ProDiagnose is its configurability. The BN representation of each part (a physical component or sensor) in an environment is configured individually, and this data is used by ProDiagnose to initially calibrate itself to the environment and guides its behavior when receiving $\mathbf{s}(t)$ or $\mathbf{c}(t)$. In addition, ProDiagnose is controlled by several global parameters, including:

- Diagnosis Delay, t_{DD}: Measured in milliseconds, this parameter gives the delay to start diagnosis output $\mathbf{dg}(t)$. In other words, $\mathbf{dg}(t)$ is empty for $t < t_{DD}$. Diagnosis delay is used at the beginning of environment monitoring, and is useful to filter out false positives (often due to transients) during the initial calibration process.
- Fault Delay, t_{FD}: This parameter delays the output of a new diagnosis (for a short while). Suppose that $\mathbf{sa}(t_i)$ contains, for the first time, a fault state f for a health node H. Then we hold off until time t_j, such that $t_j - t_i > t_{FD}$, to include f in $\mathbf{dg}(t_j)$. In many environments, one can get spurious diagnoses because of system transients (perhaps due to mode switching, perhaps due to faults), and this is a way to filter them out.

3.2 Bayesian Network Structures

A Bayesian network model of a system, for example an EPS, consists of structures modeled after physical components of the EPS (Ricks & Mengshoel, 2009a, 2009b). We discuss the following BN node types:

- $S \in \mathbf{S}$: All sensors utilize a *sensor* node S to clamp a discretized value of the physical sensor's reading $s(t)$. S nodes typically consist of three states, and thus have lower and upper thresholds.
- $H \in \mathbf{H}$: Consider $H \in \mathbf{H}$, namely a *health* node H. A health state $h \in \Omega(H)$ is output in $\mathbf{dg}(t)$, based on the result of BEL or MPE computation, to describe the health state of the component represented by H. We assume that $\Omega(H)$ is partitioned into nominal (or normal) states $\Omega_n(H)$ and faulty (or abnormal) states $\Omega_f(H)$. Only a faulty state $h \in \Omega_f(H)$ is output in $\mathbf{dg}(t)$, This is done for all $H \in \mathbf{H}$.
- $CH \in \mathbf{CH}$: A *change* node CH is used to clamp evidence for the purpose of change detection. CH nodes have a varying number of thresholds, depending on the purpose of the node (see Section 3.4).

- $DR \in \mathbf{DR}$: The *drift* node DR is used to clamp evidence about drift-type behavior. Typically, DR nodes use four thresholds, though this is configurable (see Section 3.5).
- $A \in \mathbf{A}$: An *attribute* node A is used to represent a hidden state, such as voltage or current flow.

Note that there are other nodes types besides S, CH, and DR nodes used to input evidence $\mathbf{e}(t)$ (Ricks & Mengshoel, 2009a, 2009b; Mengshoel et al., 2010; Ricks & Mengshoel, 2010). Since they are not the focus in this paper, we represent these by nodes $E_1, \ldots, E_n$ in Figures 6 and 9.

3.3 Sensors, CUSUM, and Bayesian Network Discretization

To help filter out sensor noise, we take a weighted average of the raw sensor readings, and these weighted averages are then the basis for CUSUM computation. Let $\bar{s}_p(t)$ be the weighted average of readings $\{s_p(t-n), \ldots, s_p(t)\}$ for sensor p at time t. Specifically, we have:

$$\bar{s}_p(t) = \sum_{i=0}^{n} s_p(t-i)w(t-i), \tag{4}$$

in which $s_p(t-i)$ is the raw sensor reading and $w(t-i)$ is the weight at time $t-i$. The summation is over all sensor readings from time t to time $t-n$. In other words, we keep a history of the past n sensor readings. The values of n and of all weights $\{w(t-n), \ldots, w(t)\}$ are configurable and set based on experimentation.

The weighted sensor averages in (4) can be used when calculating CUSUM (Ricks & Mengshoel, 2009b), and (1) can be modified accordingly:

$$\bar{\delta}_p(t) = [\bar{s}_p(t) - \bar{s}_p(t-1)] + \bar{\delta}_p(t-1), \tag{5}$$

in which $\bar{\delta}_p(t)$ is the weighted average CUSUM of sensor p at time t. Weighted averages help to smooth spikes in sensor readings that could otherwise lead to false positives or negatives, and (5) is the CUSUM variant used in PRODIAGNOSE.

Figure 4 shows, for voltage sensor E240, the weighted CUSUM values $\bar{\delta}_p(t)$ overlaid with the raw sensor readings for the same time period. A downward trend after the offset fault occurrence is visually seen when looking at the CUSUM values, which makes setting appropriate thresholds to catch the fault possible. After the lower threshold, $v(0)$, is reached, the voltage sensor's CH node changes state. The CUSUM values in Figure 4 are calculated by keeping a history of the past $n = 6$ sensor readings, for any time t.

In this paper, CUSUM intervals are mapped to BN node states in a natural way. If we have k CUSUM intervals defined on the real line $\Re$ (so $k-1$ thresholds), then the corresponding BN evidence node E has k states $\Omega(E) = \{e_1, \ldots, e_k\}$ also. If a CUSUM $\delta_p(t)$ crosses a threshold into an interval $[v(i), v(i+1))$, the corresponding BN evidence node E_p has a corresponding transition into state e_{i+1}.

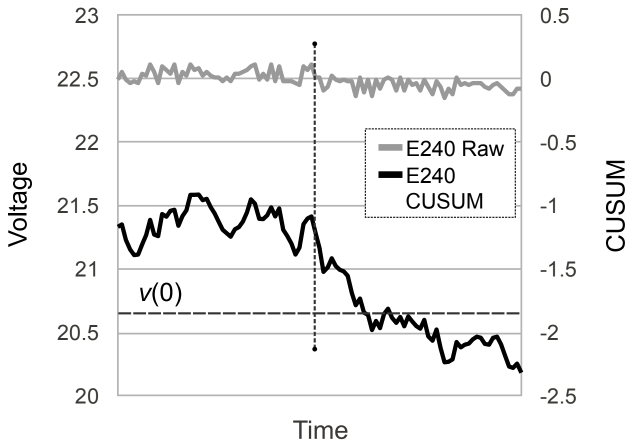

Figure 4. Graph illustrating raw voltage sensor readings, for sensor E240, and corresponding CUSUM values over a time span of 10 seconds. The vertical dotted line on the graph indicates when a very small offset fault occurred, and the horizontal dashed line represents the lower threshold, $v(0)$. The E240 CUSUM shows a very distinct downward trend after the fault.

We now describe CUSUM's characteristics and benefits, taking (5), the discretization, and Figure 4 as starting points. First, CUSUM *amplifies* a small offset, along the y-axis, by making it larger such that it becomes easier to create a threshold for and detect. Second, CUSUM *normalizes* by shifting offsets that can take place at different y-values to a normalized y-value, such that offsets can be detected using thresholds that apply to many y-values. Please see Figure 5 for an example. There is a clear impact on IT240, which can easily be detected with upper and lower thresholds, while the impact on IT267 is minimal since this sensor is downstream of a compensating inverter.

CUSUM's normalization works in combination with weighting the sensor values to give better discretization points. A key point here is that our algorithm calibrates in the beginning of a scenario. The algorithm computes a zero (or nominal) line based on initial sensor readings, and does not flag diagnoses. To the left in Figure 5, both IT240 and IT267 are close to this zero line. This zero line can help compensate for early transients, which may trigger diagnoses, but more importantly, it makes the nominal value of a sensor something we do not need to know ahead of time. Without CUSUM, we would have to know this nominal value ahead of time, which becomes difficult as a system naturally ages. With CUSUM, we use the first time period t_{DD} of a scenario to figure this out (calibration). After t_{DD}, CUSUM is relative number to the nominal reading, with zero being the nominal (weighted) value of the sensor. After the fault injection, we see in Figure 5 that IT267 CUSUM stays well above the $v(0)$ lower thresh-

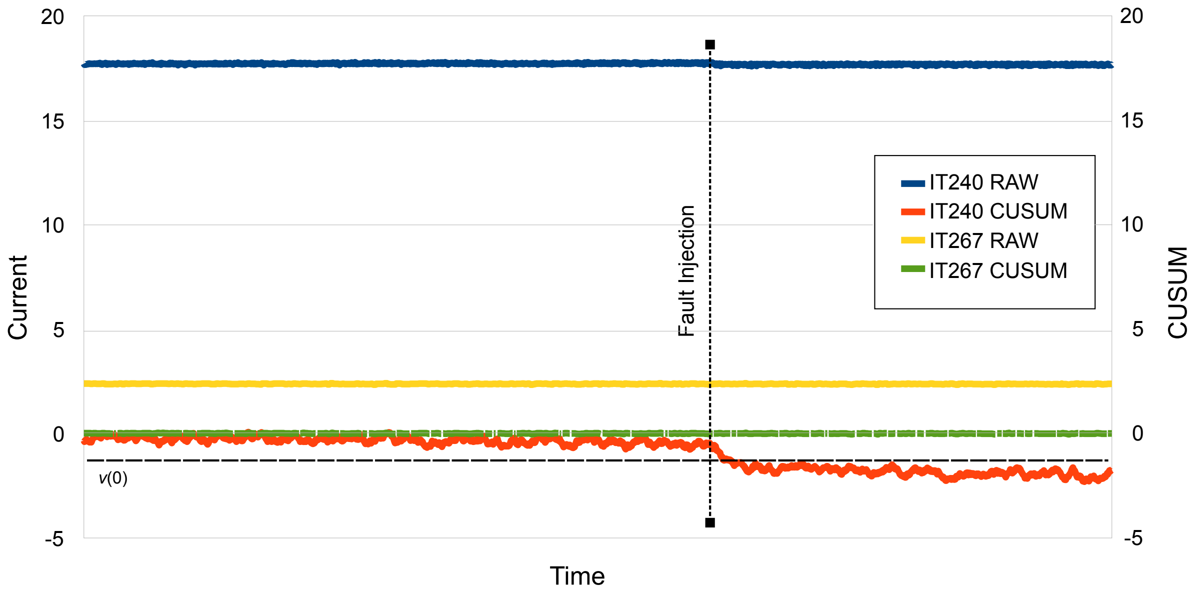

Figure 5. Graph illustrating the impact of an abrupt battery degradation offset fault on two current sensors. One sensor is immediately downstream from the battery (IT240), while the other sensor is farther away, downstream of an inverter (IT267). The vertical dotted line on the graph indicates when the fault occurred during the 3 minute time inteval shown.

old, while IT240 CUSUM drops below $v(0)$.

3.4 Handling Continuous Offset Faults: Change Nodes and CUSUM

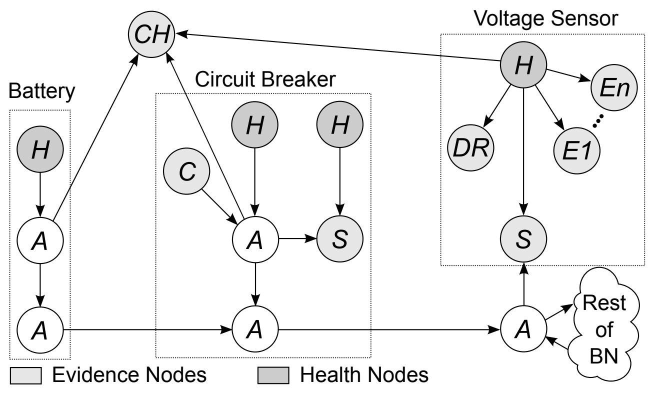

Figure 6. Bayesian network structure for a battery. The CH node provides additional evidence to the health state of the battery.

Suppose that we have a battery, a circuit breaker, a voltage sensor, and a load (say, a light bulb or a fan) connected in series. We assume that all these components and loads may fail, in different ways, and a realistic EPS will contain additional components and sensors that may also fail. Now consider the case of a continuous offset fault in the battery. There is the challenge of diagnosing an offset fault, which might be very small, using a discrete BN with relatively few states per node. In addition, there is the challenge of detecting an offset fault

Change_voltage_e140				
CH				
nominal	drop	A (Battery)	A (Breaker)	H (Sensor)
0.95	0.05	enabledHigh	closed	healthy
0.05	0.95	enabledLow		
0.5	0.5	disabled		
0.5	0.5	enabledHigh	open	
0.5	0.5	enabledLow		
0.5	0.5	disabled		
0.5	0.5	enabledHigh	closed	(not healthy)
0.5	0.5	enabledLow		
0.5	0.5	disabled		
0.5	0.5	enabledHigh	open	
0.5	0.5	enabledLow		
0.5	0.5	disabled		

Table 1. Conditional probability table (CPT) of the CH node from Figure 6. Each row shows the probabilities for the CH node's *nominal* and *drop* states (first two columns) for each combination of states from the battery's A node, circuit breaker's A node, and voltage sensor's H node (columns 3-5). Since the probabilities remain identical for all rows when the H node is unhealthy, we simplified the table by combining all these unhealthy states into *(not healthy)*.

in an upstream component (the battery) using a downstream sensor (the voltage sensor). Clearly, we want to retain the capabilty of diagnosing other types of faults (see (Ricks & Mengshoel, 2009a, 2009b; Mengshoel et al., 2010; Ricks & Mengshoel, 2010)) in the battery, the circuit breaker, and the voltage sensor.

Figure 6 illustrates how we meet these challenges using BNs and CUSUM computation. Specifically, Figure 6 shows the BN representation of a battery, a circuit breaker, and a voltage sensor connected in series. Under nominal conditions, power flows from the battery, through the circuit breaker and voltage sensor, and to one or more loads (not depicted). Traditionally, voltage sensor readings $s_p(t)$ for a sensor p lie on the real line $\Re$, but can be discretized so they end up being in one of the BN states $\Omega(S) = \{voltageLo, voltageHi, voltageMax\}$ for BN node S. The resulting set of thresholds will contain two levels for the S node in the BN (the lower and upper threshold). In the case in which the current state is $S = voltageHi$, the transition from this state to another occurs when $s_p(t) < v(0)$ or $s_p(t) \geq v(1)$ (see Section 2.2). If the current state is $S = voltageLo$, a state change would occur when $s_p(t) \geq v(0)$.

While the simple discretization approach discussed above is sufficient in many cases, in other cases it does not help, and this is the case for example for offset faults. If Δp (Equation 2) is small such that no state change among $\Omega(S)$ occurs, then it is possible that an offset fault may go undiagnosed.

To handle this problem, we use CH nodes and CUSUM. Intuitively, the purpose of the CH node in Figure 6 is to provide additional information about the battery, and specifically continuous offsets in sensor readings, while at the same time be influenced by the circuit breaker and voltage sensor. Offsets in sensor readings (Δs_p) that are too small in magnitude to cross a threshold are handled using CH nodes and CUSUM techniques. Evidence $e(t)$ that is clamped to a CH node is derived from the readings $s_p(t)$ of a sensor assigned to it, called the *source*. The readings from this source sensor are converted to CUSUM values, which are then discretized and clamped to the CH node.

In Figure 6, it is the battery, and specifically its health node H, that the CH node should influence when clamped with evidence $e(t)$ that indicates a continuous offset fault. We will call the battery the *target*. To understand in more detail how this BN design works, consider in Figure 6 the parent nodes of the CH node. Both the circuit breaker and voltage sensor parent nodes have evidence nodes as children or parents. This is not true, however, for the battery's A node, which is also a parent node of the CH node. This design makes sure that evidence $e(t)$ clamped to CH has less influence on the health nodes of the circuit breaker and voltage sensor compared to the influence on the battery's health node. And it is after all battery health, specifically offset faults, that we are targeting with this CH node.

More generally, any component between a source sensor and a target could affect the relevance of the CH node's evidence $e(t)$ for the target. For these intermediary components, it is the physical state that we are concerned about. For a circuit breaker, this is either *open* or *closed*. An *open* state should increase the probability that any voltage drop downstream is the result of this circuit breaker state and not due to a degradation of the battery. The physical state of these components are usually represented as the state of an A node that belongs to the component structure, as in Figure 6. These A nodes will have a parent-child relationship to the CH node that is similar to the relationship of the source sensor's H node.

Table 1 shows the CPT for the CH node depicted in Figure 6. Notice that for all rows in which the voltage sensor (column 5) is *not healthy*, the probabilities for all CH node states are equal. Therefore, despite the discretized CUSUM value that gets clamped to the CH node, the impact on the posterior distribution of the battery's health node H (from the additional evidence provided by the CH node) will be very small in this case.

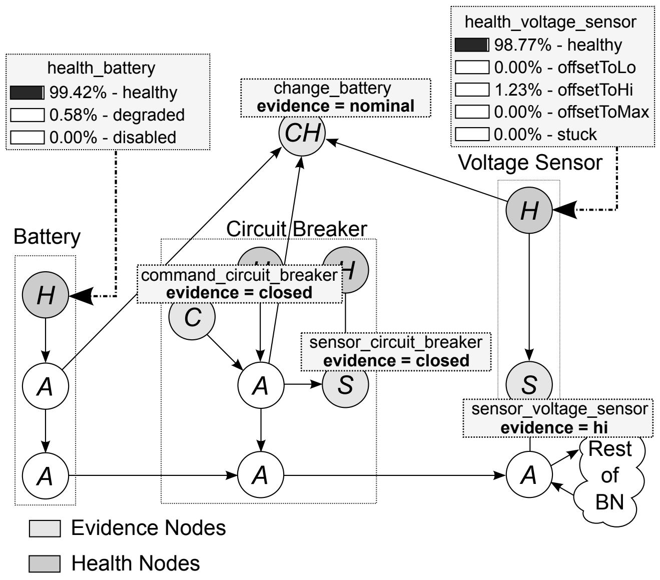

Figure 7. The marginal posterior distributions for the H nodes of the voltage sensor and battery from Figure 6 when no fault is present.

A CH node used for offset fault diagnosis is typically discretized into the same number of states as its source. For instance, the CH node for Figure 6 only has three states, $\Omega(CH) = \{drop, nominal, rise\}$, to indicate a downward change, no change, or upward change, respectively. Thus, as its source sensor, the CH node will utilize a lower and upper threshold. In the case of a battery degradation, the state of the voltage sensor's S node (Figure 6) may not change at all, but the CH node's state will become $CH = drop$ after crossing $v(0)$ to indicate the slight voltage drop due to the degradation.

Figure 7 shows the marginal posterior distributions for the H nodes of the source sensor and battery (Figure 6) under nominal conditions. In this example, the circuit breaker's state is *closed*, and the source sensor is deemed to be *healthy*. The state of the CH node is *nominal*. Thus, the state of the battery's health node is *healthy* with very high probability.

Now suppose the source sensor's value drops, but the mag-

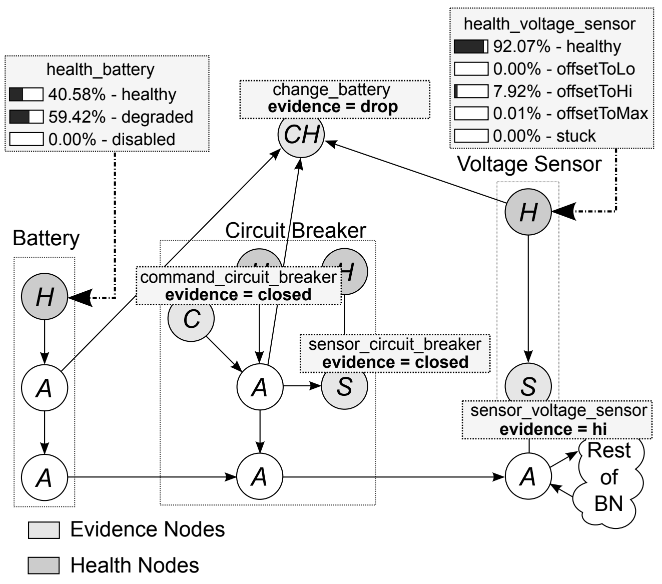

Figure 8. The marginal posterior distributions for the H nodes of the voltage sensor and battery from Figure 6, after a slight voltage drop.

nitude of the drop is too small to cross a threshold $v(i)$ in the source sensor. In such a scenario, the state of the source sensor would not change, and assuming all other evidence clamped in the BN stays the same, this slight drop would not cause the target's health state to change in the absense of the CH node. With the CH node present however (Figure 6), this voltage drop could be detected. In Figure 8, the state of the CH node changes from *nominal* to *drop*, and thus, the probability of the *degraded* state for the battery's H node increases to become the most probable explanation.

Figure 9 shows another use for CH nodes and CUSUM. Here, we have a bank with many loads, but very few sensors to provide evidence concerning their state. Some of these loads have no sensors at all, and hence diagnosing these loads becomes difficult. Here, we create additional CUSUM evidence $e(t)$, and clamp it to a single CH node. The CUSUM values are derived from the single current flow sensor (the source) that measures current flow into the load bank (the target). The CH node provides discretization of this CUSUM at a higher resolution (with more states) compared to the CH node from our previous example (Figure 6).

In a configuration such as this, the CH node will have many thresholds, $\{v(0), v(1), \ldots, v(n)\}$, that correspond to $n+1$ states. Taking $v(i-1)$ and $v(i)$ to be the bounds on a nominal range (nominal state) for the CH node's CUSUM values, each sequential crossing of a threshold $\{v(i), v(i+1), \ldots, v(n)\}$ or $\{v(i-1), v(i-2), \ldots, v(0)\}$ represents an offset of increasing magnitude, and typically corresponds to a specific fault in a component in the bank. These faults are usually offset faults or component failures.

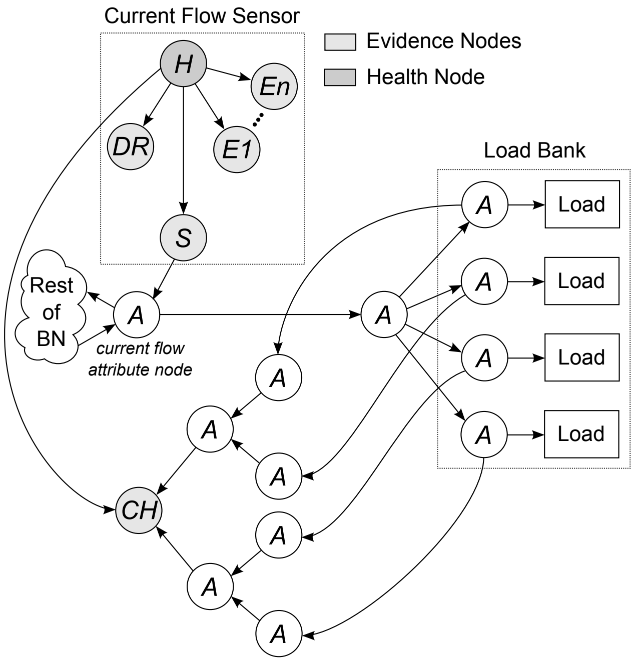

Figure 9. A simplified model of a BN model for a load bank. Evidence $e(t)$ clamped to the CH node is derived from the current sensor's readings. The CH node forms the leaf of a tree, which is structured to limit the CPT size of CH.

Figure 9 contains a tree-like structure that connects the components in the bank to the CH node. This serves to sum the current flows (providing the source sensor measures current flow) of each component, so that the CPT size of the CH node is minimized (Table 2). Note that this does not affect the number of states, $|\Omega(CH)|$, but rather the number of parents of a CH node, and hence the size of its CPT. This technique serves to combine similar states along the tree that would otherwise be present in the CH node's CPT if each load bank component was a parent of the CH node. For example, all components in the load bank have a state that corresponds to no current draw. In Figure 9, this would equal a total of four *no current* states. The two parent A nodes of the CH node (Figure 9) themselves have two parents. Each of these parents has a state which corresponds to *no current* and thus the CH node's CPT only has to have probabilities for *no current* states pertaining to its two parents (see Table 2) rather than all four parents if all load bank component A nodes were parents of the CH node.

Using Figure 9 as an example, consider a simple situation in which all loads in the bank are healthy with a state set of $\Omega(X) = \{healthy, failed\}$. Assuming this corresponds to an A state of *w60* for each load and the source sensor is healthy, we would see the most probable state of the CH node as *w240*. This state would be based on the CUSUM generated from the source sensor's values. Now suppose one of the loads failed. Since the source sensor is the sole sensor

Change_current_it167									
CH									
w0	w30	w60	...	w420	w450	A	A	H
0.95	0.05	0.0	...	0.0	0.0	w0	w0	healthy
0.0	0.05	0.9	...	0.0	0.0	w60		
0.0	0.0	0.05	...	0.0	0.0	w90		
...						...		
0.0	0.0	0.0	..	0.0	0.0	w240		
0.0	0.0	0.0	...	0.0	0.0	w270		
0.05	0.9	0.05	...	0.0	0.0	w0	w30	
0.0	0.0	0.05	...	0.0	0.0	w60		
0.0	0.0	0.0	...	0.0	0.0	w90		
...						...		
0.0	0.0	0.0	...	0.0	0.0	w240		
0.0	0.0	0.0	...	0.0	0.0	w270		
...						...	...	
0.0	0.0	0.0	...	0.0	0.0	w0	w150	
0.0	0.0	0.0	...	0.0	0.0	w60		
0.0	0.0	0.0	...	0.0	0.0	w90		
...						...		
0.0	0.0	0.0	...	0.05	0.0	w240		
0.0	0.0	0.0	...	0.9	0.05	w270		
0.06	0.06	0.06	...	0.06	0.06	w0	w0	(not healthy)
...						...		
0.06	0.06	0.06	...	0.06	0.06	w270		
...						...	...	
0.06	0.06	0.06	...	0.06	0.06	w0	w150	
...						...		
0.06	0.06	0.06	...	0.06	0.06	w270		

Table 2. Conditional probability table (CPT) of the CH node from Figure 9. This table is laid out in the same format as Table 1. Some simplifications were made to the CPT so it would fit, including taking out some intermediary states (represented by '...') from the CH node and its parents.

for the entire bank, and considering it only has three states, this load failure may not cause enough of a current drop to cause a change of state in the source sensor. If this were the case, the fault would be missed completely. Fortunately, the CH node would detect this drop and its state would change to *w180*.[5]

3.5 Handling Continuous Drift Faults: Drift Nodes and CUSUM

Drift faults are characterized by a gradual, approximately linear change of behavior (see Section 2.3), though sensor noise may disrupt strict linearity. While abrupt offset faults are diagnosed as soon as a threshold is crossed, due to their near vertical slope at the moment the fault occurs, drift faults usually do not cross these same offset thresholds immediately, and our diagnosis of them utilizes time t alongside thresholds that are specific for drift faults. Utilizing time and new thresholds help to differentiate an abrupt fault that trips a threshold, from a drift fault that would also trip this same threshold after a certain period of time.

[5]The additional evidence from the CH node would be enough to determine that a failure had occurred, but not for a specific load. We assume that a few more sensors would be available to provide additional evidence to the load bank.

A DR node (see Figure 6, voltage sensor) is used to clamp a boolean drift state, $\Omega(DR) = \{nominal, faulty\}$ for a component structure. By default, it is clamped to $DR = nominal$, or no drift present.

Drift tracking uses threshold values and times on multiple levels, defined as $i \in \{0, \ldots, n\}$. For the i-th level we have:

$$\lambda(i) = [v(i), t_{\min}(i), t_{\max}(i)], \tag{6}$$

where $v(i)$ is a threshold value (see Section 2.2), $t_{\min}(i)$ is a threshold minimum time, and $t_{\max}(i)$ is a threshold maximum time. Here, $v(i)$ represents the threshold that must be reached to move to the next level. Level $i = 0$ is the initial level and has these thresholds associated with it: $\lambda(0) = (v(0), 0, \infty)$. Once $v(i)$ is reached, PRODIAGNOSE moves to level $i + 1$ only if $t_{\min}(i) \le t(i) < t_{\max}(i)$, in which $t(i)$ is the time elapsed since the last threshold was reached.[6] If $v(i)$ is reached but $t(i) < t_{\min}(i)$ or $t(i) \ge t_{\max}(i)$, drift tracking resets to $i = 0$. Once the maximal level n has been reached, there is no reset. Pseudo-code for the DRIFTTRACKER is provided in Algorithm 1.

The number of levels is configurable. Currently, $i \in \{0, 1, 2, 3\}$; thus $\lambda(3)$ represents evidence $e(t)$ of a drift fault, and $DR = faulty$ at this point. In other words, DR is clamped to *faulty* once level $i = 3$ has been reached.

Algorithm 1 DRIFTTRACKER($\lambda(i), t(i), i, n$)

```
$th \leftarrow v(i) \in \lambda(i)$
$min \leftarrow t_{\min}(i) \in \lambda(i)$
$max \leftarrow t_{\max}(i) \in \lambda(i)$
if $i = n$ then
    return $faulty$
end if
if $|\bar{\delta}_p(t)| > th$ and $min \le t(i) < max$ then
    if $i < n$ then
        $i \leftarrow i + 1$
    else
        return $faulty$
    end if
else
    $i \leftarrow 0$
end if
return $nominal$
```

To bring out the underlying behavioral patterns in a sensor's readings and help filter out noise, a threshold $v(i)$ is compared against the sensor's CUSUM value, $\bar{\delta}_p(t)$, at each timestep t, and not its raw reading $s_p(t)$.

Algorithm 1 is integrated into PRODIAGNOSE for DR nodes similar to how CUSUM is integrated for CH nodes (Ricks & Mengshoel, 2009b, 2010). The raw sensor data for S nodes are handled first, which includes assigning the raw sensor values and discretization of these values (Ricks & Mengshoel,

[6]To handle both upward and downward drift faults, we take the absolute value of $\bar{\delta}_p(t)$. We can safely do that here, under the assumption that drift faults are consistently one or the other.

2009b). We then process CUSUM values for all CH nodes and call the DriftTracker (Algorithm 1) for all DR nodes (which also uses CUSUM values and thus shares much code with that for CH node processing). CH and DR node types take input from their S node source sensors. Similar to S node processing, the CUSUM value for each CH and DR node is discretized after being calculated. These discretized values are then clamped to their respective evidence nodes in the BN before inference is performed.

Drift_voltage_e140		
DR		
nominal	faulty	H (Sensor)
0.99	0.01	healthy
0.99	0.01	offsetToLo
0.99	0.01	offsetToHi
0.99	0.01	offsetToMax
0.01	0.99	drift
0.99	0.01	stuck

Table 3. Conditional probability table (CPT) of the DR node from Figure 6. If the DR node is clamped to *nominal*, then the H node (column 3) has a high probability of being in any state except for *drift*. Conversely, if *faulty* is clamped to the DR node, the H node has a high probability of being in the *drift* state.

Using Figure 6 as an example, consider a situation in which no drift fault is present. This would result in a DR state of *nominal*. According to the DR node's CPT (Table 3), with a clamped state of *nominal*, the parent H node has equal probability of being in any state except for the *drift* state. Now if a drift fault were to be detected for the voltage sensor, the DR clamped state would become *faulty*. This would greatly increase the probability of the voltage sensor's health state changing to *drift* (Table 3). Note that since *drift* is an unhealthy state, the CH node from Figure 6 would no longer have much influence over the battery under this condition (see Section 3.4).

4. Experimental Results

For experimentation, the ADAPT EPS was used. ADAPT is a real-world EPS, located at the NASA Ames Research Center, that reflects the characteristics of a true aerospace vehicle EPS, while providing a controlled environment for injecting faults and recording data (Poll et al., 2007). Data from each ADAPT run is stored in a scenario file, which can later be ingested by the diagnostic software. This design means the diagnostic algorithm can be repeatably run on any scenario file, supporting iterative improvement of the BN and diagnostic system during the development process.

In this section, we report on experimental results for ProDiagnose using two ADAPT data sets, namely DXC-09 and DXC-10 data used for competitions arranged as part of the DX Workshop Series.[7]

4.1 Experiments Using DXC-10 Training Data

The Diagnosis Workshop's 2010 competition (DXC-10)[8] was divided into two tiers: Diagnostic Problem 1 (DP1) and Diagnostic Problem 2 (DP2). A main difference, compared to the 2009 competition (DXC-09), was the inclusion of drift (or incipient) and intermittent faults in DXC-10. Abrupt faults (including abrupt offset faults) were included in DXC-10, as in DXC-09. Consequently, these data sets test the performance of ProDiagnose on drift and abrupt offset faults, which is where our CUSUM-based technique are intended to help. These experimental results were obtained by running training set scenarios provided to all DXC-10 competitors.

4.1.1 Diagnostic Problem 1

DP1 uses a subset of the ADAPT EPS. This subset consists of one battery, connected to a DC load, and an inverter with two AC loads. ADAPT is in a fully powered-up state throughout a scenario. Scenarios generated from this configuration of the EPS are either single-fault or nominal. DP1 contains both offset faults and drift faults, both of which test our CUSUM-based diagnosis technique.

DP1 consists of 39 scenarios in its training set. Of these, 5 are nominal (no fault injection), 12 involve sensor faults, and 22 involve component faults. Of these 39 scenarios, 7 contain offset faults, and 7 contain drift faults. Note that 9 other scenarios in the DP1 training set contain intermittent offset faults. While ProDiagnose handles these similar to the abrupt case, details have been discussed previously (Ricks & Mengshoel, 2010) and are beyond this paper's scope.

The DP1 Bayesian network currently has a total of 148 nodes, 176 edges, and a cardinality range of [2, 10]. The DP1 BN has the same overall structure as the DP2 BN (see Section 4.1.2). Some notable differences are the inclusion, in DP1, of additional evidence nodes (such as DR nodes) for fault types that are not present in DP2, specifically intermittent and drift faults, and additional CH nodes to aid in load fault diagnosis of fault types such as drift faults.

The metrics in Table 4 are briefly summarized here to aid interpretation of the results. *Mean Time To Isolate* refers to the time from when a fault is injected until that fault is diagnosed. *Mean Time To Detect* refers to the time from when a fault is injected until any fault is detected. *False Positives* occur when ProDiagnose diagnoses a fault that is not actually present. *False Negatives* occur when ProDiagnose fails to diagnose a fault that is present. Low *False Positive Rates* are important because it is undesirable to perform corrective action when the system is operating correctly. A low *False Negatives Rate*

[7]More information about the diagnostic competitions, including these data sets, can be found here: `http://www.dx-competition.org/`.

[8]More information on DXC-10, including scenario files, can be found here: `https://www.phmsociety.org/competition/dxc/10`.

	CUSUM	
Metric	**Enabled**	**Disabled**
Detection Accuracy	92.31%	46.15%
False Positives Rate	0%	0%
False Negatives Rate	8.82%	61.76%
Mean Time To Detect	17.97 s	28.36 s
Mean Time To Isolate	72.27 s	51.14 s

Table 4. Experimental results with CUSUM enabled and disabled using electrical power system scenarios for DP1.

	CUSUM	
Metric	**Enabled**	**Disabled**
Detection Accuracy	90.91%	87.88%
False Positives Rate	3.03%	3.03%
False Negatives Rate	7.69%	11.54%
Mean Time To Detect	5.74 s	10.56 s
Mean Time To Isolate	36.78 s	39.97 s

Table 5. Experimental results with CUSUM enabled and disabled using electrical power system scenarios for DP2.

indicates that few system faults will remain undetected.

In these experiments, ProDiagnose achieved an impressive *False Positives Rate* of 0% and a *False Negatives Rate* of 8.82% when CUSUM was enabled. When CUSUM was disabled, on the other hand, detection accuracy plummeted to 46% with a false negative rate of almost 62%. Detection times also increased with CUSUM disabled, due to increased detection time of certain offset faults that now must rely solely on an S node state change. Note that when CUSUM is disabled, drift faults are difficult to diagnose correctly (they will appear as abrupt offset faults) due to drift tracking's dependence on CUSUM. However, this actually lowers isolation times due to no isolation time being recorded for a mis-diagnosis.[9]

4.1.2 Diagnostic Problem 2

DP2 represents the entire ADAPT EPS. ADAPT consists of three batteries as the source, connected to two DC load banks, and two inverters each connected to an AC load bank. Scenarios generated from the full ADAPT EPS can be single, double, or triple-fault; or nominal. ADAPT is initially in a powered-down state, and various relays are closed and opened through a scenario to provide power to various components of the EPS. DP2 contains offset faults, but no drift faults, and thus our CUSUM-based diagnosis approach is not as extensively tested as in DP1.

DP2's training set contains 34 scenarios in total: 7 nominal, 9 with sensor faults, and 21 with component faults (some scenarios have both sensor and component faults). Among DP2 scenarios, 6 contain offset faults.

Since DP2 does not contain scenarios with drift and intermittent faults, the DP2 Bayesian network does not implement support for all the fault types seen in DP1. Thus, additional evidence nodes (such as DR nodes) for these fault types are omitted from the DP2 BN. The DP2 BN currently has a total of 493 nodes, 599 edges, and a cardinality range of [2, 16].

Experimental results for DP2 are summarized in Table 5. Compared to DP1, the DP2 data set did not have as many scenarios that might benefit from CUSUM (though it is worth noting that 13 scenarios which involved component faults are diagnosed by catching offsets in sensor readings). Consequently, DP2's increase in accuracy when using CUSUM is not as pronounced, although it does improve from 87.88% to 90.91%. Most of DP2's faults could be diagnosed without needing the additional evidence provided by CH nodes. DP2 also does not contain drift faults, for which ProDiagnose is dependent on CUSUM techniques to diagnose. In addition, with CUSUM enabled, the mean detection time decreased by almost half, due to the role it plays in load bank component diagnosis (see Section 3.4). This is significant, as quick diagnosis is very important in aircraft and spacecraft. Finally, using CUSUM should not adversely impact the overall diagnostic performance of ProDiagnose, and we see that all metrics in Table 5 are equally good or better when CUSUM is enabled compared to when it is disabled.

4.2 DXC-09 and DXC-10 Competition Results

ProDiagnose had the best performance in three of four of the Diagnosis Workshop's industrial track competitions in 2009 and 2010 (DXC-09 and DXC-10). In both DXC-09 and DXC-10 the CUSUM techniques discussed in this paper played a crucial role. DXC-10 competition data indicate strong performance of ProDiagnose (Kurtoglu et al., 2010), implementing the CUSUM approach for diagnosis of offset and drift faults, against algorithms relying on alternate techniques. In the official competition, ProDiagnose achieved an overall scenario detection accuracy of 82.5% in DP1 and 89.2% in DP2, surpassing the second-best DP2 entrant by 19%. In the DP2 category, ProDiagnose also had the fewest fault classification errors and the quickest fault detection time. Data from the 2009 competition (DXC-09) indicate ProDiagnose as the top performer with detection accuracies of 96.7% and 88.3% in Tier 1 and Tier 2, respectively (Kurtoglu et al., 2009).

5. Conclusion

For fault diagnosis in complex and resource-constrained environments, we would like a diagnosis algorithm to be exact, fast, predictable, able to handle hybrid (discrete and continuous) as well as dynamic behavior, and easy to verify and validate (V&V).

[9]This is according to the DXC definition of *Mean Time To Isolate*; one could certainly make the argument that a mis-diagnosis should be punished more harshly.

Fulfilling all these requirements is certainly a tall order. However, we have in this paper extended previous work on the ProDiagnose diagnostic algorithm (Ricks & Mengshoel, 2009a, 2009b; Mengshoel et al., 2010; Ricks & Mengshoel, 2010), and discussed the promise of using static arithmetic circuits, compiled from static Bayesian networks. In particular, we have shown how fault diagnosis using static arithmetic circuits can be augmented with a cumulative sum (CUSUM) technique, resulting in dramatically improved performance in situations with continuous fault dynamics. In experiments with data from a real-world electrical power system, we have observed that our CUSUM-based technique leads to significantly improved performance in situations with continuous offset and drift faults. In addition, the CUSUM techniques discussed in this paper played a crucial role in the strong performance of ProDiagnose in the Diagnosis Workshop's industrial track competitions in 2009 and 2010 (DXC-09 and DXC-10). In DXC-09 and DXC-10, ProDiagnose achieved the best performance in three of four industrial track competitions.

Acknowledgments

This material is based, in part, upon work supported by NSF awards CCF0937044 and ECCS0931978.

References

Chan, H., & Darwiche, A. (2005). On the revision of probabilistic beliefs using uncertain evidence. *Artificial Intelligence*, *163*(1), 67–90.

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

Choi, A., Darwiche, A., Zheng, L., & Mengshoel, O. J. (2011). Data Mining in Systems Health Management: Detection, Diagnostics, and Prognostics. In A. Srivastava & J. Han (Eds.), (chap. A Tutorial on Bayesian Networks for System Health Management). Chapman and Hall/CRC Press.

Darwiche, A. (2003). A Differential Approach to Inference in Bayesian Networks. *Journal of the ACM*, *50*(3), 280–305.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge, UK: Cambridge University Press.

Dechter, R. (1999). Bucket Elimination: A Unifying Framework for Reasoning. *Artificial Intelligence*, *113*(1-2), 41-85. Available from `citeseer.nj.nec.com/article/dechter99bucket.html`

Kozlov, A., & Koller, D. (1997). Nonuniform Dynamic Discretization in Hybrid Networks. In *In Proc. UAI* (pp. 314–325). Morgan Kaufmann.

Kurtoglu, T., Feldman, A., Poll, S., deKleer, J., Narasimhan, S., Garcia, D., et al. (2010). *Second International Diagnostic Competition (DXC10): Industrial Track Diagnostic Problem Descriptions* (Tech. Rep.). NASA ARC and PARC. Available from `http://www.phmsociety.org/competition/dxc/10/files`

Kurtoglu, T., Narasimhan, S., Poll, S., Garcia, D., Kuhn, L., deKleer, J., et al. (2009, June). First International Diagnosis Competition - DXC'09. In *Proc. of the Twentieth International Workshop on Principles of Diagnosis (DX'09)* (pp. 383–396). Stockholm, Sweden.

Langseth, H., Nielsen, T. D., Rumí, R., & Salmeron, A. (2009). Inference in hybrid Bayesian networks. *Reliability Engineering & System Safety*, *94*(10), 1499–1509.

Lauritzen, S., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society series B*, *50*(2), 157–224.

Lerner, U., Parr, R., Koller, D., & Biswas, G. (2000). Bayesian Fault Detection and Diagnosis in Dynamic Systems. In *Proceedings of the Seventeenth national Conference on Artificial Intelligence (AAAI-00)* (p. 531-537). Available from `citeseer.ist.psu.edu/lerner00bayesian.html`

Mengshoel, O. J. (2007). Designing Resource-Bounded Reasoners using Bayesian Networks: System Health Monitoring and Diagnosis. In *Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07)* (pp. 330–337). Nashville, TN.

Mengshoel, O. J., Chavira, M., Cascio, K., Poll, S., Darwiche, A., & Uckun, S. (2010). Probabilistic Model-Based Diagnosis: An Electrical Power System Case Study. *IEEE Trans. on Systems, Man, and Cybernetics*, *40*(5), 874–885.

Mengshoel, O. J., Poll, S., & Kurtoglu, T. (2009). Developing Large-Scale Bayesian Networks by Composition: Fault Diagnosis of Electrical Power Systems in Aircraft and Spacecraft. In *Proc. of the IJCAI-09 Workshop on Self-$\star$ and Autonomous Systems (SAS): Reasoning and Integration Challenges.*

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, *41*, 100 - 115.

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

Poll, S., Patterson-Hine, A., Camisa, J., Garcia, D., Hall, D., Lee, C., et al. (2007). Advanced Diagnostics and Prognostics Testbed. In *Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07)* (pp. 178–185). Nashville, TN.

Ricks, B. W., & Mengshoel, O. J. (2009a). The Diagnostic

Challenge Competition: Probabilistic Techniques for Fault Diagnosis in Electrical Power Systems. In *Proc. of the 20th International Workshop on Principles of Diagnosis (DX-09).* Stockholm, Sweden.

Ricks, B. W., & Mengshoel, O. J. (2009b). Methods for Probabilistic Fault Diagnosis: An Electrical Power System Case Study. In *Proc. of Annual Conference of the PHM Society, 2009 (PHM-09).* San Diego, CA.

Ricks, B. W., & Mengshoel, O. J. (2010). Diagnosing Intermittent and Persistent Faults using Static Bayesian Networks. In *Proc. of the 21st International Workshop on Principles of Diagnosis (DX-10).* Portland, OR.

Brian Ricks received the Bachelor's of Science degree in Computer Science from the University of Texas at Dallas, in 2010.

He is currently a graduate student at the University of Texas at Dallas, majoring in Computer Science. He previously worked at the NASA Ames Research Center, Intelligent Systems Division, as an intern with Carnegie Mellon University - Silicon Valley, and also at the NASA Ames Research Center as an intern with the Universities Space Research Program. He will graduate in Spring of 2012 with a Master's in Computer Science. Mr. Ricks performed part of the research reported here during both prior internships, under the leadership of Dr. Ole J. Mengshoel.

Ole J. Mengshoel received the B.S. degree from the Norwegian Institute of Technology, Trondheim, Norway, in 1989, and the Ph.D. degree from the University of Illinois at Urbana-Champaign, Illinois, in 1999, both in computer science.

He is currently a Senior Systems Scientist with Carnegie Mellon University (CMU), Silicon Valley, CA, and affiliated with the Intelligent Systems Division, National Aeronautics and Space Administration (NASA) Ames Research Center, Moffett Field, CA. Prior to joining CMU, he was a Senior Scientist and Research Area Lead with USRA/RIACS and a Research Scientist with the Decision Sciences Group, Rockwell Scientific, and Knowledge-Based Systems, SINTEF, Norway. His current research focuses on reasoning, machine learning, diagnosis, prognosis, and decision support under uncertainty – often using Bayesian networks and with aerospace applications of interest to NASA. He has published more than 50 papers and papers in journals, conference proceedings, and workshops. He is the holder of four U.S. patents.

Dr. Mengshoel is a member of the Association for the Advancement of Artificial Intelligence, the Association for Computer Machinery, and IEEE.