Deriving Bayesian Classifiers from Flight Data to Enhance Aircraft Diagnosis Models

Daniel L.C. Mack [1], Gautam Biswas [1], Xenofon D. Koutsoukos [1], Dinkar Mylaraswamy [2], and George Hadden [2]

[1] *Vanderbilt University, Nashville, TN, 37203, USA*
daniel.l.mack@vanderbilt.edu
gautam.biswas@vanderbilt.edu
xenofon.koutsoukos@vanderbilt.edu

[2] *Honeywell Aerospace, Golden Valley, MN 55422, USA*
dinkar.mylaraswamy@honeywell.com
george.d.hadden@honeywell.com

ABSTRACT

Online fault diagnosis is critical for detecting the onset and hence the mitigation of adverse events that arise in complex systems, such as aircraft and industrial processes. A typical fault diagnosis system consists of a reference model that provides a mathematical representation for various diagnostic monitors that provide partial evidence towards active failure modes, and a reasoning algorithm that combines set-covering and probabilistic computation to establish fault candidates and their rankings. However, this approach often suffers from incompleteness in the reference models and simplifying assumptions made by the reasoning algorithms. Incompleteness in the reference models take several forms, such as absence of evidence, errors and incompleteness in the mapping between evidence and failure modes, while inaccuracies in the reasoning algorithm arise from simplifying noise models and independence assumptions. In this paper, we describe a Tree Augmented Naive Bayesian Classifier (TAN) classifier that forms the basis for systematically extending reference models using data from systems operating with and without faults. We investigate the performance of the TAN models starting from the expert supplied reference model using airline data, and demonstrate that the generated TAN structures can be used by the expert to identify areas of improvement through the addition of new causal links, and updated thresholds for classification among the system monitors. This is then translated into an improvement for the reference model which in turn benefits the reasoner of the aircraft.

1. INTRODUCTION

An important challenge facing aviation safety is early detection and mitigation of adverse events caused by system or component failures. Take an aircraft for example, which consists of several sub-systems such as propulsion, aviation, bleed, flight control, and electrical; each of these subsystems consists of several dozens of interacting components. Faults can arise in one or more aircraft subsystem; their effects in one system may propagate to other subsystems, and faults may interact. To detect these faults, an onboard fault diagnosis solution must be able to handle these interactions and provide an accurate diagnostic and prognostic state for the aircraft with minimal ambiguity.

The current state of online fault diagnosis is focused on installing a variety of sensors onboard an aircraft along with a reasoning software to automatically interpret the evidence generated by them to indicate the presence of faults. One such state of the art system is the Aircraft Diagnostic and Maintenance System ADMS (Spitzer, 2007) that is used on the Boeing B777. ADMS can be broadly categorized as a model-based diagnoser that separate system-specific knowledge and the inferencing mechanism.

Consider characteristics of some typical faults arising in aircraft subsystems. Turbine blade erosion is a natural part of turbine aging and wearing of the protective coating due to microscopic carbon particles exiting the combustion chamber. As the erosion progresses over time, it starts to affect the ability of the turbine to extract mechanical energy from the hot expanding gases. Eventually this fault manifests itself as increase in fuel flow and gradual degradation of engine performance. This causal propagation of faults is usually known to a domain expert and captured mathematically using a static system reference model. As evidence get generated by aircraft installed sensors, a reasoning algorithm "walks" the relevant causal paths and concludes the current state of the aircraft—in this case, turbine erosion of the propulsion engine.

The ADMS uses a fault propagation system reference model that captures the interactions between aircraft components under various operating modes. A Bayesian belief propagation network together with the Bayesian update rule provides an ideal framework for this onboard diagnostic reasoning. It provides the necessary transparency for certification as a safety system, while allowing the sub-system manufacturer to encode proprietary fault models. Having said that, it is equally important to note that generation of this reference model

is a manual process and often the most tedious step in the practical development and deployment of an ADMS. While most of the knowledge about fault propagation can be derived from earlier aircraft designs, upgrades to component design (for example using active surge control rather than passive on-off surge prevention) create gaps in the knowledge base. As the engineering teams "discover" such knowledge from an operating fleet, these translate as expert heuristics rather than a systematic upgrade to the reference model that was generated at design time.

Many of the shortcomings of the ADMS can be attributed to incomplete and incorrect information in the system reference model. In other words, a gap exists for systematic upgrades and increments to the reference model as vast amount of operational data is collected by operating airlines. We look at this problem as a "causal structure discovery" problem. Specifically, learning causal structures in the form of a Bayesian Network wherein the nodes represent system failures (cause) and diagnostic evidence (symptom). Unlike associations, Bayesian networks can be used to better capture the dependencies among failures (failure cascade from one subsystem to another) and evidence cascade (failure mode in one system triggering a symptom in a nearby component).

This paper presents a case study, an adverse event surrounding an in-flight shutdown of an engine, which was used to systematically augment an existing ADMS reference model. Section 2. describes the basic principles and the constituents of a model-based onboard fault reasoner. Section 3. describes the problem statement wherein we formally define the model augmentation we seek to derive using operational data. Next in section 4. we describe the historic data surrounding the adverse event. Section 6. then discusses the data mining algorithms we have employed for constructing the diagnostic classifiers using Tree-Augmented Bayesian Networks (TANs). Section 7. then discusses experimental results describing how an expert would utilize aircraft data with the classifiers to improve a reference model. Metrics are defined for evaluating classifier performance, and a number of different experiments are run to isolate where improvement in the model may be possible. For these experiments, the derived classifier models provide information to the expert to update the reference model. Section 8. presents a summary of our approach, and outlines our directions for future work for diagnostic and prognostic reasoning using the data mining algorithms.

2. BACKGROUND ON REFERENCE MODELS

Model-based strategies that separate system-specific knowledge and the inferencing mechanism are preferred for diagnosing large, complex, real-world systems. An aircraft is no exception to this, wherein individual component suppliers provide system-specific knowledge that can be represented as a bipartite graph consisting of two types of nodes: failure modes and evidence. Since this knowledge acts as a baseline for diagnostic inferencing, the term "reference model" is also used to describe this information. The set F defines all *distinct* failure modes defined or enumerated for the system under consideration. A failure mode $fm_i \in F$ may be occurring or not occurring in the system, i.e. it may exist in a 1 (occurring) or 0 (not occurring) state. Often a -1 an unknown state is also included as an initial state. We use the following shorthand notations regarding these assertions.

$$\begin{aligned} fm_i = 0 &\Leftrightarrow \text{The failure mode is not occurring} \\ fm_i = 1 &\Leftrightarrow \text{The failure mode is occurring} \end{aligned} \tag{1}$$

Every failure mode has a priori probability of occurring in the system. This probability is given by $P(fm_i = 1)$. A failure mode fm_k can occur (or not occur) independently of another failure mode fm_j occurring. That is, $P(fm_k = 1|fm_j = 1) = P(fm_k = 1)$. This corresponds to a Naive Bayes assumption for the the independence of fault hypothesis.

To isolate and disambiguate the failure modes, component suppliers also define an entity called "evidence" in the system model. The jth evidence is denoted by e_j and the set E denotes all distinct monitors defined for the system under consideration. The diagnostic monitor associated with the ith evidence can either *indict* or *exonerate* a subset of failure modes called its ambiguity group. The monitor m_i can take three mutually exclusive values allowing a monitor to express indicting or exonerating or unknown support for the failure modes in its ambiguity group. The notations are described in equation (2).

$$\begin{aligned} m_i = 0 &\Leftrightarrow \text{Exonerating evidence} \\ m_i = 1 &\Leftrightarrow \text{Indicting evidence} \\ m_i = -1 &\Leftrightarrow \text{Unknown evidence} \end{aligned} \tag{2}$$

An ideal monitor associated with evidence e_i fires only when one or more failure modes in its ambiguity group are occurring. Given the fact that the i'th failure mode is occurring in the system, d_{ji} denotes the probability that there will be a monitor providing an indicting evidence under this condition.

$$d_{ji} \Leftrightarrow P(m_j = 1|fm_i = 1), \tag{3}$$

d_{ji} is called the detection probability of the jth evidence with respect to failure mode monitor fm_j. A monitor may fire when there is no failure mode present in the system. False alarm probability is the probability that an indicting monitor is present when there are no failure modes occurring in the system. That is,

$$\epsilon_j \Leftrightarrow P(m_j = 1|fm_i = 0, \forall fm_i \in F) \tag{4}$$

Designing a monitor requires deep-level domain knowledge. This inner working of the monitor is not important from a reasoning point of view. However, an abstract view of the monitor helps the reasoning algorithm. This abstraction is shown in Figure 1. With few exceptions, most diagnostic monitors are derived by thresholding a time-series signal. This signal can be a raw sensor value or a derived quantity. We call this a condition indicator and denote it as $x(t)$. Without loss of generality one can assume an upper threshold such that $m \Leftrightarrow x(t) > \theta$. A diagnostic monitor may specify the underlying condition indicator and the threshold or simply provide the net result of applying a hidden threshold.

Figure 2 illustrates an example reference model graphically, with fault modes (hypotheses) as nodes on the left, and diagnostic monitors (DM) on the right.Each link would contain a detection probability, i.e., conditional probability $P(m_j = 1|fm_i = 1)$. In addition, fault

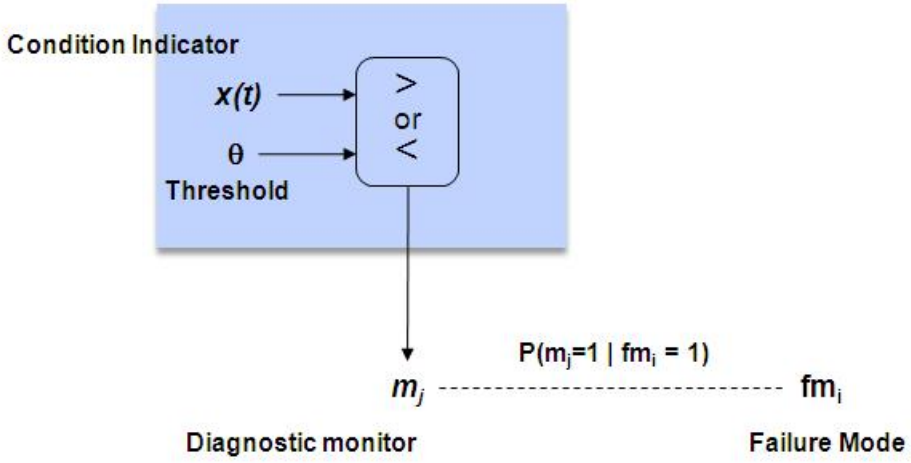

Figure 1: Abstraction of Diagnostic monitor

nodes on the left contain the a priori probability of fault occurrence, i.e., $P(fm_i)$. Probabilities on the DM nodes indicate the likelihood that a particular monitor would indicate a fault in a nominal system. Bayesian methods are employed to combine the evidence provided by multiple monitors to estimate the most likely fault candidates.

The reasoner algorithm (called the W-algorithm) combines an abductive reasoning algorithm with a forward propagation algorithm to generate and rank possible failure modes. This algorithm operates in two steps: (1) *Abductive reasoning step*: Whenever a diagnostic monitor m_1 fires, it provides either indicting (if $m_1 = 1$) or exonerating (if $m_1 = 0$) evidence for the failure modes in its *ambiguity set*, $AG = \{fm_1, fm_2, \ldots fm_k\}$. This step assumes that the firing of a DM implies at least one of the faults in the ambiguity set has occurred; and (2) *Forward reasoning step*: For each fm_i belonging to AG, this step calculates all other diagnostic monitors that may fire if any of the failure modes are indeed occurring. These are called the evidence of interest. Let $m_2, m_3, \cdots$ denote this evidence of interest set. Some of these monitors may be in an indicating state, for example $m_2 = 1$ or exonerating state, for example $m_3 = 0$. The reasoning algorithm calculates the joint probability $P(fm_1 = 1, m_1 = 1, m_2 = 1, m_3 = 0, \ldots)$ of a specific failure mode fm_1 occurring in the system. As additional monitors fire, the numeric values of these probabilities increase or decrease, till a specific failure mode hypothesis emerges as the "winner". The reasoning algorithm can generates multiple single fault hypothesis, each hypothesis asserting the occurrence of exactly one failure mode in the system.

The reasoning algorithm may not not reduce the ambiguity group to a single fault element. This can happen for various reasons. A modest aircraft has over 5000 monitors and failure modes; estimating the detection probabilities d_{ji} for a modest aircraft is a challenging offline design task. Errors in d_{ji} or more specifically lack of a link between an evidence and a failure mode can adversely affect the reasoner performance. Further, to keep things simple, a modeler may assume monitor firing events to be independent. This eliminate the modeler from calculating the joint probability $P(m_j = 1, m_k = 1|fm_i = 1)$ and approximate it as $P(m_j = 1|fm_i = 1) \times P(m_k = 1|fm_i = 1)$. Designing a good set of monitors is yet another challenging task. For example, the modeler may have overlooked a new monitor m_p that could have differentiated between failure modes fm_1 and fm_2.

Given the complexity of large systems such as an aircraft, incompleteness in the reference model is expected. However, as one collects more operational data, some of these gaps can be addressed. The independence assumption implies that the reasoning algorithm treats the reference model as a Naïve Bayes classifier. Consequently, the direct correspondence between the reference model and the simple Bayesian structure provides opportunities to use a class of generative Bayesian model algorithms to build these missing links structures from data. It is this systematic approach for updating the system reference model is the theme of this paper. We now formally define the problem statement.

3. PROBLEM STATEMENT

The reference model when viewed as a single fault diagnoser can be interpreted as a Noisy-OR classifier, which is a simplified form of a standard Bayesian Network. A number of Machine Learning techniques for building Bayesian networks from data have been reported in the literature (Friedman, Geiger, & Goldszmidt, 1997). For example, state-based hidden Markov Models (HMMs) and even more general Dynamic Bayesian Network (DBN) formulations can be employed to capture the dynamics of aircraft behavior and effects of faults on system behavior and performance. However, rather than addressing the problem as a traditional data mining problem, we approached it as an extension to the existing ADMS. In other words, the data mining algorithms should be designed to provide information that supplements existing expert-generated reference models, as opposed to providing different formulations and different reasoner structures. Verifying the model enhancements by experts becomes relatively straightforward.

A systematic approach (unlike looking for a needle in haystack approach) to data mining to precisely define the

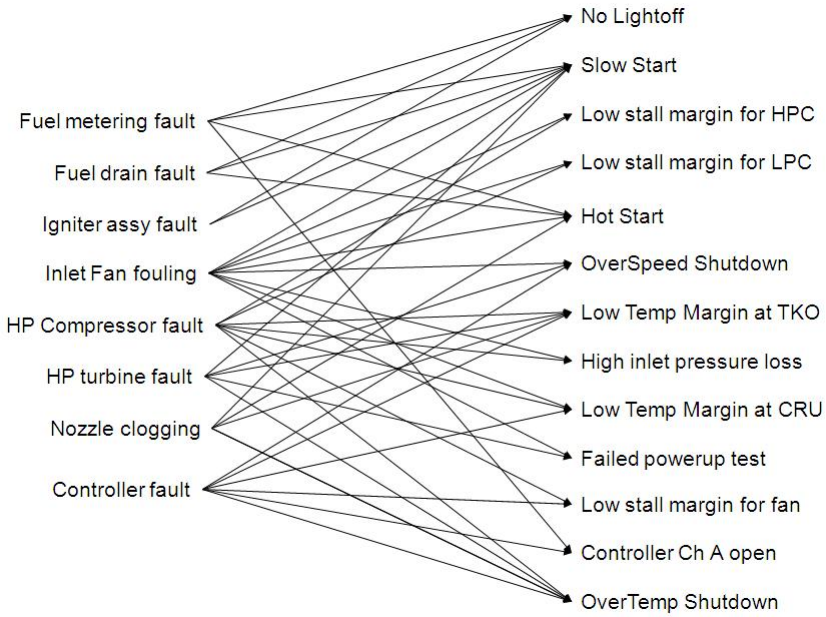

Figure 2: Example Reference Model

missing elements in the reference model that we need to learn. Specifically:

1. Updating the relations between monitors and failure modes. Specifically updating d_{ji} for monitor-failure mode pairs that already exist, as well as assigning a non-zero number for d_{ji} if the link did not exist.
2. Updating the threshold θ associated with a diagnostic monitor given a trace of condition indicators. In other words, the data mining is indirectly updating the detection probability and the false alarm rate ϵ_j associated with a monitor.
3. Creating new monitors that combines m_1 and m_2 such that either that assert the combination asserts a stronger evidence for a specific failure mode fm_i. That is, calculate a stronger value for $P(m_j = 1, m_k = 1|fm_i = 1)$ which is greater than $P(m_j = 1|fm_i = 1) \times P(m_k = 1|fm_i = 1)$.

Besides formulating the output of the data mining as an extension to the existing reference model, the computational complexity of the data mining algorithms should be manageable, so that they can be used as exploratory analysis tools by the domain experts. We envision a successive refinement process, where the expert may request a number of experimental runs, each using a specific data set from an operating fleet of aircrafts and the results from the nth experiment augments the reference model from the $(n-1)^{th}$ experiment. This will result in a continuous learning loop wherein the observations from the fleet are incorporated systematically to understand the causal relations between failure modes, their manifestations (monitors) and, also to study the dependence amongst failure modes under various adverse event situations. Over time, this learning loop will increase the accuracy (while reducing false positives) in the diagnostic reasoner.

Having established this framework, we stay within the Bayes net paradigm, and describe the data available to us for enhancing an existing expert-supplied reference models for the next step.

4. AIRCRAFT FLIGHT DATA

Demonstrating and validating the proposed upgrades to the system reference model needs data. A good data set would span several contiguous flights as well as contain multiple aircrafts. This data set will contain the statistical richness arising from aircraft-to-aircraft variation as well as heterogeneity of flight patterns. One such data set came from a fleet of regional airlines operating in North America. The fleet consisted of 30+ identical aircrafts, each aircraft operating 2–3 flights each day. Data spanning three years was made available to to support our work.

The Aircraft Condition Monitoring System (ACMS) is an airborne system that collects data to support fault analysis and maintenance. The Digital ACMS Recorder (DAR) records airplane information onto a magnetic tape (or optical) device that is external to the ACMS. This data is recorded in raw, uncompressed form. The DAR can record information a maximum of 512 12-bit words per second via a serial data stream modulated in either Harvard Bi-Phase or Bi-Polar Return-to-Zero code. The recorded data is then saved permanently to a compact flash card. The ACMS can be programmed to record parameter data from the propulsion subsystem, the airframe, the aircraft bleed subsystem, and the flight management system at a maximum rate of 16Hz. It is this raw time-series data that we use for designing the reference model and the reasoner enhancements.

The second source of information we need are adverse event annotations. One such source is the FAA developed Aviation Safety Information Analysis and Sharing (ASIAS) system. ASIAS is a database collection of ad-

verse events reported by various airline operators. A simple cross check with this database revealed an engine shutdown event. One of the engines aboard the aircraft shutdown automatically and the flight crew declared emergency; returning back to the airport where the flight originated with no casualties. It is this event that forms the focus of our reasoner and the data mining exercise.

Typically an adverse event such as an engine shutdown is the culmination of several things, our objective was to analyze the ACMS data surrounding this event and hence suggest improvements to the existing ADMS—the primary intent is to detect the root cause earlier and possibly avoid the onset of the adverse event. Investigation with the airline maintenance crew revealed the root cause as a faulty fuel metering hydro-mechanical unit. The fuel metering unit is a controller-actuator that meters fuel into the combustion chamber so that the engine produces the aircraft desired thrust. While the ASIAS database defined the date when the adverse event was detected, and the maintenance crew identified the root cause, we needed a rough estimate when the fault initiated. Dialogues with the expert indicates that early indication of a failing fuel metering unit could be visible as early as 50 flights *before* the flight crew observed the engine shutdown adverse event. We used a $[-50, 0]$ flight interval for our analysis, here 0 indicates the flight when the adverse event occurred and -50 indicates 50 flights before this one.

As noted in the Introduction, most modern planes are equipped with an Aircraft Diagnostic and Maintenance System or ADMS. We further noted that evidence are provided by diagnostic monitors, which in turn are derived by setting appropriate threshold to condition indicators. The following condition indicators and diagnostic monitors were available from this aircraft.

StartTime This CI provides the time duration till the engine reaches its idling speed. Appropriate threshold generates the *no start* diagnostic monitor.

IdleSpeed This CI provides the idling steady state speed. Appropriate threshold generates the *hung start* diagnostic monitor.

peakEGTC This CI provides the peak exhaust gas temperature within an engine start-stop cycle. Appropriate threshold generates the *overtemp* diagnostic monitor.

N2atPeak This CI provides the speed of the engine when the exhaust gas temperature achieves its peak value. Appropriate threshold generates the *overspeed* diagnostic monitor.

timeAtPeak This CI provides the dwell time when the exhaust gas temperature was at its peak value. Appropriate threshold generates the *overtemp* diagnostic monitor.

Liteoff This CI provides the time duration when the engine attained stoichiometry and auto-combustion. Appropriate threshold generates the *no lightoff* diagnostic monitor.

prelitEGTC This CI provides the engine combustion chamber temperature before the engine attained stoichiometry. Appropriate threshold generates the *hot start* diagnostic monitor.

phaseTWO This CI provides the time duration when the engine controller changed the fuel set-point schedule. There are no diagnostic monitors defined for this CI.

tkoN1, tkoN2, tkoEGT, tkoT1, tkoPALT These CI provides the fan speed, engine speed, exhaust gas temperature, inlet temperature and pressure altitude respectively, averaged over the time interval when aircraft is operating under take off conditions. There are no diagnostic monitors defined for these CI.

tkoMargin This CI provides the temperature margin for the engine during take off conditions. Appropriate threshold generates the *medium yellow* and *low red* diagnostic monitors.

Rolltime This CI provides the time duration of the engine's roll down phase. Appropriate threshold generates the *abrupt roll* diagnostic monitor.

resdTemp These CI provide the engine exhaust gas temperature at the end of the engine's roll down phase. Appropriate threshold generates the *high rtemp* diagnostic monitor.

N2atDip, dipEGTC These CI provide the engine speed, exhaust gas temperature at the halfway point in the engine's roll down phase. There are no diagnostic monitors defined for these CI.

N2cutoff These CI provide the rate of change of the engine speed at the halfway point in the engine's roll down phase. There are no diagnostic monitors defined for these CI.

With this data set, we set out to achieve our reasoner enhancements as described earlier in section 3.. While this large volume of data provides with opportunities to study a number of different operating scenarios in much greater depth and detail, the data is not in a form that be directly processed by machine learning algorithms. Curation methods, therefore, have to be developed to prepare the data sets that can be analyzed by machine learning algorithms. This data curation step is discussed next.

5. DATA CURATION

An important requirement for the success of data driven techniques is the need for relevant and well-organized data. Well-organized typically implies getting rid of unwanted details, being able to structure the data on a timeline or a sequence of events, and applying filtering algorithms if the sensors used are known to be noisy. Relevance is a very important concept, since it is important to extract sequences of data, which contain information about the particular situation being modeled. For example, if the goal is to design a classifier that can identify a faulty situation from one in which there is no fault, it is important to provide the two sets of data, so that the classifier can learn the discriminating features from the data. The systems under study are complex, they operate in different modes and under different circumstances. This information is likely to be important for the classification tasks, and the data needs to be appropriately annotated with this information. Overall, unreliable data quality makes it difficult improve an already effective reference model. The data curation problem is often as complex as

and sometimes even more complex than running a classifier or a clustering algorithm. It requires having a good understanding of the nature of the data and how it was acquired, before the analysis methods can be established.

The raw data is contained in binary files, each file containing the ACMS recording for the entire flight from a specific aircraft tail number. Several thousands of these files were organized first by the aircraft tail number. Within each aircraft, the data was then organized chronologically using the flight timestamp. Further, since our case study involves an engine shutdown, we further classified the data based on the engine serial number. While this organization supports the specific case-study, we did want this case-study to restrict the data curation step.

For practical reasons, given the size of the data, and the need to extract specific sub-sequences for the data mining task, it makes sense to include the formatted and organized data into a centralized database with a table structure that makes for easy retrieval. A typical data analysis session involves formulating the appropriate data base queries and collecting the resulting data segments. With a general structure in place, the data can be processed more efficiently and cleansed, if necessary. Cleansing data is not exact and requires a definition of an anomalies. For our analysis, all ground-test (then the aircraft is on the ground and the maintenance crew does some test) were defined as anomalies and removed during the cleansing step.

At the end of the curation process, the original aircraft data is ready for wide-spread distribution and more specifically for applying the data mining. We discuss this next.

6. TREE AUGMENTED NAIVE BAYESIAN NETWORKS

The choice of the data driven techniques to apply to particular problems is very much a function of the nature of the data and the problem(s) to be solved using the data. As mentioned previously, the diagnostic models employed in hierarchical framework lend themselves to Bayesian Methods. As was also noted, the independence assumption used in the VIPR model is one that can be systemically relaxed to capture more information that will be useful for diagnosis. There are several interesting alternatives, but one that fits well with our reference model structure is the Tree Augmented Naive Bayesian Method (Friedman et al., 1997) abbreviated as the TAN algorithm. The TAN network is a simple extension to the Naive Bayesian network formulation. The Root (the fault mode) is casually related to every evidence node. However, as a slight relaxation of the independence assumption, the evidence nodes have limited causality with respect to one another. The limitation is that every node may have at most one evidence node as its parent. This maintains the directed acyclic graph requirements and produces a more nuanced tree that captures relationships among some of the variables (i.e., the system sensors and monitors). Generation of this structure is not as computationally expensive as a general Bayes network.

An example TAN structure is illustrated in Figure 3. The root node, labeled class, is the fault hypothesis under consideration. The other nodes represent supporting evidence for the particular fault hypotheses. In this particular structure, Rolltime, associated with the shutdown

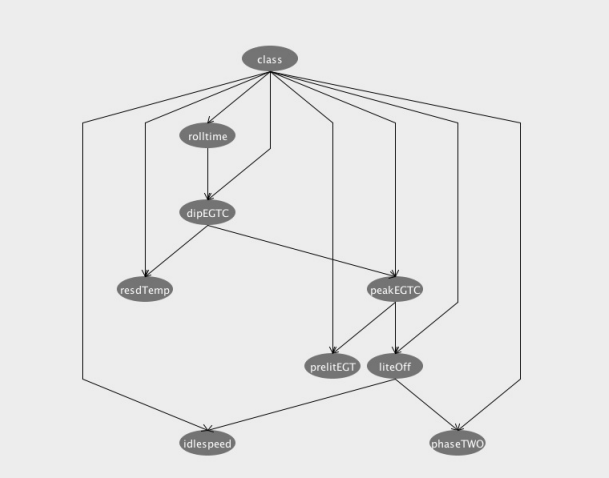

Figure 3: Example TAN Structure

phase of the aircraft is the root observational node. Note that like a Nave Bayesian classifier, the fault hypothesis node, class, is linked to all of the observational monitor nodes. Dependencies among some of the monitors, e.g., Rolltime and dipEGTC, are captured as additional links in the Bayesian network.

The TAN Structure can be generated in several different ways, such as a greedy search that constrains the graph from building "illegal" edges (i.e., a node having more than one parent from the evidence nodes)(Cohen, Goldszmidt, Kelly, Symons, & Chase, 2004). The other procedure and the one adopted for this work is to build a Minimum Weighted Spanning Tree (MWST) of the Evidence Nodes and then connect the fault mode to all of the evidence after the tree has been constructed (Friedman et al., 1997). In either case, a decision has to be made about the anchor node (i.e., the observational root) of the derived tree structure: (1) whether it is an evidence node, or (2) it is the fault mode, which has no parents. Lastly, since this structure is still static, it doesn't have an explicit temporal component. One can address this in a few different ways, as we discuss later.

A standard algorithm (e.g., Kruskal's algorithm(Kruskal, 1956)) is applied to generate the Minimum Weighted Spanning Tree. This algorithm could be substituted by any other that generates similar output, but within the specified theoretical bounds. The values that are used as the edge weights all utilize a form of a log-likelihood such as a Bayesian value (Chickering, Heckerman, & Meek, 1997) or a Bayesian Information Criterion (BIC) (Schwarz, 1978). If the values are discrete (either by nature or discretized continuous values), the use of the Bayesian likelihood metric is preferred. This is a simple metric that can quickly calculate the likelihood that a variable is dependent on another. If the values are continuous, then the BIC is better since it deals with continuous distributions (like a Gaussian Normal) easily. The values are calculated for every pair of evidence nodes and stored in a matrix. The order of the nodes matter, since the graphs are directed. Kruskal's algorithm is used to construct the tree.

The choice of the observational root node is important; the rest of the MWST linked from this node will be

Algorithm 1 TAN Algorithm Using MWST

INPUT:Dataset D of N Features and a label C
INPUT:Observational Root Node FRoot
INPUT:CorrelationFunction
OUTPUT:TAN Structure with Adjacency Matrix, CLassAdjMat, describing the Structure
OUTPUT:Probability Values ProbVec for each Node {**Note:** Corr is a matrix of the likelihood that feature i is causally related to feature j (different values can be found for i to j and j to i)} {Count(Node,ClassAdjMat,D) is a counting function, that takes the Data, the Class, the Full Adjacency Matrix of the TAN and for the Node finds either the CPT for discrete-valued features, or the set of means and covariances to describe the Gaussian Normal Distributions of the Node for continuous valued variables.} {AdjMat describes the parents so that correct data slices can be isolated and used in the counting. }
for $feature_i = 0$ to $feature_i = N$ **do**
  for $feature_j = 0$ to $feature_i = N$ **do**
    if $feature_i \neq feature_j$ **then**
      Corr (i,j) = CorrelationFunction(fi, fj ,D)
    end if
  end for
end for
AdjMat = MWST(Corr, FRoot){ Build a Minimum Weighted Spanning Tree using the Correlation Matrix and the Root chosen}
for $feature_i = 0$ to $feature_i = N$ **do**
  $ClassAdjMat(feature_i, C) = 1$ {Connect every feature to the Class Node to build the TAN}
end for
ProbVec(C) = Count(C,ClassAdjMat,D) {Estimate the parameters, starting with the class}
for $feature_i = 0$ to $feature_i = N$ **do**
  $ProbVec(feature_i) = Count(feature_i, ClassAdjMat, D)$
end for
RETURN: $(AdjMat, ProbVec)$

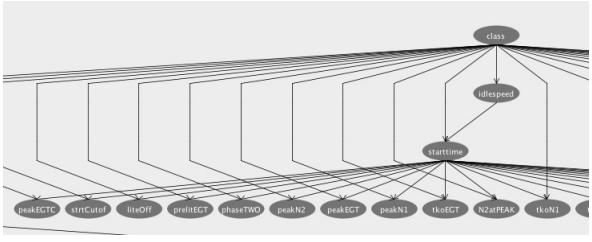

Figure 4: TAN Structure with idlespeed root

conditioned both in structure and the conditional probability based on this root. Given similar data, the nature of the values used to identify stronger causal links should not change drastically on the choice of the observational root, but the choice may reorganize the tree. Therefore, choice of the root can influence the structure of the generated TAN. The pattern of nodes will be similar but their general ordering will be the difference in the structure. For example in Figure 4, when the data is centered on flights near a fault, the root node is idlespeed. This then connects to the starttime feature, to which the rest of the nodes are connected. In Figure 5 with data further away from the incident, we see PeakEGTC as the root node; however, notice the similar structure of idlespeed causally related to starttime with a further connection to a large group of nodes. This structure is similar, but is not as focused near the top of the observational tree. These shifts in the structure requires that a domain expert examine the TAN structure and look for similar structures in the tree of observational nodes. The lessons that can be learned from these changes to the structure will be discussed in Section 7.

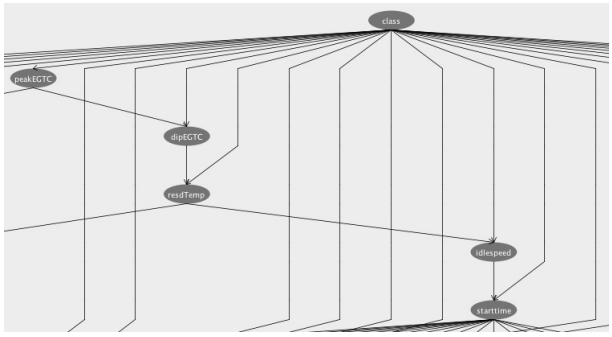

Figure 5: TAN Structure with peakEGTC Root

As this structure shifts, one of the more important changes is in the CPTs or Sets of Gaussian distributions learned from the data. This shift makes the node chosen as the observational root to have only one parent; the class. This means this split is what initially biases the TAN towards one label or another. This shift also changes some causal relationships and may impact how the counting algorithm for parameter estimation groups the data and produces probabilities for the evidence. As will be discussed in a later section, these choices can be used by the domain expert to more effectively improve a reference model for the AHM.

This choice of the root node for the observation portion of the tree as shown in the pseudocode is an input parameter to the algorithm. This choice is normally based on a ranking computed using a heuristic. In a strict data driven method approach, using a heuristics that are purely statistical in nature is an effective method. In fact, among the implementations discussed below, Weka (Hall, Eibe, Holmes, Reutemann, & Witten, 2009) uses just such a method. In discovering the appropriate root node, it builds a TAN with every feature as the root node of the MWST. It compares them using a scoring metric, such as a log-likelihood for the training data. The structure with the best score is then used as the classifier. However, it is possible to use domain

knowledge to choose this node instead. For example, using expert knowledge of the system one may choose the sensor that is closest to the fault under consideration because it is not likely to be causally dependent on other sensors. The implication in the classifier is that it will be closest to indicating a fault.

6.1 Implementations Used for Building TANs

Two different implementations can be employed for the TAN algorithms used in the experiments. The first is one that attempts to maintain the continuous nature of the features and build Gaussian Normal distributions for the nodes. It is implemented in MATLAB using the Bayesian Network Toolkit (Murphy, 2011).

The second implementation included in the data mining toolkit is called Weka (Hall et al., 2009) It does not handle continuous values and instead uses a discretization algorithm which looks to bin each of the features into sets that unbalance the classes to provide as much of a split as possible. This produces better classifiers, but it may create very fine splits for features that results in excessive binning(thus building very large conditional probability tables).

7. EXPERIMENTS

To evaluate our data mining techniques and check their abilities to improve the reference models, we have conducted a case study that includes a systematic application of the data mining algorithms to find information that may be used to enhance the reference model. To set up the systematic analysis, we have established an initial set of metrics that reflect the structure and performance requirements from the reference models and the reasoners. The experiments focus on the case study of the FuelHMA problem discussed earlier, and was conducted from the curated aircraft data .

7.1 Experiment 1

Our first task was to investigate how effective the classifier structures would be in isolating the fault condition using the condition indicator data. We used condition indicators (CIs) rather than the health indicators (HIs) in this analysis, because the intuition was that the expert-chosen thresholds were too conservative, and therefore result in unnecessary delays in the fault detection and isolation. With the fault isolated to being present between the incident and 50 flights prior, the initial dataset consisted of these features being calculated over each flight in this range. Since the indicators operate over a single engine, one could look at the features for each engine together (100 features with 50 samples), or look at the engines as each providing a sample for the flight (25 features and 200 samples). We chose this approach, since only one engine was showing degradation caused by the fault under study. We labeled the other three engines as nominal with respect to the faulty engine. The classifiers were trained and evaluated using 10-Fold Cross validation (180 samples for training, and 20 for testing) with engine data being agnostic of which engine produced the data. We utilized the TAN algorithms from the Weka implementation to produce the necessary classifiers. The fact that the TANs are discretized will be useful in identifying thresholds for otherwise continuous values. This is important in Section 7.3. This fault versus no fault classifier used all of the condition indicators described in 4. The resultant TAN structure is illustrated in Figure 6.

The classification accuracy for the TAN classifier was quite high. The TAN achieved on average 99.5% accuracy, with a .7% false positive rate and no false negative rate. These initial results were encouraging and to better understand them, we extended this experiment to determine that it was the fault and not a particular engine characteristic that was being implicitly coded into the classifier (e.g. the classifier structure implied differences between engine 2 and engine 4). To do this, the training was done with one Nominal Engine and the Faulty Engine. Then the other two nominal engines were used as the test data. If the engine data was split between the nominal engine and the faulty, that would indicate whether the classifier identified sympathetic effects due to the engine placement on the aircraft and not between nominal and faulty behavior. Experimental combinations run with all three nominal engines show that the behavior most likely being discovered was faulty behavior with a nominal baseline. These results allowed us to move along and try and help the domain expert better understand the fault and the relationship to the CIs.

7.2 Experiment 2

The success of the classification task led to the next step of working with the domain expert to understand and refine the model that accounted for the evolving degradation phenomena. To start, the expert examined the TAN created over the 50 flight set used in Experiment 1 and shown in Figure 6. He noted the complex relationship between CIs (rolltime,dipEGTC) from the Shutdown phase of the flight and the CIs (PeakEGTC and Starttime) from the Startup phase. He determined it likely there was a cycle starting in the shutdown at flight n, and leading to the startup of flight $n + 1$.

To investigate this possibility, we designed an experiment to track how the causal structure and accuracy changed as the data used to train the classifier got further away from the incident. The idea was that the effects that would most explicitly show a failure would be less emphasized. The 50 flights were divided into 5 bins of 10 flights each. A test set was constructed of the remaining 40 flights (nominal and faulty labels) as well as the samples of engine 3s CIs from every flight after the incident labeled as nominal, since no other occurrence of this fault was found in the ASIAS database. From this test set, we measured the accuracy, the false positive rate(FP%), as well as the Observational Root Node, and its immediate children. Table 1 shows the results generated from this experiment.

The domain expert expected the accuracy to be high for bins closer to the incident and false positives to be lower as compared to the bins further away from the incident. The results for both show partial agreement. Bin 1 indeed has the highest accuracy, and also the lowest false positive rate, the next lowest is indeed further away from the incident but also has accuracy that compares with Bin 1. The domain expert's opinion was that it was best to look at the results between Bin 1 and Bin 4, where the accuracy and false positive rates are the best. From this, he then proceeded to look at the structure of the TAN for each bin. The results showed two CIs that seemed to be causally related to one another, startTime and the peakEGTC. This connection is observed directly in Bin

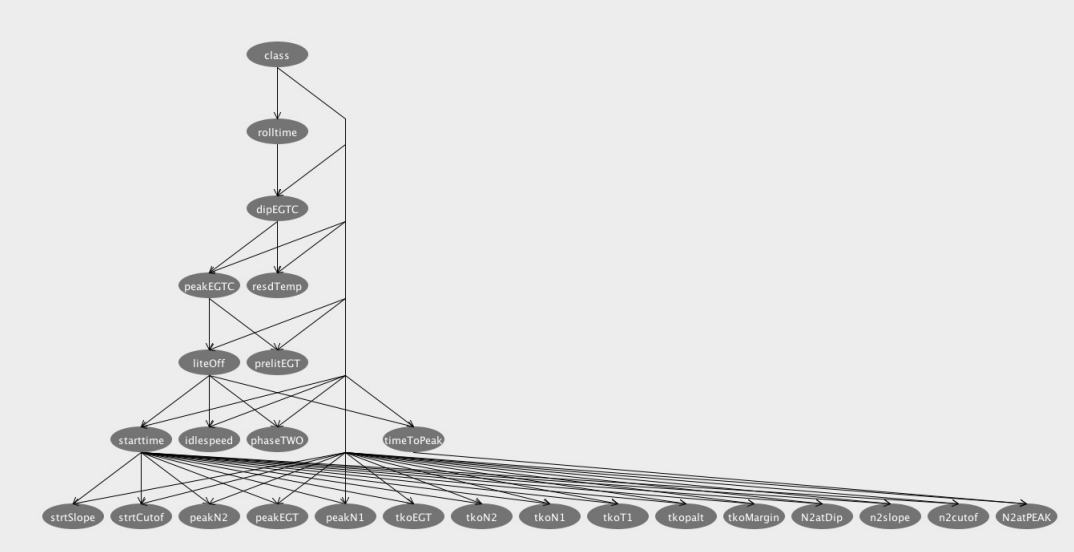

Figure 6: TAN Structure from the 50 Flights

Bin	Training Flights	Acc.on Holdout Set	FP%	Obs. Root Node	Children of ORN	Notes
1	1 to 10	97.65%	2.30%	IdleSpeed	StartTime	Thresholds Chosen from this Bin due to low FP
2	11 to 20	93.90%	5.70%	peakEGTC	liteOff,dipEGTC	peakEGTC Important Node
3	21 to 30	94.65%	5.30%	peakEGTC	liteOff,dipEGTC	peakEGTC Important Node
4	31 to 40	96.62%	3.50%	startTime	peakEGTC	Links startTime and PeakEGTC
5	41 to 50	96.06%	4.10%	liteOff	phaseTwo,RollTime	Links Startup and Rolldown CI

Table 1: Accuracy, False Positive Rate and Notes on the Different Classifiers

4, but startTime is a highly ranked node in Bin 1 and PeakEGTC is the root node in Bins 2 and 3. From this information, the domain expert believes that their causal connection, implemented as a new monitor, would lead to improvements in the reference model performance.

7.3 Experiment 3

The results from Experiment 2 provided the domain expert with explicit changes in the form of updates to the threshold values as well as additional links that the expert could add to the current reference model. The threshold values were applied to generating the HIs from the CIs. The CPTs generated by the learned classifier were discretized bins that use split points that correspond to the threshold. Looking at the bins, the lowest false positive rate occurred in Bin 1. For updating the thresholds, the originals are compared with the CPT splits in the TAN. For the rest, their causality with respect to a observational parent is removed by marginalizing the table to remove that dimension. From that point, the table will list the probability that the values based on the split indicates a nominal versus a faulty split. These splits are collected and a domain expert could elect to add them to the reference model, thus updating the thresholds and potentially improving the accuracy of catching the fault without becoming too noisy.

From this data, the domain expert discovered something else about the health indicators. While there was a Slow Start HI, the problem with this fault is that the startTime during start up wasn't that it was too slow; it's that there was a split that showed the fault occurring if the time was also too fast. From this, a new HI was made and added to the reference model. The fastStart HI looks to see if it is below the threshold outlined by the effect startTime has on the TAN for this fault.

Also, as mentioned in Experiment 2 above, there was a causal relationship appearing between startTime and peakEGTC. The domain expert suggested adding this as a "super monitor". This new HI would effectively use logic to see when the fastStart HI and the HighTemp HI both indict the failure mode in the reference model. If they both fired, then this new monitor would also fire implying that there was a simple causality that could be captured. It is worth noting that due to the type of monitors and when they are evaluated, this type of "super monitor" could contain even more complex interactions. In other words, joint occurrence of these two monitors provides stronger evidence of the fault than if one considers the effect of the two monitors individually. For example, in the original structure that showed a possible relationship between monitors in flight N and flight N+1, the causality might cause this new monitor to fire only when the two HI involved fire with that explicit timing. This would help capture that both HI are useful in indicting the failure mode, but that they also have a cycle which is useful to capture.

To show that these two new monitors as well as the updated thresholds are useful, traces were made of these 50 flights, including 10 nominal flights after the problem was caught and corrected. The first is only a recording of the original monitors. The second trace includes the new material as well the updated information. Run separately, they can tell us if the reasoner finds the fault sooner in the trace and indicates that maintenance is more than likely needed for the aircraft.

The results from the reasoner simulations are shown in Figures 7 and 8. The traces illustrate the reasoner's conclusions at different points in time as it gets closer to the incident. These points indicate how far away before the adverse event the reasoner would be able to indicate that there is a problem and preventive maintenance could be applied. With the original reference model the reasoner was unable to disambiguate between three potential fault candidates at any point leading up to the event. All of the fault candidate hypotheses were waiting for more evidence to support the isolation task. This outcome would be the same as the situation from which the data was extracted, where the engine would suffer from an unfortunate shutdown and in-flight emergency. Figure 8 describes the trace of the reasoner using the new reference model. Using the updated thresholds and the two new monitors suggested by the data mining algorithms brought a correct isolation of the fault as a fuel HMA problem. In this case, the reasoner originally hypothesized five fault conditions: four of these were at the faulty engine and one was at the vehicle level. As further monitor information became available, fuel metering remained the only plausible candidate, which means that the reasoner was able to disambiguate. That this isolation by the reasoner occurred 30 flights before the incident is significant. Not only does the aircraft capture this problem, but with enough time to give the maintenance crews time to fix the problem before an emergency has to occur.

This experiment provides encouraging results in the area of model improvement through data mining. It indicates that it may be possible to uncover new information about the relationship between components on a vehicle and how they can be harnessed to increase diagnostic reasoning. Not only can it help isolate faults, but also potentially catch them earlier in the cycle. These three experiments provide a general direction to assisting a domain expert in improving their work, and giving them access to new or missing information.

8. CONCLUSIONS AND FUTURE WORK

The overall results the single case study conducted on the data indicate positive results and show the promise of the methodology and process we have been developing. To further validate our work, we have identified a number of directions and tasks we need to pursue as we move forward in this project.

- Validate the approach and classifier structures generated by looking at additional engine data sets from data that report the same and related adverse events. To establish the robustness of our work, it is important to extend our analysis to looking at multiple occurrences of the same adverse event, and comparing the thresholds, relations, and monitor structures generated by the extended data analysis.
- Need to extend the analysis beyond single systems and subsystems. A rich source of information about fault effects involves looking at the interactions between subsystems, especially after fault occurrence. Of particular interest is looking at cascades of monitors and cascades of faults. In this framework, studying the response of the avionics systems under different fault conditions would be very useful.

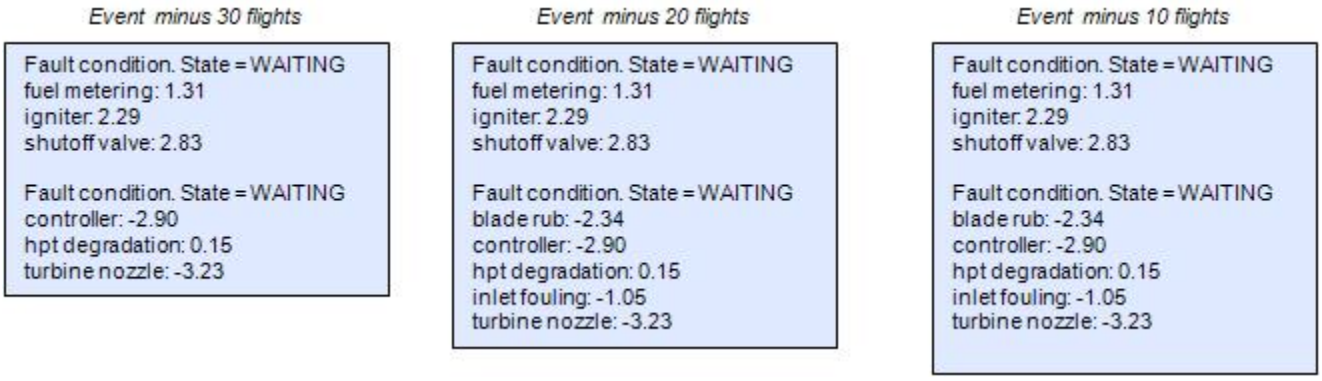

Figure 7: Trace of the Reasoner on the Original Reference Model

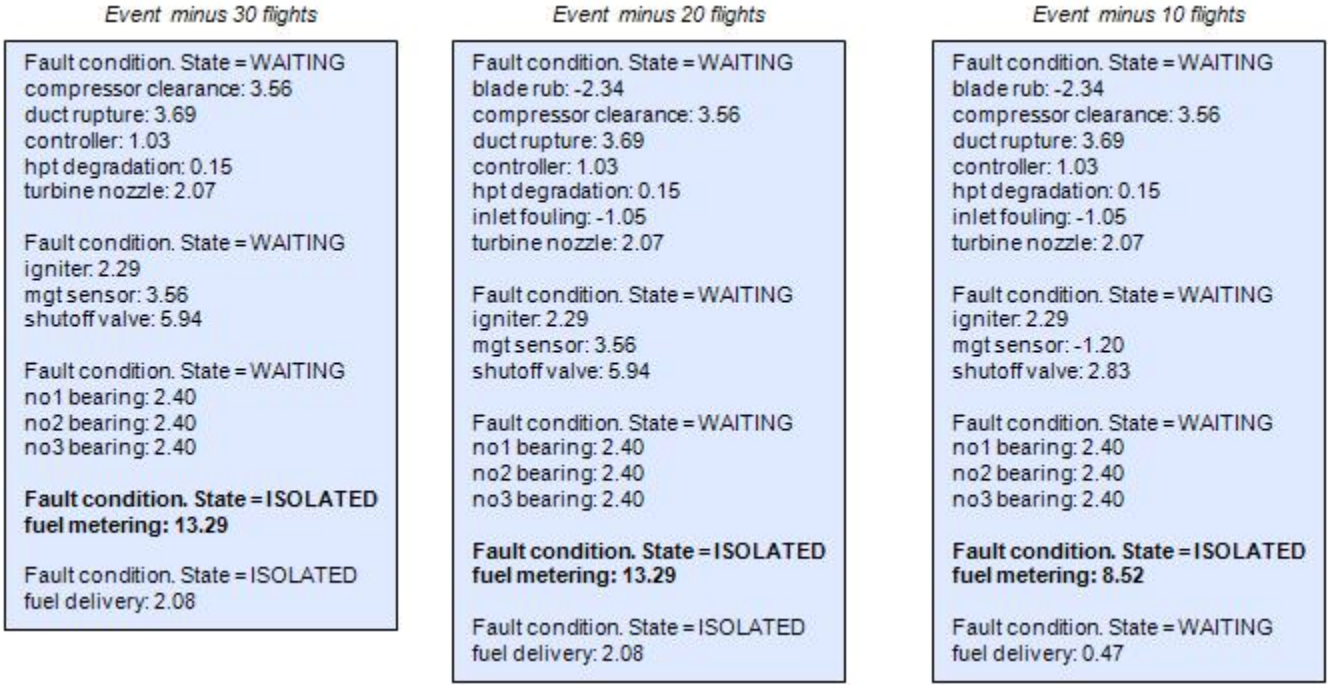

Figure 8: Trace of the Reasoner with the improved Reference Model

- Similarly, use data mining techniques to extract causal relations between avionics and other subsystem reports, as well as correlating the avionics and engine reports to adverse vehicle events, such as the in-flight engine shutdown and bird strikes would be very useful.

ACKNOWLEDGMENTS

The Honeywell and Vanderbilt researchers were partially supported by the National Aeronautics and Space Administration under contract NNL09AA08B. We would like to acknowledge the support from Eric Cooper from NASA; Joel Bock and Onder Uluyol at Honeywell for help with parsing and decoding the aircraft raw data.

REFERENCES

Chickering, D. M., Heckerman, D., & Meek, C. (1997). A Bayesian approach to learning Bayesian networks with local structure. In *In Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence.* Morgan Kaufmann.

Cohen, I., Goldszmidt, M., Kelly, T., Symons, J., & Chase, J. S. (2004). Correlating instrumentation data to system states: a building block for automated diagnosis and control. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6* (pp. 16–16). Berkeley, CA, USA: USENIX Association.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning*, *29*, 131–163.

Hall, M., Eibe, F., Holmes, B., Geoffrey amd Pfahringer, Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, *11*(1), pp. 10-18.

Kruskal, J., Joseph B. (1956). On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, *7*(1), pp. 48-50.

Murphy, K. (2011). *Bayesian Net ToolBox @ONLINE.* Available from `http://code.google.com/p/bnt/`

Schwarz, G. (1978). Estimating the Dimension of a Model,. *Annals of Statistics*, *6*.

Spitzer, C. (2007). Honeywell Primus Epic Aircraft Diagnostic and Maintenance System. *Digital Avionics Handbook*(2), pp. 22-23.