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: (1) a reference model that provides a mathematical representation for various diagnostic monitors that provide partial evidence towards active failure modes, and (2) a reasoning algorithm that combines set-covering and probabilistic computation to establish fault candidates and their rankings. However, for complex systems reference models are typically incomplete, and simplifying assumptions are made to make the reasoning algorithms tractable. Incompleteness in the reference models can take several forms, such as absence of discriminating evidence, and errors and incompleteness in the mapping between evidence and failure modes. Inaccuracies in the reasoning algorithm arise from the use of simplified noise models and independence assumptions about the evidence and the faults. Recently, data mining approaches have been proposed to help mitigate some of the problems with the reference models and reasoning schemes. This paper describes a Tree Augmented Naïve Bayesian Classifier (TAN) that forms the basis for systematically extending aircraft diagnosis reference models using flight data from systems operating with and without faults. The performance of the TAN models is investigated by comparing them against an expert supplied reference model. The results demonstrate that the generated TAN structures can be used by human experts to identify improvements to the reference model, by adding (1) new causal links that relate evidence to faults, and different pieces of evidence, and (2) updated thresholds and new monitors that facilitate the derivation of more precise evidence from the sensor data. A case study shows that this improves overall reasoner performance.

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, which consists of several subsystems such as propulsion, avionics, bleed, flight control, and electrical; each of these subsystems consists of several dozen interacting components within and between subsystems. Faults can arise in one or more aircraft subsystems; 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 deal with 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 reasoning software to automatically interpret the evidence generated by them, and infer 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 separates 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 gets generated by aircraft installed sensors, a reasoning algorithm "walks" the relevant causal paths and infers 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 onboard diagnostic reasoning using the reference model. It provides the necessary transparency for certification as a safety system, while allowing the subsystem manufacturer to encode proprietary fault models. The generation of this reference model is mostly 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" the new knowledge from an operating fleet, they are translated into expert heuristics that are added to the specific aircraft model, rather than applying systematic upgrades to the overall 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, there is a missing link in making 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 built for classical fault diagnosis, wherein the nodes represent system faults and failures (causes) and available diagnostic evidence (symptoms)(Pearl, 1988). 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). We adopt this approach, and develop a data mining approach to updating existing reference models with new causal information.

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 that formally defines the model augmentation to be derived using operational data. Next, section 4. describes the available historic data surrounding the adverse event. Section 5. briefly discusses the challenges in taking operational data and transforming it into a form that can be used by the data mining algorithms. Section 6. then discusses these data mining algorithms employed for constructing the diagnostic classifiers as Tree-Augmented Naïve Bayesian Networks (TANs). Section 7. presents experimental results of this case study to show how a human expert could utilize the classifier structure derived from flight data to improve a reference model. Metrics are defined for evaluating classifier performance, and a number of different experiments are run to determine when improvements can be made in the existing model. Section 8. presents a summary of the approach, and outlines directions for future work for diagnostic and prognostic reasoning using the data mining algorithms.

2. BACKGROUND ON REFERENCE MODELS AND REASONERS

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, as 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 captures 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, which is indicated by a 1 (occurring) or 0 (not occurring) state. Often a -1 an unknown state is also included in the initial state description. The following are 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 an a priori probability of occurring in the system. This probability is given by $P(fm_i = 1)$. Failure modes are assumed to be independent of one another, i.e., given any two failure modes fm_k and fm_j, $P(fm_k = 1|fm_j = 1) = P(fm_k = 1)$.

To isolate and disambiguate the failure modes, component suppliers also define an entity called "evidence" that is linked to sensors and monitors in the system. The set DM denotes all distinct diagnostic monitors defined for the system under consideration. A diagnostic monitor associated with $m_j \in DM$, can either *indict* or *exonerate* a subset of failure modes called its ambiguity group. In other words, each monitor m_i in the system is labeled by three mutually exclusive values allowing a monitor to express indicting, 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 m_j 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 monitor m_j will provide 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 j^{th} monitor with respect to failure mode fm_i. A monitor may fire when none of the failure modes in its indicting set are present in the system. *False alarm probability* is the probability that an indicting monitor fires when its corresponding failure modes in its ambiguity group are not present in the system. That is,

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

Designing a monitor often requires deep domain knowledge about the component or subsystem, but the details of this information may not be important from the reasoner's viewpoint. A more abstract view of the monitor is employed in the reasoning algorithm. This abstraction is shown in Figure 1. With few exceptions, most

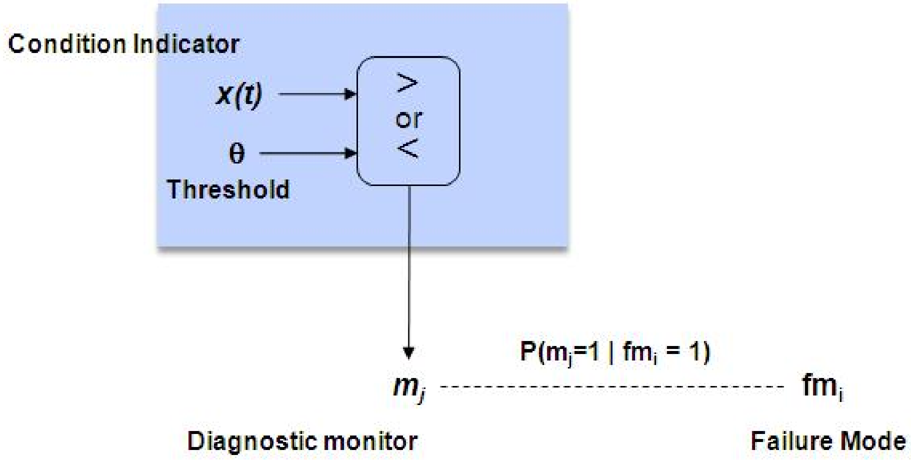


Figure 1: Abstraction of Diagnostic monitor

diagnostic monitors are derived by applying a threshold to a time-series signal. This signal can be a raw sensor value or a derived quantity from a set of one or more sensor values. We call this a condition indicator (CI) and denote it as $x(t)$. Assuming a pre-defined threshold value θ, we set $m = 1 \Leftrightarrow x(t) \lesseqgtr \theta$. 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 $m_i \in DM$ on the right. Each link has an associated detection probability, i.e., conditional probability $P(m_j = 1|fm_i = 1)$. In addition, fault 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 scheme 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 indicting evidence, for example $m_2 = 1$ or they may be exonerating evidence, 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 highest-ranked or the most likely hypothesis. The reasoning algorithm can generate multiple single fault hypotheses, each hypothesis asserting the occurrence of exactly one failure mode in the system.

The reasoning algorithm may not succeed in reducing the ambiguity group to a single fault element. This can happen for various reasons: (1) incompleteness and errors in the reference model; (2) simplifying assumptions in the reasoning algorithm; and (3) missing evidence (monitors) that support or discriminate among fault modes. For example, a modest aircraft has over 5000 monitors and failure modes; estimating the detection probabilities, d_{ji}, even for this aircraft is a challenging offline design task. Errors in d_{ji}, and more specifically missing the link between a monitor and a failure mode (incompleteness) can adversely affect the reasoner performance. Further, to keep things simple for the reasoner, a modeler may assume that the firing events for monitors are independent. This eliminates the need for the modeler to provide joint probability values of the form, $P(m_j = 1, m_k = 1|fm_i = 1)$ (say for 4000 monitors and 1000 faults the modeler would have to provide 1.56×10^{10} probability values), and instead approximate it as $P(m_j = 1|fm_i = 1) \times P(m_k = 1|fm_i = 1)$. This reduces the total number of probabilities to 4×10^6, which is still a large number but orders of magnitude less than the number required for the joint distributions and the order of the joint distributions grow exponentially when additional monitors fire. 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. As one collects enough operational data, some of these gaps can be addressed. The collection of assumptions made about the fault hypotheses and monitors results in the probability update function for each fault hypothesis, $fm_i \forall i \in F$, being computed using a Naïve Bayes model, i.e., $P(fm_i|m_j, m_k, m_l \cdots = \alpha \times P(m_j, m_k, m_l \cdots |fm_i) = \alpha \times P(m_j|fm_i) \times$

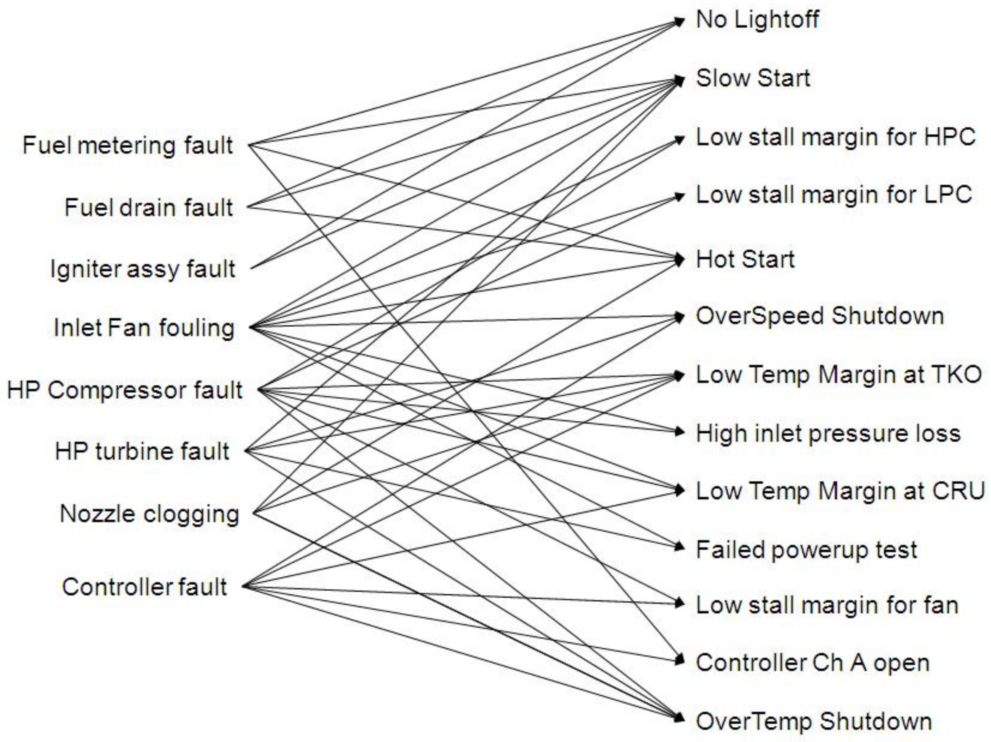


Figure 2: Example Reference Model

$P(m_j|fm_i) \times P(m_j|fm_i) \times \cdots$ where α is a normalizing constant. 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 structures that are relevant for diagnostic reasoning from data. These newly learned structures can then be used with a systematic approach for updating the system reference model. The following work focuses on this systematic approach.

3. PROBLEM STATEMENT

The combination of the reference model and reasoner 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. These networks that model diagnostic information (i.e., monitor-fault relations) can be built from data itself as a practical application of data mining. A number of Machine Learning techniques for building Bayesian networks from data have been reported in the literature (Friedman, Geiger, & Goldszmidt, 1997), (Cheng, Greiner, Kelly, Bell, & Liu, 2002),(Grossman & Domingos, 2004). For example, state-based hidden Markov Models (HMMs) (Smyth, 1994) and even more general Dynamic Bayesian Network (DBN) (Dearden & Clancy, 2001), (Lerner, Parr, Koller, & Biswas, 2000), (Roychoudhury, Biswas, & Koutsoukos, 2008), (Verma, Gordon, Simmons, & Thrun, 2004) 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, it is approached as an application that works to extend an existing ADMS. In other words, the output of the data mining algorithms have to be designed to provide information that supplements existing expert-generated reference models, as opposed to providing replacement forms of the reference model with corresponding reasoner structures. This technique can be construed as a method for supporting human experts, by having a human in the loop to interpret the findings generated by the data mining algorithm and make decisions on how to modify and update the existing reference model and reasoner structures. By including humans, who verify and integrate the model enhancements, we turn the verification into a straightforward task for the human to either approve the selected changes, or ignore them.

A systematic approach to the data mining task in this context is to discover elements of the reference model that can be augmented by comparing the results of a data driven model produced from using a learning algorithm against an existing structure extracted from the reference model. The extraction of this existing structure from the Reference Model begins by isolating a specific failure mode. The failure mode chosen is often guided by the available data where the mode was active. Isolating a single failure mode from the reference model and the monitors that indict the mode produces a tree structure where the class node describes the binary presence of that fault. The indicators(the leaves of the tree) have probabilities for the indictment of the mode and false alarm rates from the reference model that can be used to construct the complete probabilistic space. This structure and probabilistic information is the classical definition of a Naïve Bayes classifier. With data and the use of algorithms to build a Bayesian structure, the model can be leveraged to improve this very specific structure. Limiting it to a structure that preserves the general tree of the Naïve Bayes classifier eases the transition of information from the learned structure back to the original model. This is balanced with our desire to add limited causality into the network to help the expert understand if there are health indicators which are correlated. This information could be added back in limited ways to the isolated structure, without requiring the entire reference model to be changed from the Noisy-OR model. The structure that is chosen for learning is a Tree Augmented Naïve Bayesian network, which we will discuss in more detail in section 6.

The information added back to the reference model falls into three areas:

1. **Update Monitors** Update the threshold θ associated with a diagnostic monitor. The idea is to make the monitor i more sensitive to failure mode j (so that it can be detected earlier, if it is present) without sacrificing the false alarm rate, ϵ_j for the monitor.
2. **Add Monitors to Failure Mode** Update the isolated structure by adding a monitor that helps indict the failure mode. Specifically this could take two forms: (a) creating a new monitor with the requisite probabilistic information, and adding a new d_{ji} to associate it with the failure mode, and (b) assigning a non-zero number for d_{ji} if the link did not already exist with a previously created monitor.
3. **Create Super Monitors** Creating new monitors that combine existing monitors, say m_i and m_j such that the combination of monitor indictments asserts stronger evidence for a specific failure mode fm_k. That is, calculate a stronger value for $P(m_i = 1, m_j = 1|fm_k = 1)$ which is greater than $P(m_i = 1|fm_k = 1) \times P(m_j = 1|fm_k = 1)$.

In addition to establishing areas of improvement found by comparing the data mining results with the 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. Potentially, the experts may apply a successive refinement process by requesting a number of experimental runs, each using a specific data set from an operating fleet of aircraft, and the results from the nth experiment augments or confirms the reference model from the $(n-1)^{th}$ experiment. This will result in a continuous learning loop wherein historical observations from the fleet are analyzed systematically to understand the causal relations between failure modes and their manifestations (monitors). In addition, building models from the data may also reveal unknown (or currently unmodeled) dependencies among failure modes that are linked to the adverse event situations under consideration. Over time, this learning loop will increase the accuracy and time to detection (while reducing false positives) in the diagnostic reasoner.

The next step is to explore and pre-process the available aircraft flight data for the data mining task. The preprocessing plays a major role in determining the nature of information derived from the data, and, using prior knowledge of the aircraft domain the pre-processing algorithms can be tailored to avoid the "needle-in-the-haystack" search problem.

4. AIRCRAFT FLIGHT DATA

It is important to extract flight data of the right type and form that will potentially help to find and validate new diagnostic information. Since the goal is early and reliable detection of an evolving fault in the aircraft, it is important that the data set formed for analysis span several contiguous flights. This set should also include multiple aircraft to account for aircraft-to-aircraft variations and the heterogeneity of flight conditions and flight paths. Our data set comes from a fleet of aircraft belonging to a regional airline from North America. The fleet consisted of 30+ identical four engine aircraft, each operating 2–5 flights each day. This work examines data spanning three years of the airline's operations.

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 typically stored in raw, uncompressed form. The DAR can record information at a maximum rate 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 16 Hz. We apply our initial data retrieval and pre-processing algorithms to this raw time-series data that was made available to us in the form of multiple CDs.

A second source of information we referenced for this study was adverse event annotations that is available in a FAA developed Aviation Safety Information Analysis and Sharing (ASIAS) database system. The ASIAS database is a collection of adverse events reported by various airline operators. On searching this database for the time period of the flight data available to us revealed that an engine shutdown event had occurred for one of the aircraft in our list. On one of the flights of this aircraft, the third engine(out of four) aboard the aircraft shutdown automatically. As a result, the flight crew declared an emergency situation and returned back to the airport where the flight originated. Fortunately, there were no casualties or serious injuries. For this study, we decided to focus on this failure mode, mainly because the on board reasoner or the mechanics who serviced the aircraft were unable to detect any anomalies in the engine till the adverse event occurred.

In more detail, an adverse event such as an engine shutdown typically evolves as a sequence of anomalous events and eventually leads to a situation, such as over heating, that causes the shutdown. For this case study, our objective was to analyze the ACMS flight data from the aircraft prior to adverse event with the goal of defining anomalies in the system monitors that were not defined for the existing ADMS. The primary intent was to use these anomalous monitor reports to detect and isolate the root cause for the failure as early as possible, so that the onset of the adverse event could be avoided.

Investigation of the airline maintenance crew reports after the adverse event revealed that the root cause for this adverse event was a faulty fuel metering hydromechanical unit(Fuel HMA) in the third engine, which was the engine that shut down. The fuel metering unit is a controller-actuator that meters fuel into the combustion chamber of the engine to produce the desired thrust. Given that we now knew the time of the adverse event and the root cause for the particular engine failure, knowledge of the fuel metering unit impliedthat this was a slowly evolving (i.e., *incipient*) fault that could very likely start manifesting about 50 flights *before* the actual engine shutdown adverse event. Therefore, we extracted the $[-50, 0]$ flight interval for the analysis, where 0 indicates the flight number for which the adverse event occurred and -50 indicates 50 flights before this one. We assumed that the particular aircraft under study had one faulty and three nominal engines. We collected relevant data (discussed below) for all of the engines, and then ran Bayesian classifiers to discover the differences between the faulty and the nominal engines.

As we discussed earlier, all of the aircraft for the regional airline were equipped with ADMS. The diagnoser receives information at pre-defined rates from the diagnostic monitors. In this case study, we also assume that we had access to the sequence of condition indicators(CI) values that were generated. As discussed earlier, the binary monitor output is produced by applying a predefined threshold to the CI values. In our data mining analysis, we use the CI's as features, and compare the thresholds derived by the classifier algorithms against the thresholds defined in the reference model. The following condition indicators and diagnostic monitors were available from this aircraft flight dataset.

StartTime This CI provides the time the engine takes to reach its idling speed. Appropriate threshold generates the *no start* diagnostic monitor.

IdleSpeed This CI provides the steady state idling 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 CIs provide 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 CIs.

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 CIs provide the engine speed and the 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.

This large volume of CI data (multiple aircraft, multiple flights) provides opportunities to study aircraft engines in different operating scenarios in great depth and detail. However, the data as extracted from the raw flight data DAR files was not in a form that could be directly processed by our classification algorithms. We had to develop data curation methods to generate the data sets that could be analyzed by the machine learning algorithms.

5. DATA CURATION

An important requirement for the success of data driven techniques for knowledge discovery is the need to have relevant and well-organized data. In our study, well-organized implied getting rid of unwanted details, being able to structure the data on a timeline (having all of the CI's aligned in time, and the monitor output inserted into the time line as a sequence of events), and applying filtering algorithms to the noisy sensor data. Relevance is an important concept, since it is necessary to extract sequences of data that 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 that the classifier be provided with both nominal and faulty data, so that it can derive the discriminating features from the data. Further, the systems under study are complex, and they operate in different modes and under different circumstances. This information is likely to be important for the classification task, so the data needs to be appropriately annotated with this information. It is clear that unreliable data is unlikely to provide useful information to an already effective reference model. Our (and others) experiences show that the data curation task is often more time consuming and sometimes quite difficult (because of noisy, missing, and unorganized data) as compared to the data mining task, which involved running a classifier or a clustering algorithm on the curated data. A good understanding of the nature of the data and how it was acquired is critical to the success of the data mining task.

In our study, the DAR files represented single flights encoded in binary format. As a first step, we organized several thousands of these files by the aircraft tail number. For each aircraft, the data was then organized chronologically using the time stamp associated with the particular flight. Since the case study involves an engine shutdown situation, the data was further classified based on the engine serial number so that the data associated with each engine could be easily identified.

For practical reasons, given the size of the data, and the need to extract specific sub-sequences for the data mining task, we designed a relational database to create an organized representation for the formatted data. This made it easier to access the relevant data for different experimental analyses. For a data analysis session, step 1 involved formulating data base queries and collecting the extracted data segments into the form required by the classification algorithm. Step 2 was the data curation step. A primary task at this step was removing all extraneous non-flight data. For this analysis, all ground-test information (data generated when the maintenance crew ran a test when the aircraft was on the ground) was defined as anomalous and removed during the cleansing step. Step 3 involved running the classification algorithms.

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 addressed using the data. The extracted portion of the reference model discussed earlier can be modeled as a Naïve Bayes reasoner. The independence assumptions of the model may also be systematically relaxed to capture more discriminatory evidence for diagnosis. There are several interesting alternatives, but one that fits well with the isolated structure is the Tree Augmented Naïve Bayesian Method (Friedman et al., 1997) abbreviated as the TAN algorithm. The TAN network is a simple extension to the Naïve Bayes network formulation. The Root (the fault mode) also know as the class node is causally related to every evidence node. In addition, the independence assumption for evidence nodes is relaxed. An evidence node can have at most two parents: one is the class node, the other can be a causal connection to another evidence node. These constraints maintain the directed acyclic graph requirements and produce a more nuanced tree that captures additional relationships among the system sensors and monitors. At the same time, the learning algorithm to generate the parameters of this structure is computationally simpler than learning a general Bayes net structure.

The TAN Structure can be generated in several different ways. One approach uses 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). Another procedure, sketched out in Algorithm 1, builds a Minimum Weighted Spanning Tree (MWST) of the evidence nodes and then connects the fault mode to all of the evidence nodes in the tree (Friedman et al., 1997). We use this algorithm in our work. A standard algorithm (e.g., Kruskal's algorithm(Kruskal, 1956)) is applied to generate the MWST. The edge weight computation for the tree structure utilizes a log-likelihood criterion, such as the Bayesian likelihood value (Chickering, Heckerman, & Meek, 1997) and the Bayesian Information Criterion (BIC) (Schwarz, 1978). If the values are naturally discrete or they represent discretized continuous values, the Bayesian likelihood metric is preferred. This is a simple metric, which calculates the likelihood that two variables are dependent. The BIC is better suited for data sets whose features are derived from continuous distributions (like a Gaussian Normal). For either measure, the values are calculated for every pair of evidence nodes and stored in a matrix. Note that the value calculated for node i to node j is different for the value calculated for node j to node i, Therefore, the directed edges of the MWST represent the implied direction of causality, and the derived structure includes preferred causal directions (and not just correlational information).

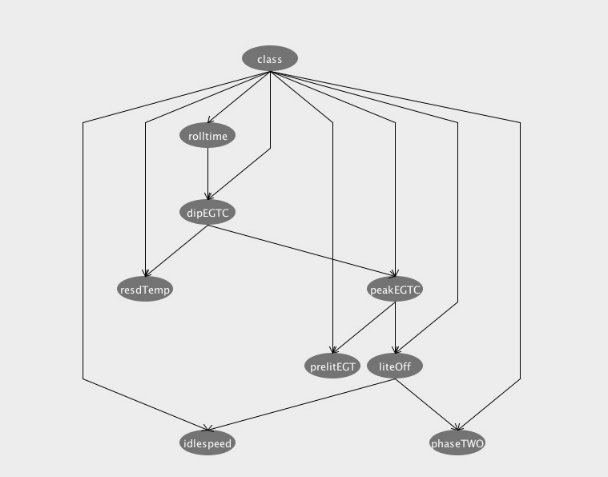


Figure 3: Example TAN Structure

An example TAN structure is illustrated in Figure 3. The root node, labeled class, is the fault hypothesis of interest. The other nodes represent evidence supporting the particular fault hypotheses. For the structure in Figure 3, rolltime, a monitor associated with the shutdown phase of the aircraft is the anchor evidence node in the TAN structure, called the *observation root node*. Like a Naïve Bayesian classifier, the fault hypothesis node (class) is linked to all of the relevant monitor nodes that support this hypothesis. Dependencies among some of the monitors, e.g., rolltime and dipEGTC, are captured as additional links in the Bayesian network. Note that the TAN represents a static structure; it does not explicitly capture temporal relations among the evidence. The choice of the observation root node is important; in some ways, it represents an important monitor for the fault hypothesis, since it is directly linked to this node. This means the distribution used in the observation root node(whether it be a discrete CPT, or a continuous distribution) is conditioned only on the priors of the class distribution. The rest of the MWST structure is also linked to this node. All other conditional probability tables(CPTs) generated for this TAN structure include the class node and at most one other evidence node. The choice of the observation root node may determine the overall structure of the tree, but for the same data, the procedure for identifying the stronger causal links should not change in any significant way, i.e., the strong links will appear on all TANs, irrespective of the choice of the observation root node.

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: $(AdjMat, ProbVec)$
```

When the inference is used to assign a faulty or nominal label to the observed flight data, the result will be biased towards one class(fault) over another based on the CI value of the observation root node. 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. In a later section we discuss how these choices can be used by the domain expert to make effective improvements to the reference model for the AHM.

This choice of the observation root node, as shown in Algorithm 1 is an input parameter to the algorithm. This choice is normally based on a ranking computed using a heuristic, such as the highest BIC value. The algorithm in Weka (Hall et al., 2009) builds TANs with every feature as the root node of the MWST. It compares the generated structures, using a scoring metric such as the log-likelihood for the training data. The structure with the best score is then chosen as the classifier structure. Another approach could use domain knowledge to choose this node. 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.

Consider the example TAN shown in Figure 4. When the data for constructing the TAN is extracted from flights just before the adverse event occurred, the root node chosen by the Weka scheme is idlespeed. This node connects to the rest of the MWST, which in this case is the starttime feature, to which the rest of the feature nodes are connected. Using data from flights that were further away (before) from adverse event occurrence, the Weka algorithm picked PeakEGTC as the root node. This is illustrated in TAN structure in Figure 5. However, the derived causal link from idlespeed to starttime to a large group of nodes is retained at the bottom right of Figure 5. The similarities and shifts in the TAN structures from different segments of data typically informs the domain expert about the underlying phenomena due to the fault that is captured by the monitors. We discuss this in greater detail when we develop our case study in Section 7..

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 method from the Weka (Hall et al., 2009) toolkit, uses a discretization algorithm which looks to bin each of the features into sets that unbalance the classes to provide the best possible split. For this case study, it produced more accurate classifiers, however, there were situations where it created a number of very fine splits in the feature values to define all of the class structures. The result was excessive binning, which produced very large conditional probability tables. When considering a more general view, methods that produce excessive binning are likely to be less robust to noise. Therefore, one has to consider these trade offs when choosing between these approaches.

7. EXPERIMENTS

To evaluate the data mining approach and demonstrate its ability to improve a diagnoser reference model, we conducted a case study using real flight data from the regional airline described earlier. We defined three standard metrics: (1) classification accuracy (2) false positive rate, and (3) false negative rate to systematically evaluate the TAN learning algorithm. Starting from the

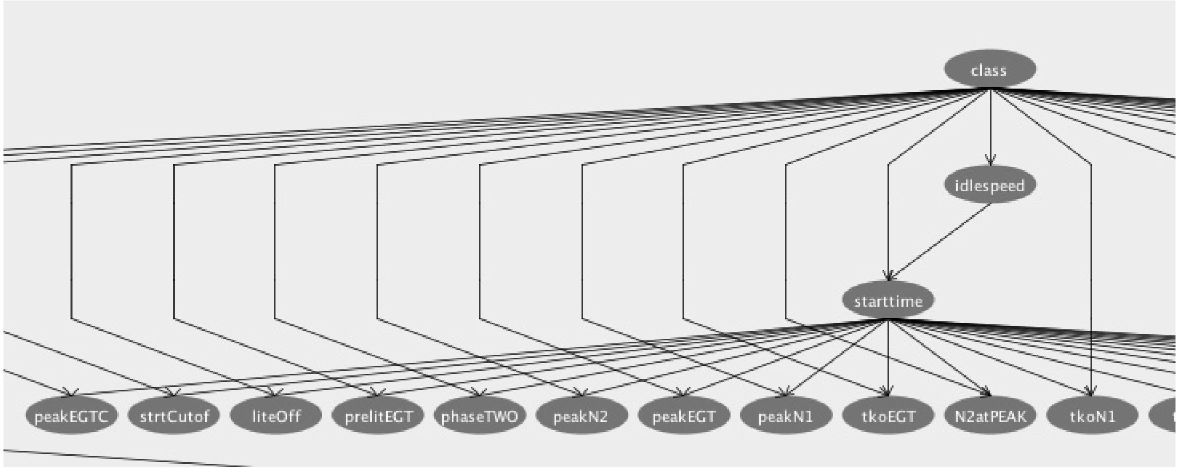


Figure 4: TAN Structure with idlespeed as observation root node

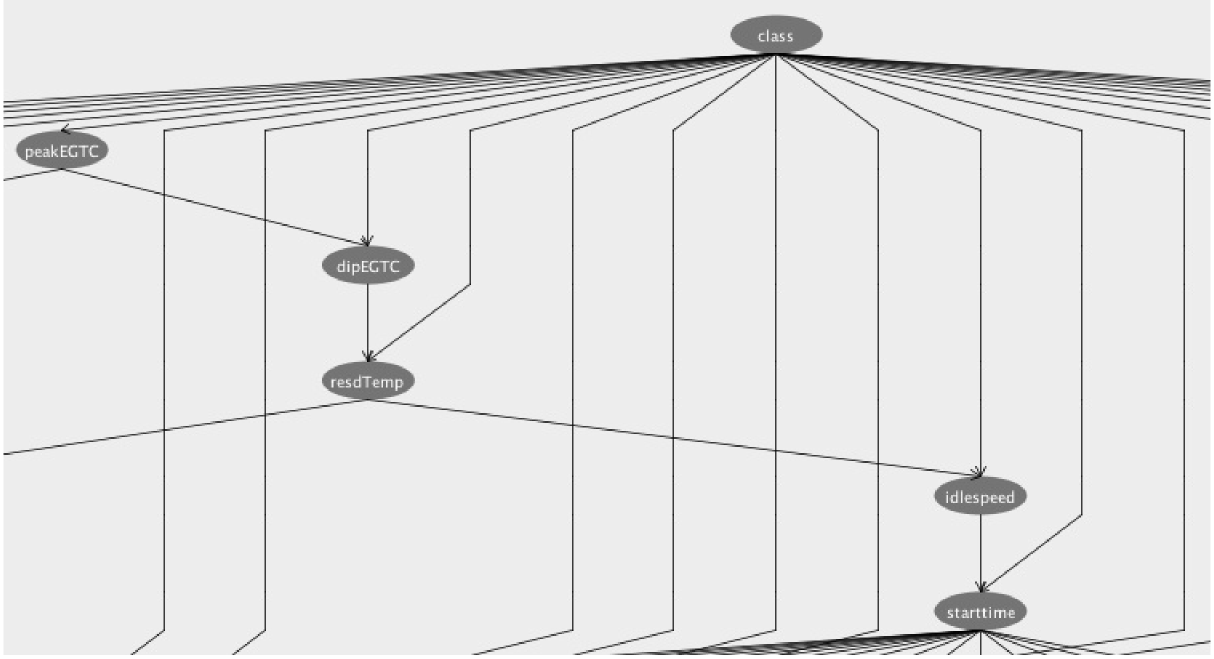


Figure 5: TAN Structure with peakEGTC as observational root node

flight in which the adverse event occurred for a particular aircraft, we used the earliest time to detection as another metric to evaluate the improvement in the system reference model after it has been updated with new information generated from the TAN classifier structure. The particular case study discussed here was an aircraft, where an overheated engine caused the engine to shutdown. This was considered to be a serious adverse event, and the pilots decided to return to the originating airport. By comparing the data from the faulty engine against the three other engines on the aircraft, which operated normally, starting from 50 flights before the adverse event, we were able to generate additional monitor information that reliably pointed to a FuelHMA problem (i.e., leak at the fuel meter) 40 flights before the actual incident. We break up the case study into three experiments and discuss their details next.

7.1 Experiment 1

The first task was to investigate the effectiveness of the generated classifier structures in isolating the fault condition using the condition indicator information derived from the flight data. We used condition indicators (CIs) rather than the health indicators (HIs) in this analysis because they make fewer assumptions about the nature of the data. We hypothesized that the original expert-supplied thresholds for the HIs were set at conservative values to minimize the chances for false alarms, and our derived classifier structures could potentially provide better thresholds without sacrificing the accuracy and false alarm metrics. This would lead to faster detection times.

From the ASIAS database, we extracted the aircraft and flight number in which the adverse event occurred. The report also indicated the nature of the fault that caused the adverse event, and knowledge of the fault provided context to our domain expert as to when this fault could be detected in the engine system. Our expert surmised that the fault would most likely start manifesting about 50 flights prior to the adverse event. The initial dataset that we then formulated consisted of the calculated CIs for all 50 of the identified flights. Each engine has its set of monitors, therefore, we had four sets of CIs, one for each engine. For analysis, we considered two ways for organizing this data. The first approach combined the four engine CI's as four separate features associated with one data point (i.e., one flight). Since we had 25 different CIs, this meant the dataset consisted of 50 data points, with each data point defined by 100 features. The second approach looked at each engine as a separate data point. Therefore, we formed four datasets, each with with 50 data points and 25 features. From the problem definition, it was clear that one of the four engines was faulty, and other three were most likely nominal for the 50 flights that we were analyzing. Therefore, we chose the latter approach for representing the data. To label the dataset appropriately for the classification study that we applied in this experiment, the three engines of the aircraft that showed no abnormalities (1, 2, and 4) were labeled as nominal, and the data points corresponding to engine 3, where the shutdown incident occurred, was labeled as faulty.

The classifiers were trained and evaluated using 10-Fold Cross validation (180 samples for training, and 20 for testing) with the nominal engine data being agnostic of which engine produced the data. All of the CIs described in 4. were used as features in the classifier algorithm. The TAN generation algorithm from the Weka toolkit were used to derive the necessary classifiers. The fact that the discretized representation of the conditional probabilities were employed made it easier to find the threshold values for the diagnostic monitors that were linked to each CI. This is discussed in greater detail in Section 7.3. The derived TAN structure is illustrated in Figure 6.

The classification accuracy for this TAN structure was high, the average accuracy value was 99.5% with a .7% false positive rate and 0% false negative rate. These initial results were encouraging and to better understand them, the experiment was extended to confirm that the classifier results were be attributed to the evolving fault in engine 3 and it was not just an artifact of the differences between the different engine characteristics. The above experiment was repeated with the training data including one of the nominal engines (1, 2, or 4) and the faulty engine, 3. The other two nominal engines were used as the test data. If the classifier split the remaining nominal engine data between the nominal and faulty classes derived, this would indicate that its structure more likely an artifact of engine placement on the aircraft. This experiment was repeated two more times, each time using a different nominal engine providing the training data and the other two being used as the test data. For all 3 experiments, the fault classification accuracy remained high, indicating that the classifier was truly differentiating between the fault and no-fault conditions.

7.2 Experiment 2

The positive results from the classification task led to the next step, where we worked with a domain expert to determine which of the CIs in the classifier provided the best discrimination between the faulty and nominal conditions. This information would provide the necessary pointers to update the current reference model. As a first step, the expert examined the TAN created using data from the 50 flight set used in Experiment 1. The expert's attention was drawn to the complex relationship between certain pairs of CI's during different phases of the flight:(1) rolltime and dipEGTC during the Shutdown phase, and (2) PeakEGTC and Starttime from the Startup phase. The expert concluded that there was a likely dependence between the shutdown phase of flight n and the startup of the next flight, $n + 1$. The reasoning was that an incomplete or inefficient shutdown in the previous flight created situations where the startup phase of the next flight was affected. The expert hypothesized that this cycle of degradation from previous shutdown to the next startup resulted in the fault effect becoming larger and larger, and eventually it would impact a number of CIs of the faulty engine.

This phenomena was investigated further by designing an experiment to track how the causal structure and accuracy of the classifiers derived from different segments of data. The different segments were chosen as intervals of flights before the flight with the adverse event occurrence as shown in Table 1. The 50 flights were divided into 5 bins of 10 flights each. A test set was constructed from the remaining 40 flights (data with nominal and faulty labels) as well as the samples of CIs from engine 3 after it had been repaired (after engine 3 was repaired,

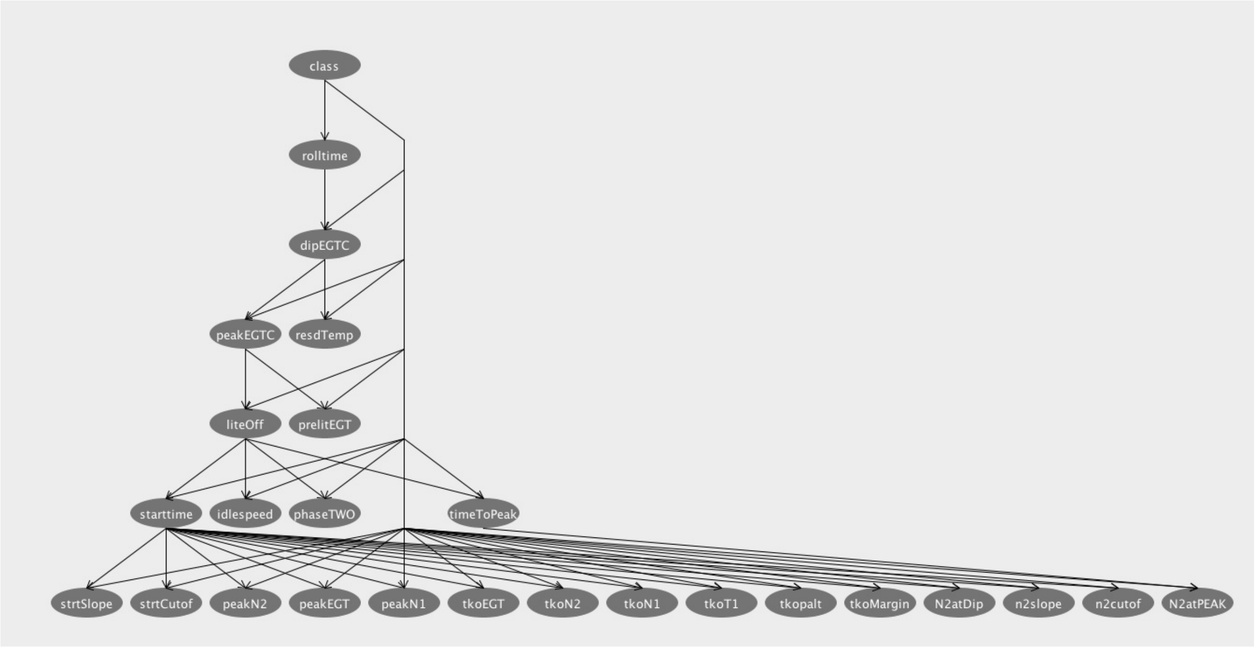


Figure 6: TAN Structure Generated using Data from all 50 Flights

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, Observational Root Node and Immediate Child Node for Classifiers Created from different data segments

no anomaly reports were generated by mechanics, and no further adverse events were reported for engine 3 in the ASIAS database, therefore, it was considered to be nominal). Table 1 shows the accuracy and false positive rate(FP%) metrics reported for the five experiments. The observation root node, and its immediate child in the generated TAN structures are also shown.

The conventional wisdom was that the accuracy and false positive metrics would have the best values for the classifiers generated from data close to the adverse event, and performance would deteriorate for the TAN structures derived from bins that were further away from the incident. The actual results show partial agreement. The bin 1 experiment produced the highest accuracy and lowest false positive rate, but the next best result is produced for TAN classifiers generated from the bin 4 data. This prompted the domain expert to study the bin 1 to bin 4 TANs more closely. The expert concluded that two CIs, startTime and peakEGTC showed a strong causal connection for bin 4, and startTime was highly ranked for the bin 1 TAN. On the other hand, PeakEGTC was the root node for bins 2 and 3. This study led the domain expert to believe that a new monitor that combined startTime and peakEGTC would produce a reference model with better detection and isolation capabilities. The process of designing and testing the diagnoser with the new monitor is described as Experiment 3.

7.3 Experiment 3

Working closely with the data mining researchers, the domain expert used the framework in the problem statement to reconcile the results from Experiment 2 to suggest explicit changes in the reference model; this included: (1) updates to the threshold values that specified the diagnostic monitors (i.e., updated HIs from the CIs), (2) a new monitor that could be added to the current reference model and (3) the addition of a "Super Monitor" to the reference model.

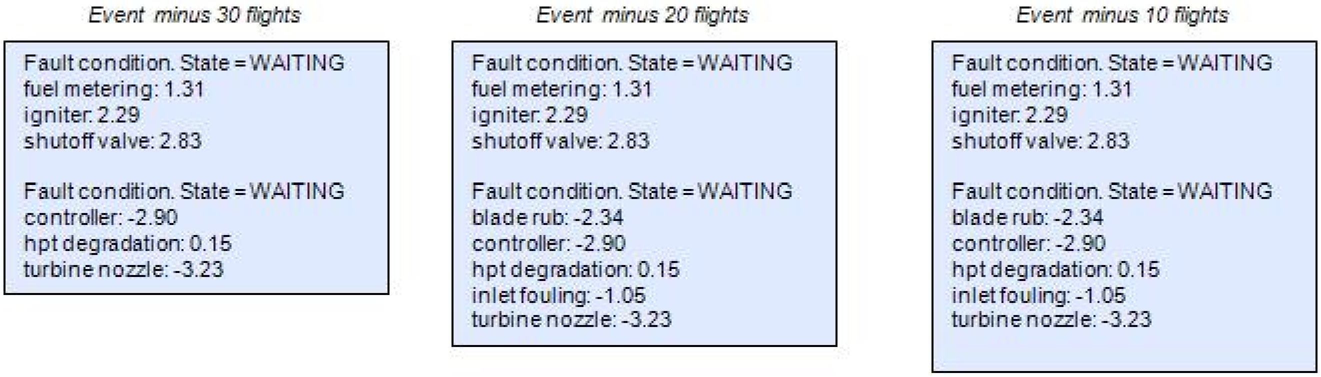
The CPTs generated by the learned classifier were defined as discretized bins with split points that could be interpreted as thresholds. Looking at the bins, the lowest false positive rate occurred in bin 1. For the observation root node, the thresholds were updated using the results for bin 1 by comparing the split values with the original thresholds. For the remaining nodes, their causality with respect to the observation parent was removed by marginalizing to remove that particular variable. Once marginalization is applied, the CPT lists the probability values at the nominal versus faulty split points. The domain expert studied these split values and made decisions on whether the new split values should update the thresholds in the reference model. The trade off was to improve the accuracy of fault detection without introducing too much noise (uncertainty) into the decision process.

Studying the TAN structures provided additional information to the domain expert. When the slowStart HI fired, the expert discovered that this was not because the startTime during start up was slow; sometimes the fault occurred when the startTime was too fast. This implied a new HI could be added to the failure mode that now examines if startTime is under a threshold and too fast. The addition of this HI(called fastStart) to the reference model would be to speed up detection by adding new evidence to indict the fault.

Experiment 2 also showed a causal relationship appearing between startTime and peakEGTC. The domain expert suggested adding this as a "super monitor". This new HI would combine information from the fastStart HI and the HighTemp HI to identify the fuelHMA fault in the reference model. In other words, if both monitors fired, then this new monitor would also fire directly implicating the fault hypothesis. 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 in that explicit sequence, flight n and flight $n + 1$. Not only does this super monitor combine the results from other monitors, but it also indicates cyclic behaviors that again provide very useful diagnostic information. In general, these "super monitors" could model complex interactions thus increasing the overall discriminability properties of the reasoner. The consequence of using a super monitor, is that the usefulness of the two monitors used in the construction are lost. These are removed from the links to the failure mode being examines(however they remain for any other failure mode). In this situation, the just created monitor for fast startTimes would be removed as well as the HighTemp HI in place of a new super monitor for a fast start and a high engine temperature on start up.

To show that this new super monitor and the updated thresholds produce better results, multiple traces of the monitor output were made for the 50 flight data set. This included 10 nominal flights after the problem was caught and corrected. The first trace is only a recording of the original monitors designed by the expert. The second trace includes the new monitors(both the fast startTime monitor and "super monitor") derived by the data mining analyses, as well the updated information(thresholds). Run separately, they can be analyzed to determine if the reasoner finds the fault sooner in the trace and indicates that maintenance is more than likely needed for the aircraft.

These results from the reasoner simulations are shown in Figures 7 and 8. The traces illustrate the reasoner's inferences at different flight numbers before the actual incident occurrence. This analysis demonstrates how far before the adverse event the reasoner would reliably detect the fault, and potentially generate a report that would lead to preventive maintenance, and, therefore, avoidance of the adverse event. 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 required more evidence to support the isolation task. This would not avoid the unfortunate shutdown and the emergency return to the originating airport. Figure 8 shows the reasoner trace for the new reference model. Using the updated thresholds and the new super monitor(which is derived from a new monitor itself and one original monitor) suggested by the data mining algorithms led to a correct isolation of the fault, i.e, the fuelHMA problem. In this case, the reasoner originally hypothesized five fault conditions: four of these were linked to the faulty engine and one was a vehicle level hypothesis. As further monitor information became available, fuel metering re-



Event minus 30 flights

Fault condition. State = WAITING
fuel metering: 1.31
igniter: 2.29
shutoff valve: 2.83

Fault condition. State = WAITING
controller: -2.90
hpt degradation: 0.15
turbine nozzle: -3.23

Event minus 20 flights

Fault condition. State = WAITING
fuel metering: 1.31
igniter: 2.29
shutoff valve: 2.83

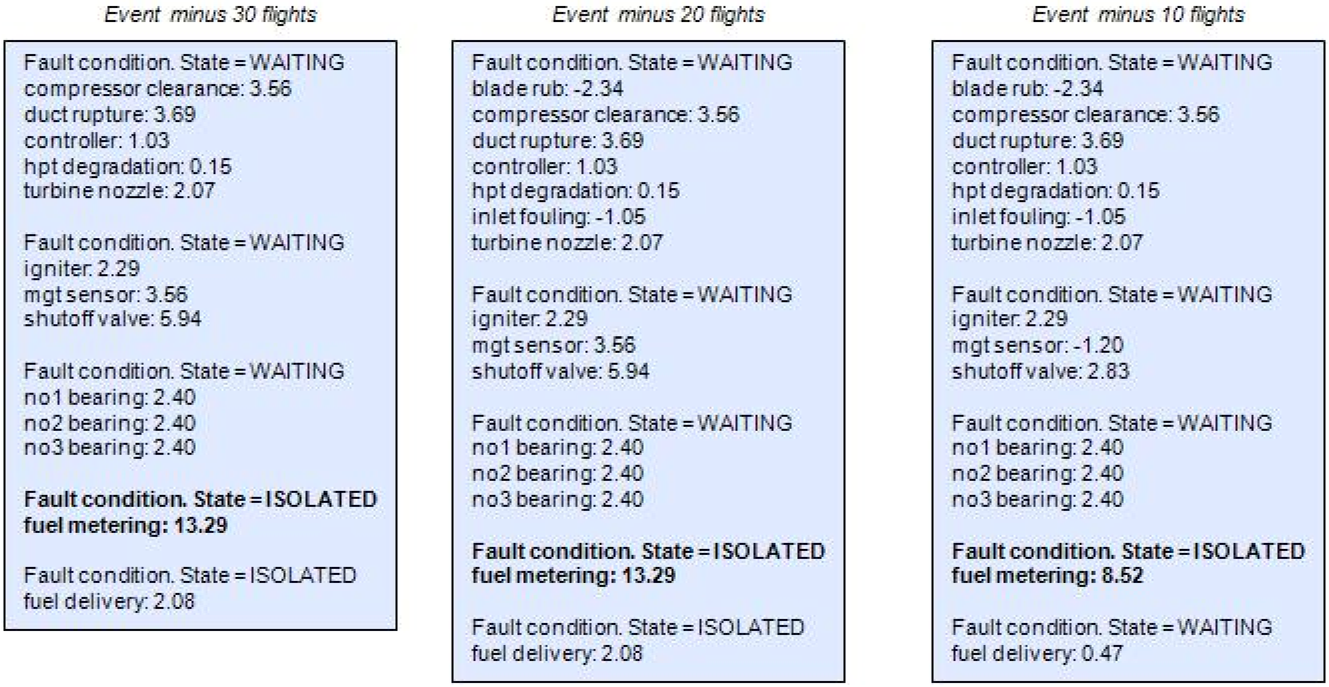
Fault condition. State = WAITING
blade rub: -2.34
controller: -2.90
hpt degradation: 0.15
inlet fouling: -1.05
turbine nozzle: -3.23

Event minus 10 flights

Fault condition. State = WAITING
fuel metering: 1.31
igniter: 2.29
shutoff valve: 2.83

Fault condition. State = WAITING
blade rub: -2.34
controller: -2.90
hpt degradation: 0.15
inlet fouling: -1.05
turbine nozzle: -3.23

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



Event minus 30 flights

Fault condition. State = WAITING
compressor clearance: 3.56
duct rupture: 3.69
controller: 1.03
hpt degradation: 0.15
turbine nozzle: 2.07

Fault condition. State = WAITING
igniter: 2.29
mgt sensor: 3.56
shutoff valve: 5.94

Fault condition. State = WAITING
no1 bearing: 2.40
no2 bearing: 2.40
no3 bearing: 2.40

Fault condition. State = ISOLATED
fuel metering: 13.29

Fault condition. State = ISOLATED
fuel delivery: 2.08

Event minus 20 flights

Fault condition. State = WAITING
blade rub: -2.34
compressor clearance: 3.56
duct rupture: 3.69
controller: 1.03
hpt degradation: 0.15
inlet fouling: -1.05
turbine nozzle: 2.07

Fault condition. State = WAITING
igniter: 2.29
mgt sensor: 3.56
shutoff valve: 5.94

Fault condition. State = WAITING
no1 bearing: 2.40
no2 bearing: 2.40
no3 bearing: 2.40

Fault condition. State = ISOLATED
fuel metering: 13.29

Fault condition. State = ISOLATED
fuel delivery: 2.08

Event minus 10 flights

Fault condition. State = WAITING
blade rub: -2.34
compressor clearance: 3.56
duct rupture: 3.69
controller: 1.03
hpt degradation: 0.15
inlet fouling: -1.05
turbine nozzle: 2.07

Fault condition. State = WAITING
igniter: 2.29
mgt sensor: -1.20
shutoff valve: 2.83

Fault condition. State = WAITING
no1 bearing: 2.40
no2 bearing: 2.40
no3 bearing: 2.40

Fault condition. State = ISOLATED
fuel metering: 8.52

Fault condition. State = WAITING
fuel delivery: 0.47

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

mained the only plausible candidate, and the fault hypothesis was established unambiguously. The fact that this isolation by the reasoner occurred 30 flights before the incident is significant because it gave sufficient advanced warning to the maintenance crews to fix the problem before an emergency situation developed.

This case study provides encouraging results in the area of diagnoser 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 improve 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 from this case study generated positive results and show the promise of the data mining methodology and the overall process that starts from data curation and ends with systematic updates to and verification of the system reference model. The results presented clearly demonstrate that the data mining approach is successful in: (1) discovering new causal relations in the reference model, (2) updating monitor thresholds, and (3) discovering new monitors that provide additional discriminatory evidence for fault detection and isolation. Experiment 3 demonstrated that the new knowledge leads to better diagnoser performance in terms: (1) early detection, and (2) better discriminability. An immediate next step will be to generalize this methodology by applying it to other adverse event situations. In the longer term, to further validate this work, we plan to advance this research in a number of different directions.

- Validation of the approach and classifier structures generated by looking at additional engine data sets from other flight data that report the same and related adverse events. To establish the robustness of the work, it is important to extend the analysis to looking at multiple occurrences of the same adverse event, and to compare the thresholds, relations, and monitor structures generated by the extended data analysis.
- Extension of the analysis methodology beyond single systems and subsystems. A rich source of information about fault effects involves looking at the interactions between subsystems, especially after fault occurrence begins to manifest. 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.
- Advance our data mining techniques to extract causal relations between avionics and other subsystems, as well as study correlations between the combined avionics and engine features and adverse vehicle events, such as in-flight engine shutdowns and bird strikes. Understanding what features of a flight differentiate situations when a pilot starts compensating for what may be a slowly degrading component that originates from a bird strike will help us gain a better understanding of how to monitor the actual operation of the aircraft with its subsystems under various conditions. This also presents interesting problems from the application and development of machine learning algorithms to utilize in this data mining problem.

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

Cheng, J., Greiner, R., Kelly, J., Bell, D., & Liu, W. (2002). Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, *137*(1-2), 43 - 90.

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.

Dearden, R., & Clancy, D. (2001). Particle filters for real-time fault detection in planetary rovers. In *Proc. of the 12th International Workshop on Principles of Diagnosis* (p. 1-6).

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

Grossman, D., & Domingos, P. (2004). Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the twenty-first international conference on Machine learning* (pp. 46–). New York, NY, USA: ACM.

Hall, M., Eibe, F., Holmes, G., Pfahringer, B., 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.

Lerner, U., Parr, R., Koller, D., & Biswas, G. (2000). Bayesian Fault Detection and Diagnosis in Dynamic Systems. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence* (p. 531-537). San Mateo, CA, USA: AAAI Press.

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

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc.

Roychoudhury, I., Biswas, G., & Koutsoukos, X. (2008). Comprehensive diagnosis of continuous systems using dynamic Bayes nets. In *Proc. of the 19th International Workshop on Principles of Diagnosis* (p. 151-158).

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

Smyth, P. (1994). Hidden Markov models for fault detection in dynamic systems. *Pattern Recognition*, *27*(1), pp. 149-164.

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

Verma, V., Gordon, G., Simmons, R., & Thrun, S. (2004). Real-time fault diagnosis. *IEEE Robotics and Automation Magazine*, *11*(2), pp. 56-66.