A probabilistic model for information and sensor validation

Ibargüengoytia, PH, Vadera, S and Sucar, LE

http://dx.doi.org/10.1093/comjnl/bxh142

A Probabilistic Model for Information Validation

Pablo H. Ibargüengoytia
Instituto de Investigaciones Eléctricas, México, pibar@iie.org.mx

Sunil Vadera
University of Salford, UK, S.Vadera@salford.ac.uk

L. Enrique Sucar
ITESM, Campus Cuernavaca, México, esucar@itesm.mx

Abstract

This paper develops a new theory and model for information validation. The model represents relationships between variables using Bayesian networks and utilises probabilistic propagation to estimate the expected values of variables. If the estimated value of a variable differs from the actual value, then an apparent fault is detected. The fault is only apparent since it may be that the estimated value is itself based on faulty data.

The theory extends our understanding of when it is possible to isolate real faults from potential faults and supports the development of an algorithm that is capable of isolating real faults without deferring the problem to the use of expert provided domain specific rules.

To enable practical adoption for real-time processes, an any time version of the algorithm is developed, that, unlike most other algorithms, is capable of returning improving assessments of the validity of the sensors as it accumulates more evidence with time.

The developed model is tested by applying it to the validation of temperature sensors during the start up phase of a Gas turbine when conditions are not stable; a problem that is known to be challenging. The paper concludes with a discussion of the practical applicability and scalability of the model.

1 Introduction

Artificial intelligence (AI) is playing an increasingly important role in domains like communications, medicine, finance and industry. Examples of industrial applications include the control of advanced manufacturing plants [2], control in the power generation process [24], network management for power distribution, and chemical processes. In medicine, several diagnosis systems have been designed and tested with increasingly satisfying results [4, 7]. In finance, forecasting systems have been used for decision making in all the operations of the field [1].

In general, AI methods are moving towards more realistic domains that require co-operation between several fields of research. Examples of these methods include probabilistic reasoning, planning, case based reasoning, model based reasoning and artificial neural networks. All these methods require a model for the representation of the domain knowledge. The models are fed through the input variables, with the information obtained from the process.

In general, information can be obtained from several sources depending on the application domain. For example, many applications receive information provided by human agents. Other applications receive information through sensors or instrumentation. In industrial processes, a decision based on faulty data could lead to a disaster. Consider for example the case of temperature sensors in a gas turbine of a thermoelectric power plant. The temperature is considered the most important parameter in the operation of a turbine since it performs more optimally at higher temperatures. However, a little increase in the temperature, over a permitted value, may cause severe damage in the turbine itself and in the process. Now, imagine that one of the sensors is faulty, then the following situations might arise:

1. The sensor indicates no change in the temperature even if it increases to dangerous levels.

2. The sensor reports a dangerous situation even if it is normal.

The first situation may cause a disaster, with possible fatal consequences for the whole plant, including human life. The cost of this type of failure can not be easily calculated. The second situation, as described above, may cause a false shut down, and loss of time and money. In this case, the cost can be calculated based on the fuel spent and the cost of the energy not produced while the plant is idle.

As another example, consider an incident that occurred on Canadian Air Transit Airbus from Toronto to Lisbon in 2001. Both Engines of the Airbus shut down, one after the other, as it crossed the Atlantic but the pilots doubted the consistency of the sensors readings and had to deduce that the problems were due to fuel loss. The pilots were able to divert the flight, fly the Aircraft as a glider and land it safely [13]. This illustrates the potential impact of the lack of proper information validation, which contributed to risking the lives of 306 passengers.

Besides the need for validated information, these types of applications require real time behaviour. By definition, the response of a real time system depends not only on the logical result of the computation but also on the time at which the results are produced [31]. Usually, real applications possess a time limit by which some actions must be performed. This paper presents a new theory and an *any time* algorithm for probabilistic information validation.

Section 2 presents a description of the information validation problem in intelligent systems. Section 3 develops a theory and a model for validation based on probabilistic methods. Section 4 extends the algorithm to make it appropriate for performing in real time environments. Section 5 presents an empirical evaluation of the algorithm by applying it to the validation of sensors in an industrial plant. Section 6 concludes the paper with a discussion of the practical applicability of the model.

2 Information Validation

What do we mean by information validation? In the context of this work, information refers to the set of variables that some application requires for its decision process. These variables are considered the input variables to an intelligent system. Additionally, this work considers only those applications where the process can be modelled using input information and some mechanism, where relations between the different variables can be described, e.g., dependency relations. In this context, input information can be considered as provided by sensors.

The input of a sensor is the value V_s which is considered unknown and inaccessible, and the output is the measurement V_m (Fig. 1). A sensor is declared *faulty* if the output measurement V_m gives an incorrect representation of the V_s [37].

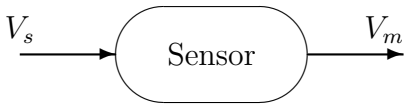

Figure 1: Basic model of a sensor.

Thus, in this paper, information validation is the process of determining the validity of the readings reported by a collection of variables or sensors.

Typical approaches for the detection of incorrect representations of a sensor include the use of:

- *Hardware redundancy and majority voting:* where hardware instrumentation (or the information agent) is duplicated and a voting algorithm is used to exclude erroneous information.
- *Analytical redundancy:* in which all process, actuators and sensors are monitored centrally. This approach requires the development of mathematical or analytical models whose solution require expensive computer power.
- *Temporal redundancy:* in which repeated measurements are taken and related with some statistical mechanism.

Hardware redundancy is not always feasible for economical or availability reasons. On the other hand, analytical and temporal redundancy presents several problems including [14]:

- The relationships between the process variables needs to be identified, and represented with differential equations. This can be difficult, and sometimes impossible.
- The approach is very sensitive to modelling errors. That is, the effects of modelling errors obscures the effects of faults and is therefore a source of false alarms.
- It requires the development of a domain dependent fault diagnosis process.

Utilising analytical redundancy demands an enormous amount of expertise to use it in a different process or even make a modification of the monitored

system. Additionally, some application domains have no useful model at all, such as medicine and finance. These problems have encouraged the development of the alternative approach to information validation that is presented in this paper.

3 Model, Theory and Algorithm

This section begins with a description of the problem and then develops the theory and validation algorithm. An illustrative example is used throughout the section to motivate the theory and algorithm.

The information validation problem can be summarised as: *how do we validate the input variables of an application, or equivalently, how do we identify any faulty sensor readings of a process?*

A reasonable starting point towards solving this problem, that is taken by most existing algorithms, is to estimate the value of a variable using the other variables and then reporting a failure if the actual reading differs significantly from the estimated value. This may be adequate for simple cases, such as validating height against age, but is not adequate for other examples such as sensor validation. To appreciate the problem, suppose we have a process or application that has five input variables: $\{m, t, p, g, a\}$ and that, somehow, we are able to relate these variables. Suppose in our example, that the variable t is incorrect (or faulty). Use of the other variables may find that t is faulty. However, use of the faulty t will also lead to poor estimates of other variables and hence lead to correct variables being reported as faulty. Thus the main questions, which are tackled in this section are:

- How can we represent the relationships between the variables and estimate the expected values of variables?
- Is it possible to isolate the real faults, and if so, under what circumstances can they be isolated?
- Can we develop a suitable algorithm that does not rely on redundancy?

How can we obtain the relationships between the variables so as to enable us to estimate the other variables? Some authors have used neural networks [20], one for each variable. However, this doesn't help much in overcoming the problem of distinguishing between the real and apparent faults. Instead, this research uses Bayesian networks [26] to represent the

relationships, which as we will see below, helps with the problem of isolating real from apparent faults and is a more appropriate representation given that estimates of variables are not precise but imply a probabilistic distribution. As an example, Fig.2 shows how the relationships between the five variables $\{m, t, p, g, a\}$ can be represented in a Bayesian network. A Bayesian network also includes the conditional probability distribution defining $P(X|parents(X))$ for each node X and a prior probability distribution for root nodes.

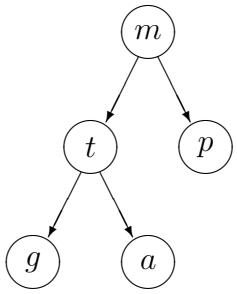

Figure 2: A Bayesian network representing a simple process.

But how could such a network be obtained? The use of Bayesian networks means that we can capitalise on the significant amount of research on learning Bayesian networks, from the early algorithms of Chow and Liu [10] to the more recent work of Chickering [9] that are able to learn such networks given the data. A further benefit is that probabilistic propagation (or inference methods) can be used to estimate the probability distribution of a particular variable given the other variables. For example, we could set the values for the variables m, p, g, a and then use propagation methods to estimate the probability distribution of t, which could then be used to assess if the value reported for t is valid.

This still leaves the main problem:

How can the real faults be isolated from the apparent ones?

The key insight of this research is that adoption of Bayesian networks enables us to take advantage of a useful property: that a faulty reading leads to the observation of potential faults in a particular set of variables. This set is known as the *Markov blanket* of the variable which is defined as follows [26].

Definition 3.1 *A* Markov blanket $MB(X)$ *of any variable* $X \in V$ *is a subset* $S \subset V$ *where* $X \notin S$ *for which*

$$I(X, S, V - S - X).$$

where $I(A, B, C)$ *denotes the conditional independence of* A *and* C *given* B*.*

For example, in Fig. 2, $I(t, \{m, a, g\}, p)$ so t is conditionally independent from p given the set $\{m, a, g\}$. Thus, $MB(t) = \{m, a, g\}$. The following corollary defines a Markov blanket that can be used in Bayesian networks [26].

Corollary 3.1 *In any Bayesian network, the union of the following three types of neighbours is sufficient for forming a Markov blanket of a node* X*: the direct parents* $PA(X)$*, the direct successors,* $SU(X)$ *and all the spouses* $SP(X)$ *(ie, direct parents of* X*'s direct successors).*

The reader is referred to Gieger [16] for a proof. Although there may be other Markov blankets, only this type of blanket is considered and assumed in the theory below. A useful symmetry property of Markov blankets that we will need later is that a variable X is in the Markov blanket of all the variables Y that are in the Markov blanket of X and it is only in these Markov blankets.

Lemma 3.1 *(symmetry)*
Let X *be a node in a Bayesian network* $G = (V, E)$ *with a Markov blanket* $MB(X)$*,* $X \in MB(Y)$ *iff* $Y \in MB(X)$*,* $\forall Y \in V$*.*

Proof:
First, the proof that if $Y \in MB(X)$ then $X \in MB(Y)$. Given that $MB(X) = PA(X) \cup SU(X) \cup SP(X)$, then $Y \in PA(X)$ or $Y \in SU(X)$ or $Y \in SP(X)$, so $X \in SU(Y)$ or $X \in PA(Y)$ or $X \in SP(Y)$, respectively. In any case, $X \in MB(Y)$.

Next, the proof that if $Y \notin MB(X)$ then $X \notin MB(Y)$. By Definition 3.1:

$$I(X, MB(X), V - MB(X) - X).$$

By the *Symmetry* axiom (from [27])

$$I(V - MB(X) - X, MB(X), X).$$

Now, if $Y \notin MB(X)$ and $Y \neq X$, then

$$Y \in V - MB(X) - X.$$

Hence, by the *Decomposition* axiom (from [27])

$$I(Y, MB(X), X), \quad \forall\, Y \in V - MB(X) - X.$$

Thus X is not in $MB(Y)$.$\diamond$

Returning to the validation problem, the process can begin by assuming that the input variables, one by one, are suspect. By probabilistic propagation, the system can decide if the value of an input is correct based on the values of the most related variables. So, to see if the value of t is correct, the other variable values can be used to predict t. This is done by instantiating the other nodes in the model and obtaining the posterior probability distribution of the value of t. From this probability distribution, an estimate of t is calculated. Then, if the predicted value is different from the observed value, a fault is detected (The precise detection criterion will be explained below). This process can be repeated for all the input variables resulting in a list of apparently faulty variables.

Notice that when t fails, the list of apparently faulty nodes is $\{m, t, g, a\}$ which corresponds to the Markov blanket of t plus the variable itself. This is called the extended Markov blanket (EMB) of a variable. Table 1 shows the EMBs for each variable. That is, if the input of a variable is erroneous, then the above validation process will report a fault in all the variables in its extended Markov blanket. This is a general result that is now formailised and proved.

Table 1: Extended Markov blankets for the model of Fig. 2.

process variable	Extended Markov Blanket
m	$\{m, t, p\}$
t	$\{m, t, g, a\}$
p	$\{m, p\}$
g	$\{t, g\}$
a	$\{t, a\}$

Formally, given a probability distribution P on a set of variables V, a DAG (Directed Acyclic Graph) G is a *Bayesian network* if it represents the

dependency model M for a the probability distribution P [26]. Like other sensor validation models, the theory developed makes the following assumptions:

1. **Observability**: all the variables can be measured directly.
2. **Fault detection**: if there is an error in variable X it can always be detected. This is a *real fault* denoted by $Fr(X)$.
3. **Fault propagation**: if a variable Y has a real fault $Fr(Y)$, and $Y \in MB(X)$, a fault in X will be detected. This is called an *apparent fault*.

Assumption 1 is common across any application. Assumption 2 is implicit in most sensor validation models and the extent to which it is valid for a particular application depends on how accurately a variable's expected value can be estimated and the extent to which its reading departs from the expected value. Bayesian networks are developed specifically for representing probability distributions of the data and have been used successfully in many applications [25], and as such, one would expect them to be as good as any other model for estimating the expected distribution of variables. Assumption 2 would not, of course, hold for very mild faults where the readings would not depart enough from the expected value.

The extent to which assumption 3 holds depends on how well the dependencies between the variables have been captured. Weak dependencies would violate the assumption. However, learning algorithms rarely produce Bayesian networks that include weak dependencies and if the Bayesian networks are constructed manually, there are techniques, such as those developed by van Engelen [34] that enable the removal of weak links since they contribute least to representing the distribution of the data. We discuss these assumptions further in the conclusion of the paper when we consider the practical applicability and scalability of the model.

Given the above model and its assumptions, the key property that enables identification of the real faults can be stated as the following lemma.

Lemma 3.2 *If there is an error in variable X, it will produce a* potential fault *in X, and all the variables in $MB(X)$, and no other variable.*

Proof:

From assumption 2, an error in X produces a potential fault in X. From Lemma 3.1, X is an element of the MB of all variables $Y \in MB(X)$. So by assumption 3, an error in X produces potential faults in all variables in $MB(X)$. Finally, from Lemma 3.1, X is not an element of any other MB, so it will not produce a potential fault in any other variable.$\diamond$

The following corollary follows from directly from this lemma.

Corollary 3.2 *If there is an error in variable X with $EMB(X)$, and also an error in Y with $EMB(Y)$ they will produce* potential faults *in all nodes $Z \in EMB(X) \cup EMB(Y)$. In general, if there are errors in variables X_i, $i = 1, \ldots, m$, they will produce potential faults in all nodes $Z \in EMB(X_1) \cup \ldots \cup EMB(X_m)$.*

Given the above lemmas and corollary, we can state and prove the following theorem for distinguishing a single fault when the set of apparent faults is S.

Theorem 3.1 *(Single Faults)*

If $S = EMB(X)$, and there is no $Y \neq X$ such that $EMB(Y) \subseteq S$, then there is a single real fault in X.

Proof

If $Fr(X)$ then theorem 3.1 leads to $S = EMB(X)$, i.e., there are apparent faults in all the $EMB(X)$.

Now, by contradiction, suppose there is a $Y \neq X$ such that $Fr(Y)$. Then, by Lemma 3.2, there are apparent faults in $EMB(Y)$. Here, since S contains all the potential faults, there is a contradiction because from Corollary 3.2 $EMB(Y) \subseteq S$.$\diamond$

In practice, there may also be multiple faults, so a valid question, addressed by the following theorems is: under what circumstances can we distinguish multiple faults?

Theorem 3.2 *(Multiple Faults)*

If there is a unique combination:

$$S = EMB(X_1) \cup EMB(X_2) \cup \ldots \cup EMB(X_n)$$

then $Fr(X_1) \wedge Fr(X_2) \wedge \ldots \wedge Fr(X_n)$ can be identified.

Proof:

A *unique combination* means that $\forall X_k \neq X_i,\ i = 1, \ldots, n$ then

$$EMB(X_k) \not\subseteq EMB(X_1) \cup EMB(X_2) \cup \ldots \cup EMB(X_n) \text{ and}$$

$$EMB(X_j) \not\subseteq \cup_{i \neq j} EMB(X_i),\ j = 1, \ldots, n$$

Thus, by definition, $S = EMB(X_1) \cup EMB(X_2) \cup \ldots \cup EMB(X_n)$ so, an additional faulty variable X_k implies $EMB(X_k) \subseteq S$, and there is a contradiction of the uniqueness condition.

Now, assuming that one of the X_i is not faulty, i.e., $\neg X_j$ where $1 \leq j \leq n$, then by Lemma 3.2, the potential faulty set will be

$$S = \cup_{i \neq j} EMB(X_i).$$

Since the combination is unique, then $EMB(X_j) \not\subseteq S$, which is a contradiction.$\diamond$

If the conditions of this theorem are not met, it is relatively easy to show that the multiple faults can not be distinguished.

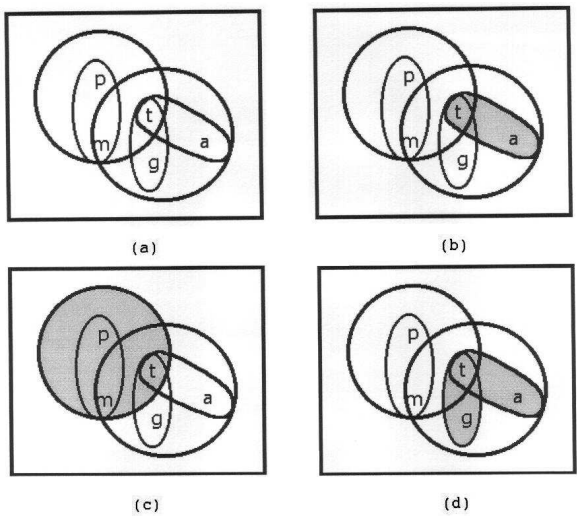

Figure 3: Representation of the EMBs as sets indicating the different cases: (a) no failure, (b) a single distinguishable fault in a, (c) a indistinguishable double fault in m and p, (d) a multiple distinguishable fault in g and a.

Figure 3 illustrates the theorems on the above example by viewing $EMBs$ as sets of variables in a Venn diagram. In (a) a representation of all the

$EMBs$ is shown when there is no fault (no dark area). In (b), a representation of the case when $S = \{t, a\}$ is shown. It is clear that this is a unique set, i.e., there is a single distinguishable fault. In (c), a representation of the indistinguishable double fault is given since it is not possible to distinguish the set $\{m, t, p\}$ and its union with the set $\{m, p\}$. The case of multiple distinguishable faults is shown in (d). This case shows a unique union of two sets. As pointed out by a referee, although the notion of a Markov blanket is not probabilistic, the dependencies are probabilistic and judgement of whether a variable has an error remains probabilistic.

This theory leads to the information validation algorithm given in Fig. 4 which is illustrated on the probabilistic model of Fig. 2, from step 2 onwards. Step 2 produces the EMB table given in Table 1. Suppose steps 3 to 5 give the results shown in Table 2. Then, step 6 indicates that $S = \{m, p\} = EMB(p)$ so it can be concluded that the real fault is in variable p. Notice that there is no other $EMB \subseteq S$ so according to step 6(b), it is a single fault. As another example, suppose that t is faulty. In such a case, the potentially faulty list would be $S = \{m, t, g, a\} = EMB(t)$. Steps 6(c) therefore applies, i.e., there is a real fault in t but possibly (and indistinguishable) real faults in variables g and a.

Table 2: Steps 3, 4 and 5 of the algorithm for the example of Fig. 2.

validating	result	faults list S
m	fails	$\{m\}$
t	correct	$\{m\}$
p	fails	$\{m, p\}$
g	correct	$\{m, p\}$
a	correct	$\{m, p\}$

4 Any Time Validation

The information validation process described in the previous section takes all the inputs, performs the validation and produces a list of faulty sensors – i.e., no intermediate results are available. This may be adequate for a number of applications; however, for real time applications, these characteristics are

1. Obtain the Bayesian network of the information variables of the application process.
2. Make a list of the variables to be validated (usually all) and build a table of EMBs.
3. Take each one of the variables to be checked as the hypothesis, instantiate the variables that form the Markov blanket of the hypothesis, and propagate the probabilities to obtain the posterior probability distribution of the variable given the evidence.
4. Compare the predicted value (the maximum posterior probability) with the current value of the variable and decide if an error exists.
5. Repeat steps 3 and 4 until all the variables in the list have been examined and the set of variables with apparent faults (S) is obtained.
6. Compare the set of apparently faulty variables obtained in step 5, with the table of the EMB for each variable:
   (a) If $S = \phi$ there are no faults.
   (b) If S is equal to the EMB of a variable X, and there is no other EMB which is a subset of S, then there is a single real fault in X.
   (c) If S is equal to the EMB of a variable X, and there are one or more EMBs which are subsets of S, then there is a real fault in X, and possibly, real faults in the variables whose EMBs are subsets of S.
   (d) If S is equal to the union of several EMBs and the combination is *unique* (defined below), there are multiple distinguishable real faults in all the variables whose EMB are in S.
   (e) If none of the above cases is satisfied, then there are multiple faults but they can not be distinguished. All the variables whose EMBs are subsets of S could have a real fault.

Figure 4: Basic information validation algorithm.

inadequate, for example when an urgent decision to stop a costly process is required.

This section describes how the algorithm developed above is extended to an any time algorithm that is more appropriate for real time applications. *Any time* algorithms are those that can be interrupted at any point during computation, and return an answer whose *quality* increases as it is allocated additional time [12, 18, 8, 19].

The key to developing an any time information validation algorithm is to recognize that the knowledge about the state of the variables (faulty or correct) becomes more certain and complete as time progresses. Certainty about the state of a variable refers to the degree of belief in the correctness of a variable, and completeness is characterized by the number of variables whose state is known. Thus, it is required to be able to monitor the state of the variables during all the validation process. This is done by using a vector whose elements $P_f(s_i)$ represent the probabilities of failure for the variables s_i. Given that the any time validation process needs to be cyclic, the top level of the algorithm can take the form shown in Fig. 5. The following subsections develop and describe the steps of the any time algorithm.

1. Initialize $P_f(s_i)$ for all variables s_i.
2. While there are unvalidated variables do:
   (a) select the next variable to validate
   (b) validate it
   (c) update the probability of failure vector P_f
   (d) measure the quality of the partial response

Figure 5: Top level of the any time validation algorithm.

4.1 Selection of next variable

This section develops a model for choosing the best variable to validate next given the history of the validation process and the current state of the system.

Also, the model proposed here will be used for measuring the quality of the response in order to obtain the performance profile of the validation algorithm.

The central idea is that the validation of a variable provides information about the validity of the variables, and different variables will provide different degrees of information. Hence, if we can measure the amount of information that can be provided by each variable, the most promising one can be selected. The measure of information utilised is the standard one, defined as follows by Shannon[30].

Definition 4.1 *Given a finite probability distribution*

$$p_i \geq 0 \text{ for } (i = 1, \ldots, n), \text{ and } \sum^n p_i = 1$$

Shannon's entropy measure is defined as

$$H_n = H_n(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{1}$$

Since the validation of a variable s has two possible outcomes (correct or faulty), the entropy function $H(s)$ is then defined as:

$$H(s) = \begin{cases} 0 & \text{if } p = 0 \text{ or } p = 1 \\ -plog_2(p) - (1-p)log_2(1-p) & \text{otherwise} \end{cases} \tag{2}$$

where p represents the probability of failure of the variable. Thus, considering each single variable validation as an independent experiment, this function can be used to measure the amount of information provided by that validation. Then, the average amount of information $\mathcal{E}$ for the system S can be defined as follows:

$$\begin{aligned} \mathcal{E}(s_1, \ldots, s_n) &= \frac{1}{n} \sum_{i=1}^{n} H(s_i) \\ &= -\frac{1}{n} \sum_{i=1}^{n} [P_f(s_i) log_2 P_f(s_i) + (1 - P_f(s_i)) log_2 (1 - P_f(s_i))] \\ &= -\frac{2}{n} \sum_{i=1}^{n} P_f(s_i) log_2 P_f(s_i) \end{aligned} \tag{3}$$

Where the last step is due to the symmetry of $P_f log_2 P_f$, n is the number of variables and $P_f(s_i)$ represents the current probability of failure assigned to variable s_i.

Given this measure, the any time validation algorithm needs to select a variable $X \in S$ that gives the best improvement in the average entropy of the system S. Hence the following conditional version of equation 3 can be written:

$$\begin{aligned}\mathcal{E}(S \mid X) &= \mathcal{E}(S \mid X = correct) + \mathcal{E}(S \mid X = faulty) \\ &= \frac{1}{n}\left(\sum H(s_i \mid X = correct) + \sum H(s_i \mid X = faulty)\right). \quad (4)\end{aligned}$$

This function can be evaluated for each variable and the one which gives the most information (the minimum $\mathcal{E}(S \mid X_i)$) can be selected as the next variable X_i to be validated. However, the computation suggested by the above formulae could be too expensive for a real time validation process. To overcome this problem, a pre-compilation of the variable selection mechanism is developed as follows. The above formulae are used to select the variable, s_r which gives the most information. This selected variable forms the root of a binary decision tree. A fault is simulated in this variable and the formulae are again used to select the next variable s_{r-}. Then, the root s_r is assumed to be correct, and the formulae are used to select the variable s_{r+} in this case. This results in the partial decision tree shown in Fig. 6.

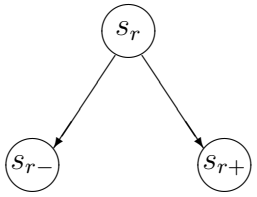

Figure 6: Partial decision tree.

This process is repeated recursively on the nodes s_{r-} and s_{r+} to obtain a complete decision tree, so that each path in the tree includes all the variables.

As an example, consider again the network shown in Fig. 2. This process results in the decision tree shown in Fig. 7. This decision tree can be used to

select the next variable more efficiently in real time than by performing the calculations. Thus, the selection step of the algorithm of Fig. 5 consists of simply traversing the tree one level after every single variable validation. The cycle starts at the root, and the decision tree points to the next node in the tree according to the result of validating the current variable. Notice that the binary tree for n variables contains n levels and up to 2^{n-1} nodes[1]. Then, if a system has 21 input variables, the binary tree would require 1,048,575 nodes, and assuming 10 bytes per node this tree requires more than 10 Mbytes of memory. Hence, to accommodate situations when memory is short, an

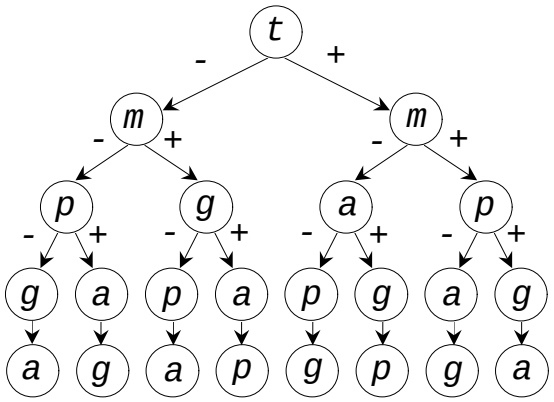

Figure 7: Binary tree indicating the order of validation given the response of the validation step.

alternative approach to the pre-compilation of the decision tree is used. This considers the sequence of validations when single faults are simulated in all the variables of the system. Of course, the case of no faults at all must also be considered. Table 3 presents the valid trajectories followed in the case of failures (first column) of the example of Fig 2. The plus sign represents the correct validations while the minus sign represents a faulty condition in the variable. The first variable to validate is always t since it has the lowest conditional entropy. The first row indicates the case of no failures. The second row presents the sequence of validation when m is simulated as faulty. Here, all $m'sEMB$ will be faulty. Representing the information of Table 3 in a decision tree results in a pruned tree with n levels and at most

[1]The exact number is $(2^n - 1) - 2^{n-2}$.

$n \times (n+1)$ nodes as shown in Fig. 8. This is because only the rows in Table 3 would be trajectories in the tree from the root to the leaves.

Table 3: Trajectories of validation in the case of single faults. The + represents the validation as correct while - represents a fault in the variable.

case	first	second	third	fourth	fifth
no fault	$t+$	$m+$	$p+$	$g+$	$a+$
m	$t-$	$m-$	$p-$	$g+$	$a+$
t	$t-$	$m-$	$p+$	$g-$	$a-$
p	$t+$	$m-$	$g+$	$p-$	$a+$
g	$t-$	$m+$	$g-$	$a+$	$p+$
a	$t-$	$m+$	$g+$	$p+$	$a-$

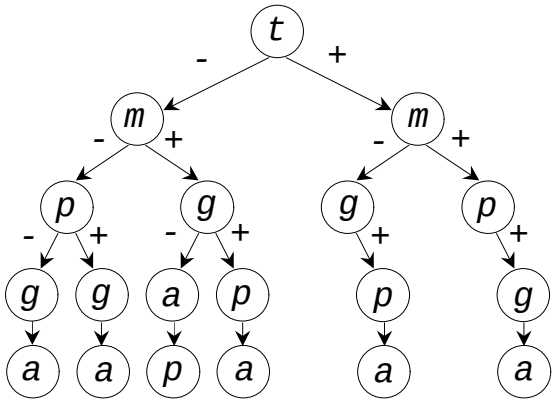

Figure 8: Reduced decision tree.

Notice the sequence of validations when no fault is found, i.e. t, m, p, g and a, which corresponds specifically to the right hand branch in Fig. 8. This trajectory is considered as *default* when an invalid sequence is found. For example, suppose that the sequence is: t correct, and m faulty. According to Fig. 8 the next variable to validate is g but the tree only considers a valid trajectory when g is correct. If g is faulty, then the default trajectory would be followed (p would be the next node in this example).

Notice that, even if the decision tree was defined following the single fault assumption, the existence of multiple faults will also be detected.

4.2 Isolation

The validation step provides only a list of potentially faulty variables. Thus, a comparison is made between the set of potentially faulty variables with the table of extended Markov blankets of all the variables. When a match is found, a real fault is determined. However, the set of potentially faulty variables is obtained after all the variables have been validated. Therefore, in order to extend that algorithm for any time behaviour, a different mechanism for distinguishing real faults from apparent ones is required. This new mechanism is achieved by using the fact that real faults lead to the observation of apparent faults and defining a two level Bayesian network.The first level consists of the nodes that represent the events of real failure in every variable and the second level is formed by nodes representing apparent failures in all the variables. Arcs are included between every root node, and the corresponding nodes of the extended Markov blanket. For example, the causal network shown in Fig. 9 can be obtained directly from the Bayesian network given in Fig. 2.

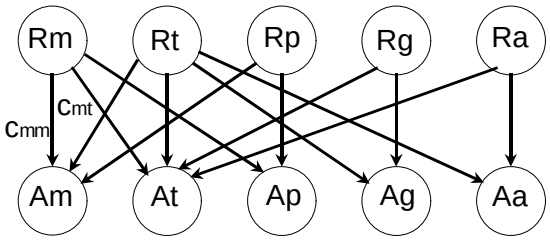

Figure 9: Probabilistic causal model for fault isolation in the example of Fig. 2. R_i represents a real fault in variable i while A_j represents an apparent fault in variable j. c_{mt} represents the conditional probability of obtaining an apparent fault in variable t (A_t) given a real fault in m (R_m).

The network of Fig. 9 is multiply connected and in general, $O(2^n)$ conditional probabilities would be required (for a node with n parents). However, the *noisy or* model can be adopted here. Two assumptions need to hold in order to use this model [26]:

1. No apparent fault occurs without being caused by some real fault (accountability).

2. If an apparent fault A_j is a consequence of two real faults R_1 and R_2, then the inhibition of the occurrence of A_j under R_1 is independent of the mechanisms of inhibition of A_j under R_2 (exception independence).

The accountability assumption holds by the way the model is constructed, i.e., a variable is apparently faulty only if there is a fault in its MB. The exception independence assumption holds by assumption 3 on page 9 since the relationship between the real and apparent faults is obtained from a Bayesian network that represents the dependencies between variables. Hence, the probability of a real fault not resulting in an apparent fault is small. Further, the mechanism by which a real fault in one variable does not result in an apparent fault is even less likely to be dependent on another real fault.

Hence, given that these assumptions are reasonable, the noisy-or model can be adopted and enables us to utilise the above Bayesian network structure where the only parameters required are $c_{ij} = P(A_j \mid R_i \ only)$.[2]

In the case of the information validation problem, in an ideal case, all the parameters $c_{ij} \approx 1$. In the rare cases where this assumption is not valid, we can still utilise the model by estimating c_{ij} by use of simulation.

The network of Fig. 9 is initialized with the following information: (i) the prior probability of all the root nodes in the model is 0.5 (assuming ignorance at the beginning of a cycle), and (ii) the parameters $c_{ij} = 0.99$[3] for all present combinations of i, j. Of course, if additional information about the prior probability is available, it could be incorporated.

Having described how real and apparent faults can be related, the fault isolation model can now be summarized. It receives as an input, a validated variable with its detected state (faulty or correct) and updates the probability of failure of all the variables. It does this by instantiating the value of the corresponding apparent failure node and using a propagation algorithm to obtain the posterior probabilities of the real faulty nodes. A vector P_f of these posterior probabilities represents the current state of knowledge about the variables, and can be viewed as the output of the system at any time. Table 4 illustrates how this process works if there was a fault in g for the network of Fig. 2.

[2]The reader is refereed to Pearl [26] for details of the noisy or model.

[3]An arbitrary number close to 1.0 was used.

Table 4: Example of the values of the probability vector P_f.

Step	$P_f(m)$	$P_f(t)$	$P_f(p)$	$P_f(g)$	$P_f(a)$
$t = faulty$	0.534	0.534	0.5	0.534	0.534
$m = correct$	0.013	0.013	0.009	0.663	0.663
$g = faulty$	0.009	0.019	0.009	0.99	0.502
$a = correct$	0.009	0.0	0.009	0.999	0.009
$p = correct$	0.0	0.0	0.0	0.999	0.009

5 Experimental results

This section tests the validation algorithm by applying it to the validation of temperature sensors of the gas turbine at the *Gómez Palacio* power plant in México. The experiments are carried out for a subset of the sensor readings for the start up phase of the plant where thermodynamic conditions change considerably – a problem that has been acknowledged as very difficult for sensor validation, even when expert rules and a model based approach are used [23].

However, the aim of the experiments is not to suggest that on its own, the model is adequate for fault diagnosis of Gas turbines, which is a large, complex task that continues to be the subject of significant investment and research effort, for example, as part of the European Community Network of Excellence known as Monet [35]. Instead, the model is meant as a component to be used by higher level layers, that for example, can perform risk assessment and fault diagnosis.

The experiments were carried out on a set of 21 temperature sensors over the first 15 minutes of the start up phase. The instrumentation of the plant provides the readings of all the sensors every second, resulting in 870 instances. A Bayesian network representing the *dependencies* between the sensors of the plant is shown in Fig. 10. The dependency model was obtained by utilising an automatic learning program that uses real data from the start up phase of the turbine [32]. A tree was utilised given the low computational time for inference in this kind of network [4].

The data set was partitioned in two subsets: one partition for training the

[4]Propagation in trees is known to be significantly more efficient than in arbitrary networks

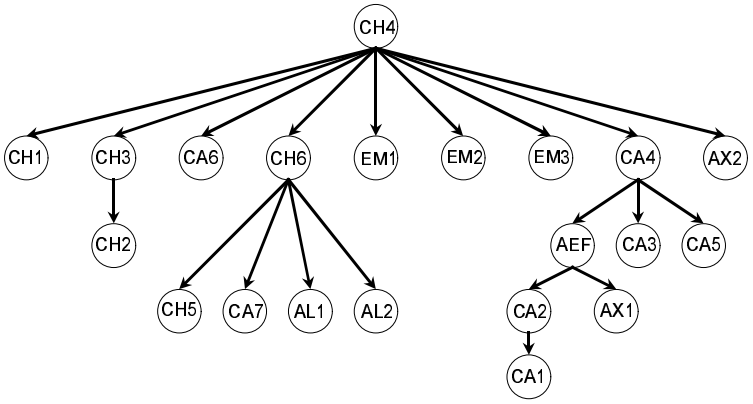

Figure 10: Probabilistic tree of the application. Nodes represent temperature signals of a gas turbine.

network, and the other partition for testing. The training/testing partition used was 70-30 % of the original data set, i.e., 610 instances for training the model (calculating the prior and conditional probabilities), and 260 instances for testing.

Two experiments were carried out: the first to see how well the approach works in terms of accuracy, and the second was to examine the information theoretic any time selection scheme. Given the absence of other any time sensor validation algorithms, the information theoretic selection scheme is compared against one that selects the next sensor to validate at random. The following subsections present and discuss the results.

5.1 Accuracy

Theoretically, the system should always detect and isolate single faults correctly. However, in reality, some errors may occur since in practice it is unlikely that the dependency model will be perfect. Consequently, two types of errors could occur: a correct reading might be considered faulty, and a real fault might not be detected. These two possible errors are called type I and type II errors in the literature, and defined as follows [11]:

type I: rejection of the *null hypothesis* when it is true, i.e., a correct sensor is reported as faulty.

type II: acceptance of the *null hypothesis* when it is actually false, i.e., a

faulty sensor is not detected.

The criteria for deciding if a reading is faulty or not is to calculate the distance of the real value from the expected value, and map it to faulty if it is beyond a specified threshold and to correct if it is less than a specified threshold. The threshold values considered are 2, 2.5 and 3 times the standard deviation σ. It's worth noting that these thresholds are adopted only to provide an evaluation of the accuracy and that an application that utilises the model could use the probability distributions instead, for example to perform risk assessment, minimise costs of misclassification, or develop ROC curves for a particular application [5, 22, 36]. Two different single faults were simulated:

Severe. The sensor value selected is substituted by one which differs by 50 %.

Mild. The real value is replaced by one which differs by 25 %.

The test procedure consisted of simulating a fault in the 260 instances described above. One set of experiments was made with severe faults, and then repeated with mild faults. If 260 failures were simulated, then every one of the 21 sensors was simulated as faulty at least 12 times.

Table 5: Results: percentage errors for severe and mild faults.

Criteria	2σ	2.5σ	3σ
Severe fault			
Type I	17.3 %	4.8 %	2.9 %
Type II	0.5 %	0.1 %	0.9 %
Mild fault			
Type I	21.5 %	10.3 %	11.4 %
Type II	8.0 %	11.7 %	14.9 %

Table 5 presents the percentage of type I and II errors for severe and mild faults. Overall, if the 2.5σ criteria is chosen for this application, then the results suggest over 95% accuracy for severe faults and about 90% accuracy for mild faults, which provides good empirical support for the theory, model and algorithm developed. The results indicate that type I errors are the most

common in this application. That is, there are cases where the existence of an invalid apparent fault, together with the valid ones, completes the EMB of a misdiagnosed sensor. Hence, a type I error is produced. In contrast, type II errors are detected at this stage when most of the sensors of a EMB present misdiagnosed apparent faults. This is very improbable as the results of Table 5 confirm.

5.2 Any time performance

Section 4.1 developed an any time information validation algorithm that utilises an entropy function as a criterion for selecting the next variable to validate. This entropy function calculates the amount of information that any single validation provides for diagnosing all the variables. Hence, to evaluate the any time algorithm, we compare the performance profile of the algorithm as a function of time when the entropy based measure is used and when it is not used (i.e., the next sensor is selected at random).

Figure 11 shows the resulting performance profiles. The measure of quality used is the average entropy $\mathcal{E}$ of the variables given in equation 3 but normalised by the maximum value. That is, if the current quality measure is:

$$Q(s_1, \ldots, s_n) = -\frac{2}{n} \sum_{i=1}^{n} P_f(s_i) log_2 P_f(s_i) \tag{5}$$

then, the reported quality function is calculated with the formula

$$Q = \frac{Q_{max} - Q_{current}}{Q_{max}} \tag{6}$$

where Q_{max} is the maximum value of the quality measure (i.e., when all the n nodes have been validated).

The results confirm that using the information-theoretic measure to select the next variable to validate results in significant improvements to the profile of the any time sensor validation algorithm.

6 Discussion and Future Work

This paper has presented a novel theory and algorithms for information validation based on the use of Bayesian networks. Other approaches, such as those that use neural networks, can estimate the expected values of variables

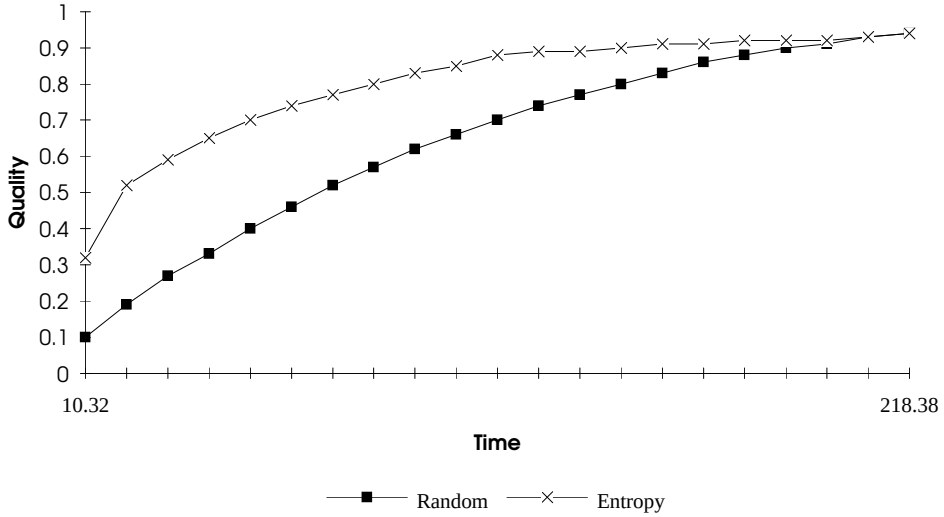

Figure 11: Performance profile of the any time information validation algorithm for sensors ($time \times 10^{-2} sec.$).

but are not able to distinguish apparent faults from real ones. A key insight of the research is that, unlike other approaches, Bayesian networks enable the use of a property known as a Markov blanket that leads to a theory for when it is possible to distinguish real from apparent faults. The theory is used to develop a validation algorithm that can operate in batch mode and an extended, any time algorithm. The any time algorithm goes beyond existing sensor validation algorithms in that it is capable of returning improving assessments of the validity of the sensors as it accumulates more evidence with time. The rate at which the any time algorithm improves the quality of its assessment is accelerated by selecting the next variable to validate on the basis of the amount of information a variable would contribute.

The theory and algorithms were tested by applying it to the validation of temperature sensors during the start-up phase of a gas turbine, when readings are not necessarily stable and for which sensor validation is known to be very difficult. The accuracy of the model was evaluated by carrying out experiments to determine the number of sensors incorrectly reported as faulty or working. In terms of accuracy, the results obtained provide good supporting evidence for the theory and model, with about 95% accuracy for severe faults and 90% accuracy for mild faults.

The any time algorithm was evaluated by carrying out experiments to determine the performance profile of the algorithm. The experiments were run with and without the entropy based selection scheme. The results confirm a profile that continuously improves the quality of the assessment of the validity of the sensors, with the the entropy based selection scheme performing significantly (16 %) better than a scheme that selects the next variable at random.

The empirical evaluations do not provide evidence of the practical applicability to problems with, say, hundreds of variables. This requires discussion of the following questions:

- To what extent do the assumptions of the model hold in practice?
- How does the model scale up as the number of variables increases?

The model makes two non-trivial assumptions to enable the development of the theory. One assumption is that if there is an error in a variable, it will be detected and the second is that if there is a real fault it will lead to the detection of apparent faults in related variables. The first assumption is made by most models of validation that depends on the process of estimating an expected value and the extent to which a reading departs from it. The diverse range and scale of successful applications of Bayesian networks on scheduling, medical diagnosis, vehicle control and weather forecasting suggests they are as good as any other approach for developing estimation models (see Chapter 12 of Neapolitan [25]). Their use does, however, offer the advantage that they were developed specifically to represent the probability distribution of the data, their use enables a sound estimate of the expected probability distribution of a variable and that there is a substantial body of knowledge and experience of utilising learning algorithms for producing Bayesian networks. Like other models, the assumption does break down if a reading does not depart sufficiently from an expected value, for example when there is a very mild failure.

The extent to which the second assumption is likely to be valid for a particular application depends heavily on the Bayesian networks of dependencies that are obtained by the use of a Bayesian learning algorithms. Given a data set, these algorithms aim to construct a Bayesian network that is a compact and accurate representation of the joint probability distribution of the data. To achieve this aim, many of these learning algorithms (e.g., Chow

and Liu [10], Friedman et al. [15]) utilise the mutual information measure to select which nodes to link. For example, Chow and Liu's algorithm evaluates the mutual information measure between all pairs of nodes, selects and links the best pair and repeats the process until the network is an adequate representation of the distribution of the data. This measure is known to be highly correlated to the degree of dependence of the variables [29] and it is therefore unlikely that these algorithms would learn a Bayesian network that includes weak dependencies. Hence, although, there are techniques, such as those developed by van Engelen[34] for removing weak links, it is unlikely that any refinement would be necessary to meet the assumptions if a Bayesian learning algorithm is used.

In terms of time, the main operation is the propagation algorithm used to estimate the expected distribution of a variable given the other values. There are several different propagation algorithms and the time-complexity of the one we utilise is known to be dependent on the size of the largest clique of the DAG [33]. Hence, scalability will not be an issue for applications where the size of the biggest Markov blankets is not large, even when the number of sensors is large. Indeed, a major advantage of Bayesian networks is that they enable efficient representation of joint probability distributions, avoiding unnecessary dependencies. However, if there are many interdependencies (i.e. large EMBs), then propagation would be slow for applications with many (say hundreds) of sensors. One option for such problems could be to utilise one of the many parallel propagation algorithms that has been developed [21, 28], although this would imply additional processors. A better option might be to partition the problem into more manageable subsets of sensors, each of which can be validated separately. This could be done by using knowledge about the domain or, if this is not feasible, utilising the kind of techniques suggested by the work of Gyftodimos and Flach [17] for restructuring large Bayesian networks into hierarchical forms.

To conclude, the development of a probabilistic theory of validation has led to an important result that allows the isolation of real faults from apparent faults, something that is normally deferred to the use of domain specific rules. The theory has also led to the development of a new any time sensor validation algorithm that provides an improving assessment of the validity of the sensors with time.

Future extensions of this work need to address how higher layers could utilise such a capability to achieve process diagnosis and system diagnosis. This implies the need to adopt a methodology for representing processes and

fusing alternative measures, which is investigated in the work of several other studies (e.g., Satnam et al. [3], and Barigozzi et al. [6]). The model also opens up the possibility of higher layers using the probabilities of failure, which are available in this model, to carry out risk assessment [5]. Future work also needs to apply the algorithm in other domains, such as finance and medicine, and to test the work on a larger scale.

Acknowledgements

The authors would like to thank the annonymous reviewers for their comments that have led to a number of improvements to this paper.

References

[1] B. Abramson. The design of belief network-based systems for price forecasting. *Comp. Elect. Eng.*, 20(2):163–180, 1994.

[2] A. Agogino, S. Srinivas, and K. Schneider. Multiple sensor expert system for diagnostic reasoning, monitoring and control of mechanical systems. *Mechanical Systems and Signal Processing*, 2(2):165–185, 1988.

[3] Satnam Alag, Alice Agogino, and Mahesh Morjaria. A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM), Special Issue on AI in Equipment Service*, 15(4):307–319, 2001.

[4] S. Andreassen. Planning of therapy and tests in causal probabilistic networks. *Artificial Intelligence in Medicine*, 4(3):227–242, May 1992.

[5] K.J. Arrow. *Essays in the theory of risk-bearing*. North Holland, 1974.

[6] A. Barigozzi, L. Magni, and R.Scattolini. A probabilistic approach to fault detection and diagnosis of industrial systems. *IEEE Transactions on Control Systems Technology*, 2004.

[7] C. Berzuini, R. Bellazzi, S. Quaglini, and D.J. Spiegelhalter. Bayesian networks for patient monitoring. *Artificial Intelligence in Medicine*, 4(3):243–260, May 1992.

[8] M. Boddy and T.L. Dean. Decision theoretic deliberation schedulling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–286, 1994.

[9] D.M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[10] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory*, 14:462–467, 1968.

[11] P.R. Cohen. *Empirical methods for artificial intelligence*. MIT press, Cambridge, Mass., U.S.A., 1995.

[12] T. Dean and M. Boddy. An analysis of time dependent planning. In *Proc. Seventh Natl. Conf. on AI*, St. Paul, MN, U.S.A., 1988.

[13] F.Fiorino. A330 overwater flameout raises etops issues. *Aviation week and Space Technology*, www.aviationnow.com /content /publication /awst/20010903/airbus.htm, September 3, 2001.

[14] P.M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge based redundancy- a survey and some new results. *Automatica*, 26:459–470, 1990.

[15] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[16] D. Geiger and J. Pearl. On the logic of causal models. In *Proc. Fourth Workshop on Uncertainty in AI*, pages 136–147, St. Paul, Minn, U.S.A., 1988.

[17] E. Gyftodimos and P. Flach. Hierarchical bayesian networks: an approach to classification and learning for structured data. In *Proceedings of the ECML/PKDD - 2003 Workshop on Probablistic Graphical Models for Classification*, pages 25–36, 2003.

[18] E.J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proc. Third Conference on Uncertainty in Artificial Intelligence*, pages 301–324, Seatle, WA, U.S.A., 1987.

[19] P.H. Ibargüengoytia, L.E. Sucar, and S. Vadera. Any time probabilistic reasoning for sensor validation. In G.F. Cooper and S. Moral, editors, *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence UAI-98*, pages 266–273, Madison, Wisconsin, U.S.A., 1998.

[20] M. Khadem, F.J. Alexandro, and R.W. Colley. Sensor validation in power plants using neural networks. In D.J. Sobajic, editor, *Proc. Inss summer workshop*, pages 51–54, Stanford, Calif. U.S.A., 1992.

[21] A.V. Kozlov. Parallel implementations of probabilistic inference. *Computer*, 29(12):33–40, 1996.

[22] Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. The DET curve in assessment of detection task performance. In *Proceedings of Eurospeech '97*, pages 1895–1898, Rhodes, Greece, 1997.

[23] R. Milne. Minutes of the Monet Task Group Meeting, [monet.aber.ac.uk: 8080/monet/monetinfo/bridge_mins.htm], 2003.

[24] R. Milne and C. Nicol. TIGER: knowledge based gas turbine condition monitoring. *AI Communications*, 9:92–108, 1996.

[25] R.E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2004.

[26] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, Palo Alto, Calif., U.S.A., 1988.

[27] J. Pearl, D. Geiger, and T. Verma. Conditional independence and its representation. In G. Shafer and J. Pearl, editors, *Readings in Uncertain Reasoning*, pages 55–60. Morgan Kaufmann, San Mateo, California, U.S.A., 1990.

[28] D.M. Pennock. Logrithmic time parallel Bayesian inference. In *Proceedings of the fourteenth annual conference on Uncertainty in Artificial Intelligence*, Madison, WI, USA, 1998.

[29] R.G.Gallagher. *Information theory and Reliable Communication*. Wiley, New York, 1968.

[30] C.E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois press, Urbana, Ill., U.S.A., 1949.

[31] J.A. Stankovic. Misconceptions about real time computing: a serious problem for next generation systems. *Computer*, 21(10):10–19, 1988.

[32] L.E. Sucar, J. Pérez-Brito, J.C. Ruiz-Suarez, and E. Morales. Learning structure from data and its application to ozone prediction. *Applied Intelligence*, 7:327–338, 1997.

[33] H.J. Suermondt and G.F. Cooper. A combination of exact algorithms for inference on bayesian belief networks. *Journal of Approximate Reasoning*, 5:521–542, 1991.

[34] R. A. van Engelen. Approximating bayesian belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):916–920, 1997.

[35] Monet Webmaster. MONET:European Network of Excellence. [monet.aber.ac.uk:8080 /monet/ index.html], accessed 28 February 2005.

[36] G. Weidl, A.L. Madsen, and E. Dahlquist. Object oriented bayesian networks for industrial process operation. In *Workshop on Bayesian modelling, Uncertainty in AI, [citeseer.ist.psu.edu/705216.html]*, 2003.

[37] S.K. Yung and D.W. Clarke. Local sensor validation. *Measurement & Control*, 22(3):132–141, 1989.