ANY TIME PROBABILISTIC SENSOR VALIDATION

A thesis submitted to the University of Salford

for the degree of Doctor of Philosophy

November 1997

By

Pablo Héctor Ibargüengoytia González

Department of Computer & Mathematical Sciences

TIME Research Institute

University of Salford

Contents

List of Tables

List of Figures

Abstract

Many applications of computing, such as those in medicine and the control of manufacturing and power plants, utilize sensors to obtain information. Unfortunately, sensors are prone to failures. Even with the most sophisticated instruments and control systems, a decision based on faulty data could lead to disaster. This thesis develops a new approach to sensor validation. The thesis proposes a layered approach to the use of sensor information where the lowest layer validates sensors and provides information to the higher layers that model the process. The approach begins with a Bayesian network that defines the dependencies between the sensors in the process. Probabilistic propagation is used to estimate the value of a sensor based on its related sensors. If this estimated value differs from the actual value, then a potential fault is detected. The fault is only potential since it may be that the estimated value was based on a faulty reading. This process can be repeated for all the sensors resulting in a set of potentially faulty sensors. The real faults are isolated from the apparent ones by using a lemma whose proof is based on the properties of a Markov blanket. In order to perform in a real time environment, an any time version of the algorithm has been developed. That is, the quality of the answer returned by the algorithm improves continuously with time. The approach is compared and contrasted with other methods of sensor validation and an empirical evaluation of the sensor validation algorithm is carried out. The empirical evaluation presents the results obtained when the algorithm is applied to the validation of temperature sensors in a gas turbine of a power plant.

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Partial results of this thesis have been presented jointly with the supervisors in the following papers:

- *A probabilistic model for sensor validation*, Proc. Twelfth Conference on Uncertainty in Artificial Intelligence, UAI-96, Portland, Oregon, U.S.A., pp 332-339, 1996.

- *Real time probabilistic reasoning for sensor validation*, AAAI Fall Symposium - Flexible Computation in Intelligent Systems, Working Notes, M.I.T., Cambridge, MA., U.S.A., pp 100-105, 1996.

- *A layered, any time approach to sensor validation*, Proc. European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty, ECSQARU-97, Bad Honnef, Germany, pp 336-349, 1997.

Acknowledgements

The development of this thesis was supported by a grant from CONACYT and IIE under the In-House IIE/SALFORD/CONACYT doctoral programme.

Special thanks are due to my supervisor in México, Dr. Enrique Sucar and my supervisor in Salford, Dr. Sunil Vadera.

I am grateful to Professor F.A. Holland and Dr. E. Wilde for their advice and continuous help during the development of this thesis.

I wish to express my thanks to the Instituto de Investigaciones Eléctricas, (IIE), for giving me the opportunity and for supporting my participation in the programme, specially to Dr. Pablo Mulás del Pozo, Dr. Roberto Canales R. and Dr. David Nieva.

I am grateful to Professor Tom Dean, from Brown University, U.S.A., for his valuable advice in the research proposal.

I also wish to thank the members of the informatics module of the programme for their continuous advice and encouragement, specially to Dr. Luis A. Pineda and Dr. Eduardo Morales. Thanks are also due to Dr. Guillermo Rodríguez and my fellow investigators Andrés Rodríguez, Pablo de Buen, and Sergio Santana.

Thanks are also due to Mrs. M.E. Calderon and to the late Fis. Andrés Estebaranz for their untiring effects to make the In-House IIE/Salford CONACYT programme increasingly successful.

Chapter 1

INTRODUCTION

1.1 Motivation

The generator of a thermoelectric power plant starts rotating commanded by a distributed control system. The microprocessor based control system monitors the plant status through acquisition boards which are connected to the sensors. The velocity sensor indicates a hundred, then a thousand, and then two thousand revolutions per minute (rpm). The control system commands an increment of the fuel and air supplies, taking care at the same time, of other important variables like the temperature and pressure of the turbine. Eventually, after two hours approximately, the velocity sensor indicates almost three thousand six hundred rpm so the start up phase is about to finish. Consider the situation if suddenly, the control system receives a signal indicating a zero velocity. Since the computer program performs no additional reasoning, the control system increases the supply of gas and air which increases the temperature and pressure.

Of course, a plant trip occurs and the system is shut down. The system is reset, the plant is re initialized and started again after some hours. This results in loss of time and money. When the shut down is analyzed, the engineers conclude that the process was fine and it was only the velocity sensor that was faulty. Given

the readings of temperature, pressure and gas supply, a zero velocity reading was highly unlikely. Consider now the case of the temperature sensors in a turbine. 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 delivers an erratic measure. Two situations can occur:

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 the above situations suggest, the validation of sensors is an important problem that requires the development of modern techniques. Specifically, this problem represents a challenge to the areas of:

- **uncertainty** management,
- **real time performance**,
- continuous operations,
- inputs provided by sensors,
- temporal reasoning,

- decision theory.

Uncertainty arises because the information provided by the sensors may be unreliable. Real time performance is required since, a decision taken, based on faulty data, could lead to a disaster. These decisions are generally made on a real time basis. Finally, this problem contains the typical characteristics of an industrial application, e.g., continuous operation, and inputs and outputs through sensors and actuators respectively. That is, although the motivation for this research project is based on a consideration of the difficulties that can arise in power plants, it is a generic problem that has a wide range of applications. Example are refineries, transportation (train lines), and fusion of sensors in intensive care units.

1.2 The Problem: Sensor Validation

The validation of sensors has been a concern since automatic control has been implemented in plants. One approach has been the hardware redundancy and majority voting. The classical approach, called *analytical redundancy*, exploits the static and dynamic relationship between measurements using a mathematical model. This technique predicts a sensor's value by using values from others in the form of known or empirical derived relations among the sensor values. However, hardware redundancy is not always possible since, for example, adding further sensors might weaken the walls of pressure vessels. The analytical redundancy approach becomes inefficient when the number of sensors increases, and when the complexity of the model increases. Additionally, the validation of sensors using analytical redundancy is adapted exclusively for each process. A slight modification is extremely expensive and demands an enormous amount of expertise.

Among all the fields of computer science, artificial intelligence (AI) contains

an extensive variety of techniques for dealing with these kinds of problems. Specifically, AI has been extensively used in diagnosis applications in several different domains. For example, the TIGER project [Milne & Nicol 1996] utilizes model-based reasoning for condition monitoring of gas turbines in chemical plants. That is, TIGER possesses a mathematical model of the process and runs a simulator in order to compare the observed output with that estimated by the simulator. Another related research project is the automated decision analytic diagnosis of thermal performance in gas turbines [Breese et al. 1992]. In this project, Breese and co-workers described the utilization of probabilistic models for the diagnosis of gas turbines for an auxiliary power unit of a commercial aircraft. They aimed to model the whole process by using a belief network that includes sensor validation as well as the fault diagnosis process.

Although both the above systems have been successfully installed in real applications, their main objective is to diagnose the whole process. In general, they assume that the sensors are working properly, or that the sensors are modelled as an integral part of the process. This way of modelling sensors, by integrating them within a model of a process, results in a larger model that can be more complex. More significantly, the sensor validation process is combined with the diagnostic process, making it less generic.

This thesis therefore develops a sensor validation model which can be utilized as a separate module that works together with other functions. In other words, it is assumed that a layered scheme is used in which the lowest level concentrates on validating the signals transmitted by the sensors as presented in Fig. 1.1 [Yung & Clarke 1989]. Faults in the sensors are detected in a decentralised and hierarchical approach, so that they can be easily isolated and repaired. Additionally, suppose that the higher layers of the system represent other important and critical functions, e.g., the fault diagnosis of a nuclear plant. The intermediate layer (loop

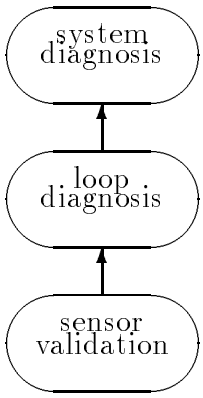

Figure 1.1: Layered diagnosis architecture.

diagnosis) may be using *model-based reasoning* to diagnose a control loop in the plant, whereas the system diagnosis layer may be utilizing a different approach. In order to focus on the sensors validation layer of Fig. 1.1, a further description of sensors is now given.

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. 1.2). A sensor is declared *faulty* if the output measurement V_m gives an incorrect representation of the V_s [Yung & Clarke 1989]. A fault is detected when the output of a sensor V_m exceeds some threshold, or deviates from a characteristic trend. But, what exactly is a characteristic trend?

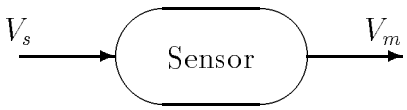

Figure 1.2: Basic model of a sensor.

This question is being answered differently by many investigators. However, in all the approaches, the central idea is to *estimate* the value that a sensor must deliver based on its environment. Some examples of these environments are the

following:

- history of a single signal in time,
- history of the state of the process in time,
- state of all related signals at a specific point in time.

This estimation process is what makes the various validation approaches different. Given that there is uncertainty about the reliability characteristics of a sensor, this thesis uses probabilistic methods for estimation based on all the related signals at specific time instants. Figure 1.3 shows some simplified models that can be used to represent sensor information in a physical process. These models are dependency models indicating *causality*. In (a), either the variable state V_s *causes* the variable measure V_m, or V_m *depends* on the value of V_s. This is the most obvious and basic model of a single sensor. Figure 1.3(b) shows a model

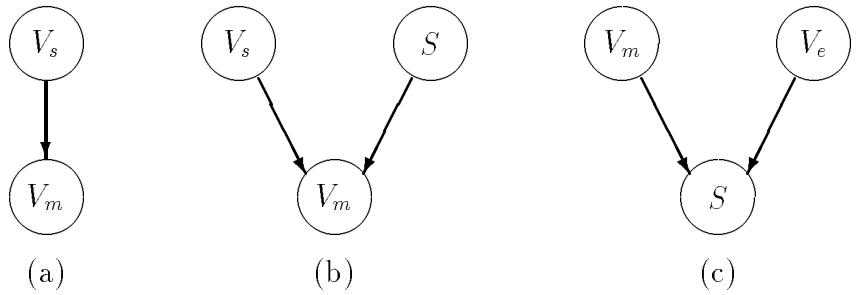

Figure 1.3: Basic model of a sensor performance. (a) represents that V_m depends on V_s. (b) represents an enhanced model where V_m depends on V_s and the state of the sensor S. (c) displays a model where S can be inferred with the values of the measure V_m and the estimated real value V_e.

including three nodes: the measure V_m depends on the variable state V_s and on the sensor state S, i.e., V_m displays a realistic representation of the variable state if the sensor is working properly ($S = correct$). Finally, since V_s is unknown and inaccessible, it is replaced with its estimation V_e in Fig. 1.3(c). Here, the inference on the sensor state S is dependent on the measure and the estimation. In fact,

Fig. 1.3(c) represents the goal of this thesis, namely to obtain the state of the sensor based on the reading and the estimated value. In other words, this model makes explicit the conditional probability of a fault in a sensor, given the measure and the estimation, i.e., $P(S \mid V_m, V_e, \delta)$, where δ represents previous knowledge about the sensor. For example, δ might represent the mean time between failures reported by the manufacturer, the physical location of the sensor in the plant, the time between the last maintenance, etc.

1.3 Objective of the Thesis

Given the above motivation, and the dependency model presented in Fig. 1.3(c), it is reasonable to suggest that probabilistic reasoning has an important role in sensor validation. Hence, the objective of this thesis can be stated as follows.

- **Develop a theory for sensor validation using probabilistic reasoning.**
- **Develop an algorithm for sensor validation, suitable for use in a layered, real time process.**

1.4 Organization of the Thesis

To accomplish the above objective, this thesis is organized as follows.

Chapter 2 describes an approach for dealing with uncertainty in artificial intelligence, namely Bayesian networks. It starts by describing the basis of probability theory up to the definition of Bayes rule. Then, it formally defines the knowledge representation and inference techniques implicit in Bayesian networks. Two different mechanisms for propagation of probabilities are described: propagation in trees and in multiply connected networks.

Chapter 2 also describes a basic algorithm for learning a probabilistic model from data obtained from the process being modelled.

Chapter 3 develops a theory and a model for sensor validation based on the probabilistic methods described in the previous chapter. Specifically, it develops a validation algorithm that, based on probabilistic propagation in Bayesian networks, detects the set of potentially faulty sensors. Then, based on a dependency property of every variable, it develops a mechanism that distinguishes between the real faults and the set of apparent faults.

Chapter 4 extends the algorithm developed in the previous chapter in order to make it appropriate for performing in real time environments. To obtain this behaviour, this thesis utilizes the *any time* algorithms mechanisms, i.e., an algorithm that provides a response whenever required and whose quality increases with time. It describes how the any time behaviour can be obtained by deciding which sensor to validate next.

Chapter 5 presents an empirical evaluation of the sensor validation algorithm. It starts by describing the application domain and the test environment developed for the experiments. Then, it presents an evaluation of the different aspects of the model, the sensor validation algorithm and the any time sensor validation algorithm.

Chapter 6 places this thesis in the context of other related work. First, it describes related approaches focusing on the sensor validation problem. Some of these approaches have been applied in different application domains. Second, it describes the related work that uses artificial intelligence for industrial applications, and specifically in gas turbines. Finally, it comments some work in *any time* algorithms used in probabilistic reasoning.

Chapter 7 presents the conclusions of this thesis and describes the fields of research that have arisen during the development of this theory. Also, possible enhancements to the algorithm are briefly outlined.

This thesis is complemented with an appendix that explain the details of the operation of the algorithm.

Chapter 2

PROBABILISTIC REASONING

The aim of artificial intelligence (AI) is to provide a computational model of intelligent behaviour. The aim of probability theory is to provide a coherent account of how belief should change in the light of partial or uncertain information [Pearl 1991]. This chapter presents one approach for the use of probability theory in AI, namely Bayesian networks that is used in this thesis. Bayesian networks, also known as probabilistic, causal or belief networks, are graphical representations of the dependencies between random variables in a specific application domain. This representation allows the codification of knowledge in the form of dependencies and independencies, and also allows inferences in the form of probabilistic propagation based on a graphical representation.

This chapter explains the basic concepts which are utilized in this thesis. Section 2.1 describes the definitions and basic concepts up to the definition of Bayes rule, i.e., the heart of Bayesian reasoning. Section 2.2 formally defines the Bayesian networks and section 2.3 describes the inference or probability propagation mechanism in simple networks called trees. Section 2.4 presents the algorithm for probability propagation in more complex representation of networks. Next, section 2.5 describes part of a more specialized Bayesian model commonly utilized

in diagnosis problems, and which is used in this thesis. Finally, section 2.6 describes an algorithm for constructing or learning probabilistic models from data.

Much of the material presented in this chapter is based on the texts by Pearl (1988), Neapolitan (1990), and the article by Pearl et al. (1990). Readers, who are familiar with these concepts may omit the details of this chapter.

2.1 Basic Concepts: Bayes Rule

Probability is formally defined as follows [Neapolitan 1990].

Definition 2.1 *Let Ω be the set of outcomes of an experiment, $\mathcal{F}$ a set of events relative to Ω, and P a function which assigns a unique real number to each $A \in \mathcal{F}$. Suppose P satisfies the following axioms:*

$$\begin{aligned} 0 \leq P(A) &\leq 1 \\ P(\Omega) &= 1 \\ P(A \text{ or } B) &= P(A) + P(B) \end{aligned} \tag{2.1}$$

if A and B are disjoint subsets of $\mathcal{F}$. Then the triple $(\Omega, \mathcal{F}, P)$ is called a probability space and P is called a probability measure on $\mathcal{F}$.

Now, since any event A can be written as

$$A = (A, B) \text{ or } (A, \neg B),$$

then, by the third axiom, $P(A)$, i.e., the probability of these two joint events can now be written as[1]

$$P(A) = P(A, B) + P(A, \neg B) \tag{2.2}$$

In general, if B has n different elements, equation 2.2 can be written as:

$$P(A) = \sum_{i}^{n} P(A, B_i) \tag{2.3}$$

[1] In the following, the notation A, B the conjunction of both events A and B.

Conditional probability is defined with the following formula:

$$P(A \mid B) = \frac{P(A, B)}{P(B)} \tag{2.4}$$

with its equivalent for the probability of joint events A and B as:

$$P(A, B) = P(A \mid B)P(B) \tag{2.5}$$

Now, from equation 2.3 and the definition of conditional probability (eq. 2.5), the following formula is obtained:

$$P(A) = \sum_i P(A \mid B_i)P(B_i) \tag{2.6}$$

which provides the basis for hypothetical reasoning, i.e., the probability of an event A is a weighted sum over the probabilities in all the distinct ways that A might be realized.

Given a set of n events, the probability of a joint event $(E_1, E_2, \ldots, E_n)$ can be written as a product of n conditional probabilities:

$$P(E_1, E_2, \ldots, E_n) = P(E_n \mid E_{n-1}, \ldots, E_2, E_1) \ldots P(E_2 \mid E_1)P(E_1) \tag{2.7}$$

This is called the *chain rule* and can be derived by the repeated application of equation 2.5.

Then, applying the chain rule to the joint probability $P(A, B)$ (i.e., $P(A, B) = P(B \mid A)P(A)$), and the definition of conditional probability (eq. 2.5):

$$P(A, B) = P(B \mid A)P(A) = P(A \mid B)P(B) \tag{2.8}$$

so the formula called the *Bayes rule* is obtained as:

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)} \tag{2.9}$$

which establishes that the probability of the hypothesis H given certain evidence E is obtained by multiplying the conditional probability $P(E \mid H)$ by $P(H)$.

Both these probabilities, the conditional probability $P(E \mid H)$ and the hypothesis prior probability $P(H)$ can be obtained from experts or from data based on the previous knowledge. $P(E)$ is a normalizing constant. In the following, $P(H)$ is called the *prior probability*, and $P(H \mid E)$ is called the *posterior probability*. This rule can be extended so that a recursive updating of the posterior probability can be made, once new evidence has been obtained. This is calculated with the formula:

$$P(H \mid E(n), E) = P(H \mid E(n)) \frac{P(E \mid E(n), H)}{P(E \mid E(n))} \tag{2.10}$$

where $E(n)$ denotes the evidence observed in the past, and $P(H \mid E(n))$ assumes the role of prior probability in order to compute the new posterior $P(H \mid E(n), E)$, i.e., the probability of H given all the past evidence and the new data observed E.

The generalization of Bayes rule of equation 2.9, for a set of n mutually exclusive and exhaustive hypotheses $\{H_1, H_2, \ldots, H_n\}$ is referred to as the Bayes theorem in the literature and expressed as:

$$P(H_j \mid E) = \frac{P(E \mid H_j)P(H_j)}{\sum_{i=1}^{n} P(E \mid H_i)P(H_i)} \tag{2.11}$$

The Bayes theorem and formula (eqs. 2.11 and 2.9) were very popular in the first expert systems utilized for diagnosis [Gorry & Barnett 1968, de Dombal et al. 1974]. However, two assumptions were made in order to keep the approach practical: (i) all the hypothesis or diseases are mutually exclusive and exhaustive, and (ii) all the pieces of evidence or manifestations are conditionally independent from each other given a disease. These assumptions restricted the expressivity of probabilistic reasoning for more realistic applications. In general, probabilistic knowledge on propositions $x_1, x_2, \ldots, x_n$ would require the definition of a *joint distribution function* $P(x_1, x_2, \ldots, x_n)$. To store this function requires a table with 2^n entries. The next section presents the Bayesian network mechanism that

allows the representation of more realistic assumptions, i.e., dependencies and independencies from which practical inference can be made.

2.2 Bayesian Networks

The main goal of Bayesian networks is to represent dependencies and independencies employing a directed acyclic graph (DAG). First, this section presents the set of axioms for the probabilistic relation: X **is independent of** Y **given** Z where X, Y and Z can be single variables or sets of variables. Second, the relation between probabilistic models and graphical representations of DAGs is established. Finally, this section presents a formal description of the properties of Bayesian networks.

First, an explanation of the notation followed in this thesis is given. Capital letters, e.g., X, represent variables while lower case letters designate the values that the variables may have, for example $X = x$ and $Y = y$.

Definition 2.2 *Let* **U** *be a finite set of variables with discrete values. Let* X, Y, *and* Z *be three disjoint subset of variables of* U. X *and* Y *are said to be* **conditionally independent given Z** *if*

$$P(x \mid y, z) = P(x \mid z) \; whenever \; P(y, z) > 0 \tag{2.12}$$

This independence will be denoted as $I(X, Z, Y)$. *Thus,*

$$I(X, Z, Y) \; iff \; P(x \mid y, z) = P(x \mid z) \tag{2.13}$$

where x, y, *and* z *are any assignment of values to the variables in the sets* X, Y *and* Z *respectively.*

This definition holds in a numeric representation of the probability P. It is interpreted as follows. Knowing the state of Z, the knowledge of Y does

not change the belief already gained in X. Now, in order to characterize the conditional independence relation as a logical condition, the following axioms are required[2] [Pearl et al. 1990]:

Symmetry:

$$I(X, Z, Y) \Rightarrow I(Y, Z, X) \tag{2.14}$$

Decomposition:

$$I(X, Z, Y \cup W) \Rightarrow I(X, Z, Y) \ \& \ I(X, Z, W) \tag{2.15}$$

Weak union:

$$I(X, Z, Y \cup W) \Rightarrow I(X, Z \cup W, Y) \tag{2.16}$$

Contraction:

$$I(X, Z \cup Y, W) \ \& \ I(X, Z, Y) \Rightarrow I(X, Z, Y \cup W) \tag{2.17}$$

Intersection (for P strictly positive):

$$I(X, Z \cup W, Y) \ \& \ I(X, Z \cup Y, W) \Rightarrow I(X, Z, Y \cup W) \tag{2.18}$$

These axioms allow the derivation of theorems that may not be obvious from the numerical representation of probabilities. Now, the next step is to relate these axioms with graphical representations.

A directed acyclic graph (DAG) $D = (V, E)$ is characterized by a set of nodes V and a set of edges E that connect certain pairs of nodes in V. Nodes in V represent the random variables while the edges or arcs represent conditional dependence relations between the nodes linked. A model M is said to be graphically represented by D if there exists a direct correspondence between the elements in the set of variables U of M and the set of vertices V of D such that the topology

[2]Normal logical operators are needed, e.g., $\Rightarrow$ is the implication, and & is the conjunction.

of D reflects the properties of M. The correspondence between $I(X, Z, Y)$ and a DAG is made through a separability criterion called *d separation* defined next.

Definition 2.3 *If X, Y, and Z are three disjoint subsets of nodes in a DAG D, then Z is said to* **d separate** *X from Y, denoted $< X \mid Z \mid Y >_D$ if along every path between a node in X and a node in Y there is a node W satisfying one of the following two conditions: (i) W has converging arrows and none of W or its descendants are in Z, or (ii) W does not have converging arrows and W is in Z.*

For example, consider the DAG of Fig. 2.1. If $X = \{B\}$ and $Y = \{C\}$, they are *d separated* by $Z = \{A\}$ but they are not by a $Z = \{A, E\}$. In both cases, there are two trajectories between B and C, namely through A and through D. Consider the trajectory through A. It has no converging arrows so, according to condition (ii), B and C are *d separated* since $A \in Z$. Consider now the trajectory through D. Since it has converging arrows, condition (i) is not satisfied if D's descendant E is in Z. Thus, X and Y are *d separated* if $Z = \{A\}$, but they are not if $Z = \{A, E\}$.

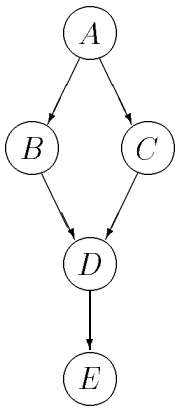

Figure 2.1: A DAG for exemplifying *d separation*.

The following definitions complete the formal description of Bayesian networks.

Definition 2.4 *A DAG D is said to be an* **I map** *of a dependency model M if every d separation condition displayed in D corresponds to a valid conditional independence relationship in M, i.e., if for every three disjoint sets of nodes X, Y, and Z, the following holds:*

$$< X \mid Z \mid Y >_D \Longrightarrow I(X, Z, Y)_M. \tag{2.19}$$

A DAG is a **minimal I map** of M if none of its arrows can be deleted without destroying its *I mapness.*

Definition 2.5 *Given a probability distribution P on a set of variables V, a DAG $D = (V, E)$ is called a* **Bayesian network** *of P iff D is a minimal I map of P.*

In other words, given a set of variables with a probabilistic model P, a Bayesian network is a graphical representation which permits the representation of the dependencies and independencies between the variables. The structure of the network represents knowledge about the variables of the process. This knowledge consists of two sets of probabilities: (i) conditional probabilities of every node given all its parents, and (ii) prior probabilities of the root nodes. Figure 2.2 presents an elementary Bayesian network and its relation with Bayes rule (eq. 2.9). In this case, the hypothesis happens to be the root node, and the evidence is represented by the leaf nodes but this is not a restriction in Bayesian networks. In this case, prior probabilities $P(H)$ are required in the roots of the networks. The other nodes require an associated matrix of conditional probabilities between each one of them and their parents (the upper extreme of the arcs). Thus, the evidence nodes are observed, and the question is to *infer* the new value

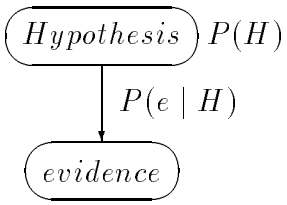

Figure 2.2: A simple Bayesian network.

of the probability of the hypothesis, i.e., $P(H \mid e)$. Notice that the hypothesis and evidence nodes can be any of the network. Different algorithms have been developed to propagate these probabilities given new evidence.

Beyond the definitions, several theorems have been published in order to formalize the Bayesian networks (e.g. [Geiger & Pearl 1988], [Geiger et al. 1989]). The following theorem, called *Strong completeness* [Geiger & Pearl 1988] includes many of the previous theorems and legitimizes the use of DAGs as a language for representing probabilistic dependencies. The complete proofs can be found in the indicated reference.

Theorem 2.1 *Strong completeness*

For every DAG D, there exists a distribution P such that for every three disjoint sets of variables X, Y, and Z the following holds:

$$< X \mid Z \mid Y >_D \quad iff \; I(X, Z, Y)_P \tag{2.20}$$

Summarizing the formal definition of Bayesian networks. Definition 2.2 introduces the notion of conditional independence and establishes the notation $I(X, Z, Y)$. In graphical representations, definition 2.3 establishes a condition that holds between nodes (or subsets of nodes) in a directed graph. Next, definition 2.4 relates the notion of conditional independence $I(X, Z, Y)$ in a model M with the *d separation* property of directed graphs. Finally, definition 2.5 explains

what a Bayesian network is. Finally, the theorem 2.1 offers a mathematical proof of the properties of Bayesian networks as reasoning methodology.

It is important now to distinguish three kinds of Bayesian networks:

Tree: This is a DAG where any node can have at most one parent. Figure 2.3(a) shows a typical network considered as a tree.

Singly connected (polytree): This is a DAG which contains one and only one path between any pair of nodes in the network. An example is shown in Fig. 2.3(b).

Multiply connected: This is a DAG without the restrictions of trees or polytrees. Figure 2.3(c) is multiply connected since there are two paths between two nodes.

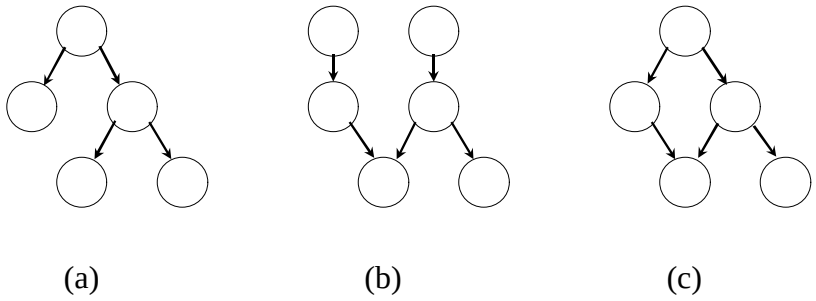

Figure 2.3: Examples of Bayesian networks. (a) is a tree, (b) is singly connected and (c) is multiply connected.

The multiply connected network is the most general and expressive when modelling specific processes. However, propagation (and therefore, reasoning in multiply connected networks) is known to be NP hard [Cooper 1990]. Trees and singly connected networks are less expressive but the probability propagation is more efficient.

Both trees and multiply connected Bayesian networks are utilized in this work and their propagation mechanism is outlined below. Trees are utilized in the

validation while multiply connected are utilized in the isolation of faults. The following section describes the propagation in trees [Pearl 1988].

2.3 Propagation in Trees

Consider the node X from Fig. 2.4 which can take n discrete values $x_1, x_2, \ldots, x_n$. Suppose that some nodes have been instantiated, i.e., their values have been observed. Let $e = e_X^- \cup e_X^+$ denote the evidence, where e_X^- stands for the evidence contained in the subtree rooted at X, and e_X^+ represents the evidence from the rest of the network. In Fig. 2.4, the subtree rooted at X is a portion of the network containing only the nodes X, D and E. The rest of the network corresponds to the structure formed by nodes A, B, and C.

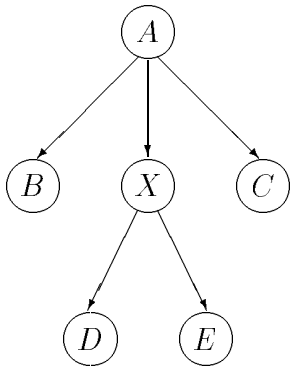

Figure 2.4: A DAG typical example of a tree.

Thus, the problem is to obtain $P(x \mid e)$, i.e.,

$$BEL(x) = P(x \mid e) = P(x \mid e_X^-, e_X^+).$$

Using Bayes rule gives

$$= \frac{P(e_X^-, e_X^+ \mid x)P(x)}{P(e_X^-, e_X^+)}$$

since e_X^- and e_X^+ are independent given x, this becomes

$$= \frac{P(e_X^- \mid x)P(e_X^+ \mid x)P(x)}{P(e_X^-, e_X^+)}$$

by Bayes rule again and the definition of conditional probability,

$$= \frac{P(e_X^- \mid x)P(x \mid e_X^+)P(e_X^+)}{P(e_X^-, e_X^+)}$$

$$= \alpha P(e_X^- \mid x)P(x \mid e_X^+) \tag{2.21}$$

where $BEL(x)$ represents the posterior probability of $X = x$ given all the evidence provided, and $\alpha = [P(e_X^- \mid e_X^+)]^{-1}$ is a normalizing constant to obtain $\sum_x BEL(x) = 1$.

Notice that this formula corresponds to a vector, with one element for each possible value of X. Now, let the following functions be defined:

$$\lambda(x) = P(e_X^- \mid x) \tag{2.22}$$

and

$$\pi(x) = P(x \mid e_X^+) \tag{2.23}$$

Vector $\lambda(X)$ represents the diagnostic support that node X receives from its descendants, while $\pi(X)$ represents the causal support attributed by all non descendants of X, and received through its parent. Then, the updated belief in $X = x$ can be obtained by *fusing* these two supports and equation 2.21 becomes:

$$BEL(x) = \alpha\lambda(x)\pi(x) \tag{2.24}$$

Since $\lambda(x)$ represents the support that X receives from all its descendants, it is necessary to fuse the support from each one of its descendants. For example in Fig. 2.4, $\lambda(x)$ corresponds to the evidence provided by nodes D and E. Thus, equation 2.22 can be rewritten as:

$$\begin{aligned} \lambda(x) &= P(e_X^- \mid x) \\ &= P(e_D^-, e_E^- \mid x) \\ &= P(e_D^- \mid x)P(e_E^- \mid x) \end{aligned} \tag{2.25}$$

since e_D^- and e_E^- are conditionally independent given x. Furthermore, renaming these terms as:

$$\lambda_D(x) = P(e_D^- \mid x),\ \lambda_E(x) = P(e_E^- \mid x) \tag{2.26}$$

then, equation 2.25 can be expressed as:

$$\lambda(x) = \lambda_D(x)\lambda_E(x) \tag{2.27}$$

Similarly, the causal support that X receives from its parent A (eq. 2.23) can be expressed as:

$$\begin{aligned} \pi(x) &= P(x \mid e_X^+) \\ &= \sum_a P(x \mid a)P(a \mid e_X^+) \\ &= \sum_a P(x \mid a)\pi_X(a) \end{aligned} \tag{2.28}$$

where $P(x \mid a)$ is an element of the matrix obtained as previous knowledge, and stored in the arc from A to X. $\pi_X(a) = P(a \mid e_X^+)$ is calculated in node A and sent as causal support to X. Thus, substituting equations 2.27 and 2.28 in equation 2.24, the following is obtained:

$$BEL(x) = \alpha\lambda_D(x)\lambda_E(x)\sum_a P(x \mid a)\pi_X(a) \tag{2.29}$$

Equation 2.29 summarizes Pearl's algorithm for probability propagation. It is best known as the message passing algorithm since $\lambda_D(x), \lambda_E(x)$ and $\pi_X(a)$ can be seen as messages that other nodes send to node X in order to update its probability vector. Thus, this posterior probability can be calculated from the previous knowledge $P(x \mid a)$, the messages $\lambda_D(x), \lambda_E(x)$ from its children and a message $\pi_X(a)$ from its parent A. Figure 2.5 shows a portion of Fig. 2.4 and the message passing for calculating node X posterior probability.

Equation 2.29 can be generalized for singly connected networks as follows. Consider a typical node X having m children, $Y_1, Y_2, \ldots, Y_m$, and n parents,

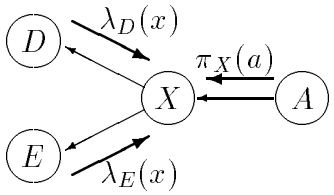

Figure 2.5: A portion of the tree of Fig. 2.4 showing the message passing algorithm.

$U_1, U_2, \ldots, U_n$. The posterior probability of X given the evidence is:

$$BEL(x) = \alpha\lambda(x)\pi(x) \tag{2.30}$$

where

$$\lambda(x) = \prod_j \lambda_{Y_j}(x), \tag{2.31}$$

$$\pi(x) = \sum_{u_1,u_2,\ldots,u_n} P(x \mid u_1, u_2, \ldots, u_n) \prod_i \pi_X(u_i) \tag{2.32}$$

and α is a normalizing constant to obtain $\sum_x BEL(x) = 1$. The term $\lambda_{Y_j}(x)$ represents the λ message that the j^{th} sends to node X, and $\pi_X(u_i)$ represents the π message sent by the i^{th} parent.

The detailed algorithm can be consulted in the book by Pearl (1988), and is easily readable in the book by Neapolitan (1990). These textbooks also present the algorithm for singly connected networks. In this later case, the main difference is that any node can have more than one parent. But, since there is only one path between any two nodes, the model and the propagation algorithm are similar, as noted in equation 2.32.

2.4 Probability Propagation in Trees of Cliques

This section presents an approach for probability propagation in multiply connected networks called propagation in trees of cliques [Lauritzen & Spiegelhalter

1988]. Other algorithms for propagation in networks are given by Cooper (1984), and by Horvitz et al. (1989). The propagation algorithm presented in this section is used in Chapter 4. A reader already familiar with this propagation algorithm may skip this section.

The basis of this method is the following formula[3]:

$$P(V) = K \prod_{i=1}^{p} \psi(W_i) \tag{2.33}$$

where V designates a finite set of propositional variables, and P represents a joint probability distribution on V. K represents a constant and let $\{W_i$ such that $1 \leq i \leq p\}$ be a collection of subsets of V. Also, ψ is a function which assigns a unique real number to every combination of values of the propositional variables in W_i. Then ($\{W_i$ such that $1 \leq i \leq m\}$,ψ) is called a potential representation of P, and W_i are called *cliques*. A clique is defined as a subset of nodes in which every pair of nodes of the clique is connected. Also, the subset must be maximal, i.e., there is no other complete set which is a subset.

The algorithm developed by Lauritzen & Spiegelhalter (1988) indicates: (i) how to obtain the collection W_i of subsets of V, and (ii) how to compute the functions $\psi(W_i)$. In other words, this method modifies the original multiply connected network in order to obtain a tree of cliques, from which probability propagation can be made utilizing the functions $\psi(W_i)$. This propagation is similar to Pearl's algorithm for trees described in section 2.3. The following sections describe these two parts of the algorithm.

2.4.1 Tree of cliques

The cliques W_i of equation 2.33 must follow a series of conditions. The procedure of Fig. 2.6 obtains the set of cliques with the required properties.

[3]The material of this section was taken from the book by Neapolitan (1990).

1. Delete the direction of the arcs, i.e., the DAG is converted to an undirected graph.
2. *Moralize* the graph.
3. Triangulate the graph.
4. Order the nodes according to a criterion called the maximum cardinality search.
5. Determine the cliques of the triangulated graph.
6. Order the cliques according to their highest labelled vertices to obtain an ordering of the cliques with the running intersection property.

Figure 2.6: Procedure to convert a network in a tree of cliques.

These steps are better explained with the aid of an example taken from the book by Neapolitan (1990). Figure 2.7 presents the original Bayesian network.

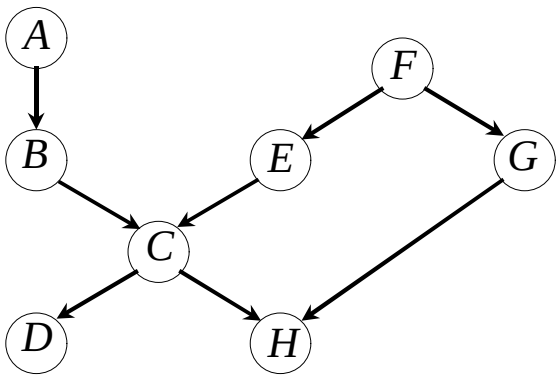

Figure 2.7: Original multiply connected network.

Notice that it is multiply connected since there is more than one path between node F and H. This network requires, as all the Bayesian networks, the prior probability of the roots and the conditional probability matrices of the other nodes given their parents. The first step in the procedure of Fig. 2.6 is trivial, i.e., only delete the direction of the arcs. The second step, the moralization, is obtained when the pairs of parents of all nodes (if they exist) are *married*. This is

done with the addition of an arc between these parent nodes. Figure 2.8 presents the moral DAG which is obtained by adding the arc between nodes B and E (parents of C), and the arc between C and G (parents of H).

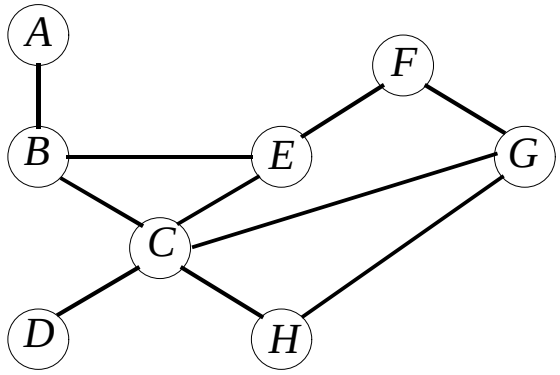

Figure 2.8: Undirected moralized graph.

Next, the triangulation step takes place. An undirected graph is called triangulated if every simple cycle of length strictly greater that 3 possesses a chord. In the original network, after the moralization, the nodes $[F, E, C, G]$ form a simple cycle of size 4. Thus, in order to triangularize the undirected graph, the arc between E and G is added. Figure 2.9 shows the triangularized graph. This

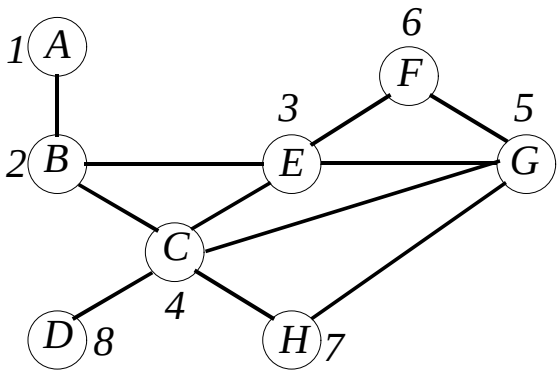

Figure 2.9: Triangulated and ordered undirected graph.

figure also show the ordering step indicated in the procedure of Fig. 2.6 which is now explained. An order of the nodes, according to a criterion known as the maximum cardinality search, is obtained as follows. First, 1 is assigned to an

arbitrary node. To number the next node, select a node that is adjacent to the largest numbered node, breaking ties arbitrary. In Fig. 2.9, number 1 was assigned to node A. Number 2 must be assigned to node B since it is the only adjacent node to A (two nodes are adjacent if there is an arc between them). Now, number 3 has to be assigned to one of the adjacent nodes to B, i.e., C or E. Node E was chosen arbitrarily. Number 4 was assigned to node C (could be F), and so on until all the nodes are numbered. The next step, determining the cliques of the triangulated graph, is now described. A clique is a subset of nodes which is complete, i.e., every pair of nodes of the clique is adjacent. Also, the subset must be maximal, i.e., there is no other complete set which is a subset. In the triangulated graph of Fig. 2.9, the following cliques are found: $\{A, B\}$, $\{B, E, C\}$, $\{E, G, F\}$, $\{C, D\}$, $\{E, C, G\}$, and $\{C, G, H\}$. Notice that the subset $\{C, E, G, H\}$ is a complete subset but it is not maximal since $\{E, C, G\}$ is a complete subset.

Finally, the ordering of the cliques is required. An ordering $[Clq_1, Clq_2, \ldots, Clq_p]$ of the cliques has the running intersection property if for every $j > 1$ there exists an $i < j$ such that

$$C_j \cap (C_1 \cup C_2 \cup \ldots \cup C_{j-1}) \subseteq C_i. \tag{2.34}$$

In the example, an ordering of the cliques is the following: $Clq_1 = \{A, C\}$, $Clq_2 = \{C, D, F\}$, $Clq_3 = \{D, E, F\}$, $Clq_4 = \{B, D, E\}$, $Clq_5 = \{F, E, H\}$, and $Clq_6 = \{F, G\}$. This ordering has the running intersection property. For example:

$$Clq_4 \cap (Clq_1 \cup Clq_2 \cup Clq_3) = \{D, E\} \subseteq Clq_3 \tag{2.35}$$

$$Clq_5 \cap (Clq_1 \cup Clq_2 \cup Clq_3 \; cup Clq_4) = \{E, F\} \subseteq Clq_3$$

Before defining the structure of the tree of cliques, two parameters need to be

defined.

$$S_i = Clq_i \cap (Clq_1 \cup Clq_2 \cup \ldots \cup Clq_{i-1})$$
$$R_i = Clq_i - S_i.$$

These parameters will be used in the propagation of probabilities and in the definition of the structure. As an example, $S_4 = \{E, F, G\} \cap \{A, B, C, E, G\} = \{E, G\}$ and $R_4 = \{E, F, G\} - S_4 = \{F\}$.

Once the set of ordered cliques has been obtained, the next step is the definition of the structure of the tree of cliques. The first clique is the root of the tree. Now, for the rest of the nodes, i.e., for each i such that $2 \leq i \leq p$, there exists at least one $j < i$ such that

$$S_i = Clq_i \cap (Clq_1 \cup Clq_2 \cup \ldots \cup Clq_{i-1}) \subseteq Clq_j. \tag{2.36}$$

Then Clq_j is a parent of Clq_i. In case of more than one possible parent, the choice is arbitrary. Figure 2.10 shows the final modification of the Bayesian network of

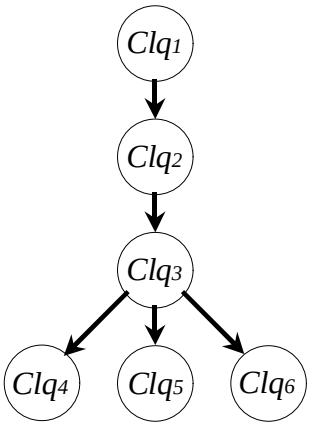

Figure 2.10: Resultant tree of cliques.

Fig. 2.7 into the tree of cliques. The next section briefly describes the algorithm for probability propagation in this tree of cliques.

2.4.2 Probability propagation

The cliques obtained in the previous section are the W_i subsets indicated in equation 2.33. The functions ψ are defined by the following theorem.

Theorem 2.2 *Let G be the DAG representing a Bayesian network, G_m the moral graph relative to G, G_u a graph formed by triangulating G_m as discussed in the previous section. Let $\{Clq_i \text{ such that } 1 \leq i \leq p\}$ be the cliques of G_u. For each node $v \in V$, assign a unique clique Clq_i such that*

$$v \cup parents(v) \subseteq Clq_i. \tag{2.37}$$

This is always possible, since parents of the original graph are married and therefore $\{v\} \cup parents(v)$ is a complete set in G_m and thus in G_u. If a complete set is a subset of more than one clique, choose one of them arbitrarily but keeping each node v assigned to only one clique. Denoting as $f(v)$ the clique assigned to v, and for $1 \leq i \leq p$,

$$\psi(Clq_i) = \prod_{f(v)=Clq_i} P(v \mid parents(v)). \tag{2.38}$$

where $f(v) = Clq_i$ represents only the nodes v that are represented in the clique Clq_i. If there is no v represented in the clique, it is assigned the value 1. Then

$$(\{Clq_i \text{ such that } 1 \leq i \leq p\}, \psi) \tag{2.39}$$

is a potential representation of P.

The complete proof can be found in the text by Neapolitan (1990). The function $parents(v)$ represents the set of nodes which are parents of node v in the original network.

For example, assigning A and B to the clique $\{A, B\}$, C to the clique $\{B, E, C\}$, D to the clique $\{C, D\}$, E, F and G to the clique $\{E, G, F\}$, and H to the clique

$\{C, G, H\}$:

$$\begin{aligned}
\psi(A, B) &= P(B \mid A)P(A) \\
\psi(B, E, C) &= P(C \mid B, E) \\
\psi(C, D) &= P(D \mid C) \\
\psi(E, G, F) &= P(E \mid F)P(G \mid F)P(F) \\
\psi(C, G, H) &= P(H \mid C, G) \\
\psi(E, C, G) &= 1. \qquad (2.40)
\end{aligned}$$

When the new tree is defined, it is ready to accept the instantiation of variables as evidence, and to compute the posterior probability of all the nodes through probability propagation in the tree of cliques. This is done in a similar way to the message passing algorithm for trees and polytrees described in section 2.2. The λ message that a node sends to its parents is calculated with the formula:

$$\lambda_{Clq_i}(S_i) = \sum_{R_i} \psi(Clq_i) \qquad (2.41)$$

where the sum is made over all the possible values of the variables in the set R_i The π message that the nodes send to their children is computed as:

$$\pi_{Clq_j}(S_i) = \sum_{Clq_j - S_i} P'(Clq_j) \qquad (2.42)$$

where the sum is made over all the possible values of the variables in the set $Clq_j - S_i$. The ψ function is updated when a clique Clq_j receives a λ message from its child Clq_i as:

$$\psi(Clq_j) = \lambda(S_i)\psi(Clq_j) \qquad (2.43)$$

For the root clique, the posterior probability once all the λ messages have been received from its children is given by

$$P'(Clq_{root}) = \psi_{root}(Clq_{root}) \qquad (2.44)$$

Finally, the posterior probability of a single variable, when the probabilities of all the cliques have been determined, is calculated with the formula:

$$P'(v) = \sum_{\substack{w \in Clq_i \\ w \neq v}} P(Clq_i). \tag{2.45}$$

The complete algorithm can be consulted in Neapolitan (1990).

2.5 Probabilistic Causal Method

This thesis also uses a special kind of network that was first studied by Peng & Reggia (1987), and then further developed by Pearl (1988) and by Neapolitan (1990). It consists of a two level DAG where the roots are considered the causes of the manifestations of the leaf nodes. Figure 2.11 shows a network known as the probabilistic causal model.

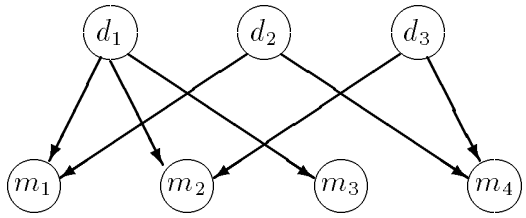

Figure 2.11: A DAG representing a probabilistic causal model.

In this network, $D = \{d_1, d_2, d_3\}$ represents the set of diseases, and $M = \{m_1, m_2, m_3, m_4\}$ represents the set of manifestations[4] respectively. Notice that there are two types of relationships between the nodes in Fig. 2.11. Figure 2.12(a) shows a common relationship where one disease has many manifestations. However, Fig. 2.12(b) shows a relation where one manifestation can be caused by several diseases. For example, the high fever event in medicine is caused by many different diseases, e.g., influenza, tuberculosis, and kidney infection. Any of these

[4]the names are traditionally taken from the medical domain.

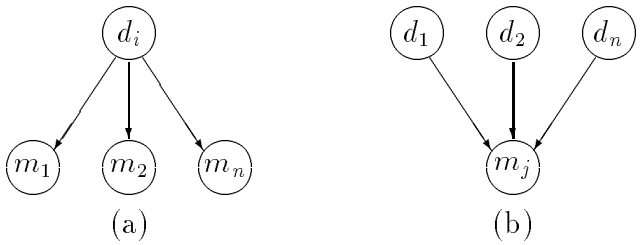

Figure 2.12: Causal relation between hypotheses or causes, and manifestations.

diseases is likely to cause high fever, but the presence of two of these diseases is only more likely to cause fever. This relation between a manifestation and several causes is known as the *noisy or* since it remains the *or gate* utilized in digital electronics. In the probabilistic case, the noisy or relation is used when any member of a set of diseases is likely to cause a specific event, but this likelihood does not significantly change when a patient suffers several of these diseases.

One of the problems in this kind of networks is the initialization of the network with the prior and conditional probabilities. Normally, the conditional probability for describing the arcs of the network in Fig. 2.12(b) contains 2^n independent parameters. It would be very difficult for a physician to estimate the probability of high fever given influenza, no tuberculosis and infection, or the probability of no influenza nor tuberculosis but with infection, and so on with the 8 combinations. A method for computing the conditional probability matrix of a disease given a set of manifestations is now explained. This method is based on the following two assumptions:

Accountability. An event m_j is false, $P(m_j) = 0$, if all conditions listed as causes of m_j are false.

Exception independence. If an event m_j is a consequence of two conditions d_1 and d_2, then the inhibition of the occurrence of m_j under d_1 is independent of the mechanisms of inhibition of m_j under d_2.

Consider the example mentioned above. Influenza alone is a cause of high fever unless an inhibitor is present. If tuberculosis alone also causes fever except when another inhibitor is present, then the exception independence mechanism assumes that both these inhibitors are independent. Then, let q_{ij} denote the probability that a manifestation m_j is inhibited when only disease d_i is present, i.e., $q_{ij} = P(\neg m_j \mid d_i \; alone)$. Then, by the exception independence assumption:

$$\begin{aligned} P(\neg m_j \mid d_1, d_2, \ldots, d_n) &= P(\neg m_j \mid d_1) P(\neg m_j \mid d_2) \ldots P(\neg m_j \mid d_n) \\ &= \prod_{i:d_i = true} q_{ij} \end{aligned}$$

In general, let $\mathbf{d}$ be the set of assignments of the set of diseases, and let $T_d = \{i : d_i = true\}$, i.e., the set of all diseases actually present. Then, the conditional probability matrix can be calculated with the following formula:

$$P(m_j \mid \mathbf{d}) = \begin{cases} \prod_{i \in T_d} q_{ij} & \text{if } \neg m_j \\ 1 - \prod_{i \in T_d} q_{ij} & \text{if } m_j \end{cases} \tag{2.46}$$

For example, in the network of Fig. 2.11, the following are the formulas of equation 2.46:

$$\begin{aligned} P(\neg m_1 \mid +d_1, +d_2) &= q_{11} q_{21} \\ P(\neg m_1 \mid +d_1, \neg d_2) &= q_{11} \\ P(\neg m_1 \mid \neg d_1, +d_2) &= q_{21} \\ P(\neg m_1 \mid \neg d_1, \neg d_2) &= 1. \end{aligned}$$

The quantities for m_1 are 1 minus the conditional for $\neg m_1$.

These equations will be utilized to obtain the parameters needed in the fault isolation phase described in Chapter 4.

2.6 Learning Algorithm

The previous section described Bayesian networks as a knowledge representation and reasoning scheme. This knowledge was in principle, assumed to be provided by an expert in the application domain. However, sometimes it can be difficult for a human to recognize all the dependencies and independencies between the variables in a process. For example, this thesis tests the developed model by applying it to the validation of temperature sensors in a gas turbine. In this problem, even for an expert in gas turbines of power plants, it may be difficult to establish the dependencies between all the temperature sensors.

In this case, the structure of the network, and the previous knowledge have to be discovered from the data. The techniques for making this discovery are known in the literature as *machine learning*. Specifically for Bayesian networks, the learning process can be divided into *qualitative* and *quantitative* learning. The first corresponds to the discovery of the structure of the Bayesian network. That is, once that all the variables (nodes) have been specified, the deduction of all the arcs needs to be obtained. The second corresponds to parametric discovery. That is, the prior and conditional probabilities have to be calculated from the data and based on the structure.

This section presents a mechanism that, based on a complete set of data, enables a tree and the prior and conditional probabilities to be obtained. This technique was first developed by Chow & Liu (1968) and then, extended by the machine learning and uncertainty communities [Fung & Crawford 1990], [Spiegelhalter & Lauritzen 1990], [Dawid & Lauritzen 1993], [Cooper & Herskovitz 1992].

Given a probability distribution P from a set of n variables $\mathbf{x} = \{X_1, X_2, \ldots, X_n\}$, the idea is to design a tree dependent probability distribution P^t that best approximates P. So, in order to evaluate a possible P^t, the following formula is

utilized:

$$P^t(\mathbf{x}) = \prod_{i=1}^{n} P(X_i \mid X_{j(i)}) \tag{2.47}$$

where $X_{j(i)}$ is the variable designated as the parent of X_i in the proposed tree. Thus, in order to evaluate how well P^t represents P, a distance criterion between the two distributions is utilized. Chow & Liu (1968) proposed the Kullback & Leibler (1951) cross entropy measure defined as:

$$distance(P, P^t) = \sum_{\mathbf{x}} P(\mathbf{x}) log \frac{P(\mathbf{x})}{P^t(\mathbf{x})} \tag{2.48}$$

This measure reaches 0 when $P^t(\mathbf{x})$ coincides with $P(\mathbf{x})$.

Two tasks are required to find the optimum tree. The first is to find a structure and second, to find the conditional probabilities $P^t(X_i \mid X_j)$ such that P^t becomes the best approximation of P. The second task is carried out with the following formula:

$$P_P^t(X_i \mid X_{j(i)}) = P(X_i \mid X_{j(i)}) \tag{2.49}$$

where P_P^t is called the projection of P on the tree t.

The problem now is to find an optimal structure. Chow & Liu (1968) used a technique called a *maximum weight spanning tree* (MWST) which chooses the best set of branches of the tree. This algorithm measures the weight of the branches between all pairs of variables (nodes) utilizing the mutual information measure defined as:

$$I(X_i, X_j) = \sum_{X_i, X_j} P(x_i, x_j) log \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \tag{2.50}$$

Chow & Liu (1968) showed that maximising the weights of the branches, minimizes the distance between both models. Figure 2.13 describes the complete algorithm. It represents a minimization problem that can be solved in $O(n^2)$ steps. A complete description of the algorithm together with an application can

MWST learning algorithm.

Input: Set of r samples of the n variables.

Output: The optimal tree Bayesian network.

1. Compute the joint distributions $P(X_i, X_j)$ for all variable pairs.
2. Compute all $n(n-1)/2$ branch weights using the distributions of step 1, and order them by magnitude.
3. Assign the largest two branches to the tree to be constructed.
4. Examine the next largest branch, and add it to the tree unless it forms a loop, in which case discard it and examine the next largest branch.
5. Repeat step 4 until $n-1$ branches have been selected, and a spanning tree has been constructed.
6. Compute $P_P^t(\mathbf{x})$ with eq. 2.49 by selecting an arbitrary root node. That is, once that the structure has been decided, calculate the prior and conditional probabilities.

Figure 2.13: Chow and Liu *maximum weight spanning tree* algorithm [Pearl 1988].

be found in the paper by Sucar et al. (1995). Chapter 5 describes the tree obtained with this algorithm from a data set of the temperature sensors of a gas turbine in a power plant.

General techniques for learning Bayesian networks, i.e., including multiply connected networks, are being developed by several authors including Heckerman et al. (1994), Heckerman & Geiger (1995), and Friedman & Goldszmidt (1996). An extensive survey of recent work in this area can be found in the paper by Buntine (1994).

2.7 Summary

This chapter presented the background knowledge required to follow the techniques developed in this thesis. Section 2.1 presented the bases of probability theory until the deduction of Bayes rule (eq. 2.9). The Bayes rule, and its utilization in the first expert systems, provided the motivation for the development of a Bayesian networks as described in section 2.2. This section presented the axioms and theorems that allow the utilization of DAGs as a language for knowledge representation and inference. Section 2.2 concluded with a brief description of the propagation algorithms for trees and singly connected networks. Section 2.4 described the algorithm for probability propagation in multiply connected networks.

The propagation method for trees will be utilized in Chapter 3 where the sensor validation model will be developed, and the more general propagation algorithm will be utilized in Chapter 4 which describes the development of the any time sensor validation algorithm. Section 2.5 described Peng & Reggia (1987) technique for the computation of conditional probabilities in causal models utilized in diagnosis. This technique will also be used in Chapter 4.

Section 2.6 described an algorithm that, based on data from a specific process,

is able to develop a tree to approximate the dependencies between the variables in the process.

The next chapter presents the utilization of the techniques presented in this chapter, for the development of a theory for sensor validation. Chapter 4 extends this algorithm for real time environments, by utilizing the causal model presented above.

Chapter 3

PROBABILISTIC SENSOR VALIDATION

The previous chapters presented the motivation for this work, and introduced probabilistic reasoning. This chapter develops the sensor validation model and its associated theory. Section 3.1 develops the algorithm with the aid of an illustrative example taken from the validation of sensors in a power plant. Section 3.2 develops the supporting theory for the sensor validation model. Next, section 3.3 concludes and discusses the application of the developed algorithm in the example considered throughout the chapter. Finally, section 3.4 summarizes the chapter.

3.1 Probabilistic Validation

In general, a sensor is considered to be faulty if it deviates from its expected behaviour. Hence, all the approaches for sensor validation work by first estimating a sensor's expected behaviour and reporting a failure if the actual behaviour is different. This thesis develops a model in which the relationships between sensors are represented by a Bayesian network. This network is utilized to predict the behaviour of the sensors and perform sensor validation as described with the aid

of the following simple example.

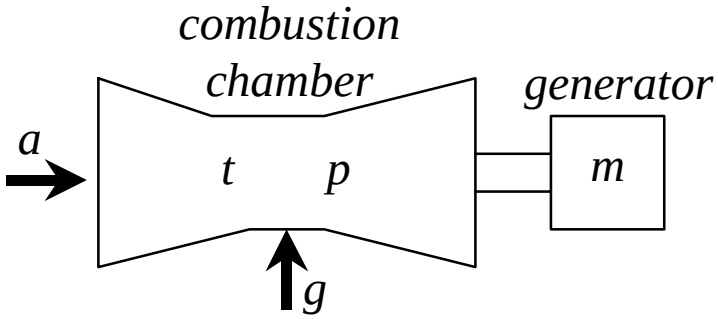

Figure 3.1: Simplified diagram of a gas turbine.

Consider the problem of validating sensors in a gas turbine of a power plant. Figure 3.1 shows a very simple diagram of a turbine. It consists fundamentally of four main parts: the compressor, the combustion chamber, the turbine itself and the generator. The compressor feeds air to the combustion chamber, where the gas is also fed. Here, the combustion produces high pressure gases at high temperature. The expansion of these gases in the turbine produces the turbine rotation with a torque that is transmitted to the generator in order to produce electric power as output. The air is regulated by means of the *inlet guide vanes* (IGV) of the compressor, and a control valve does the same for the gas fuel in the combustion chamber. The temperature at the blade path, which is the most critical variable, is taken along the circumference of the turbine. Other important variables, measured directly through sensors are the pressure in the combustion chamber and the Megawatts generated by the turbine.

A Bayesian network can be used to relate the sensors as shown in Fig. 3.2[1]. The node m represents the reading of the Megawatts generated. The temperature is represented by a node t and the pressure by p. Finally, g and a represent the fuel and air supplied to the combustion chamber respectively. The model in Fig. 3.2 represents the dependency of the temperature on the gas and air supplied. The

[1]This is a simplified model of the gas turbine. The directions of the arcs do not imply causality.

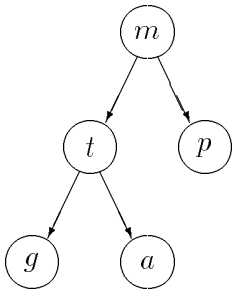

Figure 3.2: A reduced Bayesian network of a gas turbine.

generation depends on the temperature and pressure in the combustion chamber. The validation process starts by assuming that the sensors, one by one, are suspect. By probabilistic propagation, the system decides if the reading of a sensor is correct based on the values of the most related variables. For example, suppose that the sensors a and g indicate a normal supply of gas and air to produce good combustion. Also suppose that the instruments show that the power generated increases but the temperature sensor indicates the ambient temperature. This extreme situation shows how other signals can be used to *infer* if a sensor is working properly. So, to see if the sensor t is correct, the other sensor readings 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 estimation of t is calculated. Thus, if the predicted value is different from the observed value, then a fault is detected. The precise detection criterion will be explained below. This process can be repeated for all the sensors.

Now, consider the same example but suppose the sensor t is damaged. If t is used to validate m, then the process will also indicate a fault in m even when it is fine. The same happens with g and a since they also utilize t for calculating its predicted value. Finally, the validation of p may indicate that it is working properly. So, there is some confusion and an interesting problem:

which are the real and which are the apparent faults?

This question is answered by using a property based on what is known as a Markov blanket of a node. The rest of this section explains and uses this property to develop the sensor validation algorithm. The next section will develop the property more formally.

A Markov blanket is defined as the set of variables that make a variable independent from the others. In a Bayesian network, the following three sets of neighbours are sufficient to form a Markov blanket of a node: the set of direct predecessors, direct successors, and the direct predecessors of the direct successors (i.e. parents, children, and spouses). The set of variables that constitutes the Markov blanket of a variable can be seen as a protection of this variable against changes of variables outside the blanket. This means that, in order to analyze a variable, only the variables in its blanket are needed. For example, the Markov blankets of the model shown in Fig. 3.2 are shown in Fig. 3.3. In (a), the equivalent model of m is shown, where the absence of g and a indicates that these variables are out of m's Markov blanket. In (b), the Markov blanket of t indicates that changes in p do not affect t. The same goes for p, g and a in (c), (d) and (e).

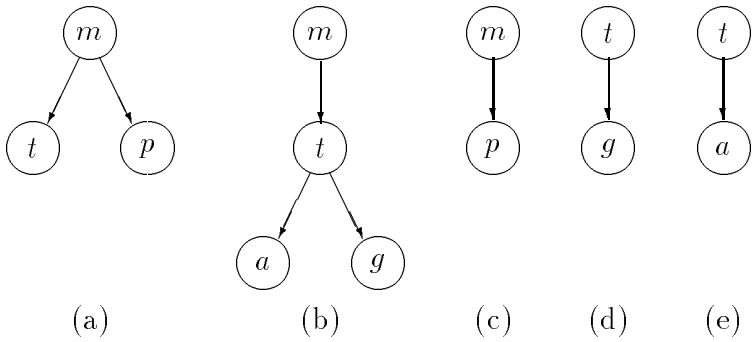

Figure 3.3: Equivalent models (Markov blankets) for the variables in the reduced Bayesian network model of a gas turbine. (a) for m, (b) for t, (c) for p, (d) for g and (e) for a.

Returning to the above question, 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 3.1 shows the EMBs for each variable. This observation can be generalized to the following property (proved in section 3.2):

> *If a sensor is faulty, then the above validation process will report a fault in all the sensors in its extended Markov blanket.*

Table 3.1: Extended Markov blankets for the simple turbine model.

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

This property can be used to distinguish between real and apparent faults as follows. Given a set S of apparently faulty sensors obtained from the above validation process, there are four cases that can arise (proofs are given in section 3.2):

Single distinguishable fault: 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. For example, if $S = \{t, a\}$, this corresponds exclusively to the $EMB(a)$.

Indistinguishable double fault: if a variable X's EMB is a superset of another variable Y's EMB, then this mechanism does not distinguish between a single failure in X, and a double failure in X and Y. In fact, this situation holds between the leaves and their parents in a Bayesian tree. For example, consider the entries of Table 3.1. The $EMB(m) = \{m, t, p\}$ while

$EMB(p) = \{\text{m,p}\}$, i.e., $EMB(p) \subseteq EMB(m)$. If $S = \{m, t, p\}$, this mechanism can not distinguish between a failure in m and a double failure in m and p since $EMB(m) = EMB(m) \cup EMB(p)$.

Multiple distinguishable faults: multiple failures can be correctly distinguished if there is a unique combination of the resultant union of $EMBs$. That is, the union of the EMBs does not include any other sensor's EMBs.

Multiple indistinguishable faults: if S does not correspond to any of the situations mentioned before, then it signifies that there are multiple indistinguishable faults among the sensors. The indistinguishable double fault case given earlier is a special case of this situation but is listed as a separate case since it can be common.

These considerations lead to the sensor validation algorithm given in Fig. 3.4.

Step 4 of the algorithm, which decides if a value differs from its predicted value (represented by the posterior distribution) requires some further explanation. The problem is to map the observed value and the distribution to a binary value: $\{correct, faulty\}$. For example, Fig. 3.5(a) shows a posterior probability distribution, and Fig. 3.5(b) shows a wider distribution. In both cases, the observed value is shown by an arrow. Intuitively, the first case can be mapped as correct while the second can be taken as erroneous.

In general, this decision can be made in a number of ways including the following.

1. Calculate the distance of the real value from the average or *mean* of the distribution, and map it to faulty if it is beyond a specified distance and to correct if it is less than a specified distance.

2. Assume that the sensor is working properly and establish a confidence level at which this hypothesis can be rejected, in which case it can be considered

1. Obtain the model (i.e., the Bayesian network) 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 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 sensors with apparent faults (S) is obtained.
6. Compare the set of apparently faulty sensors 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 sensors whose EMBs are subsets of S.
   (d) If S is equal to the union of several EMBs and the combination is unique, there are multiple distinguishable real faults in all the sensors 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 sensors whose EMBs are subsets of S could have a real fault.

Figure 3.4: Basic sensor validation algorithm.

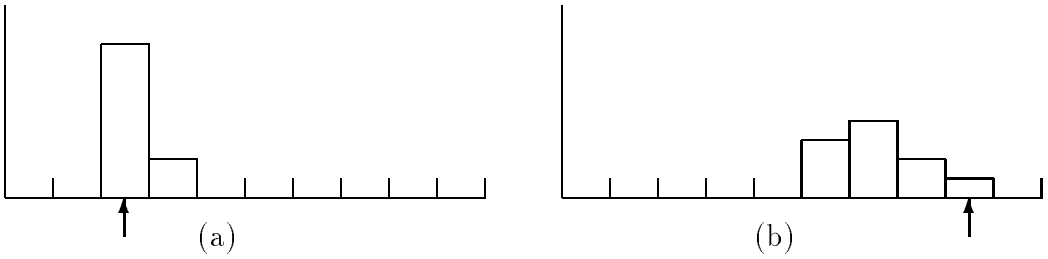

Figure 3.5: Description of some possible results. The arrows indicate the interval of the real value of a sensor.

faulty.

The first criterion can be implemented by estimating the mean μ and standard deviation σ of the posterior probability of each sensor, i.e., the distribution that results after the propagation. Then, a sensor can be assumed to be correct if it is in the range $\mu \pm n\sigma$, where $n = 1, 2, 3$ means that 68%, 95% or virtually all the cases are included respectively. This criterion allows one to work with wider distributions where the standard deviation is high and the real value is far from the mean μ value as shown in Fig. 3.5(b). However, this technique can have problems when the highest probability is close to one, i.e., the standard deviation is close to zero. In such situations, the real value *must* coincide with that interval.

The second criterion assumes as a *null hypothesis* that the sensor is working properly. The probability of obtaining the observed value given this null hypothesis is then calculated. If this value, known as the *p value* [Cohen 1995], is less than a specified level, then the hypothesis is rejected and the sensor considered faulty. Both criteria were evaluated experimentally and the results are given in Chapter 5. Here, it is worth mentioning that using the *p value* with a 0.01 rejection level, works well.

3.2 A Theory of the Sensor Validation Model

This section develops the theory that is the basis of the sensor validation algorithm given in Fig. 3.4.

The probabilistic model for sensor validation consists of a Bayesian network as defined by Pearl (1988) and described in Chapter 2. That is, given a probability distribution P on a set of variables V, a DAG D is a Bayesian network if G represents the dependency model M for a probability distribution P. For this representation, it is assumed that the model M represents strong dependencies between the sensors and from which the following assumptions can be made:

1. **Observability**: all the variables (sensors) can be measured directly [2].

2. **Fault detection**: if there is an error in sensor X it can always be detected. This is a *real fault* denoted by $Fr(X)$.

3. **Fault propagation**: if a sensor 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.*

The previous section described the importance of a Markov blanket for the sensor validation algorithm. A theory is now developed by first defining a Markov blanket formally, and then presenting theorems that corresponds to steps 6(b) to 6(e) of the algorithm given in Fig. 3.4.

A *Markov blanket* for any node X in a Bayesian network is a subset of V which makes it independent from the other variables. More formally, the following defines a Markov blanket of a variable.

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*

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

[2] A one to one correspondence between nodes, variables, and sensors is considered.

For example, in Fig. 2.1 $MB(B)$ can be the set $\{A, C, D\}$ since it satisfies $I(\{B\}, \{A, C, D\}, \{E\})$. Imagine now an additional node in Fig. 2.1 which is a parent of A, say K. In this case, the set {A,C,D,K} would also be considered a valid $MB(B)$. The following corollary [Pearl 1988] defines a Markov blanket that can be used in Bayesian networks.

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, the direct successors of X, and all direct parents of X's direct successors (i.e., spouses).*

This follows from the axioms of conditional independence, the definition of *d separation*, and the *Strong completeness* theorem [Geiger & Pearl 1988] presented in Chapter 2. Although there may be other Markov blankets, only this type of blanket is considered, and assumed in the theory below.

The *extended Markov blanket* $EMB(X)$ is defined as the union between the Markov blanket of a variable and the variable, i.e., $EMB(X) = X \cup MB(X)$.

If a sensor (variable) X has a real fault and/or apparent fault then it is called a *potential fault* $Fp(X)$. The fault detection mechanism can only tell if a sensor has a potential fault, but (without considering other sensors) it can not tell if the fault is real or apparent. So the central problem is to develop a theory for distinguishing real and apparent faults, considering that one or more sensors can fail at the same time.

The following two lemmas are needed in the proof of the main theorems. The proofs will use the notation $PA(X)$ for the parents of X, $SU(X)$ for the successors of X, and $SP(X)$ for the spouses of X.

Lemma 3.1 *(symmetry)*

Let X be a node in a Bayesian network $G = (V, E)$ with a Markov blanket[3] *$MB(X)$, $X \in MB(Y_i)$ iff $Y_i \in MB(X)$, $\forall Y_i \in V$. That is, X is in the Markov blanket of all the variables that are in $MB(X)$, and it is only in these Markov blankets.*

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 (eq. 2.14)

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

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

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

Hence, by the *Decomposition* axiom (eq. 2.15)

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

Thus X is not in $MB(Y)$.□

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

Proof:

From assumption 2, an error in X produces a potential fault in X.

[3]This lemma and the subsequent theorems apply to Markov blankets formed by the direct parents, direct successors, and direct parents of the latter.

From Lemma 3.1, X is an element of the MB of all sensors $Y_i \in MB(X)$.

So by assumption 3, an error in X produces potential faults in all sensors 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 sensor.□

Corollary 3.2 *If there is an error in sensor 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 sensors X_i, $i = 1, \ldots, m$, they will produce potential faults in all nodes $Z \in EMB(X_1) \cup \ldots \cup EMB(X_m)$.*

Corollary 3.2 follows directly from Lemma 3.2, assuming that two or more errors will not *cancel* each other (i.e., if $Z \in MB(X)$ and $Z \in MB(Y)$ and both, X and Y fail, a potential fault will still be detected in Z).

Given these two lemmas and corollary, the following main theorems can now be proved (where S is the set of potentially faulty sensors).

Theorem 3.1 *(Step 6(b) of algorithm)*

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 (by contradiction):

Suppose there is a $Y \neq X$ such that $Fr(Y)$.

Then, by Lemma 3.2, $Fr(Z)$ $\forall Z \in EMB(Y)$. Here, since S contains all the potential faults, there is a contradiction because there would be a Y such that $EMB(Y) \subseteq S$.□

Theorem 3.2 *(Step 6(c) of algorithm)*

If there is an error in sensor X with $EMB(X)$, and $Y \in EMB(X)$ with $EMB(Y) \subseteq EMB(X)$, and multiple faults (more than one sensor can fail simultaneously) are considered, then there is no distinction between $Fr(X)$ or $Fr(X) \wedge Fr(Y)$.

Proof:

By Lemma 3.2, $Fr(X)$ will produce apparent faults in $EMB(X)$.

By Corollary 3.2, $Fr(X) \wedge Fr(Y)$ will produce apparent faults in $EMB(X) \cup EMB(Y)$.

So if $EMB(Y) \subseteq EMB(X)$, then $EMB(X) \cup EMB(Y) = EMB(X)$ so both cases are indistinguishable.□

Theorem 3.3 *(Step 6(d) of algorithm)*

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

$$EMB(X_k) \not\subseteq EMB(X_1) \cup EMB(X_2) \cup \ldots \cup EMB(X_n)$$

Thus, by definition, $S = EMB(X_1) \cup EMB(X_2) \cup \ldots \cup EMB(X_n)$ so, an additional faulty sensor 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,

$$EMB(X_j) \not\subseteq \cup_{i \neq j} EMB(X_i)$$

and $EMB(X_j) \not\subseteq S$, which is a contradiction.□

Theorem 3.4 *(Step 6(e) of algorithm)*

If the set of nodes S with apparent faults in G is different from all $EMB(X_i)$, $\forall X_i \in G$, there must be multiple (at least 2) real faults *in G. Only the sensors X_i such as $EMB(X_i) \subseteq S$ can have real faults.*

Proof:

From Lemma 3.2, a real fault in X produces apparent faults in and only in the set of sensors in $EMB(X)$.

So a single fault can not produce a set S of potential faults different from all EMB in G.

From Corollary 3.2, the sensors whose EMB is a subset of S can be in fault, and by Lemma 3.2, only these sensors can be faulty.□

The above four theorems provide the basis of the algorithm given in section 3.1. Later, in Chapter 4, the following theorem is also needed.

Theorem 3.5 *If there is no potential fault in sensor X with Markov blanket $MB(X)$, then all sensors Y_i, $Y_i \in MB(X)$ have no real faults.*

Proof: By contradiction.

Suppose a Y_i is faulty. Then, by assumption 3, X will be reported faulty.□

This theorem, and indeed the above theory is dependent on assumption 3 and further comment about this assumption is therefore necessary. The assumption states that if a variable has an error, then it will have a significant effect on predicting the variables that are dependent on it. This assumption is reasonable, as mentioned before, only if the dependencies expressed in the Bayesian network are strong. In applying the model, a user must therefore provide a Bayesian network whose dependencies are indeed strong.

3.3 An Example

The example of the gas turbine of Fig. 3.1, whose probabilistic model is shown in Fig. 3.2, was utilized to introduce the algorithm developed. This section now, describes the application of the algorithm of Fig. 3.4 for that simple case from step 2 onwards. Step 2 produces the EMB table given in Table 3.1. Suppose steps 3 to 5 give the results shown in Table 3.2. Then, step 6 indicates that

Table 3.2: Steps 3, 4 and 5 of the algorithm for the example of Fig. 3.1.

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

$S = \{m, p\} = EMB(p)$ so it can be concluded that the real fault is in sensor 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 sensors g and a.

3.4 Summary

This chapter has presented a theory and an algorithm for probabilistic sensor validation. A preliminary version was presented in the paper by Ibargüengoytia et al. (1996*a*). It starts by obtaining a probabilistic model which considers the dependence relationships between the sensors. Then, by probabilistic propagation,

the value of a variable is estimated based on the readings of the related signals. These related signals correspond to a Markov blanket defined for Bayesian networks. Then, by comparing the result of the probabilistic validation, with the Markov blankets of all the sensors, it is possible to distinguish between real and apparent faults as described in section 3.2.

However, notice that this algorithm returns an answer only after all the sensors have been validated.

Chapter 4

ANY TIME SENSOR VALIDATION

The previous chapter described the approach to sensor validation taken in this thesis. Briefly, it works by modelling the relations among the sensors in a process and estimating the value of each sensor by probabilistic propagation. Then, the estimated and the real value are compared to detect a potential fault. Since this comparison can not distinguish between real and apparent faults, a fault isolation phase is used.

However, the sensor validation process described in Chapter 3 works in batch mode, i.e., no intermediate results are available, and no attempt is made to estimate the quality of the result. For a real time application, these characteristics are inadequate. By definition, a real time system is one where the correctness of a system depends not only on the logical result of the computation but also on the time at which the results are produced [Stankovic 1988]. Usually, real applications possess a time limit by which some actions must be performed.

This chapter describes how the algorithm developed in Chapter 3 is extended so that it is more appropriate for real time applications. Section 4.1 provides the context for this chapter by summarizing some previous work that combines

AI with real time systems. Specifically, it introduces *any time algorithms*, the technique used in this thesis for developing a more appropriate sensor validation algorithm for real time applications. Next, section 4.2 develops the main parts of the any time sensor validation algorithm. Finally, section 4.3 presents and comments on the resultant any time sensor validation algorithm, and section 4.4 summarizes this chapter.

4.1 Any Time Algorithms

Real time and artificial intelligence techniques started to be combined in the 1980s. One of the first surveys exploring the consequences of this combination was made by Laffey et al. (1988). They discussed the challenges of real time expert systems and described the problems that needed to be solved, e.g., a timely response, uncertainty management, continuous operation, interface to the external environment via sensors and actuators, etc. More recently, Strosnider & Paul (1994) presented their structured view of real time problem solving, structuring the problem space and the search process to reduce the variations of the problem in order to produce solutions within the time available. In another recent study, Musliner et al. (1995) described their view of the challenges of real time AI. They considered the following three basic approaches: (i) embedding AI into a real time system, (ii) embedding real time reactions into an AI system, and (iii) coupling AI and real time subsystems in a cooperative environment. Examples of applications in these categories include the following.

A real time multi tasking knowledge based system. This fully integrated real time multi tasking KBS approach [Grelinger & Morizet-Mahoudeaux 1992] adds expert system capabilities to a real time multi tasking kernel in a way that aims to achieve two main goals: (i) maintain the power of

knowledge representation schemes, and (ii) maintain the efficiency of real time software.

CROPS5. Concurrent real time OPS5 [Paul et al. 1991] architecture is a modification of Carnegie Mellon's rule based system OPS5 that enables real time reactions to be handled. It provides predictable low variance primitives for problem solving, and provides features which facilitate easy integration into real time operating environments.

CIRCA. This is a cooperative intelligent real time control architecture [Musliner et al. 1993]. This architecture separates an AI subsystem and a real time subsystem (RTS). The AI subsystem reasons about task level problems that require its powerful but unpredictable reasoning methods. The RTS uses its predictable performance characteristics to deal with control level problems that require guaranteed response times.

By their very nature, embedded approaches are closely tied to the application in which they reside. Hence, since this thesis aims to develop a more generic approach, the coupling architecture is more appropriate.

Given any of the above approaches, the main problem in a real time system is to return an answer in time for it to be used. One direction of work that aims to achieve this goal is the development of *any time* algorithms. This term was initially used by Dean in his research about time dependent planning [Dean & Boddy 1988]. At the same time, Horvitz (1987) proposed the name of *flexible computation* for this any time mechanism. Any time algorithms are those that can be interrupted at any point during computation, and return an answer whose *value* increases as it is allocated additional time [Boddy & Dean 1994]. However, how can this value be measured in a specific application? The literature contains descriptions of different dimensions that have been proposed as metrics [Strosnider & Paul 1994, Zilberstein & Russell 1996]:

Certainty. This metric reflects the degree of certainty in the results of the algorithm. This metric can be expressed as the probability of the result being correct.

Accuracy. This metric reflects the degree of accuracy in the results of the algorithm. This metric can be used when the exact solution is known, and the difference from the exact solution can be calculated. This error should decrease as the computation time increases.

Specificity. This metric reflects the level of detail in the solution. In this case, an any time algorithm always produces correct results, but the level of detail is increased over time.

Performance profiles represent the expected value of these metrics for a given procedure as a function of time. In other words, performance profiles characterize the quality of an algorithm's output as a function of computation time. Figure 4.1 illustrates three cases of performance profiles [Zilberstein & Russell 1996], [Dean & Boddy 1988]:

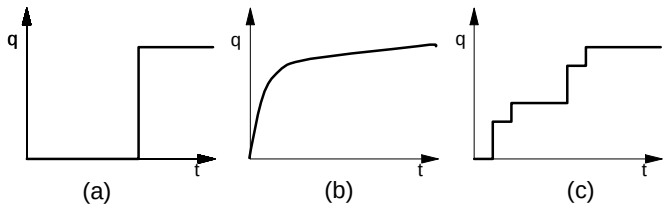

Figure 4.1: Examples of performance profiles. (a) a standard or one shot algorithm. (b) an ideal, exponential precision algorithm, and (c) a more realistic profile for an any time algorithm in practice.

Figure 4.1(a) shows a standard or *one shot* algorithm where the system produces no answer until a certain time, and then produces the answer with a bounded precision. Figure 4.1(b) shows an ideal any time algorithm whose answer has exponential and increasing precision. Finally, Fig. 4.1(c) presents a more realistic profile for an any time algorithm in practice. Clearly, all these

types of performance profiles are special cases of a superclass that can be defined as *monotonic improvement*, i.e., the quality of its intermediate results does not decrease as more time is spent to produce the result. Chapter 6 presents a more extensive description of the current use of any time algorithms in the real time artificial intelligence community and specifically in probabilistic reasoning. The next section develops the any time algorithm for the sensor validation problem.

4.2 Any Time Sensor Validation Algorithm

Any time sensor validation implies that the knowledge about the state of the sensors (faulty or correct) becomes more certain and complete as time progresses. Certainty about the state of a sensor refers to the degree of belief in the correctness of a sensor, and completeness is characterized by the number of sensors from which the state is known. Thus, it is required to be able to monitor the state of the sensors during all the validation process. This is done through a vector whose elements $P_f(s_i)$ represent the probabilities of failure for the sensors 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. 4.2.

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

Figure 4.2: Top level of the any time sensor validation algorithm.

This algorithm contains five main steps or functions: (i) initialization, (ii)

choosing the next sensor to validate, (iii) validating a single sensor, (iv) isolating the failure, i.e., updating the probability of failures, and (v) measuring the quality of the partial response. Step (iii), validating a single sensor, was already presented and discussed in Chapter 3.

Before describing the other steps, it is worth mentioning the author's previous attempt from which the any time sensor validation algorithm presented below has evolved [Ibargüengoytia et al. 1996*b*, Ibargüengoytia et al. 1997]. The intuition that the sensor with the largest Markov blanket provides more information was used to decide which sensor to select next. Also, potentially faulty sensors were preferred over other sensors in order to isolate the fault as soon as possible. Although the empirical results obtained in these earlier attempts were fairly good, the lack of a theoretical basis for the heuristics prompted the development of the algorithm whose steps are described below.

4.2.1 Initialization

The probability of failure of a new sensor is small. As it becomes older and depending on the manufacturing process, the probability of failure may increase. The inclusion of these aspects in order to initialize the model would complicate the generation of the model and its utilization. So, in a trade off between expressivity and efficiency, the mechanism proposed here considers the execution of this algorithm as an independent cycle. A cycle is defined as the validation of all sensors in the system independently of previous cycles of the history of the process. Future extensions of the work could aim to utilize the extra knowledge provided by the sensor's manufacturers (see Chapter 7). Thus, for a given cycle, the initialization is based on the assumption of ignorance [Neapolitan 1990] about the chances of failure of all the sensors, i.e., $P_f(s_i) = \frac{1}{2}$ for $i = 1, \ldots, n$.

4.2.2 Selection of next sensor: Use of information theory

This section develops a mathematical model for choosing the best sensor to validate 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 sensor provides some information and also, extra information can be inferred. Therefore, a measure of the information that a single validation produces is required. A definition of the expected amount of information that an event produces was first proposed by Shannon and used in communication theory [Shannon & Weaver 1949], and then utilized in applications like machine learning [Quinlan 1986]. Shannon proposed the following definitions.

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

$$H_n = H_n(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{4.1}$$

Thus, the entropy measures the related number of bits required to store the information. For example, in a coin tossing experiment, the entropy is defined as:

$$H(p(head), p(tail)) = H(\tfrac{1}{2}, \tfrac{1}{2}) = -\tfrac{1}{2} log_2 \tfrac{1}{2} - \tfrac{1}{2} log_2 \tfrac{1}{2} = 1$$

This means that only one bit of information is enough for storing all the information of the experiment: head/tail.

Since the validation of a sensor s has two possible outcomes, as in the coin tossing experiment, the entropy function $H(s)$ is then defined as:

$$H(s) = \begin{cases} 0 & \text{if } p = 0 \text{ or } p = 1 \\ -p log_2(p) - (1-p) log_2(1-p) & \text{otherwise} \end{cases} \tag{4.2}$$

where p represents the probability of failure of the sensor. Notice that the expression $plog_2(p) = 0$ when $p = 1$ but it is undefined when $p = 0$. However, since $plog_2(p)$ tends to zero as p tends to zero, the values defined in equation 4.2 can be safely assumed. Figure 4.3 shows the behaviour of the function [Pratt 1994]. Notice that it has its maximum when $p = \frac{1}{2}$, i.e., when the ignorance is maximum, and it is zero when either $p = 0$ or $p = 1$, i.e., when the information is maximum and ignorance is minimum. This function can be considered either as a measure

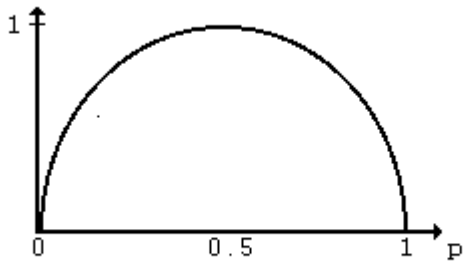

Figure 4.3: Entropy as a function of p.

of the information provided by an experiment, or as a measure of the uncertainty in the experiment's outcome. Thus, considering each single sensor validation as an experiment, this function can be used to measure the amount of information provided by that validation. Then, the average amount of information for the system can be defined as follows:

$$ENT(s_1, \ldots, s_n) = \frac{1}{n}\sum_{i=1}^{n} H(s_i) \tag{4.3}$$

$$= -\frac{1}{n}\sum_{i=1}^{n} P_f(s_i)log_2 P_f(s_i) + (1 - P_f(s_i))log_2(1 - P_f(s_i))$$

$$= -\frac{2}{n}\sum_{i=1}^{n} P_f(s_i)log_2 P_f(s_i) \tag{4.4}$$

where n is the number of sensors in the system S, and $P_f(s_i)$ represents the current probability of failure value assigned to sensor s_i. Notice that the vector whose elements are $P_f(s_i)$ provides a measure of the certainty in the validation while the sum of n individual entropies provides a specificity measure of the result.

Given this measure, the any time sensor validation algorithm needs to select a sensor that gives the best improvement in the average entropy of the system. Hence the following conditional version of equation 4.3 can be written

$$\begin{aligned} ENT(S \mid X) &= ENT(S \mid x = correct) + ENT(S \mid x = faulty) \\ &= \frac{1}{n}\left(\sum H(s_i \mid x = correct) + \sum H(s_i \mid x = faulty)\right). \quad (4.5) \end{aligned}$$

This function can be evaluated for each sensor and the one which gives the most information (the minimum $ENT(S \mid X)$) can be selected as the next sensor to be validated.

The computation suggested by the above formulae could be too expensive for a real time sensor validation process. To overcome this problem, this thesis pre compiles the sensor selection mechanism as follows. The above formulae are used to select the sensor, s_r which gives the most information. This selected sensor forms the root of a binary decision tree. A fault is simulated in this sensor and the formulae are again used to select the next sensor s_{r-}. Then, the root s_r is assumed to be correct, and the formulae are used to select the sensor s_{r+} in this case. This results in the partial decision tree shown in Fig. 4.4. This process is

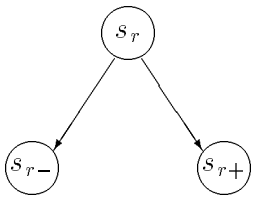

Figure 4.4: Partial decision tree.

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 sensors.

As an example, consider the network shown in Fig. 4.5. This process results in the decision tree shown in Fig. 4.6.

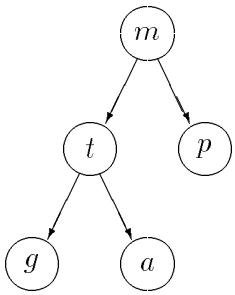

Figure 4.5: A reduced Bayesian network of a gas turbine.

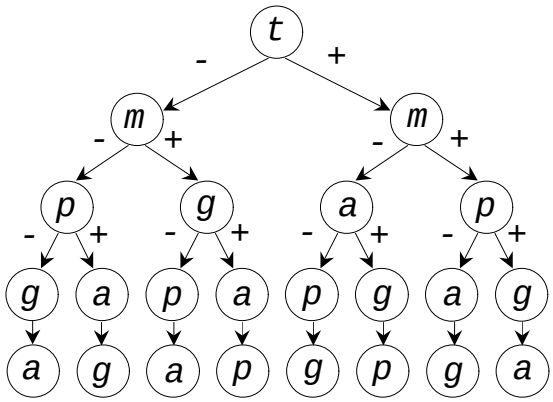

Figure 4.6: Binary tree indicating the order of validation given the response of the validation step.

This decision tree can be used to select the next sensor more efficiently in real time than by performing the calculations. Thus, the selection step of the algorithm of Fig. 4.2 consists of simply traversing the tree one level after every single sensor 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 sensor. Notice that the binary tree for n sensors contains n levels and up to 2^{n-1} nodes[1]. Then, if a system has 21 sensors, the binary tree would require 1,048,575 nodes, and assuming 10 bytes per node this tree requires more than 10 Mbytes of memory. This is a typical example of a trade off between the computing time and memory usage in a real time system.

To accommodate situations when memory is short, and only single faults are considered, this thesis proposes an alternative approach to the pre compilation of the decision tree. This considers the sequence of validations when single faults are simulated in all the sensors of the system. Of course, the case of no faults at all must also be considered. Table 4.1 presents the valid trajectories followed in the case of failures (first column) of the example of Fig 4.5. The plus sign represents the correct validations while the minus sign represents a faulty condition in the sensor. The first sensor to validate is always t since it has the lowest conditional

Table 4.1: Trajectories of validation in the case of single faults. The + represents the validation as correct while - represents a fault in the sensor.

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	$t-$	$m+$	$g+$	$p+$	$a-$

entropy. The first row indicates the case of no failures. The second row presents

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

the sequence of validation when m is simulated as faulty. Here, all $m'sEMB$ will be faulty. Representing the information of Table 4.1 in a decision tree results in a pruned tree with n levels and at most $n \times (n+1)$ nodes as shown in Fig. 4.7. This is because only the rows in Table 4.1 would be trajectories in the tree from the root to the leaves.

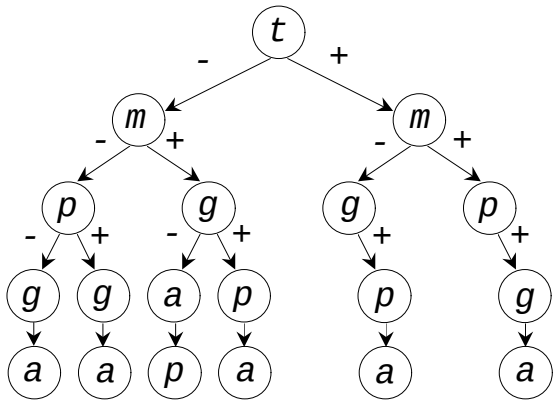

Figure 4.7: Reduced decision tree.

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. 4.7. 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. 4.7 the next sensor 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).

4.2.3 Fault isolation

Chapter 3 presented an algorithm for the isolation of one or more failures in the sensors. Briefly, a comparison is made between the set of potentially faulty sensors with the table of extended Markov blankets of all the sensors. When a match is found, a real fault is determined. However, the set of potentially

faulty sensors is obtained after all the sensors 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 provides, as the output of the isolation phase, a vector with the probability of a real fault in all the sensors. This vector is refined incrementally in time, so the any time behaviour can be achieved.

The any time fault isolation process is based on the relationship between real and apparent faults. There are two situations that arise: (i) the existence of a real fault *causes* an apparent fault (as shown in Fig. 4.8(a)), and (ii) one apparent fault is the manifestation of several possible real faults (as shown in Fig. 4.8(b)).

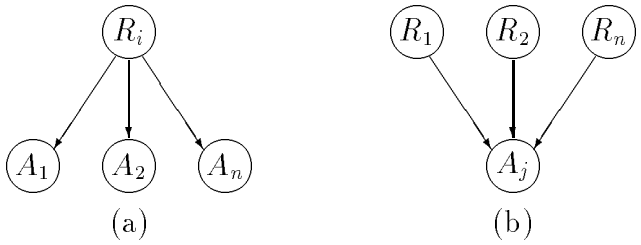

Figure 4.8: Causal relation between real faults (R) and apparent (A) faults represented as nodes.

In both figures, the relation between root nodes and leaf nodes is the same as the extended Markov blanket (EMB) of a sensor. Considering all the sensors, a causal model relating the real and apparent faults can therefore be obtained from the Bayesian network (in fact, the EMB table is sufficient to build this network). In the first level (roots), the nodes represent the events of real failure in every sensor. Then, the second level (leaves) is formed by nodes representing apparent failures in all the sensors. Arcs are included between every root node, and the corresponding nodes of the extended Markov blanket. For example, the causal network shown in Fig. 4.9 can be obtained directly from the Bayesian network

of the gas turbine given in Fig. 4.5. Thus, the consequences of observing an

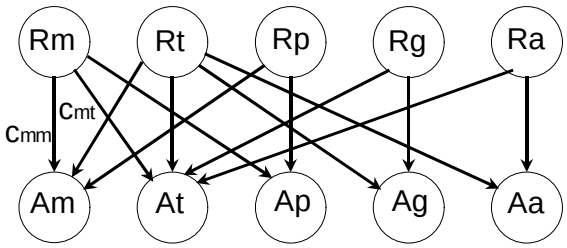

Figure 4.9: Probabilistic causal model for fault isolation. R_i represents a real fault in sensor i while A_j represents an apparent fault in sensor j.

apparent fault can be propagated in the causal network in order to obtain the probabilities of a real fault in all the sensors.

The network of Fig. 4.9 is multiply connected since several root nodes share leaf nodes. This would produce loops in the propagation algorithm of singly connected networks. Hence, the propagation method of trees of cliques presented in Chapter 2 is utilized. The adoption of this technique requires: (i) the specification of the prior and conditional probabilities, and (ii) the specification of the tree of cliques.

In general, $O(2^n)$ conditional probabilities would be required (for a node with n parents). However, the Peng & Reggia (1987) causal model can be adopted here. As described in Chapter 2, two assumptions need to hold in order to use this model.

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 sensor is apparently faulty only if there is a fault in its MB. The exception

independence assumption is concerned about a rare situation for this particular model. The relationship between the real and apparent faults is obtained from a Bayesian network in which the dependencies are assumed to be strong. 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 sensor 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 conditional probability matrix can be calculated by utilizing equation 2.46. That is, the only parameter required is defined as:

$$c_{ij} = 1 - q_{ij} = P(A_j \mid R_i \; only).$$

In the case of the sensor validation problem, in an ideal case, all the parameters $c_{ij} \approx 1$. Of course, these values can be obtained by simulation from the data if the problem is expected to depart from this ideal case. That is, according to the theory developed in Chapter 3, when a real fault R_i is present, it will always cause the apparent fault A_j (assuming that there is an arc from R_i to A_j).

The second problem, that of specifying the tree of cliques is addressed as follows. In general, following the procedure of Fig. 2.6 will provide a valid tree of cliques representing the original network. However, in the case of the causal network shown in Fig. 4.9, the cliques can be deduced directly. Figure 4.10 shows the five cliques deduced after the moralization and triangulation process. The

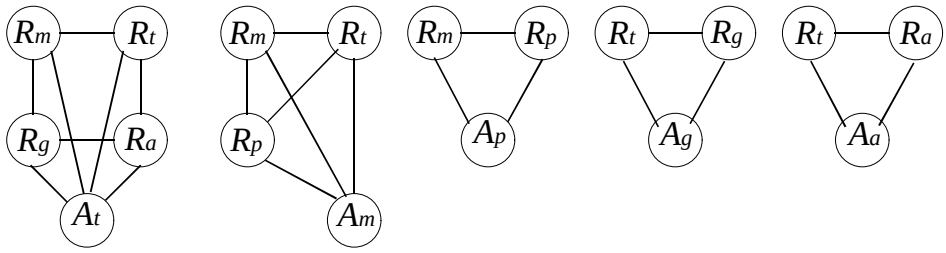

Figure 4.10: Cliques obtained from the network in Fig. 4.9.

cliques can be formed by relating each one of the leaves with all its parents. This

condition always holds in this kind of network, i.e., two level causal networks[2]. Now, numerating the nodes (as described in Chapter 2) from the left of Fig. 4.9, i.e., starting with node R_m, produces the following clique numbering:

$$\begin{aligned} Clq_1 &= \{R_t, R_g, A_g\} \\ Clq_2 &= \{R_m, R_t, R_g, R_a, A_t\} \\ Clq_3 &= \{R_t, R_a, A_a\} \\ Clq_4 &= \{R_m, R_t, R_p, A_m\} \\ Clq_5 &= \{R_m, R_p, A_p\} \end{aligned} \tag{4.6}$$

This ordering produces the tree of cliques shown in Fig. 4.11. The root is Clq_1

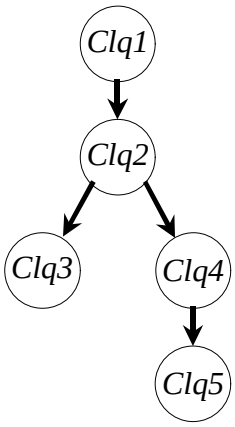

Figure 4.11: Tree of cliques obtained from the network in Fig. 4.9.

and obviously Clq_2 follows as a child. To decide the location of Clq_3 to Clq_5, equation 2.36 is applied as follows:

$$Clq_3 \cap \{R_m, R_t, R_g, R_a, A_t, A_g\} \subseteq Clq_2$$

$$Clq_4 \cap \{R_m, R_t, R_g, R_a, A_t, A_g, A_a\} \subseteq Clq_2$$

[2]Since the moralization process relate all the parents of a leave node, this clique becomes complete and maximal as established in the definition of cliques of Chapter 2.

$$Clq_5 \cap \{R_m, R_t, R_p, R_g, R_a, A_m, A_t, A_g, A_a\} \subseteq Clq_4 \tag{4.7}$$

Finally, the cliques are initialized with the information of the original network. That is, the prior probability of all the root nodes in the original model is 0.5 (the ignorance assumption of section 4.2.1) and the parameters $c_{ij} = 0.99$ for all $1 \leq i, j \leq$number of nodes.

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 sensor with its detected state (faulty or correct) and updates the probability of failure of all the sensors. It does this by instantiating the value of the corresponding apparent 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 sensors, and can be viewed as the output of the system at any time. For example, assuming a fault in g in the network of Fig. 4.5, produces the sequence of values of the probability vector as shown in Table 4.2.

Table 4.2: 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

A more complete example of the case study can be found in Appendix A.

4.2.4 Quality measure

Section 4.1 presented the proposed dimensions in which the quality of an answer can be specified: certainty, accuracy, and specificity. The question now is, *are all these measures applicable to the sensor validation algorithm?* and if so, how are they interpreted? In a previous attempt at developing an any time sensor validation algorithm [Ibargüengoytia et al. 1997], the following measure was used. Suppose there are four lists: a list FL with the number of real, detected faulty sensors; a list CL with the real correct sensors, and lists PFL, PCL for potentially faulty and correct sensors respectively. Thus, a quality function q may take the following form:

$$q(F, C, PF, PC) = \alpha FL + \beta CL + \gamma PFL + \delta PCL \tag{4.8}$$

where $\alpha, \beta, \gamma, \delta$ are weights given to the number of sensors in each one of the lists F, C, PF, PC. For example, a function with, $(\alpha, \beta, \gamma, \delta)$ = (10,10,2,2) assigns five times more quality to an answer with more sensors in the F and C lists than in the potential lists. Another application may give greater weight to the real faults and utilize $(\alpha, \beta, \gamma, \delta)$ = (20,5,1,1). That is, such a measure requires deciding the weights depending on the application.

A better measure that is independent of the application and that naturally combines the measures described in section 4.1 is the average entropy of the sensors given in equation 4.3. That is, if the current quality measure is:

$$Q(s_1, \ldots, s_n) = -\frac{1}{n}\sum_{i=1}^{n} P_f(s_i) log_2 P_f(s_i) + (1 - P_f(s_i)) log_2 (1 - P_f(s_i)) \tag{4.9}$$

then, the reported quality function is calculated with the formula:

$$Q = \frac{Q_{max} - Q_{current}}{Q_{max}} \tag{4.10}$$

where Q_{max} is the maximum value of the quality measure (i.e., n, the number of nodes). Notice that this measure captures both the certainty and specificity

measures of any time algorithms. It captures certainty since the probabilities of the sensors are used, and specificity since all the sensors are combined to give an average. Figure 4.12 shows the performance profile obtained with this quality measure for the example of Fig 4.5.

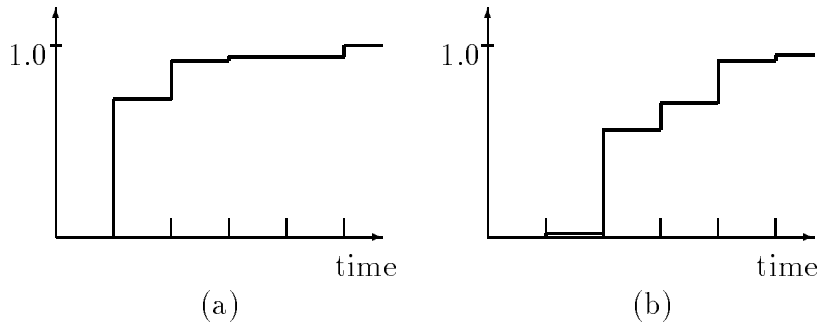

Figure 4.12: Performance profile describing the combination of certainty and specificity in one parameter against time. (a) without failure, (b) with a simulated failure in sensor g.

Again, since it is expensive to compute this quality function Q at every step of the validation, and since this entropy average can be calculated in the off line mode, a node of the decision tree of Fig. 4.6 includes the quality measured until the corresponding validation step. Additionally, since the entropy and the quality values require the computation of the probability of failure, then these probabilities are also stored in a node of the pre compiled binary decision tree. Thus, the binary tree performs: the selection of the next sensor, the current probability of failure vector, and provides the current quality provided by the algorithm. Figure 4.13 describes the fields of information in one node of the binary decision tree where ptr OK represents the pointer to the next node when the current sensor is correct, and ptr FL to the next when faulty. This allows the algorithm to perform in a time appropriate for real time applications. The next section summarizes and comments on the resultant any time sensor validation algorithm.

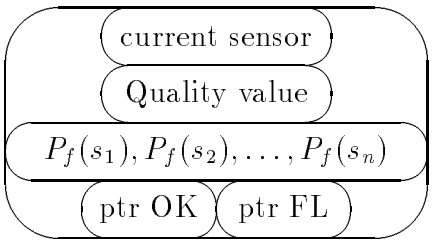

Figure 4.13: Format of a node of the pre compiled binary decision tree.

4.3 The Complete Algorithm

The above section described the basis of the steps of the any time sensor validation algorithm in some detail. This section brings together the above material and presents the whole algorithm in one place. Figure 4.14 presents the top level of the algorithm.

The initialization step is carried out off line, i.e., these are calculations that can last as long as they need. Their function is to keep the *in line* process running on an any time basis. This step obtains:

- the binary tree for the selection of the next sensor to validate, and
- the current quality measure.

With this information, and according to the layered structure presented in Chapter 1, this algorithm can be interrupted at any time yielding a partial result. This result is qualified by the quality function so that an operator or the higher layers may decide whether to stop the process and take urgent action. For example, if the algorithm indicates the presence of a fault, and the quality function reaches 80 %, it signifies that enough certainty and specificity are available in order to start corrective action, i.e., to notify the control system to ignore a faulty sensor.

Notice that, unlike the algorithm presented in Chapter 3, the algorithm of

Any time sensor validation algorithm.

Input: The Bayesian network.

Output: The vector of probabilities of failure and the quality measure, visible at any time.

- Initialization
  - Obtain the EMB table from the Bayesian network.
  - Construct the causal model relating the real and apparent faults by using the Bayesian network.
  - Pre compile the selection criteria to obtain the decision tree.
- Do continuously
  - Initialize the prior probabilities of failure.
  - While there exist unvalidated sensors do
    1. Select the next sensor using the decision tree.
    2. Validate it.
    3. Update the vector that records the probability of failure and quality function.

Figure 4.14: Complete version of the any time sensor validation algorithm.

Fig 4.14 can be combined with the real time higher layers under a real time operating system.

Figures 4.15 and 4.16, give the pre compilation process that provides the decision tree for selecting the next sensor. Figure 4.15 presents the algorithm for producing the complete binary tree (as in Fig. 4.6), and Fig. 4.16 gives the algorithm that produces the pruned decision tree (as in Fig. 4.7).

Pre compilation procedure (complete version).

Input: The set of unvalidated sensors S.

Output: The decision tree DT.

- If S is empty, return the empty tree.
- Select a sensors s_i from S that gives the most information.
- Set the apparent node for s_i in the causal model to a fault.
- Propagate the probabilities.
- Let $DTleft$ be the decision tree obtained recursively for the sensors $S - \{s_i\}$.
- Set the apparent node for s_i in the causal model as correct.
- Propagate the probabilities.
- Let $DTright$ be the decision tree obtained recursively for the sensors $S - \{s_i\}$.
- Return a decision tree DT whose left subtree is $DTleft$, whose right subtree is $DTright$, and whose root is s_i.

Figure 4.15: Complete pre compilation procedure.

The in line process runs continuously in a cyclic manner. Figures 4.17 and

Pre compilation procedure (reduced version).

Input: The set S of sensors, and the EMB table of all the sensors.

Output: The decision tree DTr.

- If S is empty, return the empty tree.
- Let *table* be a sequence that is initially empty.
- Repeat for every single failure in sensor s_j and for no failures.
  - While there exists unvalidated sensors do:
    - Select the sensors s_i from S that gives the most information.
    - If s_i is in the $EMB(s_j)$ then consider it as faulty, otherwise consider it correct.
    - Add s_i and its state to the j^{th} entry of the *table*.
- For all the sequences in the *table* do
  - While there are sensors in the sequence
    - Follow the decision tree DTr until a null pointer in the tree is found.
    - Create a new node with the current sensor (as shown in Fig. 4.13) in DTr and substitute the found null pointer with the pointer to the new node.
- Return a decision tree DTr.

Figure 4.16: Reduced pre compilation procedure.

4.18 describe the validation and isolation functions in more detail.

Function validate.

Input: A node a.

Output: Either correct or faulty.

- Instantiate all nodes except a.
- Propagate probabilities and obtain a posterior probability distribution of a.
- Read real value of sensor represented by a.
- If $P(real\ value) \geq p\ value$ then return(correct), where $p\ value$ is a specified threshold.
- Else return(faulty).

Figure 4.17: Description of the *validation* process.

4.4 Summary

This chapter has presented the development of the any time sensor validation algorithm. First, it introduced the any time algorithms as the technique utilized in this thesis for making the sensor validation algorithm appropriate for real time applications. Summarizing, the any time algorithm consists of the following operations (Fig. 4.2):

1. choose the next sensor to validate,
2. validate it,
3. update the probability of failure vector P_f, and
4. measure the quality of the partial response.

Function isolate

Input: A sensor n and the state of sensor n.

Output: The vector $P_f(sensors)$.

- Instantiate the apparent fault node corresponding to n with the indicated state.
- Propagate probabilities and obtain the posterior probabilities of all the *Real fault* nodes.
- Update vector $P_f(sensors)$.

Figure 4.18: Description of the *isolation* process.

The selection of the next sensor to validate, and the quality function, utilize Shannon's entropy function for calculating the amount of information that every validation provides. The validation utilizes probability propagation in trees, while updating the vector P_f uses propagation in a probabilistic causal model.

Finally, this chapter has presented the complete sensor validation algorithm. The algorithm has been implemented in C. The results of the experiments with real data from the temperature sensors of a thermoelectrical power plant are presented in the next chapter.

Chapter 5

EXPERIMENTAL RESULTS

Chapter 3 developed the theory for probabilistic sensor validation based on a Markov blanket property. That chapter also developed an algorithm that compares the list of apparent faults against the extended Markov blanket (EMB) in order to distinguish the real and the apparent faults. In applying the model, practice may deviate from the ideal theory and practical considerations could affect the accuracy of the results. In particular, the following points should be noted:

1. It may not be possible to obtain a dependency model that is complete and very accurate.

2. The criteria for mapping from the expected value and the actual reading to $\{correct, faulty\}$ (described in section 3.1) could affect the accuracy of the results. For instance, if a sensor is only considered to be faulty if it deviates a long way from its expected behaviour, then almost all sensors detected as faulty will really be faulty but there will also be many faulty sensors that are not detected.

Another significant part of the thesis is the development of the any time sensor validation algorithm in Chapter 4. An important part of the any time algorithm

was the development of the theory that selects the next sensor that maximised the information gained. A fair question is:

> does this selection policy result in a better quality answer more quickly in practice?

This chapter carries out an empirical evaluation of the core model and the any time algorithm. The chapter is organized as follows.

- Section 5.1 explains the application domain. That is, where the sensors to be validated are located, how they provide information and how that information is processed from raw data to obtain the probabilistic model expressed as a Bayesian tree.
- Section 5.2 describes the architecture of the implemented system and test environment.
- Section 5.3 presents the results of evaluating the accuracy of the model and the effect of the criteria for deciding if a sensor is faulty.
- Section 5.4 presents an evaluation of the any time sensor validation algorithm and presents the resultant performance profile of the system

5.1 Application Domain

The sensor validation algorithm was evaluated by applying it to the validation of temperature sensors of the gas turbine at the *Gómez Palacio* power plant in México. This is an interesting application of these techniques for many reasons. For example, since a functional model of the temperatures of a turbine is difficult to obtain, it is a good candidate for probabilistic methods. Also, the size of this problem makes it ideal for testing the development of the theory and algorithm.

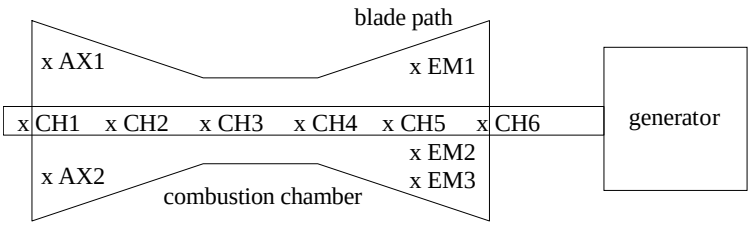

Figure 5.1: Simplified schematic diagram of a gas turbine.

Figure 5.1 shows a simplified diagram of a gas turbine. The combustion chamber receives air and gas in a specific proportion to produce high pressure gases at high temperature. These gases produce the rotation that moves the generator. Thus, the temperature is considered the most important parameter in the operation of the turbine since it performs more optimally at higher temperatures. However, a little increase in the temperature, over a permitted value, may cause severe damage. The distributed control system that governs the plant is continuously monitoring these signals in order to correct any deviation of the process. In the case of an illegal increase of a temperature parameter, the plant is stopped and taken to a safe state. Conversely, an error in a sensor's measure may cause an unnoticed increase of the temperature, or may result in an unnecessary shut down. The consequences of the former can be severe damage to the equipment and even human fatalities, and the latter could result in loss of time and fuel. Figure 5.1 shows the physical location of some of the temperature sensors used in the turbine. It shows six sensors across the beadings of the shaft ($CH1, CH2, \ldots, CH6$), three sensors on the turbine blades ($EM1$, $EM2$ and $EM3$), and two sensors of the temperature of the exciter air ($AX1$ and $AX2$). The experiments were carried out over a set of 21 sensors (though not all are shown in Fig. 5.1). These sensors can be grouped into the following sets of measurements:

- 6 beadings (CH1 - CH6),
- 7 disk cavities (CA1 - CA7),

- 1 cavities air cooling (AEF),
- 2 exciter air (AX1 - AX2),
- 3 blade paths (EM1 - EM3), and
- 2 lub oil (AL1 - AL2).

The instrumentation of the plant provides the readings of all the sensors every second. The data set utilized in the experiments corresponds to the temperature readings taken during approximately the first 15 minutes after the start of the combustion. That corresponds to the start up phase of the plant, where the thermodynamic conditions change considerably. Therefore, the data set consists of 21 variables and 870 instances of the readings.

The first step in using the sensor validation algorithm is to provide a dependency model. A dependency model for the temperatures was obtained by utilizing an automatic learning program that uses the algorithm described in Chapter 2 [Sucar et al. 1995]. Figure 5.2 shows the tree obtained with this data set. Notice

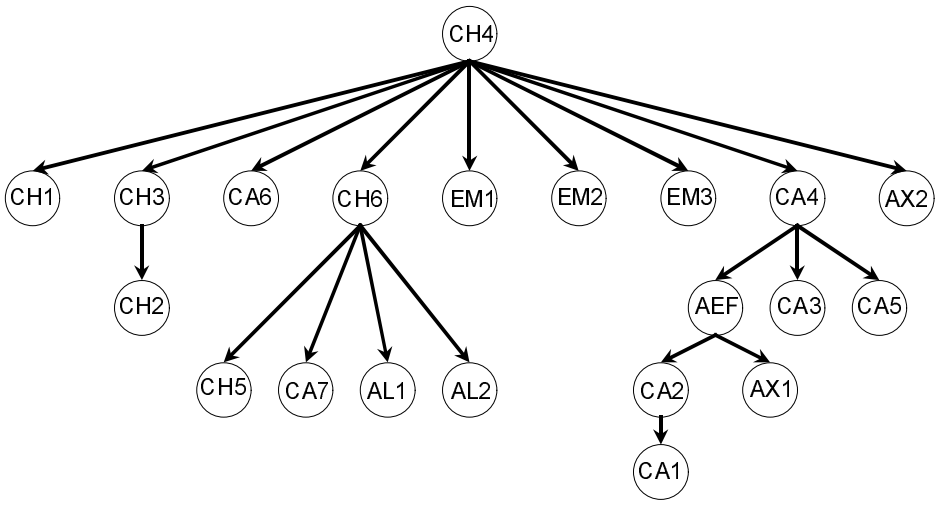

Figure 5.2: Bayesian network for this application.

that the dependencies can be explained as the *propagation* of heat from the centre

of the turbine (CH4) to the extremes. CH4 is the measure of the beading temperature which is closer to the combustion chamber, and can be modelled as the tree's root, i.e., the variable which *causes* the other variable's heating. This explanation is an intuitive interpretation of the resultant probabilistic model in terms of causality.

Table 5.1 shows the extended Markov blankets of all the sensors corresponding to the model of Fig. 5.2. Notice (in Table 5.1) that all the extended Markov

Table 5.1: EMB of all sensors in the application example.

CH4	{CH4,CH1,CH3,CA6,CH6,EM1,EM2,EM3,CA4,AX2}
CH1	{CH4,CH1}
CH3	{CH4,CH3,CH2}
CH2	{CH3,CH2}
CH6	{CH4,CH6,CH5,CA7,AL1,AL2}
CH5	{CH6,CH5}
CA4	{CH4,CA4,AEF,CA3,CA5}
AEF	{CA4,AEF,CA2,AX1}
CA2	{AEF,CA2,CA1}
CA1	{CA2,CA1}
CA3	{CA4,CA3}
CA5	{CA4,CA5}
CA6	{CH4,CA6}
CA7	{CH6,CA7}
AX1	{AEF,AX1}
AX2	{CH4,AX2}
EM1	{CH4,EM1}
EM2	{CH4,EM2}
EM3	{CH4,EM3}
AL1	{CH6,AL1}
AL2	{CH6,AL2}

blankets are different. This implies that all the single failures can be distinguished according to the theory of Chapter 3. The exception to this rule is the undistinguishable single fault or double fault between the leaves and their parents. For example, a fault in $CH3$ and a double fault in $CH3$ and $CH2$ produces the same

set of apparent faults.

According to section 2.6, in addition to the structure of a tree t, the model also requires the prior probability of the root $CH4$ and the conditional probabilities of the other nodes given their parents. However, the propagation techniques described in Chapter 2 apply only to discrete valued variables, but the temperature signals are continuous values. Hence, a discretization process is required. This discretization is achieved by simply dividing the range of values of a sensor into a fixed number of intervals:

$$\frac{maximum\ value - minimum\ value}{number\ of\ intervals} \tag{5.1}$$

More sophisticated approaches are available [Dougherty et al. 1995] according to the precision required and depending on the computational cost that is acceptable. Additionally, some work is being done on the utilization of continuous variables as nodes in Bayesian networks when the application strongly requires it [Driver & Morrell 1995]. For simplicity, the experiments are carried out with 10 intervals for the discretization. This number produces a matrix of conditional probabilities of 100 parameters per dependency, so a discretization with more intervals would result in an exponential increase in the space and computing power required.

As a more detailed note, it might be worth mentioning that in order to facilitate the operations of the discretized values, the minimum, maximum values, and the number of intervals were adjusted so that the division in equation 5.1 results in an integer. For example, the extreme values for sensor $CH4$ are 73.04 and 124.47, but they are adjusted to 71 and 126 in order to have 11 intervals of 5 degrees wide (instead of 10 intervals of 5.14).

5.2 Test Environment

In order to evaluate the sensor validation model, a test environment was designed and implemented. The environment was written in C++ and compiled in Microsoft Visual C++, version 2.0. It utilized the QuickWin application platform under Windows 3.11. The hardware consists of a Pentium 120MHz PC with 16Mbytes of RAM memory. Figure 5.3 gives a block diagram of the implemented test environment. The initial input (at the left) is the data set obtained from the power plant. With this data set, the automatic learning program provides the structure of the network in a *file.str*. Specifically, that program executed steps 1 to 5 of the learning algorithm described in section 2.6. The *file.str* provides the nodes and the arcs that form the Bayesian tree. This file also includes the extreme values of each variable, and the number of intervals for discretization. The

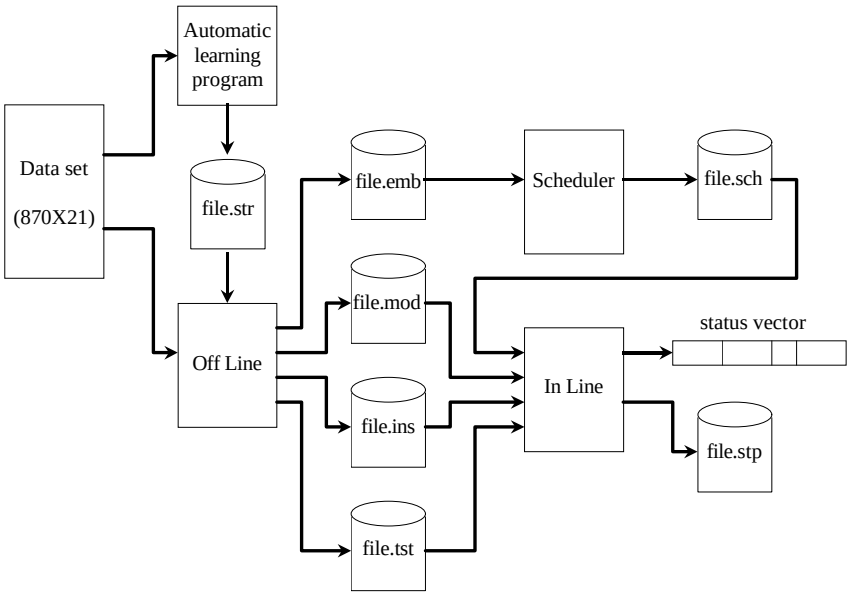

Figure 5.3: Schematic diagram of the test environment.

main modules of the test environment are the *off line* and the *in line* modules. The *off line* module has three main functions.

1. It separates the original data set into two random subsets: the training data set ($file.tst$) and the instantiation data set ($file.ins$). The training data set is utilized for calculating the prior and conditional probabilities of the initial probabilistic model. The instantiation data set is utilized to simulate the process of the gas turbine.

2. It generates a table with the extended Markov blankets of all the sensors (file.emb as illustrated in Table 5.1).

3. It generates the complete probabilistic model in $file.mod$. This is a file with the structure of the tree and the prior and conditional probabilities, i.e., P_P^t in the notation of section 2.6.

The *scheduler* module implements the pre compilation procedure described in Fig. 4.15 or 4.16. Based on the EMB table, and depending on the memory available in the system, this module designs a complete decision tree as in Fig. 4.6, or a reduced decision tree as in Fig. 4.7. Since the model consists of 21 sensors, the reduced decision tree was used for the experiments below. The *in line* module is an implementation of the any time sensor validation algorithm described in the continuous loop of Fig. 4.14. Specifically, it performs the following two functions:

1. the validation step as presented in Fig. 4.17, and

2. the isolation step as presented in Fig. 4.18.

As indicated in Chapter 4, the isolation module continuously updates the vector of probability of failure of all the sensors. This vector is shown to the external user as the current status of the sensors. Future extensions to this prototype would include a graphical interface which indicates the status of the sensors with different colours depending on the states, e.g., normal, warning, or faulty. Finally, the $file.stp$ is provided on request for debugging purposes, including partial results of the validation process.

5.3 Testing the Validation Model

This section presents the empirical evaluation of the algorithm presented in Chapter 3 by applying it to the application domain described above and by using the test environment just explained. First, section 5.3.1 classifies the kind of errors that can occur when using the system. Then, it describes the experimental method utilized in the evaluation. Second, section 5.3.2 presents the results of evaluating the accuracy of the model and the effect of the criteria for detecting faulty sensors. This is done by evaluating the probabilistic phase of the system, i.e., the accuracy of the system for detecting faults. Then, section 5.3.3 presents the results of evaluating the two phases of the model: the detection and isolation of real faults.

5.3.1 Experimental method

The test environment receives as its input, a data set from the process. Then, the *off line* module partitions the data set in two subsets: one partition for training the 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.

Theoretically, the system should always detect and isolate single faults correctly. However, as mentioned earlier, 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 [Cohen 1995]:

type I: rejection of the *null hypothesis* when it is true, and

type II: acceptance of the *null hypothesis* when it is actually false.

The *null hypothesis* used (defined in Chapter 3) refers to the hypothesis that a sensor is working properly. Thus, in other words, type I errors occur when a correct sensor is reported as faulty while type II errors occur when faulty sensors are not detected. Table 5.2 presents the four possible cases.

Table 5.2: Different cases of the status of the hypothesis and decision taken.

Choice	hypothesis true	hypothesis false
acceptance	correct	type II error
rejection	type I error	correct

As described in the introduction of this chapter, the criteria for deciding if a reading is faulty or not can result in a trade off between these two types of errors. At the end of Chapter 3, the following two criteria were mentioned:

1. 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.

2. Assume that the sensor is working properly and establish a confidence level at which this hypothesis can be rejected, in which case it can be considered faulty. This confidence level is known as the *p value*.

The accuracy of the model, i.e., the proportion of type I and II errors, is evaluated by varying the possible thresholds for each of these criteria.

A testing session includes the following steps:

1. Obtain a random partition of the data set.

2. Run the *off line* module.

3. Run the *scheduler* module.

4. Run the *in line* module utilizing the instantiation data set. This test corresponds to a simulation with no errors.

5. Modify the instantiation data set to insert a single failure in one sensor, in every one of the 260 lines of the file.

6. Run the *in line* module again.

7. Compare the results obtained with the expected results, and generate a results table.

Step 5 introduces the simulated failures and requires further explanation. A single line of the testing file includes the readings of all the sensors considered. In every line, one sensor is modified in order to represent an erroneous reading. The first line modifies the first sensor. The second modifies the second, and so on, until all the sensors have been modified. This operation is repeated, until all the lines, starting with the first sensor, have been edited. Two different faults were simulated:

Severe. The value modified is the most distant extreme value, i.e., if (maximum value - real value) is greater than (real value - minimum value) then the real value is substituted by the maximum value, and by the minimum otherwise.

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

This test procedure was used to evaluate:

- the accuracy of the validation phase, and

- the accuracy of the isolation phase.

The validation phase is an intermediate phase of the model that determines if a sensor is potentially faulty. It is therefore important to test its accuracy as well as the accuracy of the isolation phase. The following two subsections present the results of these two evaluations.

5.3.2 Accuracy of the probabilistic validation phase

To evaluate the accuracy of the validation phase, two experiments were carried out. First, a process with no faulty sensors was simulated and the number of faulty sensors incorrectly found (i.e., type I errors) was determined. Second, single failures were simulated as described in the test procedure in section 5.3.1, and the number of type I and II errors determined.

First experiment: no faulty sensors

The first of these experiments was carried out as follows. A single run consists of:

$$260 \text{ instances} \times 21 \text{ sensors} = 5{,}460 \text{ single validations.}$$

This was repeated 25 times, with random partitions of the data set with 70 % for training and 30 % for testing. Table 5.3 presents the average of the 25 runs for each criterion utilizing the testing data set without failures. For example, the

Table 5.3: Results of the experiments without simulating failures: average number of type I errors and the percentage that they represent.

Criteria	2σ	2.5σ	3σ	$p = 0.05$	$p = 0.01$
TYPE I	51.2 ± 12.4	13.4 ± 8.1	7.7 ± 6.0	64.4 ± 11.2	21.1 ± 9.2
	0.93 %	0.24 %	0.14 %	1.17 %	0.38 %

first column reports that when the $\mu \pm 2\sigma$ criterion is used, 51.2 out of the 5,430 validations were incorrect on average. For instance, the $\mu \pm 2\sigma$ criterion states that the model has a type I of 0.93 %, when all the sensors are working properly.

Second experiment: when failures are simulated

The second experiment was carried out as follows. A single fault was inserted in each one of the 260 instances as described in section 5.3.1. One set of experiments

was made with severe faults, and then repeated with a mild faults. For each single failure, and according to the theory previously presented in Chapter 3, several apparent failures are detected depending on the size of the extended Markov blanket. The sum of all the sizes of all the sensors's EMB is 61, i.e., the number of apparent faults that would ideally be detected in a validation cycle of all the 21 sensors. If 260 failures were simulated, then every one of the 21 sensors was simulated as faulty at least 12 times (12.6 times). In total, in every run of the system, a total of 769 apparent faults must be detected. Table 5.4 presents the average of the results for 25 runs with a 70/30 training/testing split for both mild and severe faults. The first column indicates that, from the 769 errors that the

Table 5.4: Results of the experiments simulating a single failure: average number of type I and type II errors and the percentages that they represent.

Criteria	2σ	2.5σ	3σ	$p = 0.05$	$p = 0.01$
Severe fault					
TYPE I	45.4 ± 11.1 5.9 %	11.6 ± 7.1 1.51 %	6.4 ± 4.8 0.83 %	50.8 ± 9.7 6.61 %	15.5 ± 7.3 2.0 %
TYPE II	8.1 ± 2.2 1.06 %	10.1 ± 2.3 1.32 %	11.7 ± 3.1 1.52 %	5.1 ± 2.1 0.66 %	5.2 ± 2.3 0.68 %
Mild fault					
TYPE I	45.4 ± 11.1 5.9 %	11.6 ± 7.1 1.51 %	6.4 ± 4.8 0.83 %	50.8 ± 9.7 6.61 %	15.5 ± 7.3 2.0 %
TYPE II	45.6 ± 6.2 5.92 %	57.6 ± 5.5 7.49 %	77.6 ± 5.7 10.1 %	22.5 ± 4.0 2.9 %	19.8 ± 4.2 2.57 %

experiments should find, 45.4 false errors (type I) and 8.1 undetected errors (type II) resulted, on average for severe errors. Also, 45.4 type I and 45.6 type II errors were found for mild faults. Figure 5.4 presents this information of severe faults of Table 5.4 graphically.

When examining the results for type I errors, both Tables 5.3 and 5.4 confirm that as the threshold for accepting normal behaviour increases, the number of type I error reduces. This is not too surprising. However, more significantly,

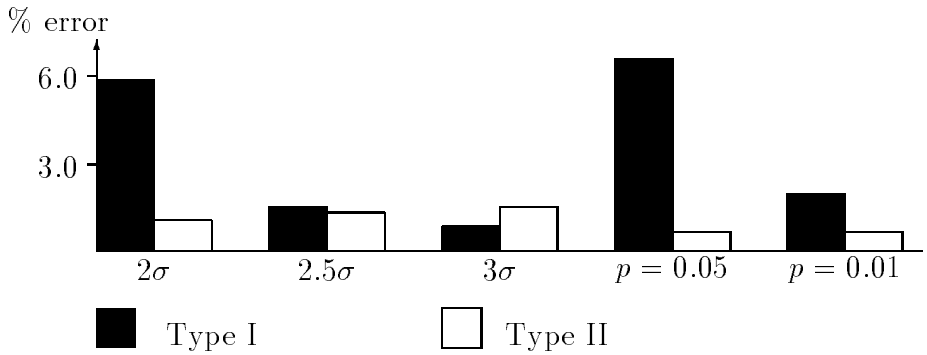

Figure 5.4: Graphical comparison of the results of the different criteria for severe faults.

increasing the threshold for normal behaviour does not result in a significant increment in the number of type II errors. For example, notice that when moving from $p = 0.05$ to the $p = 0.01$, the incidence of type I errors decreases significantly but type II errors remain very similar. The same situation holds for the 2σ to the 3σ criteria. Thus, choosing a criterion with a better type I error performance, does not sacrifice type II performance in these experiments.

Table 5.5 presents a global error rate for all the criteria analyzed, and for both intensities of errors. These numbers are obtained by taking the average of the two types and two intensities errors.

Table 5.5: Global performance measure of the first phase of the prototype

2σ	2.5σ	3σ	$p = 0.05$	$p = 0.01$
4.7 %	2.9 %	3.3 %	4.2 %	1.8 %

In general, the trade off between type I and II errors can be considered application dependent, i.e., some applications will want to avoid type I errors at all costs, whereas others may want to detect all real faults even if they obtain many false alarms. For example, for equipment in an intensive care unit of a hospital, it is important to warn the nurses about failures in the sensors even if there are

some false alarms. In contrast, if a chemical process can not proceed in the presence of a failure, the validation needs to be very strict in the alarms that it issues since frequent shut downs would be expensive. Additionally, an application may also consider differently the kind of faults that are important, i.e., severe or mild. For example, for soft and slow failures, the system may require stricter criteria in order to detect small deviations from the *normal* trend of a signal.

5.3.3 Accuracy of the fault isolation phase

The results presented in this subsection correspond to the evaluation of the validation and isolation phases. That is, the results correspond to the output produced by the probabilistic causal model described in Chapter 4. The propagation of probabilities in this model produces a probability close to one for those sensors whose extended Markov blanket is contained in the potential faulty list. Also, it produces probabilities close to zero for those sensors whose EMB's are not in the potentially faulty list.

Since the results obtained in this phase are probabilities, a mapping of the range of values between 0 and 1, to {correct,faulty} is required. The criterion for this mapping is presented in Fig. 5.5. The parameter $Hmin$ represents the

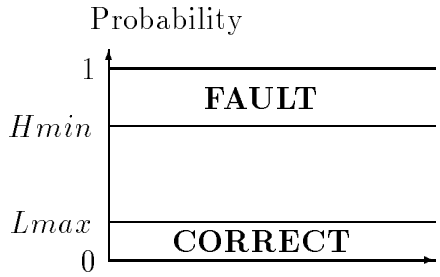

Figure 5.5: Final criteria to declare correct and faulty sensors.

minimum value for a high probability to be considered as a fault, while $Lmax$

represents the maximum value for a low probability to be considered as correct. These parameters were necessary for obtaining the numbers presented in Table 5.6. That is, type I errors were those whose value of probability was greater that $Hmin$, while type II errors were the ones with lower values than $Lmax$. The values in the zone in between do not produce errors. These values can be application dependent and can be fixed through empirical evaluation. In this thesis, $Hmin$ was fixed at 0.75 while $Lmax$ was fixed at 0.25. Notice that, in practice, only the value corresponding to $Hmin$ is required to consider a fault detected and isolated.

Table 5.6 presents the final evaluation of the prototype with the percentage of type I and II errors for severe errors. Lack of time has meant the results for mild errors are left for future experiments. However, given the results in Table 5.4, the errors for mild faults can be expected to have a similar pattern.

Table 5.6: Final evaluation: number of errors and their percentage for severe faults.

	2σ	2.5σ	3σ	$p = 0.05$	$p = 0.01$
type I	45.0 ± 11 17.3 %	12.5 ± 7.1 4.8 %	7.7 ± 5.0 2.9 %	37.8 ± 8.2 14.5 %	7.7 ± 5.6 2.9 %
type II	1.3 ± 1.2 0.5 %	2.5 ± 1.8 0.1 %	2.4 ± 1.8 0.9 %	1.9 ± 1.4 0.7 %	1.2 ± 1.1 0.4 %

Type I errors imply that most of the sensors in a EMB present apparent type I errors. This is more common as it can be seen in Table 5.6. 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. On the contrary, 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 5.6 confirms. The percentages are obtained comparing the average number of errors, with the total number of experiments.

In this case, the total number is 260, i.e., the number of different cycles where a single fault was simulated.

Before finishing this section, it is worth noting that the experiments were carried out by simulating single failures only. The theory presented in Chapter 3 shows that multiple failures can be distinguished, except in the cases explicitly mentioned in the theory. These cases were not considered in the results presented in Table 5.6 (they were not considered type I errors). Hence, although lack of time has prevented it here, an empirical evaluation of the behaviour of the sensor validation model with respect to multiple failures could be carried out in the future. The next section presents the results of the experiments when time is considered, i.e., the performance profile of the any time validation algorithm.

5.4 Any Time Validation

Chapter 4 developed an any time sensor validation algorithm that utilizes an entropy function as a criterion for selecting the next sensor to validate. This entropy function calculates the amount of information that any single validation provides for diagnosing all the sensors. Hence, to evaluate this criterion, this section compares the performance profile of the any time sensor validation algorithm as a function of time when the entropy based measure is used, and when a random selection scheme is used.

Figures 5.6 and 5.7 show the quality response obtained as a function of the number of validation steps for two sensors, $CH2$ and $CH6$. The *entropy* graph represents the resultant quality with the entropy based scheme. The *random* graph represents the experiment with a random selection scheme.

As these figures show the profile, in individual cases will, of course, vary. In Fig. 5.6, the entropy based selection scheme takes around 3 steps before its quality exceeds 80 % while the random criterion takes 8 steps. In Fig. 5.7, the quality

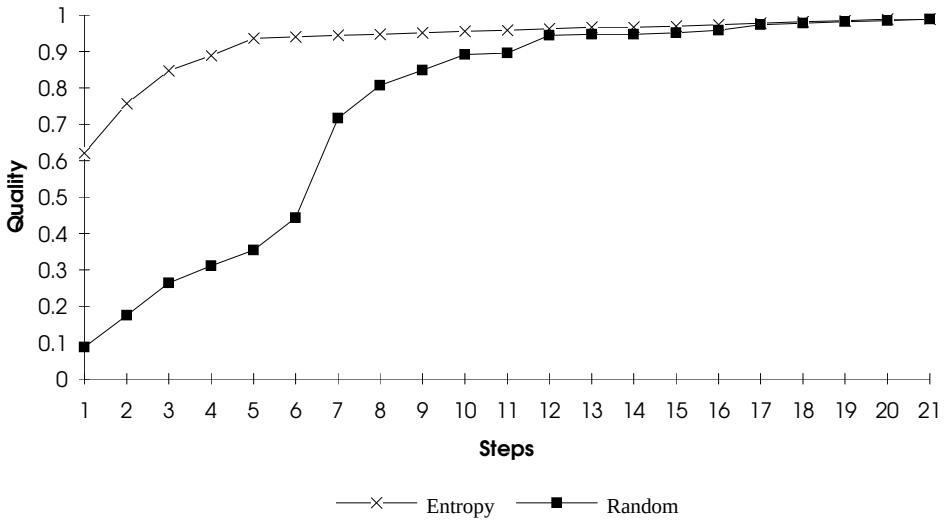

Figure 5.6: Quality response as a function of steps for the sensor $CH2$.

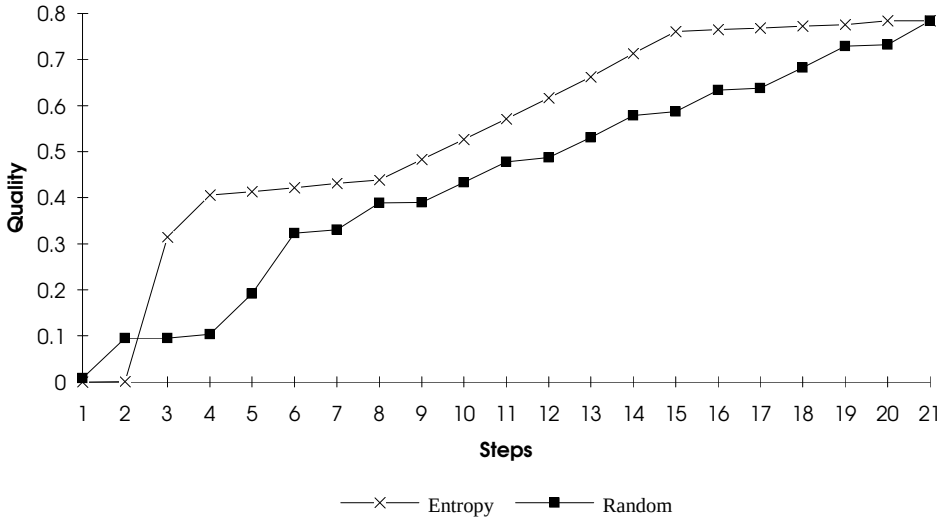

Figure 5.7: Quality response as a function of steps for the sensor $CH6$.

of the entropy based scheme takes 15 steps before its quality exceeds 80 % while the random scheme takes almost the 21 steps.

In these cases, this behaviour occurs since $CH2$ has only one other sensor that influences its diagnosis, and this is selected quite early. In the case of $CH6$, both criteria take longer to select all the sensors that influence the diagnosis of $CH6$.

Figure 5.8 shows the resultant performance profile of the any time sensor validation algorithm. That is, the quality of the response as a function of time.

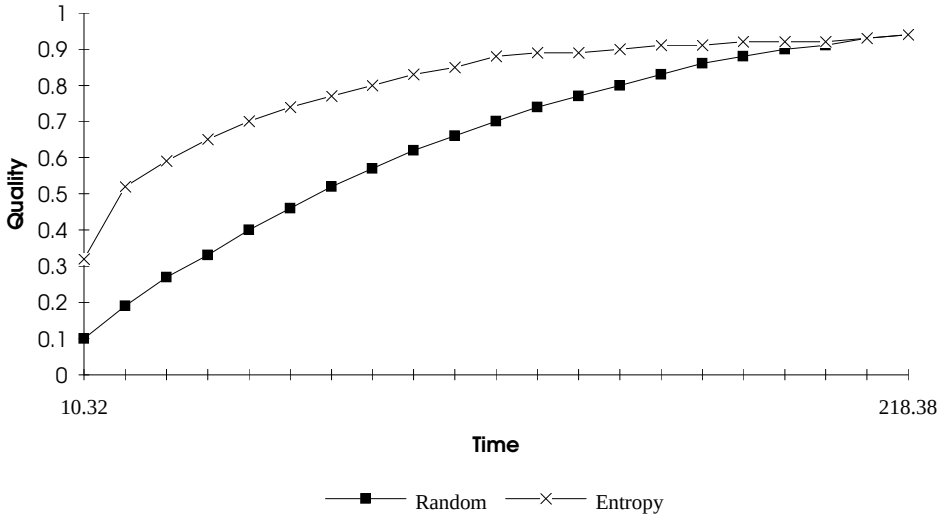

Figure 5.8: Performance profile of the any time sensor validation algorithm ($time \times 10^{-2} sec.$).

The experiments were repeated for the 260 instances of the testing file. Thus, every one of the 21 sensors was simulated as faulty at least 12 times (12.6 times). The *entropy* graph represents the average of the resultant quality with the entropy based scheme for the 21 sensors. The *random* graph represents the average of the same experiment with a random selection scheme. The time measurements consider delays inserted in the execution, so a comparison between the two graphs can be done[1]. That is, the time axis is a qualitative comparison rather than

[1]This is due to the lowest resolution of time measurements in MsDos of 10^{-2} sec.

quantitative.

Alternatively, the results can also be evaluated by comparing the time required to reach different levels of quality. For example in Fig. 5.8, when the random criterion reaches 60 % of quality, the entropy criterion has already reached more than 80 %.

5.5 Summary

This chapter presented the empirical evaluation of the theory and algorithm developed in this thesis. This evaluation was carried out on a problem whose dependency model was not necessarily perfect. First, the origin of the data was described, and how this data was utilized to obtain the probabilistic model. Notice that a tree was utilized here for simplicity. That is, in this application, the results show that the precision that a tree provides is good enough. Other applications may need more complex models. The testing environment was also described in section 5.2. Experiments were carried out to evaluate: (i) the validation phase alone, (ii) the validation and isolation phases, and (iii) the performance of the any time algorithm.

The results for the accuracy of the model were reported in terms of the type I and type II errors and with respect to detecting severe and mild faults. The results showed, that for this particular test application, more stringent criteria for detecting failures reduced type I errors but did not significantly increase type II errors.

The results of the evaluation of the validation and isolation phases together are shown in Table 5.6. Again, with a *p value* of 0.01, there are 2.9 % of type I errors, and 0.4 % type II errors. Notice that, in general, the sensor validation algorithm performs almost perfectly with respect to undetected faulty sensors, i.e., all the faults are detected. At the same time, the rate of incorrect detection

faults is satisfactory for most of the criteria analyzed.

This chapter also evaluates the any time behaviour of the algorithm presented in Chapter 4. That was done by carrying out experiments to obtain the performance profile of the entropy based selection scheme and comparing it with a random selection scheme.

Of course, the profiles will differ in other applications and the extent to which it affects an application will depend on the particular requirements.

Chapter 6

RELATED WORK

The thesis has developed a sensor validation algorithm that was evaluated, and the results presented in the previous chapter. This chapter places this work in the context of other related research. First, in section 6.1, the traditional approaches of physical and analytical redundancy for dealing with the sensor validation problem are reviewed. Their limitations are outlined and provide some motivation for the use of artificial intelligence techniques. Second, section 6.2 describes some approaches to sensor validation that use AI techniques in different application domains. Next, section 6.3 comments on related projects that have a wider objective. These are projects for intelligent monitoring and diagnosis in gas turbines. Finally, section 6.4 directs the reader to related research on any time algorithms and Bayesian networks.

6.1 Traditional Approaches for Sensor Validation

The validation of sensors has been a concern ever since automatic control has been implemented in plants. Since then, several approaches have been proposed

including [Willsky 1976], [Fox et al. 1983], [Bacchus et al. 1995], and [Brooks & Iyengar 1996]. The survey papers by [Basseville 1988], [Yung & Clarke 1989], and [Frank 1990] give an overview of the main approaches and provide further references.

The most traditional method for sensor validation has been to use *physical redundancy*, i.e., the inclusion of several instruments that measure the same parameter of the process. Thus, the validation comes from simple majority voting logic. However, this approach is prohibitive for many chemical plants where, for example, adding more sensors might weaken the walls of the pressure vessels. In addition, further techniques for fusing of information between the sensors is required.

Another technique, used in the last two decades, is based on the use of *analytical* rather than physical redundancy. This technique is inspired by the inherent redundancy contained in the static and dynamic relationships among the system inputs and measured outputs. Specifically, the analytical redundancy approach is based on the fact that the existing redundancy leads to relationships that can simply be evaluated by information processing under fault free conditions, in the control room. The idea is that faults of a dynamic system are reflected in the physical parameters as, for example, friction, mass, viscosity, etc. These parameters are modelled with differential equations, and state estimation techniques used to predict the physical parameters. The deviations from the nominal values are computed to obtain the residuals, and the faults detected if the residuals are not zero. Of course, more than one relationship may be violated in which case a model to diagnose the faults is required.

The analytical redundancy technique has been widely used for fault diagnosis in complete dynamic systems. However, there are several problems in using this approach including [Frank 1990]:

- 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.

These problems have encouraged the development of alternative approaches to sensor validation. In particular, several researchers have applied AI techniques in an attempt to solve the sensor validation problem.

6.2 Knowledge Based Approaches for Sensor Validation

This section reviews some of the approaches for sensor validation that have utilized knowledge based techniques. There are several studies that have aimed to adopt knowledge based techniques for sensor validation (e.g. [Dean & Wellman 1991], [Bacchus et al. 1995], [Brooks & Iyengar 1996]). The systems described in this section were selected to be representative of a wide range of approaches and more recent systems were preferred to other systems. The systems selected are:

- Sensor validation in space rockets [Bickmore 1993]. This was selected because it utilizes a combination of traditional and knowledge based techniques to perform real time sensor validation.

- Sensor validation using neural networks [Khadem et al. 1992]. This was selected since it uses the different technology of neural networks.

- Self validation of sensors (SEVA) [Henry 1995]. This was selected primarily because it takes a different approach from this thesis and the above approaches in that it proposes self diagnosing sensors. Its inclusion therefore provides greater contrast and wider context.

In order to make it easier to contrast and compare the different approaches, the review is structured with respect to the following characteristics:

- model utilized,
- detection of faults,
- fault isolation,
- response time, and
- application domain.

Sensor validation in space rockets

This project was developed by the NASA Lewis Research Center for detecting sensor failures on liquid rocket engines [Bickmore 1993]. The approach has been tested in the laboratory and will be installed in the main engine of the Space Shuttle.

Model utilized. Previous studies carried out by Bickmore (1993) indicated that no single algorithmic method is enough for the validation of sensors. He concluded that several methods should be used and the results integrated or *fused* into a final conclusion. For this reason, the project utilized two different models: analytical redundancy and Bayesian networks. Figure 6.1 shows a linear equation of the simplest form of the empirical relations used in this project.

The relations are integrated in a Bayesian network as shown in Fig. 6.2. There is one node for every sensor and the other nodes represent relations that hold

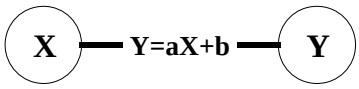

Figure 6.1: Example of a simple empirical relation.

between pairs of sensors.

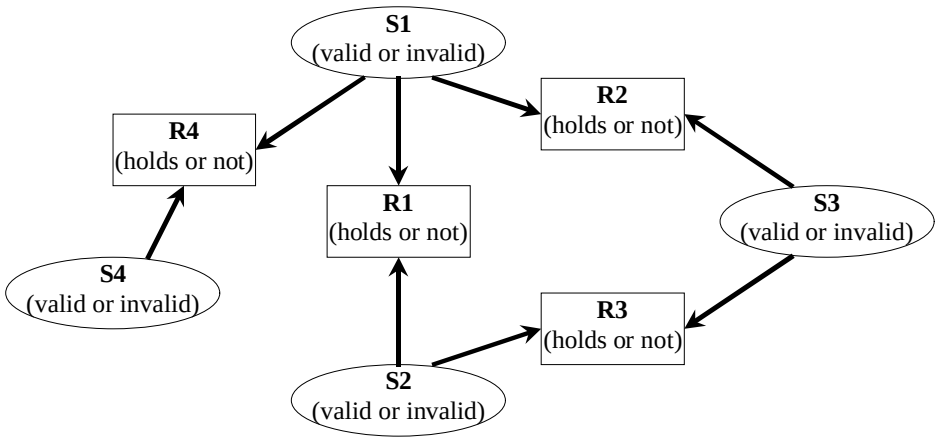

Figure 6.2: Bayesian network including the analytical redundancy relations.

All the nodes are binary, i.e., they can have only two values: node sensors can be valid or invalid, while the relation nodes hold or do not hold. For example, a failure in sensor $S1$ would influence the expected probability distribution of the status of relation $R1$.

Fault detection. The basic method for detecting faults is the estimation of a value using the relations of the analytical model. The difference between a value predicted and the directly sensed value is called a *residual*. A validation cycle is carried out by sampling the values of all the sensors and determining if each of the relations hold or not by thresholding the residuals (for example by three standard deviations).

Fault isolation. Once the residuals of all relations have been calculated, the Bayesian network is instantiated. Then, a probability propagation algorithm is utilized to evaluate the probability of each sensor being correct based on the

status of the relations in the network. For example, from Fig 6.2,

$$P(S1 \mid R1, R2, R4) = \frac{P(S1, R1, R2, R4)}{P(R1, R2, R4)}.$$

The prior probability of the root nodes (the sensor's status) is defined using the mean time between failures (MTBF) parameter given by the sensor manufacturer.

Response time. The sensor validation process in this application is in real time, i.e., the status of all the sensors is given faster than the sampling ratio. This is achieved through a pre compilation of the probabilities that participate in the propagation. For example, the probability of a binary relation holding given that one of the sensors has failed is given by the formula

$$\frac{2 \times 3 \times \ standard\ deviation}{range\ of\ the\ sensor}.$$

Application domain. As mentioned before, the purpose of this project is the validation of sensors on the space shuttle's main engine. The goal is to prevent the controller, or safety system from making critical decisions, such as the decision to shut an engine down, on the basis of data from faulty sensors.

Discussion. The approach presented in the report by Bickmore (1993) has some characteristics in common with the one developed in this thesis. Both utilize Bayesian networks and both are based on relationships between sensors. However, the nature of the relationships is quite different. In Bickmore's approach, the relationships are viewed as invariants that must hold between two sensors if the sensors are working properly. In the approach in this thesis, the relationships between sensors are defined by dependency relationships which need not be binary. The way these relationships are obtained is also quite different. The NASA approach uses experts' knowledge and empirical experimentation for obtaining the binary relationships, while this thesis utilizes a learning algorithm that provides the dependency relations between all the sensors in the process.

Sensor validation using neural networks

The nuclear industry has been very active in the development of techniques for sensor validation [Upadhyaya & Eryurek 1992]. The work by Khadem et al. (1992) utilized neural networks and was installed in a nuclear power plant in the U.S.A.

Model utilized. This project utilized artificial neural networks to detect faults in two sensors' readings. The network consists of seven input nodes, one hidden layer with four nodes, and two output nodes. The output of the network represents the sensors that are being validated and the inputs correspond to related variables. In order to decide which variables are the most related, this project utilizes differential equations that describe the dynamics of the system. With these equations, a linear approximation of the system was obtained to establish the degree to which a set of variables are related to the target sensors. A back propagation algorithm is used to learn the relationships between the input variables and the sensor readings represented in the output layer.

Fault detection. The network is trained utilizing valid data from the process. After the training process, new information is entered to the network inputs and propagated to the output nodes to estimate the expected values of the sensors. The real and the estimated values are compared and, if the difference is greater than a specific threshold, a fault is identified. The paper mentioned error rates below 2 % in both outputs.

Fault isolation. The technique proposed in this work was applied to just two sensors of a subsystem of a nuclear power plant, so fault isolation was not considered. Additionally, the related sensors used as inputs are implicitly considered as always correct.

Response time. The use of neural networks require a training period but this is done off line. The application of this technique in line is almost immediate,

with the advantage that it can be processed in parallel for more speed.

Application domain. This project was applied to the feed water flow meters in two feed water flow loops of a nuclear power plant. The training data was obtained from a plant simulator in the steady state of the plant at three different power levels.

Discussion. Artificial neural networks have proved to be an appropriate mechanism for the validation of a small set of sensors. The use of neural networks offers an interesting way of predicting the expected value of a sensor that appears to work in this application. However, when multiple sensors are involved, there appears no obvious method for isolating faults. Further, the assumption that the input nodes represent correct sensors is at odds with the overall aim of sensor validation. In contrast, in this thesis, the relationships between sensors have been used to isolate the faults. The use of neural networks for sensor validation is nevertheless worth considering if it is possible to develop additional neural networks to perform fault isolation.

Self validating sensors

The SEVA (SEnsor VAlidation) project is a collaboration between Oxford University, Foxboro GB Ltd and ICI, which began in 1989. It is a collaboration between academia, sensor manufacturers and sensor users [Henry & Clarke 1993]. This project addresses the sensor validation problem from the perspective of technological advances in two fields: the construction of the sensor itself, and the communication media between the sensor and the control room. Both fields utilize digital technology based on microprocessors.

Model utilized. This project takes the validation of a sensor to inside the sensor. Figure 6.3 [Henry 1995] presents their approach, namely, the construction of a sensor (for one or several measurements) that can communicate with the

control room through a digital communication channel called a *Fieldbus*.

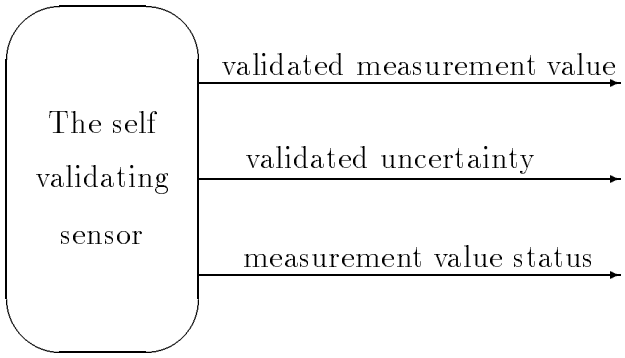

Figure 6.3: Parameters issued by the self validating sensor.

A sensor provides the following information via the fieldbus:

VU: validated uncertainty. This is a measure used to quantify the error when tracing the calibration of an individual instrument. It is a concept[1] applied in numerous international standard documents, e.g., [ANSI 1985].

VMV: validated measurement value. This signal is used to provide the *best estimate* of the measurement signal. Under normal conditions, it is the transducer measurement with some possible enhancements like expressed directly in engineering units. Under minor faults, a correction can be applied using the techniques available in the sensor. If a severe fault is detected, this signal provides an estimate based on historical data and the knowledge embedded in a sensor's hardware.

MVS: measurement value status. This signal indicates the detected status of the measurement. Even if the VMV provides a good estimate, the MVS reports that it is only an estimate, or it indicates that the signal is validated

[1]Do not confuse this engineering term with the AI concept of uncertainty.

and free of errors. It has four status values: no fault, mild fault, severe fault and severe temporal fault.

These three signals indicate the detection of a fault, and the severity of it, so the control system may take a corrective action.

Fault detection. Faults are detected with self diagnosis routines that are carried out within the sensor. These multi signal sensors utilize additional signals other than process measurements. For example, information provided by the manufacturer (e.g., physical and electric properties, spectral analysis of the signals, etc.) is included in the sensor and is used to monitor the health of the device. This knowledge can be used to estimate the value that a signal may have in the presence of a fault. In general, as Henry (1995) mentioned, the detection method in every sensor depends on the characteristics of the sensor and on the application that it is designed for. There can be analytical redundancies and/or knowledge based techniques involved. Notice that these techniques are used within the sensor, in the microprocessor, and utilize a small number of signals and parameters for the detection of faults between a small number of measurements.

Fault isolation. Since each sensor looks after itself, the problem of fault isolation does not occur.

Response time. Since the validation is carried out inside the sensors using microprocessors and digital electronic devices, response time is very quick.

Application domain. These types of sensors have a wide range of potential applications, specially in the chemical industry. However, they are still under development and have yet to be validated.

Discussion. This approach is very different from the other approaches. In concentrating on the self validation of sensors, information from other sensors is not utilized nor needed. The problem of fault isolation does not occur. The

problem of deciding if the sensor is faulty using its own knowledge is based on analytical redundancy. The main disadvantage of this approach is that the sensor itself becomes more complex, and presumably more expensive to manufacturer. Further, even if this approach is validated, traditional sensors will still be in operation for many years.

6.3 Intelligent Diagnosis

This section briefly describes two projects that apply artificial intelligence techniques for the diagnosis of gas turbines. Their focus is on the diagnosis of the performance of a turbine where the validation of sensors is not the main focus. They are described here since they can be considered as the higher layers described in Chapter 1 and therefore provide better context for the work carried out in this thesis.

Diagnosis of thermal performance

This project was developed for diagnosis of gas turbines for an auxiliary power unit of a commercial aircraft [Breese et al. 1992]. The aim of this project was to model the whole process by using a Bayesian network that includes sensor validation as well as the fault diagnosis process. A Bayesian network was used to represent the model of the whole process and contained small sections devoted to the validation of sensors. Thus, the kind of models first described in Chapter 1 and shown in Fig. 6.4 (Fig. 3.3) can be found as part of the whole model. The figure shows three cases. In (a), they utilise directly the value V_m as the unique source of information. In (b), they use a simplified model where V_s and S cause V_m. Finally, in Fig. 6.4(c) they use a sensor state S that must obtain its value ({correct, false}) from another source (human or computerised).

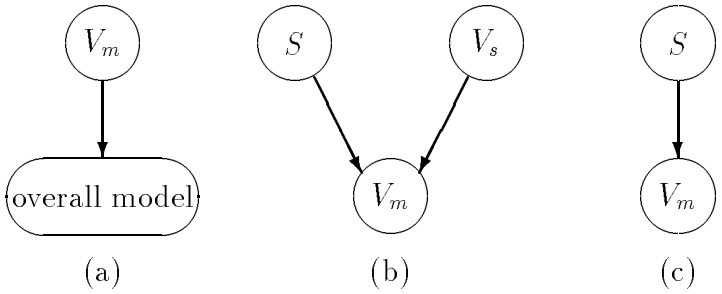

Figure 6.4: Three different uses of sensor related nodes in a overall process model.

Summarizing, this project requires the validation of only a subset of the sensors in the gas turbine. In other cases, they make an implicit assumption that the signals utilized in other aspects of the diagnosis model, are correct. For example, in the Fig. 6.4(a) the value of the variable is considered correct, while in (b) the status S of the sensor is supposed to be available. Thus, this approach can be viewed as one of the higher layers of the proposed model in Chapter 1, where the validation of all the sensors is not the main concern.

The TIGER project

TIGER is a knowledge based gas turbine condition monitoring system [Milne & Nicol 1996]. Its goal is to monitor the turbine on a regular basis in order to establish when maintenance actions need to be performed. It has been in use at Exxon Chemicals Ltd. in Scotland. TIGER consists of three systems that run independently and which are coordinated by a fault manager. These systems are:

- KHEOPS: This is a high speed rule based system utilized for limit checking and other related functions. For example, checking for different limits on different operating regimes of the turbine.

- IxTeT: This monitors the dynamic reaction of the gas turbine. It is basically a language to specify sequences of events that can encode causal relations

between the elements of the process. It is also an interpreter that can trace the operations of the turbine in order to detect deviations.

- CA-EN: This is a model based supervision system. It is based on two levels of knowledge representation mechanisms: (i) analytical, which represents differential equations, and (ii) causal graphs, which represent the flow of events caused by a perturbation.

The TIGER system covers the most common and expected faults in the following areas:

- fuel system and fuel valves,
- combustion problems,
- compressor and turbines,
- second stage nozzles,
- inlet guide vanes,
- steam injection and helper turbine.

Summarizing, the TIGER system is a complete monitoring system already installed in a plant. It utilizes several knowledge representation mechanisms as well as different reasoning methods (e.g., rule based, model based, analytical based). It assumes that the information provided by the sensors is accurate, unless an abnormal situation is found. That is, TIGER first looks for faults in the process and then, it may deduce that the fault is in a sensor.

6.4 Any Time Algorithms and Bayesian models

This thesis has developed a sensor validation algorithm utilizing Bayesian models and which has an any time behaviour. This section therefore gives an indication

of related work in these areas.

The work on any time algorithms was first introduced in the studies by Horvitz (1987) and Dean & Boddy (1988) in which they tackled the basic problem of returning an answer whose quality improves with time. More recently, Zilberstein [Zilberstein & Russell 1995, Zilberstein 1993, Zilberstein & Russell 1996] described an approach in which several sequential any time modules of a system are *compiled* and a composite quality measure is obtained. In all these studies, the measure of quality is not specified except to state that it can be defined in terms of the certainty, accuracy, and specificity of the answer. In this thesis, this measure had to be defined and a measure that combines both certainty and specificity was used. Such a combined measure may have wider applications to other any time algorithms. In particular, it may be easier to visualize and optimize the performance of an any time algorithm with this information theoretic composite measure than with the three separate measures.

Another important area of research on any time algorithms is work on intelligent planning [Boddy & Dean 1994]. They addressed the problem faced by complex intelligent systems in which the time spent in decision making affects the quality of the responses generated by the system. Their approach is called deliberation scheduling in which the intelligent system is capable of taking its own computational resources into consideration during planning and problem solving.

Also, research has been conducted in time dependent utility, as in time critical contexts, where the utility of the system's outcomes diminish significantly with delays in taking appropriate action [Horvitz & Rutledge 1991], [D'Ambrosio 1992], [Horvitz & Barry 1995].

Given a sensor validation algorithm like the one developed in this thesis, the above work can be utilized to develop the higher layers of a process model.

Any time algorithms have also been used in probabilistic reasoning. The idea

is to obtain any time behaviour in the computation for propagating probabilities. One approach is a modification of the original Bayesian network to a form in which propagation can be performed faster. For example, Wellman & Liu (1994) proposed a state space abstraction in which the states of selected nodes are merged. Another example of this network modification is proposed by Jitnah & Nicholson (1997). They eliminate some variables that have less influence on the corresponding query node. So, in these approaches, fast results are obtained and, if more time is available, further modifications closer to the original network are conducted to incrementally increase the precision of the answers.

Other approaches for any time probabilistic reasoning consist of bounding the probability values within certain intervals. This is achieved by taking a subset of the information needed to specify the complete Bayesian network [Ramoni 1995],[D'Ambrosio 1993]. Practical experiments comparing several approaches is reported in the paper by D'Ambrosio & Burgess (1996).

Chapter 7

CONCLUSIONS AND FUTURE WORK

The primary objective of this thesis was to develop a theory and an algorithm for sensor validation that could be used as part of a layered model of a real time process.

This chapter summarizes the main contributions of this thesis and future work. First, section 7.1 describes the key aspects of the developed theory and sensor validation algorithm. Then, it summarizes the results of evaluating the algorithm, and discusses the relationship with other work. Finally, section 7.2 presents the future work.

7.1 Conclusions

Sensor validation is an important problem whose solution would make a significant contribution to the use of real time systems where information is obtained via sensors. Since there is uncertainty in the readings obtained from sensors, this thesis has proposed the use of Bayesian networks as a basis for sensor validation.

A Bayesian network is used to model the relationships between the variables in

a process. Probabilistic propagation is used to obtain the posterior distribution of a variable given its related variables. If the observed value differs from the expected behaviour, the sensor is considered to be potentially faulty. The isolation of real faults from potential ones is based on a property of a Markov blanket. The property states that if a sensor is faulty, it will produce potential faults in all the sensors in its Markov blanket. The theory developed shows how this property can be used to isolate single failures from the set of sensors identified as potentially faulty. The theory also shows that multiple failures can be distinguished provided that the set of potentially faulty sensors can be obtained by forming a unique combination of the sensors' extended Markov blankets. When this is not possible, higher layers of a process model are provided with a list of sensors whose extended Markov blankets are a subset of the potentially faulty sensors. This list will normally be smaller than the set of potentially faulty sensors and could be used by the higher layers to perform diagnosis.

The developed theory results in a sensor validation algorithm that operates in batch mode but which is inappropriate for use in real time processes. Hence, this thesis developed an any time version of the sensor validation algorithm. The first problem faced was to convert the batch mode operation, where all the sensors were validated before the results can be presented, to one where the results can be presented at any time. This was done by adopting a cyclic process whose output is the probability of failure of each sensor. Within each cycle, a sensor is selected, validated, and its effect on the probability of failure of all the sensors is revised. To achieve this, the following problems had to be addressed:

- How can the effect of a potentially faulty sensor on the other sensors be calculated?

- How can the next sensor be selected so as to improve the performance of the any time algorithm?

The effect of detecting potentially faulty sensors was modelled by adopting a Bayesian network with two layers in which one layer consisted of nodes representing real faults and a second layer which consisted of nodes representing potential faults. The dependencies between the real and potential faults, namely that the real faults result in potential faults, were obtained from the extended Markov blankets of the sensors. Then, when a sensor is detected as faulty, probabilistic propagation can be used to update the probability of failure of all the sensors.

The problem of selecting the next sensor was addressed by noticing that Shannon's entropy measure could be used to represent the information provided by the state of the probabilities of failure. The amount of information that would be gained by validating each sensor can be calculated and the sensor which gives the most information is selected as the next sensor. The amount of information also provides a suitable measure for the quality of the answer given by the sensor validation algorithm.

The theory and the algorithm were evaluated by applying it to the validation of temperature sensors of a gas turbine at the Gómez Palacio power plant in México. The accuracy of the theory and the model was evaluated by carrying out experiments to determine the number of sensors incorrectly reported as faulty or working. First, a learning algorithm was used to obtain the dependencies between the variables. Then, in each experiment 70 % of the data was used for training the network and 30 % was used for testing by simulating single faults. Results were obtained for the number of incorrect classifications both for severe and mild sensor faults. In both cases, the threshold criteria for accepting normal behaviour was also varied. The main findings were as follows.

- The model works well for this application. The best results for the fault detection phase were obtained with a *p value* criterion of 0.01 in which 2 % of the correct sensors, and less than 1 % of the faulty sensors were

misdiagnosed. The poorest results were obtained when a 2σ criterion was used, in which case, almost 6 % of the correct sensors, and 1 % of the fault sensors were misdiagnosed.

- As expected, when the type of faults become less severe, it is harder to detect faulty sensors. However, the results obtained for mild faults are still reasonably good. Thus, with the *p value* criterion of 0.01, less than 3 % of the faulty sensors were not detected.

- As the criteria for accepting normal behaviour becomes less stringent, the number of working sensors incorrectly diagnosed as faulty reduces. This however, does not result in an equivalent increase in the number of faulty sensors misdiagnosed. For example, with the 2σ criterion, 5.9 % of the correct sensors, and 1.1 % of the faulty sensors were misdiagnosed. When this criterion is loosened to 3σ, 0.8 % of the correct sensors, and 1.5 % of the faulty sensors were misdiagnosed.

To evaluate the any time version of the algorithm, experiments were carried out to determine the performance profile of the algorithm. The experiments were run with the entropy based selection scheme and with a random selection scheme. The results of the experiments were presented as profiles in which the quality increases with time. The results show that on average, the entropy based selection scheme performs significantly (16 %) better than the random selection scheme.

The developed algorithm has been compared to several other approaches to sensor validation. The comparison was carried out with respect to the model utilized, the way faults are detected, the fault isolation scheme used, the response time and the domain of application. To achieve fault detection, most of the approaches rely on analytical model of the process. Such models are not easy to obtain and represent. In contrast, the approach developed in this thesis utilizes

existing work on learning Bayesian models to obtain the relationships between the variables in the process. The other approaches either ignore or have weak mechanisms for fault isolation. An exception is the work of Bickmore (1993) in which fault isolation is achieved by calculating the probability of a faulty sensor given the status of relationships that should hold if the sensors are working properly. However, unlike the approach developed in this thesis, the dependencies between the sensors and the relationships are binary. This means that the number of relationships that need to be manually identified increases exponentially. In terms of the response time, the reviewed systems all have good real time performance. For example, the system developed by Khadem et al. (1992) achieves good performance by utilizing neural networks in which propagation is quick once the network has been trained. In this thesis however, the use of Bayesian propagation has meant that pre compilation and any time techniques had to be utilized in order to achieve good performance. In terms of the domain of application of the reviewed systems, it is interesting to note that sensor validation is being applied to a wide range of domains including space rockets, nuclear power plants, and the chemical industry.

The thesis also mentions related work on any time algorithms and Bayesian networks. Within the field of any time algorithms, separate measures of quality, namely certainty, accuracy and specificity, have been proposed and used. In this thesis, an entropy based measure was utilized that combined both the certainty and specificity measures, making it easier to present performance profiles. This measure may have wider application to other any time algorithms. The way this thesis utilizes research on Bayesian networks is novel. Bayesian networks originated as a way of modelling diagnosis problems in which some of the information is known and propagation is used to calculate the probabilities of unknown events. In contrast, for the sensor validation model developed in this thesis, the values of

all the events are known, and the aim is to detect deviations from predicted values. This could have wider applications to problems where information needs to be validated, for example, to check for consistency or to identify false information.

To conclude, the main contributions of this thesis are the development of a theory of sensor validation using Bayesian networks, the development of a sensor validation algorithm, and the development of an any time sensor validation algorithm. The theory should provide a good basis for further work on sensor validation and the any time algorithm could be developed into a system that is of great practical value.

7.2 Future Work

There are a number of possible enhancements to the algorithm and the theory developed in this thesis that could lead to further research. This section summarizes some of these problems and gives preliminary directions for solving them.

Failures in the process

When the operator or the control system receives an alarm, sometimes a question remains: what is faulty, the sensor or the process? Since a failure in the process may result in abnormal readings from the sensors, the developed algorithm may report a faulty sensor. The use of a higher layer in the diagnostic system must consider this situation. By using an appropriate model of the whole process, this higher layer can take the alarm from the sensor validation layer as extra information in order to deduce a fault in the process.

Multiple indistinguishable failures

Chapter 3 discusses the problem of identifying multiple failures. In its present form, the theory only enables the identification of the real faults when a unique

combination of the sensor's Markov blankets results in the set of potentially faulty sensors. Extending the theory so that it could go further in isolating multiple faults would be interesting. However, the most likely solution is to use additional domain knowledge about the process. For example, in the validation of temperature sensors there can be other signals that may confirm the state of some temperature sensors. If a blade path temperature sensor reading is low together with a very high temperature in two of the beadings (i.e., undistinguishable double fault), then consulting the generation sensor and the speed of the turbine may give more information about the state of these sensors (parameters related to each of these temperature sensors).

One possible solution to the two problems stated above that is worth investigating, is to use probabilistic temporal networks [Nicholson & Brady 1994, Kanazawa 1991, Berzuini 1990, Dean & Kanazawa 1988, Hanks 1988].

Ignorance assumption

At the beginning of this thesis, in Fig. 1.3(c), a basic model of the sensor diagnosis was given. That is, the state S of a sensor can be inferred with the values of the measure and the estimated real value. In practice, in Fig. 4.8 shows the model for fault isolation where the roots of that model represent the state of the sensors. The prior probability of these nodes was assumed to be 0.5, representing ignorance about the chances of failure of the sensor. That assumption was taken as an approximation in order to keep the model simple. Some applications may require and support the inclusion of additional knowledge about the failure behaviour of a specific sensor. For example, the manufacturer of the sensors may provide some parameters like the *mean time between failures* (MTBF). Other important information that can be used include the location of the sensor in the plant, the importance of the signal in the control of the process, etc.

Changing models problem

The developed theory and algorithm assumes that the Bayesian network representing the process does not change. In practice, the model may change as the process moves into a new phase. For example, the probabilistic model obtained in Fig. 5.2 is valid only during the start up phase of the turbine. In this phase, the variables measured by the sensors maintain a relationship based on the dynamics of the process. However, once the generator reaches the required speed, and the generator is synchronised with the transmission network, a steady state phase continues. This steady state phase requires only to maintain the speed and to respond to the load changes. Also, the stopping phase of the plant requires a different model. Thus, different relations hold between the same set of variables during the phases of the process. If, as in this application, there are fixed points at which these transitions occur, then separate Bayesian networks could be developed and the approach developed in this thesis could be used. However, if this not the case, then a more elaborate solution needs to be found.

Appendix A

Partial Results

In order to provide a better understanding of the calculations proposed in Chapter 4, a sample of the results are included.

Assume the model presented in Fig 5.2. This appendix presents the results of a validation cycle when there is a simulated fault in sensor $CH6$. Each entry in the table consists of four parts. The first part indicates the sensor being validated and its posterior probability distribution. This part shows the probability assigned to all the intervals in which the variable has been discretized. For example:

```
Posteriors(CH4): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

describes the posterior distribution of sensor $CH4$ after the propagation. Notice that the propagation indicates that the real value read by the sensor is not possible in any interval. So there is definitely a fault. The second part shows the real value read by the sensor and the probability of its interval. For example:

```
P(85.42) = 0.00
```

indicates that sensor $CH4$ has a value of 85.42 which is not possible in this stage of the process, so a probability of 0.0 is obtained. The third part indicates the content of the 21 elements of the probability of failure vector P_f. For example:

```
Probability of failure vector:
[ 0] 0.50 [ 1] 0.50 [ 2] 0.50 [ 3] 0.50 [ 4] 0.50 [ 5] 0.50 [ 6] 0.50
[ 7] 0.50 [ 8] 0.50 [ 9] 0.50 [10] 0.50 [11] 0.50 [12] 0.50 [13] 0.50
[14] 0.50 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.50 [20] 0.50
```

describes the 21 values of the vector. Notice that, since the numbers are indicated with only two decimal digits, the probabilities shown seem to be unchanged from their initial value. Finally, the fourth part presents the normalized quality value obtained in the indicated validation step using equation 4.3. For example:

```
Quality step 0: 0.00
```

On the following page, the first entry indicates the validation of $CH4$. It indicates that the real value measured by the sensor, and the estimated value differ completely. In comparison, the validation of $CH1$ (10^{th} entry) produces a wider probability distribution.

Notice that the validation cycle reports apparent faults in the sensors $CH4$, $CH6$, $CH5$, $CA7$, $AL1$, and $AL2$. These corresponds to the $EMB(CH6)$.

The quality function starts with zero and increases after the third step. This function reaches 75 % after the step 15 when it reaches close to its final value. After this step, the validation of the remaining sensors provides no more information about the state of the system. Now, the vector P_f, is initialized to 0.50 in all its elements. Then, the validation of $CH4$ in the first step makes no significant change given the influence of $CH4$ on the system. Next, the validation of $CH6$ as faulty produces no significant changes in P_f. In the fourth step, after the validation of $CA4$ and AEF as correct, $P_f(CH4, CA4, AEF, CA2, CA3, CA5)$ goes close to zero, i.e., the system indicates that these sensors are correct. The final state of the P_f vector indicates certain failure in $CH6$, and some probability of failure in $CH5, CA7, AL1$ and $AL2$. That is, a failure in $CH6$ and in its four sons (see chapter 3).

```
Posteriors(CH4): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(85.42) = 0.00
Probability of failure vector:
[ 0] 0.50 [ 1] 0.50 [ 2] 0.50 [ 3] 0.50 [ 4] 0.50 [ 5] 0.50 [ 6] 0.50
[ 7] 0.50 [ 8] 0.50 [ 9] 0.50 [10] 0.50 [11] 0.50 [12] 0.50 [13] 0.50
[14] 0.50 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.50 [20] 0.50
Quality step 0: 0.00

Posteriors(CH6): 0.00 0.48 0.52 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(141.00) = 0.00
Probability of failure vector:
[ 0] 0.51 [ 1] 0.50 [ 2] 0.50 [ 3] 0.50 [ 4] 0.51 [ 5] 0.51 [ 6] 0.50
[ 7] 0.50 [ 8] 0.50 [ 9] 0.50 [10] 0.50 [11] 0.50 [12] 0.50 [13] 0.51
[14] 0.50 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.51 [20] 0.51
Quality step 1: 0.00

Posteriors(CA4): 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
P(313.76) = 1.00
Probability of failure vector:
[ 0] 0.53 [ 1] 0.50 [ 2] 0.50 [ 3] 0.50 [ 4] 0.51 [ 5] 0.51 [ 6] 0.52
[ 7] 0.52 [ 8] 0.50 [ 9] 0.50 [10] 0.52 [11] 0.52 [12] 0.50 [13] 0.51
[14] 0.50 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.51 [20] 0.51
Quality step 2: 0.31

Posteriors(AEF): 0.66 0.34 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(117.05) = 0.66
Probability of failure vector:
[ 0] 0.01 [ 1] 0.50 [ 2] 0.50 [ 3] 0.50 [ 4] 0.52 [ 5] 0.52 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.50 [13] 0.52
```

```
[14] 0.01 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.52 [20] 0.52
Quality step 3: 0.40

Posteriors(CH3): 0.00 0.00 0.93 0.07 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(123.37) = 0.93
Probability of failure vector:
[ 0] 0.00 [ 1] 0.50 [ 2] 0.01 [ 3] 0.01 [ 4] 0.52 [ 5] 0.52 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.50 [13] 0.52
[14] 0.01 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.52 [20] 0.52
Quality step 4: 0.41

Posteriors(CH5): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(77.07) = 0.00
Probability of failure vector:
[ 0] 0.00 [ 1] 0.50 [ 2] 0.01 [ 3] 0.01 [ 4] 0.67 [ 5] 0.67 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.50 [13] 0.50
[14] 0.01 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.50 [20] 0.50
Quality step 5: 0.42

Posteriors(CA7): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(108.92) = 0.00
Probability of failure vector:
[ 0] 0.00 [ 1] 0.50 [ 2] 0.01 [ 3] 0.01 [ 4] 0.80 [ 5] 0.60 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.50 [13] 0.60
[14] 0.01 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.50 [20] 0.50
Quality step 6: 0.43

Posteriors(AL1): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(62.64) = 0.00
```

```
Probability of failure vector:
[ 0] 0.00 [ 1] 0.50 [ 2] 0.01 [ 3] 0.01 [ 4] 0.89 [ 5] 0.56 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.50 [13] 0.56
[14] 0.01 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.56 [20] 0.50
Quality step 7: 0.43

Posteriors(AL2): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(65.20) = 0.00
Probability of failure vector:
[ 0] 0.00 [ 1] 0.50 [ 2] 0.01 [ 3] 0.01 [ 4] 0.94 [ 5] 0.53 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.50 [13] 0.53
[14] 0.01 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.53 [20] 0.53
Quality step 8: 0.48

Posteriors(CH1): 0.00 0.00 0.00 0.00 0.47 0.33 0.20 0.00 0.00
P(95.75) = 0.33
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 0.94 [ 5] 0.53 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.50 [13] 0.53
[14] 0.01 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.53 [20] 0.53
Quality step 9: 0.52

Posteriors(CA6): 0.27 0.40 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(281.67) = 0.33
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 0.95 [ 5] 0.53 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.01 [13] 0.53
[14] 0.01 [15] 0.50 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.53 [20] 0.53
Quality step 10: 0.57
```

```
Posteriors(AX2): 0.03 0.97 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(27.49) = 0.97
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 0.95 [ 5] 0.53 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.01 [13] 0.53
[14] 0.01 [15] 0.01 [16] 0.50 [17] 0.50 [18] 0.50 [19] 0.53 [20] 0.53
Quality step 11: 0.61

Posteriors(EM1): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.74 0.26 0.00
P(974.42) = 0.74
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 0.96 [ 5] 0.52 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.01 [13] 0.52
[14] 0.01 [15] 0.01 [16] 0.01 [17] 0.51 [18] 0.51 [19] 0.52 [20] 0.52
Quality step 12: 0.66

Posteriors(EM2): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.96 0.04 0.00
P(974.42) = 0.96
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 0.97 [ 5] 0.52 [ 6] 0.00
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.01 [13] 0.52
[14] 0.01 [15] 0.01 [16] 0.01 [17] 0.01 [18] 0.52 [19] 0.52 [20] 0.52
Quality step 13: 0.71

Posteriors(EM3): 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.79 0.21 0.00
P(994.64) = 0.79
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 1.00 [ 5] 0.50 [ 6] 0.00
```

```
[ 7] 0.00 [ 8] 0.01 [ 9] 0.50 [10] 0.01 [11] 0.01 [12] 0.01 [13] 0.50
[14] 0.01 [15] 0.01 [16] 0.01 [17] 0.01 [18] 0.01 [19] 0.50 [20] 0.50
Quality step 14: 0.76

Posteriors(CA1): 0.94 0.06 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(254.42) = 0.94
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 1.00 [ 5] 0.50 [ 6] 0.00
[ 7] 0.00 [ 8] 0.00 [ 9] 0.01 [10] 0.01 [11] 0.01 [12] 0.01 [13] 0.50
[14] 0.01 [15] 0.01 [16] 0.01 [17] 0.01 [18] 0.01 [19] 0.50 [20] 0.50
Quality step 15: 0.76

Posteriors(AX1): 1.00 0.00 0.00 0.00 0.00 0.00 0.00
P(23.91) = 1.00
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 1.00 [ 5] 0.50 [ 6] 0.00
[ 7] 0.00 [ 8] 0.00 [ 9] 0.01 [10] 0.01 [11] 0.01 [12] 0.01 [13] 0.50
[14] 0.00 [15] 0.01 [16] 0.01 [17] 0.01 [18] 0.01 [19] 0.50 [20] 0.50
Quality step 16: 0.76

Posteriors(CA2): 0.99 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00
P(253.54) = 0.99
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 1.00 [ 5] 0.50 [ 6] 0.00
[ 7] 0.00 [ 8] 0.00 [ 9] 0.00 [10] 0.01 [11] 0.01 [12] 0.01 [13] 0.50
[14] 0.00 [15] 0.01 [16] 0.01 [17] 0.01 [18] 0.01 [19] 0.50 [20] 0.50
Quality step 17: 0.77

Posteriors(CA3): 0.13 0.08 0.10 0.15 0.38 0.15 0.00 0.00 0.00 0.00 0.00
```

```
P(164.75) = 0.15
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 1.00 [ 5] 0.50 [ 6] 0.00
[ 7] 0.00 [ 8] 0.00 [ 9] 0.00 [10] 0.00 [11] 0.01 [12] 0.01 [13] 0.50
[14] 0.00 [15] 0.01 [16] 0.01 [17] 0.01 [18] 0.01 [19] 0.50 [20] 0.50
Quality step 18: 0.77

Posteriors(CA5): 0.00 0.00 0.33 0.36 0.31 0.00 0.00 0.00 0.00 0.00 0.00
P(290.02) = 0.36
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.01 [ 3] 0.01 [ 4] 1.00 [ 5] 0.50 [ 6] 0.00
[ 7] 0.00 [ 8] 0.00 [ 9] 0.00 [10] 0.00 [11] 0.00 [12] 0.01 [13] 0.50
[14] 0.00 [15] 0.01 [16] 0.01 [17] 0.01 [18] 0.01 [19] 0.50 [20] 0.50
Quality step 19: 0.78

Posteriors(CH2): 0.00 0.00 0.00 0.00 0.00 0.53 0.47 0.00 0.00 0.00 0.00
P(124.69) = 0.47
Probability of failure vector:
[ 0] 0.00 [ 1] 0.01 [ 2] 0.00 [ 3] 0.00 [ 4] 1.00 [ 5] 0.50 [ 6] 0.00
[ 7] 0.00 [ 8] 0.00 [ 9] 0.00 [10] 0.00 [11] 0.00 [12] 0.01 [13] 0.50
[14] 0.00 [15] 0.01 [16] 0.01 [17] 0.01 [18] 0.01 [19] 0.50 [20] 0.50
Quality step 20: 0.78
```

Bibliography

ANSI (1985), Measurement uncertainty, Technical Report 19.1-1985, ANSI/ASME.

Bacchus, F., Halpern, J. & Levesque, H. (1995), Reasoning about noisy sensors in the situation calculus, *in* 'Proc. International Joint Conf. on Artificial Intelligence', IJCAI, Montreal, Canada, pp. 1933–1940.

Basseville, M. (1988), 'Detecting changes in signals and systems', *Automatica* **24**(3), 309–326.

Berzuini, C. (1990), Representing time in causal probabilistic networks, *in* 'Proc. Sixth Conference on Uncertainty in Artificial Intelligence', Cambridge, Mass, U.S.A., pp. 15–28.

Bickmore, T. (1993), Real time sensor data validation, Technical Report NAS 3-25883, NASA Lewis Research Center, U.S.A.

Boddy, M. & Dean, T. (1994), 'Decision theoretic deliberation schedulling for problem solving in time-constrained environments', *Artificial Intelligence* **67**(2), 245–286.

Breese, J., Horvitz, E., Peot, M., Gay, R. & Quentin, G. (1992), Automated decision analytic diagnosis of thermal performance in gas turbines, *in* 'Proc.

Intl. Gas Turbine and Aeroengine Congress and Exposition', Cologne, Germany, pp. 1–9.

Brooks, R. & Iyengar, S. (1996), 'Robust distributed computing and sensing algorithm', *Computer* **29**(6), 53–60.

Buntine, W. (1994), 'Operations for learning with graphical models', *Journal of Artificial Intelligence Research* **2**, 159–225.

Chow, C. & Liu, C. (1968), 'Approximating discrete probability distributions with dependence trees.', *IEEE Trans. on Info. Theory* pp. 462–467.

Cohen, P. (1995), *Empirical methods for artificial intelligence*, MIT press, Cambridge, Mass., U.S.A.

Cooper, G. (1984), NESTOR: A computer-based medical diagnostic aid that integrate causal and probabilistic knowledge, PhD thesis, Computer Science Dept., Stanford Univ., U.S.A. Rep. No. STAN-CS-84-48.

Cooper, G. (1990), 'The computational complexity of probabilistic inference using bayesian networks', *Artificial Intelligence* **42**, 393–405.

Cooper, G. & Herskovitz, E. (1992), 'A bayesian method for the induction of probabilistic networks from data.', *Machine Learning* **9**(4), 309–348.

D'Ambrosio, B. (1992), Value-driven real time diagnosis, *in* 'Proceedings of the Third International workshop on the principles of diagnosis'.

D'Ambrosio, B. (1993), Incremental probabilistic inference, *in* 'Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence', Washington, D.C., U.S.A.

D'Ambrosio, B. & Burgess, S. (1996), Some experiments with real-time decision algorithms, *in* 'Proceedings of the Twelfth Annual Conference on

Uncertainty in Artificial Intelligence (UAI–96)', Portland, Oregon, U.S.A., pp. 194–202.

Dawid, A. P. & Lauritzen, S. L. (1993), 'Hyper Markov laws in the statistical analysis of decomposable graphical models', *Annals of Statistics* **21**, 1272–1317.

de Dombal, F., Leaper, D., Horrocks, J., J.Staniland & McCann, A. (1974), 'Human and computer aided diagnosis of abdominal pain: Further report with enphasis on performance', *British Medical Journal* **1**, 376–380.

Dean, T. & Boddy, M. (1988), An analysis of time dependent planning, *in* 'Proc. Seventh Natl. Conf. on AI', St. Paul, MN, U.S.A.

Dean, T. & Kanazawa, K. (1988), Probabilistic temporal reasoning, *in* M. Kaufmann, ed., 'Proc. Seventh National Conf. on Artificial Intelligence', AAAI, St. Paul, MN, U.S.A., pp. 524–528.

Dean, T. & Wellman, M. (1991), *Planning and control*, Morgan Kaufmann, Palo Alto, Calif., U.S.A.

Dougherty, J., Kohavi, R. & Sahami, M. (1995), Supervised and unsupervised discretization of continuous features, *in* A. Prieditis & S. Russell, eds, 'Machine Learning, Proceedings of the Twelfth International Conference', Morgan Kaufmann, San Francisco, CA, U.S.A.

Driver, E. & Morrell, D. (1995), Implementation of continuous bayesian networks using sums of weighted gaussians, *in* 'Proc. Eleventh Conference on Uncertainty in Artificial Intelligence', Montreal, Quebec, Canada.

Fox, M., Lowenfeld, S. & Kleinosky, P. (1983), Techniques for sensor based diagnosis, *in* 'Proc. eigth International Joint Conf. on Artificial Intelligence', IJCAI, Karlsruhe, West Germany, pp. 158–163.

Frank, P. (1990), 'Fault diagnosis in dynamic systems using analytical and knowledge based redundancy- a survey and some new results', *Automatica* **26**, 459–470.

Friedman, N. & Goldszmidt, M. (1996), Learning Bayesian networks with local structure, *in* 'Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)', Portland, Oregon, U.S.A., pp. 252–262.

Fung, R. & Crawford, S. (1990), Constructor: a system for induction of probabilistic models, *in* 'Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)', Vol. 2, MIT Press, Boston, Massachusetts, pp. 762–779.

Geiger, D. & Pearl, J. (1988), On the logic of causal models, *in* 'Proc. Fourth Workshop on Uncertainty in AI', St. Paul, Minn, U.S.A., pp. 136–147.

Geiger, D., Verma, T. & Pearl, J. (1989), Identifying independence in bayesian networks, Technical Report R-116, UCLA Cognitive Systems Laboratory, U.S.A.

Gorry, G. & Barnett, G. (1968), 'Experience with a model of sequential diagnosis', *Computers and Biomedical Research* **1**, 490–507.

Grelinger, G. & Morizet-Mahoudeaux, P. (1992), A fully integrated real time multi tasking knowledge based system: application to an on board diagnostic system, *in* 'Proc. IEEE Eighth Conference on Artificial Intelligence for Applications', pp. 310–316.

Hanks, S. (1988), Representing and computing temporally scoped beliefs, *in* M. Kaufmann, ed., 'Proc. Seventh National Conf. on Artificial Intelligence', AAAI, St. Paul, MN, U.S.A., pp. 501–505.

Heckerman, D. & Geiger, D. (1995), Learning Bayesian networks: A unification for discrete and Gaussian domains, *in* 'Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95)', Montreal, Quebec, Canada, pp. 285–295.

Heckerman, D., Geiger, D. & Chickering, D. (1994), Learning Bayesian networks: The combination of knowledge and statistical data, *in* 'Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)', Seattle, WA, pp. 293–301.

Henry, M. (1995), 'Sensor validation and fieldbus', *Computing and Control Engineering Journal* pp. 263–269.

Henry, M. & Clarke, D. (1993), 'The self-validating sensor: rationale, definitions and examples', *Control Engineering Practice* **1**(4), 585–610.

Horvitz, E. (1987), Reasoning about beliefs and actions under computational resource constraints, *in* 'Proc. Third Conference on Uncertainty in Artificial Intelligence', Seatle, WA, U.S.A., pp. 301–324.

Horvitz, E. & Barry, M. (1995), Display of information for time-critical decision making, *in* 'Proc. Eleventh Conference on Uncertainty in Artificial Intelligence', Montreal, Quebec, Canada, pp. 296–305.

Horvitz, E. & Rutledge, G. (1991), Time dependent utility and action under uncertainty, *in* 'Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence', Los Angeles, Calif, U.S.A., pp. 151–158.

Horvitz, E. J., Suermondt, H. J. & Cooper, G. F. (1989), Bounded conditioning: Flexible inference for decisions under scarce resources, *in* 'Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence (UAI-89)', Morgan Kaufmann, Windsor, Ontario, pp. 182–193.

Ibargüengoytia, P., Sucar, L. & Vadera, S. (1996*a*), A probabilistic model for sensor validation, *in* 'Proc. Twelfth Conference on Uncertainty in Artificial Intelligence', Portland, Oregon, U.S.A., pp. 332–339.

Ibargüengoytia, P., Sucar, L. & Vadera, S. (1996*b*), Real time sensor validation with probabilistic reasoning, *in* 'Proc. Congreso Iberoamericano de Inteligencia Artificial, IBERAMIA'96', Cholula, Puebla, México, pp. 468–468.

Ibargüengoytia, P., Vadera, S. & Sucar, L. (1997), A layered, any time approach to sensor validation, *in* 'European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty ECSQARU-97', Bad Honnef, Germany, pp. 336–349.

Jitnah, N. & Nicholson, A. (1997), treenets: A framework for anytime evaluation of belief networks, *in* 'European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty ECSQARU-97', Bad Honnef, Germany, pp. 350–364.

Kanazawa, K. (1991), A logic and time nets for probabilistic inference, *in* 'Proc. Ninth National Conf. on Artificial Intelligence', AAAI, Academic Press, pp. 360–365.

Khadem, M., Alexandro, F. & Colley, R. (1992), Sensor validation in power plants using neural networks, *in* D. Sobajic, ed., 'Proc. Inss summer workshop', Stanford, Calif. U.S.A., pp. 51–54.

Kullback, S. & Leibler, R. (1951), 'Information and sufficiency', *Ann. Math. Statistics* **22**, 79–86.

Laffey, T., Cox, P., Schmidt, J., Kao, S. & Read, J. (1988), 'Real time knowledge based systems', *AI Magazine* **9**(1), 27–45.

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

Milne, R. & Nicol, C. (1996), 'TIGER: knowledge based gas turbine condition monitoring', *AI Communications* **9**, 92–108.

Musliner, D., Durfee, E. & Shin, K. (1993), 'Circa: A cooperative intelligent real time control architecture', *IEEE Trans. on Systems, Man, and Cybernetics* **23**(6), 1561–1574.

Musliner, D., Hendler, J., Agrawala, A., Durfee, E., Strosnider, J. & Paul, C. (1995), 'The challenges of real time ai', *Computer* **28**(1), 58–66.

Neapolitan, R. (1990), *Probabilistic resoning in expert systems*, John Wiley & Sons, New York, New York, U.S.A.

Nicholson, A. & Brady, J. (1994), 'Dynamic belief networks for discrete monitoring', *IEEE Trans. on Systems, Man, and Cybernetics* **24**(11), 1593–1610.

Paul, C., Acharya, A., Black, B. & Strosnider, J. (1991), 'Reducing problem solving variance to improve predictability', *Communications of the ACM* **34**(8), 81–93.

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

Pearl, J. (1991), *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann. (Revised 2nd Edition).

Pearl, J., Geiger, D. & Verma, T. (1990), Conditional independence and its representation, *in* G. Shafer & J. Pearl, eds, 'Readings in Uncertain Reasoning', Morgan Kaufmann, San Mateo, California, U.S.A., pp. 55–60.

Peng, Y. & Reggia, J. (1987), 'A probabilistic causal model for diagnostic problem solving-parts i and ii', *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-17**(2,3), 395–406,146–162.

Pratt, I. (1994), *Artificial intelligence*, Macmillan, London, U.K.

Quinlan, J. (1986), 'Induction of decision trees', *Machine Learning* **1**(1), 81–106.

Ramoni, M. (1995), Anytime influence diagrams, *in* 'IJCAI-95 Workshop on Anytime Algorithms and Deliberation Scheduling', Montreal, Canada.

Shannon, C. & Weaver, W. (1949), *The mathematical theory of communication*, University of Illinois press, Urbana, Ill., U.S.A.

Spiegelhalter, D. J. & Lauritzen, S. L. (1990), 'Sequential updating of conditional probabilities on directed graphical structures', *Networks* **20**, 579–605.

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

Strosnider, J. & Paul, C. (1994), 'A structured view of real time problem solving', *AI Magazine* pp. 45–66.

Sucar, L., Pérez-Brito, J. & Ruiz-Suarez, J. (1995), Induction of dependence structures from data and its application to ozone prediction, *in* G. Forsyth & M. Ali, eds, 'Procedings Eight International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)', DSTO:Australia, pp. 57–63.

Upadhyaya, B. & Eryurek, E. (1992), 'Application of neural networks for sensor validation and plant monitoring', *Nuclear Technology*.

Wellman, M. P. & Liu, C.-L. (1994), State-space abstraction for anytime evaluation of probabilistic networks, *in* 'Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)', Seattle, WA, U.S.A., pp. 567–574.

Willsky, A. (1976), 'A survey of design methods for fault detection in dynamics systems', *Automatica* **12**(6), 601–611.

Yung, S. & Clarke, D. (1989), 'Local sensor validation', *Measurement & Control* **22**(3), 132–141.

Zilberstein, S. (1993), Operational rationality through compilation of Any time algorithms, PhD dissertation, University of California, Berkeley, Berkeley, California, U.S.A.

Zilberstein, S. & Russell, S. (1995), *Approximate reasoning using anytime algorithms*, Imprecise and Approximate Computation, Kluwer Academic Publishers, U.S.A., chapter 1.

Zilberstein, S. & Russell, S. (1996), 'Optimal composition of real-time systems', *Artificial Intelligence* **82**(1-2), 181–213.