Associative memories, stochastic activity networks and their application to sensor validation systems of nuclear power plants

Shen, Bin, M.S.

The University of Arizona, 1993

ASSOCIATIVE MEMORIES, STOCHASTIC ACTIVITY NETWORKS AND THEIR APPLICATION TO SENSOR VALIDATION SYSTEMS OF NUCLEAR POWER PLANTS

by

Bin Shen

A Thesis Submitted to the Faculty of the

DEPARTMENT OF NUCLEAR AND ENERGY ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN NUCLEAR ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 9 3

STATEMENT BY AUTHOR

SIGNED: [illegible]

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

[illegible] — 12/10/93

John G. Williams — Date
Professor of Nuclear Engineering

ACKNOWLEDGMENTS

This thesis would not have been possible without the emotional, financial, and technical support of Dr. John Williams, so I would like to first and foremost express my great gratitude and appreciation to him.

I am very pleased with the education I received at the University of Arizona and Shanghai Jiao Tong University. I would therefore like to acknowledge the fine work by the faculty at those two universities. I appreciate Dr. Wayne Jouse's help and incisive comments on this piece of work.

Finally, the support of my family and friends throughout my education has been of immeasurable help in attaining my degrees and finishing this thesis. To my wife, my parents, sister and to the many close friends I have made, I owe and express my deepest appreciation and gratitude.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

ABSTRACT

In this paper, the problem of designing an advanced sensor validation system (SVS) which is robust and fault-tolerant under faulty conditions is considered.

Associative memories, which provide robust pattern recognition are investigated as an information processing technology that can be applied to sensor validation. Studies of Binary Associative Memories (BAM) and Continuous Associative Memories (CAM) yield many results including 1) the stability condition of exemplars and spurious memories in BAMs, 2) the formula of choosing diagonal weights and bias that eliminates spurious memories most effectively in BAMs, 3) the convergence theory of CAMs that have asymmetric weight matrix with non-zero diagonal elements and non-monotonically increasing activation functions, 4) the energy function that explores the convergence behavior of CAMs, and 5) the hybrid learning algorithm that reduces spurious memories effectively in CAMs.

The concept of performability is introduced to the evaluation of SVS. A set of important performability variables is introduced. Stochastic Activity Networks are used as a modeling tool to evaluate the performability of SVS. An illustration example, the evaluation of the pressurizer SVS of a PWR, is provided.

CHAPTER 1
INTRODUCTION

In a nuclear power plant, outputs from several hundred instrumentation channels are used in control systems, protection systems and plant monitoring systems. The routine validation of these signals is useful in increasing the reliability of operator decisions, in improving the plant control actions and in minimizing plant downtime.

Since the incident at Three Mile Island unit 2, computerized plant status display, implementation of human factors in control room design, and plant monitoring based on expert system technology have seen a tremendous growth. One such proposed operator aid is a plant signal validation system. This system is used to check the consistency of redundant measurements (sensors) of selected process variables, estimate their expected values for plant-wide data, and detect, isolate and characterize the type of anomaly in the instrument channel outputs. Automated signal validation is necessary because of the large amount of information available, and because of the operator's inability to validate information from many diverse sources.

1.1 Sensor Validation Systems and Their Evaluations

1.1.1 Sensor Validation System

A Sensor Validation System (SVS) is a subsystem of a signal validation system. It takes the sensor measurements as its input, checks the consistency of redundant measurements, and provides the status (failure or not) of sensors and the best estimates of those measurements. It helps to provide more reliable information for control decisions and essential information for sensor maintenance.

A SVS has functions of fault detection, isolation and accommodation. Fault detection detects if there are any faults in a sensor system. Fault isolation identifies which faults have occurred. Fault accommodation recovers wrong outputs of a sensor system regardless of whether faults are detected or not.

There are many approaches for sensor validation. In section 1.2, a survey of different approaches will be presented. As we shall see, approaches based on neural networks and associative memories have shown their potential advantages over other approaches. However, since the properties of neural networks are still not thoroughly understood, their applications are only found in limited scope.

The first part of my research effort has been devoted to a better understanding of the properties of associative memories, so that how and how far they can be applied in SVSs will be understood. Analysis of the limitations of existing neural network structures also helps the pursuit of a more suitable structure of associative memory for application to sensor validation.

1.1.2 Evaluation of SVSs

The second part of my research effort has been devoted to the evaluation of SVSs. Given a SVS, we would like to know how well it performs. So, quantitative evaluation of a SVS is important and necessary. It forms a base for performance comparison among different SVSs and also points out the improvement direction for an existing SVS.

Traditionally, the evaluation of a system is divided into two aspects: 1) performance, which is the system's ability to perform under its normal condition (fault free), 2) dependability, which is the system's ability to keep in its normal condition. However, large complex systems, like nuclear power plants, often operate in a degraded condition (in presence of faults). Therefore, the concept of performability, the system's ability to perform in the presence of faults, is more suitable to represent the real situation of the system. The performability of a system is statistically measured by the performability variables.

A sensor system in a nuclear power plant is not simple. Its validation system is even more complicated. The complexity of a sensor validation system requires a sophisticated representation to account for performance and dependability in a unified way. The modern technique used here is stochastic activity networks (SANs).

In this paper, I explore the application of SANs as a modeling tool to evaluate SVSs in a nuclear power plant.

1.2 Survey of Sensor Validation

1.2.1 Introduction

Because of the increasing demands on reliability and safety of increasingly sophisticated plants and their elements, methods for improving the supervision and monitoring as part of the overall control of processes are of increasing interest. An essential prerequisite for the further development of intelligent automatic supervision is a system for early process fault detection, isolation and accommodation(FDIA). A SVS performs the functions of FDIA. The problem of FDIA of a dynamic system has received growing attention during the last 20 years, as can be seen from the survey papers (Willsky, 1976; Mironovski, 1980; Kligene and Telksnys, 1983; Isermann, 1984; Basseville, 1988; Frank, 1990).

Sensor validation is based on the redundancy of signal measurement. There are two kinds of signal redundancy, physical redundancy and analytical redundancy. Physical redundancy comes from multiple physical channels measuring the same signals. Analytical redundancy comes from quantitative relations and constraints among different signals in a system. Those relations and constraints usually have their mathematical forms based on physical laws.

The approach based on physical redundancy was traditionally used in SVSs. Recent developments of sensor validation have introduced the approach based on analytic redundancy, which can be implemented without need of additional physical instrumentation in the plant. Usually this approach makes use of mathematical models of a plant. However, recent researches have shown some potential advantage in using knowledge-based models and in using neural-network-based models when detailed mathematical models of a plant are not available.

In the rest of this section, approaches based on mathematical models, knowledge-based models and neural-network-based models will be introduced.

1.2.2 Mathematical model-based approach

A mathematical model-based approach takes advantages of the existing analytical redundancy in mathematical models of a plant. However, there is a price to pay for this benefit which arises from the need for the mathematical model of the plant. Not only is there considerably more computational expenditure required for on-line modeling of the process; a much more serious problem is that of the sensitivity of the detection system with respect to modeling errors that are by no means avoidable in practice. This problem increases the chance of malfunction of the validation system.

The robustness issue with respect to modeling error has been emphasized in Frank's survey (Frank, 1990). The following classification of mathematical approaches follows his classification . Readers who are not interested in the detailed classification of mathematical model approaches can skip to section 1.2.3 without loss of continuity.

The general procedure of model-based approaches can be roughly divided into two steps: 1) generation of residuals and 2) decision and isolation. The mathematical model-based approaches can be classified by the different methods of residual generation.

The parity space approach was developed by Potter and Suman (1977), Desai and Day (1981), and the group of Willsky et al. (1984 and 1986). The key idea is to check the parity (consistency) of the mathematical equations of the system (analytical redundancy relations) by using the actual measurements. The parity space is actually a null-space of the system model. A fault is declared to have occurred once preassigned error bounds are surpassed.

Independent from the above approach, the dedicated observer approach makes use of single or banks of Luenberger observers or Kalman filters, see for example, Clark et al. (1975), Willsky (1976), Frank (1978b; 1988). The Basic idea of the observer approach is to reconstruct the outputs of the system from the measurements or subsets of the measurements with the aid of observers or Kalman filters using the estimation error as a residual for the detection and isolation of the faults.

The fault detection filter, which was first proposed by Beard (1971) and Jones (1973), is a full-order state estimator with a special choice of feedback gain H so that the fault residuals come to lie in some certain direction.

All of the above approaches end up with state estimators. There is an alternative approach, the parameter identification approach (Isermann, 1984), which makes use of the fact that faults of a dynamic system are reflected in the physical parameters. This approach may be particularly useful for the detection of incipient faults.

1.2.3 Knowledge-based approach

Knowledge-based methods (expert systems) open a new dimension of possible fault diagnosis for complex processes with incomplete process knowledge, see S. Tzafestas in the book by Patton et at. (1989). The expert system approach makes use of qualitative models based on the available knowledge of the system instead of quantitative analytical models which are used by the mathematical model-based approach.

The knowledge-based validation system usually consists of four components, the knowledge base (knowledge of facts and rules), the data base (information about the present state of the process), the inference engine (forward or backward reasoning) and the explanation component (interface between user and the validation system). Fault diagnosis is done in the inference engine which has to combine heuristic reasoning with algorithmic operations in terms of the evaluation of analytical redundancy.

1.2.4 Neural network approach

The mathematical model-based technique is strongly dependent upon a reliable system model which may not always be attainable in a complex system. Furthermore, modeling errors may cause malfunction of the validation system and correcting these errors can be very costly. The knowledge-based techniques are based on finite rules of a sophisticated system. The performance of a knowledge-based model is limited by its knowledge, which is usually incomplete and cannot be improved from experience. The neural network approach overcomes these shortcomings by not requiring a detailed model of a plant and by continuously improving performance during learning.

An artificial neural network is a data processing system consisting of a number of highly interconnected processing elements (PE) in an architecture inspired by the structure of the cerebral cortex portion of the brain. Hence, neural networks are often capable of doing things which humans or animals do well but which conventional computers often do poorly, such as pattern recognition, learning function mapping and so on.

Systems of neural networks have shown great potential for use in environments in which robust, fault-tolerant pattern recognition and function mapping are necessary in a real-time mode, and in which the incoming data may be distorted or noisy. The function of SVS can be abstracted into fault recognition (fault detection and isolation) and fault-tolerant mapping (accommodation). Therefore sensor validation becomes one of the areas to which neural networks have shown great potential applications.

Pioneer researches have shown the feasibility and advantages of this approaches.

T. -H. Guo is one of the pioneers in this area. He first applied neural networks to sensor failure detection and recovery of the Space Shuttle Main Engine (1991). Belle R. Upadhyaya and Evren Eryurek are pioneers of neural networks in nuclear engineering. From 1987 to 1990, Upadhyaya, et al., at the University of Tennessee, developed a comprehensive signal validation system with application to current nuclear power plants and future advanced reactors. Their approach combined the cross information among process variables, sensor redundancy, and the data base containing knowledge about instrumentation, plant subsystem models, and history of instrument and plant behavior. An on-line signal validation system is in operation at the Northeast Utilities Millstone units 2 and 3 PWR plants. Recently, they applied neural networks for sensor validation and nuclear power plant monitoring (Eryrek and Upadhyaya 1990, 1992).

Eryrek and Upadhyaya (1990) at the University of Tennessee have investigated the feasibility of using neural networks for signal validation. The objective of these projects is to enhance the safety and performance of SVSs of nuclear plants through the use of neural networks. Both auto-associative and hetero-associative neural networks were applied for sensor and process monitoring in a Pressurized Water Reactor (PWR). Their report shows that neural networks offer several advantages over traditional methods for sensor validation and plantwide monitoring.

Many national laboratories have conducted their research in this area too. Their approaches are similar in terms of two-step-validation. In the first step, use a fault-tolerant mapping for state estimation (accommodation). Then, in the second step, check the deviation between estimates and corresponding sensor readings for fault detection and isolation. There are several significant developments, including development of a "universal network" (Mott et. al 1992), application of neural networks to neutron noise spectra (Korsah, Damiano and Wood 1992) and incorporation of neural networks into automated safeguards (Whiteson and Howell 1992).

A universal network is a variation of neural networks. It has been developed as a universal process modeling software which has been implemented on a dedicated computer system at EBR-II in Argonne National Lab., Idaho Falls (Mott et. al 1992). Their research has concluded that the universal network can provide extremely fast, accurate, and fault-tolerant estimation, validation, and replacement of signals in a real system.

Neutron noise data reduction, analysis, and interpretation can be used as means to diagnose degradation of reactor internals. In the Oak Ridge National Laboratory, a neural network-based method has been developed to represent neutron noise spectra (Korsah, Damiano and Wood 1992). The back-propagation learning method is applied.

An automated safeguards system involves detection of anomalous events, identification of the nature of the event, and recommendation of a corrective action. In Los Alamos National Lab, a neural network model has been applied in the first step: detection of anomalous events (Whiteson and Howell 1992). It detects anomalous events by predicting how a system should be behaving.

However, the technology of applying neural networks to SVS is not mature yet. Lack of thorough understanding of the functions, properties and limitations of different structures have prevented application and dissemination of neural networks to sensor validation, as well as to other areas.

As we know, Hopfield has shown that recurrent networks are suitable for associative memories. However, in the University of Tennessee approach (Eryrek E. and Upadhyaya B.R. 1992), the feed forward structure is still used for auto-associative networks and hetero-associative networks. In this paper, I will explore the feasibility of using associative memories for sensor validation systems.

1.3 <u>Associative Memories and Sensor Validation</u>

1.3.1 Redundancy checking and pattern recognition

In order to validate sensor signals, there must be some redundancy between signals, either physical or analytical redundancy. By checking the consistency of existing redundancy, one should be able to validate these signals, if the redundancy is sufficient. From another aspect, redundancy can also be represented as patterns. Therefore, consistency checking can be transformed into pattern recognition.

However, in a large system, such as a nuclear power plant, there are hundreds of signals that need to be validated, in real time. The pattern recognition task is tedious. Making it harder, analytic redundancy usually is implicit. A learning machine is needed to explore the implicit redundancy. Since the signals are noisy and disturbed in a real plant, the robustness of SVSs over disturbed patterns is necessary. In short, a SVS must have the ability of self-learning and fault-tolerant pattern recognition.

1.3.2 Associative memory and fault-tolerant pattern recognition

An associative memory is a neural network in which patterns can be stored and retrieved. Patterns, in general, can be represented by vectors. The main function of an associative memory is to retrieve patterns that have been stored in the memory, when some input patterns are presented, even though the input pattern may have been disturbed. If the input pattern is identical or close (when it is disturbed) to the pattern to be retrieved, the memory is called an auto-associative memory. If the input pattern is different from the pattern to be retrieved, the memory is called a hetero-associative memory.

One important feature of an associative memory is the ability to retrieve patterns regardless of noisy, even faulty input patterns. Another feature is its ability of on-line storing (learning), which makes it possible to explore more implicit redundancy based on experience. These two features make an associative memory a suitable device for self-learning and fault-tolerant pattern recognition in a SVS.

1.3.3 Contributions

One of the major types of associative memory is known as the Hopfield net(Hopfield 1982, 1984). Detailed description of associative memories will be carried in section 2.1.

One major part of this paper will be dedicated to theoretically analyzing the properties of the Hopfield type of network, so that how and how far it can be applied in the sensor validation system will be understood. In chapter 3, I will propose a series of theorems, which lead to better understanding of single layer associative memories. Furthermore, this knowledge leads to better learning algorithms as we shall see in chapter 3.

1.4 Stochastic Activity Network and Evaluation of Sensor Validation Systems

1.4.1 Performability

When we have a SVS in hand, we would like to know how it performs. More important, how does it perform in the presence of faults? Traditional performance and dependability modeling techniques require the separate evaluation of system performance and dependability, disregarding any dependencies that exist between them. The concept of "performability" overcomes the limitation (Meyer 1980). Informally, performability quantifies a system's ability to perform in the presence of faults. The performability of a system is statistically measured by the performability variables.

When a failure of a sensor is detected and isolated, it most likely will be repaired. Thus the SVS is closely related with the sensor maintenance and sensor performance. The evaluation of a SVS usually includes the evaluation of the performability of the sensor system itself, as well as of its validation system.

A sensor system in a nuclear power plant is not simple. Its validation system is even more complicated. The complexity of the sensor validation system requires a sophisticated representation to account for performance and dependability in a unified way. The modern technique used here is stochastic activity networks.

1.4.2 Stochastic activity networks

Stochastic activity networks (SANs) are a stochastic extension to Petri nets (Movaghar and Meyer 1984). They can be viewed as a modeling language which represents our knowledge of a sequence of random events of a complex system. It is machine readable so that it is feasible for computer application. Even more, its graphic interface makes it human readable so that it is easy to communicate.

As a problem solver, a SAN has its reward variables as output. The reward variables are defined based on the model and can be computed by the SAN's solver. Performability variables can be further constructed as functions of reward variables. SANs have been used successfully to evaluate a wide variety of computer systems and networks. Detailed description of SANs will be carried in section 2.2.

1.4.3 Contributions

In this paper, I explore the application of SANs as a modeling tool to evaluate SVSs in a nuclear power plant. The concept of performability is introduced into the evaluation of SVS. In Chapter 4, a set of important performability variables are defined. They construct a framework for the evaluation of SVSs. Strategies of using SANs for evaluating

SVSs are developed. As illustrative examples, the SAN models of a pressurizer sensor system are presented. The performability of the system is evaluated and analyzed.

CHAPTER 2

ASSOCIATIVE MEMORIES AND STOCHASTIC ACTIVITY

2.1 Associative Memories

2.1.1 Introduction

Through the years, two human traits have continued to elicit great interest among philosophers and researchers: the ability of humans to retrieve information on the basis of associated cues and the ability to recognize speech utterances and handwriting in a very robust manner despite major variations, distortions, or omissions.

The first ability is thought to be the power of a human's associative memory. For example, a few bars of a tune can evoke memory of the entire tune; a glimpse of the back of a head in a crowd can be sufficient basis for thinking of an old friend. Such power and other related phenomena have fueled a continuing interest of philosophers and psychologists in the "human's associative memory", even from as early as the days of Aristotle:

> "Thus memory belongs to the faculty of the soul to which imagination belongs; all objects which are imaginable are essentially objects of memory; all those that necessarily involve images are objects of memory incidentally." (Aristotle, 384-322 BC.)

In nuclear a power plant, the power of human associative memory also helps the operators to detect and identify failures in the plant based on the signature of the failures. This inspires the idea to use artificial associative memories for intelligent control in safety systems, such as sensor validation systems, fault detectors and diagnostics, and fault tolerant controller.

Since associative memories are human traits, researchers have looked to their limited knowledge of human neural nets for inspiration and guidance in computer simulation model building. Therefore many computer associative-memory models also tend to be nets of neuron like nodes functioning on the basis of distributed processing and storage.

One simple version of such a neural-net model is a binary, discrete-time, additive model, aspects of which have been studied since 1943 by a number of researchers, including Hebb [1949] and McCulloch and Pitts [1943]. In 1982, Hopfield first developed a network based on "two-state McCulloch-Pitts neurons", which will be simply called binary associative memory (BAM) because its input and output are limited to be binary numbers. Later, in 1984, he proposed another model based on "continuous neurons", which will be called continuous associative memory (CAM) because its input and output are real numbers. These models have received considerable attention because of Hopfield's detailed exposition of their characteristics.

Since the Hopfield Net was introduced, many applications of this type of network have been found and continuous research efforts have been devoted to understanding and developing its performance. Meany and Pimmel [1991] have performed a detailed analysis of how to use bias and non-zero diagonal terms to improve the performance of BAMs. Sudharsanan and Sundareshan [1990] recently have developed a special type of dynamic network (a variation of Hopfeild's CAM) which has its application to associative memories. Their simulation examples have shown that it has high accuracy in storing and retrieving information.

In the rest of this section, Hopfield's associative memory, Meany and Pimmel's work and Sudharsanan and Sundareshan's contribution will be introduced in a coherent frame of vector space, so that the notation and the structure will be consistent with later development in Chapter Three. Therefore, some refreshment of vector space and introduction to some important concepts will be necessary to be carried out first.

2.1.2 Vector space

Notation: all the vectors are underlined, like $\underline{x}$.

The inner-product of two vectors is viewed as a composition of two operators: the element by element product and the summation of the elements of their element by element product. The element by element product of two vector $\underline{z} = \underline{x}*\underline{y}$, $\underline{x}$, $\underline{y} \in R^N$ is defined as $\underline{z} \in R^N$ and its elements $z_i = x_i\ y_i$. The sum of the elements of a vector $<\underline{z}>$, $\underline{z} \in R^N$ is defined as $<\underline{z}> \in R$ and $<\underline{z}> = \sum z_i$. Therefore, the inner-product of two vector $\underline{x}^T\underline{y}$ can be defined as $<\underline{x}*\underline{y}>$.

The outer-product of two vector, $\underline{xy}^T$, can be viewed as an operator composed of two operators. It multiplies the vector $\underline{x}$ by the inner-product of $\underline{y}$ and the operand. Therefore, the outer-product, $\underline{xy}^T$, can also be denoted as $x<y*.>$.

In a vector space R^N, the magnitude $|\underline{x}|$ of a vector $\underline{x} \in R^N$ is defined as its L-2 norm, i.e., $|\underline{x}| = \sqrt{<\underline{x}*\underline{x}>}$. The angle θ between two vectors $\underline{x}$ and $\underline{y} \in R^N$ is defined as $\cos(\theta) = <\underline{x}*\underline{y}>/|\underline{x}||\underline{y}|$. On the other hand, the inner-product of two vector results in the magnitude of projection of one vector on the other, i.e., $<\underline{x}*\underline{y}> = |\underline{x}|\,|\underline{y}|\cos(\theta)$.

A binary space is a subset of a vector space of the same dimension, denoted as 2^N. A binary vector is a vector whose elements are +1 or -1. It can be shown that $|\underline{x}| = \sqrt{N}$ if $\underline{x} \in 2^N$.

We say that a vector $\underline{x}$ is greater than a vector $\underline{y}$, denoted as $\underline{x} >> \underline{y}$, if all the elements of $\underline{x}$ are greater than the corresponding elements of $\underline{y}$, i.e., $\underline{x} >> \underline{y}$ if and only if $x_i > y_i$, for all i. By analogy from two real numbers, x and y, which are of the same sign if and only if $xy > 0$, we say that two vectors x and y are of the same sign if and only if $\underline{x}*\underline{y} \gg \underline{\mathbf{0}}$ where $\underline{\mathbf{0}}$ is a vector whose elements are all zeros. Let $\underline{J}$ be the unit vector whose elements are all +1.

A hyper-quadrant is a set of all the vectors that are of the same sign. So, two vectors $\underline{x}$ and $\underline{y}$ are in the same hyper-quadrant if and only if $\underline{x}*\underline{y} \gg \underline{\mathbf{0}}$. Clearly, a hyper-quadrant is a subset of a vector space of the same dimension.

Each hyper-quadrant contains one and only one binary vector. We call this binary vector the quadrant vector of the hyper-quadrant, or loosely, the quadrant vector of the vectors in the same hyper-quadrant. In other words, each binary vector denotes a quadrant vector. Clearly, a binary space is the set of all the quadrant vectors.

The quadrant operator Q is a function which maps a vector to its quadrant vector, i.e., $Q : R^N \rightarrow 2^N$ and $Q(\underline{x})*\underline{x} \gg \underline{\mathbf{0}}$.

We will find that the concepts of hyper-quadrants, quadrant vector and the quadrant operator are quite useful and handy in describing and analyzing the properties of associative memories. The perspective of the inner-product and outer-product provides insight on the behavior of associative memories.

2.1.3 Hopfield's associative memory

In 1982, Hopfield proposed a stochastic model of associative memory based on McCulloch-Pitts neurons. It only has one recurrent layer, as depicted in Figure 2.1.

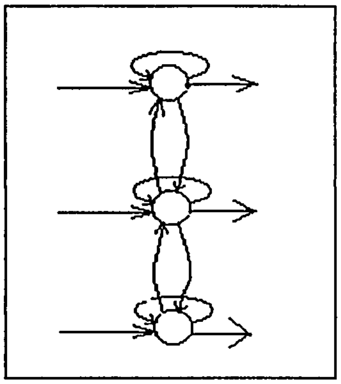


Figure 2.1 Diagram of the Hopfield Network

The state of the net is initialized by the input. Then the state in the recurrent layer will evolve until it reaches an equilibrium state. The output of the net is the equilibrium state of the net. The dynamic of the network can be mathematically described as follows:

$$\begin{aligned} &\underline{u}(0)=x \\ &\underline{v}(k)=g(\underline{u}(k)) \\ &\underline{u}(k+1)=W\underline{v}(k)+\underline{b} \qquad [2.1] \\ &\underline{y}=\underline{v}^S \\ &\textit{when} \quad \underline{v}^S(k)=\underline{v}^S(k+1) \end{aligned}$$

where k is the iteration step, $\underline{x}$ is a binary vector as initial input, $\underline{y}$ is a binary vector as final output, W is the recurrent weight matrix, $\underline{b}$ is the bias, $\underline{u}$ and $\underline{v}$ are the state vectors of the net $\underline{v}^s$and g($\underline{u}$) is the activation function. Hopfield used the quadrant operator *Q*(.) as the activation function. Since the input and the output of the memory are binary vectors, the model is called Binary Associative Memory (BAM).

The BAM stores patterns (exemplars $\underline{e}_i$, i = 1, 2, ... M), by forming its weight matrix using exterior-product formulation (EPF), the sum of outer-products of the exemplars, as described in the following equation.

$$W = \sum_{i=1}^{M} \underline{e}_i \underline{e}_i^T - MI + D_w \qquad [2.2]$$

where M is the number of exemplars to be stored in the memory and D_w is a diagonal matrix which forms the diagonal term of the weight matrix.

Hopfield prescribed zero bias ($\underline{b}=\underline{\mathbf{0}}$) in equation 2.1 and zero diagonal elements ($D_w=\mathbf{0}$) in equation 2.2.

In 1984, Hopfield proposed a deterministic, continuous model of associative memory, which can be described as the following equation set.

$$\begin{aligned} &\underline{u}(0) = \underline{x} \\ &C\dot{\underline{u}} = -R^{-1}\underline{u} + Wg(\underline{u}) + \underline{b} \\ &y = g(\underline{u}^s) \\ &when \quad \underline{u}^s = RWg(\underline{u}^s) + R\underline{b} \end{aligned} \qquad [2.3]$$

where k is the iteration step, $\underline{x}$ the initial input vector and $\underline{y}$ is the final output vector, $\underline{u}$ is the state vector of the net, W is the recurrent weight matrix, C is the matrix representing capacitance of the cell membranes, R is the matrix representing the resistance of the cell membranes, $\underline{b}$ is bias representing some fixed input and $g(\underline{u})$

is an activation function. $\underline{u}^{s}$ represents the stable steady state to which the network asymptotically evolves. Since the input and the output of the net are continuous vectors, the net is called Continuous Associative Memory (CAM).

The convergence property of the recall process in the CAM is governed by the Lyapunov function. A Lyapunov function is a non-negative continuous function of the state of the system and its time derivative is not positive. The convergence theorem states that if there is a Lyapunov function existing for a dynamic system, then the dynamic is guaranteed to converge to its stable state.

Hopfield introduced an energy function as follows:

$$E = -\frac{1}{2} g^{T}(u) W g(u) + \sum \int_{0}^{u} \frac{dg(u)}{du} R^{-1} u\, du - \underline{b}^{T} g(u) \qquad [2.4]$$

When W is a symmetric matrix with zero diagonal elements, the time derivative of the energy function is determined by the following equation:

$$\frac{dE}{dt} = -\sum_{i} C_i \frac{g(u_i)}{du_i} \left(\frac{du_i}{dt} \right)^2 \qquad [2.5]$$

Since dg/du is a monotonically increasing function, dE/dt is always not positive during state evolution and will reach zero only when $du_i/dt=0$ for all i. Therefore, the energy function is a Lyapunov function and the iteration must converge and lead to a stable state.

If a pattern is a stable state (a minimum of the energy function) in the memories, then it is stored (stabilized) in the memory. On the contrary, if a pattern is not a stable state in the memories, it is not stored (destabilized) in the memory. Since the states of the memory will converge to a stable state, the stable state is also called an attractor, which attracts its adjacent states.

The learning procedure is an attempt to store exemplars in the memories, in other words, to stabilize exemplars. Any stabilized patterns that are not exemplars are called spurious memories. The objective of learning (storing exemplars) is to stabilize all the exemplars and to eliminate spurious memories at the same time.

The Hopfield's model has three major limitations in performance: spurious memories, destabilized exemplars and limited capacity. Much research has been done to improve the performance of BAMs and CAMs by modifying the model or changing the learning algorithm. Sudharsanan and Sundareshan (1990) proposed to use Backwards Error Propagation (BEP) to store the exemplars. Meany and Pimmel (1989, 1991) investigated the effect of bias and non zero diagonal terms of the weight matrix (diagonal weights) on the performance of a BAM. Their work will be introduced in the following two sections.

2.1.4 Sudharsanan and Sundareshan's model

Sudharsanan and Sundareshan proposed a dynamic network as follows.

$$\begin{aligned}
&\underline{u}(0)=\underline{x}\\
&\dot{\underline{u}}=-\underline{u}+Wg(\underline{u})+\underline{b}\\
&\underline{y}=\underline{u}^s\\
&\textit{when } \underline{u}^s=Wg(\underline{u}^s)
\end{aligned} \qquad [2.6]$$

where $\underline{x}$ is the vector presented to the net and $\underline{y}$ is the output vector of the net, $\underline{u}$ is the state of the net, W is the weight matrix and $\underline{b}$ is the bias. $\underline{u}^s$ represents the stable steady state to which the network asymptotically evolves. They used, as the activation function, the inverse tangent function with large gain, which is a continuous function providing a good approximation to the quadrant operator.

Besides their use of identity matrices for the capacitance matrix and the resistance matrix, their network differed from Hopfield's in their use of $\underline{u}^s$ as the final output of the memory, instead of $g(\underline{u}^s)$ as used in Hopfield's model. The choice of $\underline{u}^s$ as the final output lifts the limitation that the output is in the range of the activation function.

They also used the Backwards Error Propagation (BEP) algorithm to form the weight matrix and bias. The error E of the network is defined as follows.

$$E=\frac{1}{2}\sum_i\left(\underline{y}_d(\underline{x}_i)-\underline{y}(\underline{x}_i)\right)^T\left(\underline{y}_d(\underline{x}_i)-\underline{y}(\underline{x}_i)\right) \qquad [2.7]$$

where $\underline{y}_d(\underline{x}_i)$ is the desired output pattern of the net and $\underline{y}(\underline{x}_i)$ is the actual output pattern of the net when the input pattern is $\underline{x}_i$.

The BEP algorithm uses the gradient descent rule to minimize the error. The desired patterns (exemplars) will be stored (stabilized) in the net when the error of the net is reduced to zero. The BEP algorithm yields the following update of weight matrix and bias.

$$\begin{aligned}\Delta W &= \mu(\underline{y}_d - \underline{y})g(\underline{y})^T \\ \Delta \underline{b} &= \eta(\underline{y}_d - \underline{y})\end{aligned} \qquad [2.8]$$

where μ and η are learning constants. Sudharsanan and Sundareshan have shown that the learning constants must be in appropriate ranges in order to get good training result.

2.1.5 Meany and Pimmel's analysis

Meany and Pimmel investigated the effect of bias and non zero diagonal terms of the weight matrix (diagonal weights) on the performance of a BAM (Meany 1989, Meany and Pimmel 1992). Bias is utilized to break network symmetry, eliminating exemplar complements as stable states. Positive diagonal terms can stabilize otherwise unstable exemplars, but may also stabilize otherwise unstable spurious memories. They suggested choosing diagonal weights according to the following equation which guarantees the stability of every exemplar.

$$w_{ii} = max[0, max_s((-\sum_{j \neq i} w_{ij}e_j^{\,s} - b_i) + e_i^{\,s})] + \varepsilon \qquad [2.9]$$

where ε is a small positive constant. Their simulation suggested that let ε to be 0.01.

2.2 Stochastic Activity Network

2.2.1 Introduction

Stochastic activity networks (SANs) are a stochastic extension to Petri nets. They can be viewed as a modeling language which represents our knowledge of a sequence of random events of a system. Movaghar and Meyer (1984), at University of Michigan, proposed the SAN model. The Michigan Evaluation Tool for the Analysis of Stochastic Activity Networks (METASAN) was the first SAN-based software package (Sanders and Meyer 1986). These studies illustrated SAN's usefulness in representing real systems.

UltraSAN, (Couvillion et. al 1991) recently developed in University of Arizona, is new graphical, X window-based software package for SAN. It incorporates some innovations including: 1) a class of SAN-level reward variables common to both analytical and simulation solution methods, 2) methods that use the reward variable choice and the SAN structure to greatly reduce the size of the stochastic process required for an analytical solution, and 3) methods that use the reward variable choice and the SAN structure to reduce the number of activities checked on each state change, thus speeding the simulation.

The following description of SAN is based on the documentation of the UltraSAN software package.

2.2.2 Primitives

As a modeling language, SAN has its "keywords", primitives. There are three kinds of primitives in SAN, activities, places and gates. Those primitives are interconnected by arcs to form a network. The semantics of the net represents our knowledge of a sequence of random events.

In a network, places are denoted by circles. They represent states of objects and may contain tokens. The number of tokens in a place represents the number of objects in the state. The number of tokens must be initialized when a place is constructed.

For example, the SAN model of a SVS system is plotted in Figure 2.2. In the figure, "NORMAL", "FAILED", "DETECTED" and so on, are places. The place "NORMAL" represents the normal state of sensors. The number of tokens in place "NORMAL" represents the number of normal sensors and shall be initialized with the number of normal sensors initially existing.

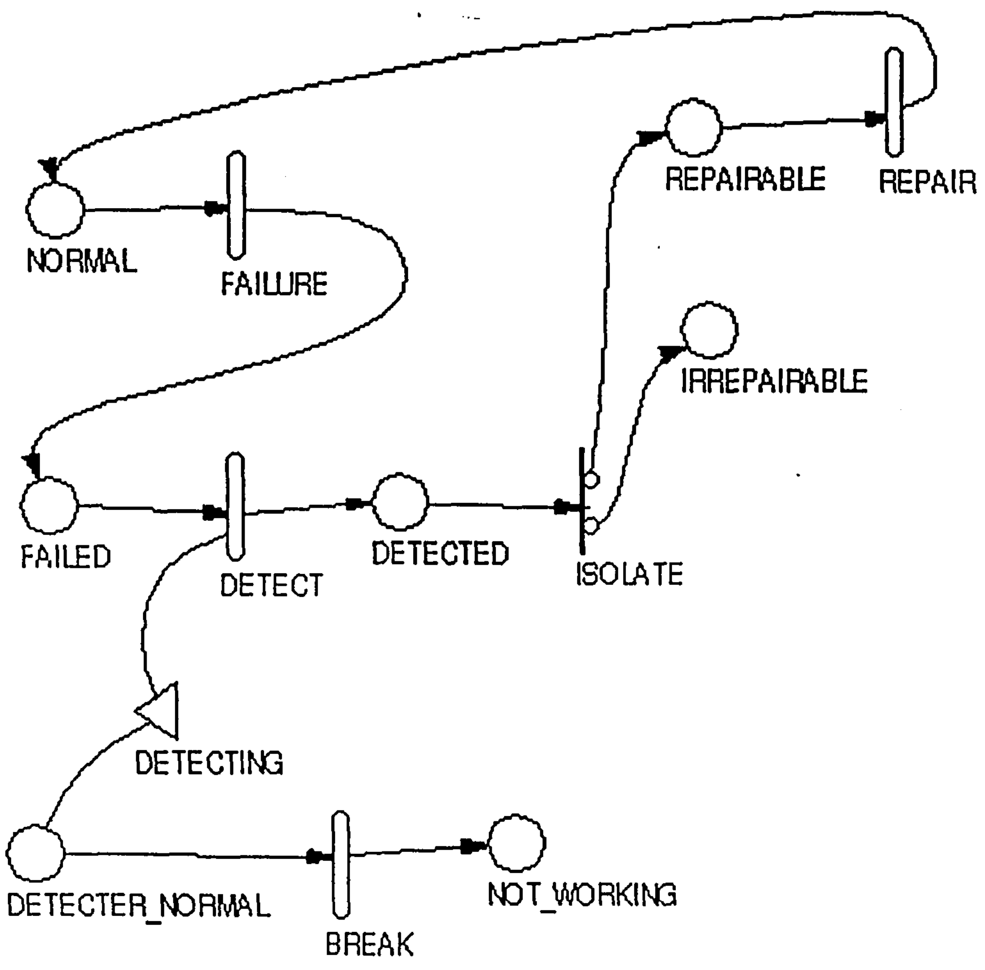


Figure 2.2 the SAN Model of a Simple SVS

Activities represent the transitions between states and are connected to places by arcs. The input places of an activity are connected to its left hand side and the output places to the right hand side. On the completion of an activity, the number of tokens in its input places all decrease by one and the number of tokens in its output places all increase by one.

In SANs, there are two kinds of activities, timed activities, denoted by ovals, and instantaneous activities, denoted by vertical bars. Timed activities represent the state transitions that complete in a considerable amount of time. The probability distributions of the elapsed time of their completion must be specified when they are constructed. Instantaneous activities represent the state transitions that complete in a negligible amount of time.

In Figure 2.2, "FAILURE", "DETECT", "REPAIR" and so on, are timed activities. The activity "FAILURE" represents the transition of a sensor from normal to abnormal. Since the elapsed time of isolating a detected sensor failure is negligible, the isolation procedure is represented by an instantaneous activity, "ISOLATE".

Cases associated with an activity are denoted by small circles on the right side of the activity. They divide output places of the activity into different groups according to the outcomes of the activity. On the completion of the activity, one of its cases will be chosen and only the group of places that are connected to that case become active output places, that is, the number of tokens in those places will increase by one. In Figure 2.2, two cases in the activity "ISOLATE" differentiate two possible states of a failed sensor, repairable and irreparable.

Gates, denoted by triangles, can be divided into input gates, which are connected to the input side (left) of activities and output gates, which are connected to the output side (right) of activities. Input gates enable or inhibit the activity to which they connected according to the value of their predicates. An activity will be enabled only when the predicates of all its input gates return TRUE and the number of tokens in neither of its input places are zero. Output gates do not have predicates, so they do not have the function of enabling or inhibiting activities. Both input and output gates have functions that may change the number of tokens in the places on the completion of the activity. In short, gates provide greater flexibility in specifying enabling and completion rules for activities.

For examples, in Figure 2.2, "DETECTING" is an input gate. It enables activity "DETECT" when there is some detector available, that is, the number of tokens of place "DETECTER_NORMAL" is not zero.

2.2.3 Reward Variables

Using a SAN model, we now can represent our knowledge of a system in a unique form, both machine readable and human readable. The next question is what information we can get from it. As a problem solver, a SAN returns reward variables as its output.

In SANs, there are two types of rewards, impulse reward associated with each state change and a rate reward associated with the duration of certain states. Activity completion can be assigned to impulse rewards while particular numbers of tokens in places are assigned to rate rewards. Since impulse rewards are not used in this project. I omit their detailed description.

A rate reward is composed of a predicate and a function. The function takes the numbers of tokens in places as its arguments and returns a real number when its predicate is TRUE. Therefore, a rate reward accumulates information of the duration of particular states. Then evaluation of a system can be done by processing the information i.e., constructing performability variables as functions of appropriate reward variables.

2.2.4 Solvers

After our knowledge of a sequence of random events is represented in a SAN model, SAN will transform it into a

Markov chain, i.e., into a Markov state transition equation as follows.

$$\underline{P}(k+1) = Q(k)\underline{P}(k) \qquad [2.10]$$

where $\underline{P}(k)$ is the vector of state-occupancy probabilities and Q(k) is the state transition matrix at time k.

Quite often, we would like to know the asymptotic state of the system, i.e., $\lim_{k\to\infty} P(k)$. The following well known theorem is used:

Theorem: If the Markov chain is irreducible and aperiodic, then the asymptotic state-occupancy probabilities exist and are equal to the steady state-occupancy probabilities which satisfy the steady state equation, i.e.,

$$\lim_{k\to\infty} P(k) = P^s$$
$$\text{where } QP^s = P^s \text{ and } \sum_i P_i^s = 1 \qquad [2.11]$$

There are two type of solvers in SAN, the analytic solver and the simulation solver.

The analytic solver solves the desired steady state-occupancy probabilities from the steady state equation, [2.11]. Then, it computes the mean, variance, probability density function and probability distribution function for each desired reward variable from known steady state-occupancy probabilities.

If the condition of the above theorem is not satisfied, asymptotic state-occupancy probabilities either do not exist or can not be computed from the steady state equation. Then, the simulation solver can be applied, instead of the analytic solver.

The simulation solver uses the Markov state transition equation to compute the state-occupancy probabilities at every instance. And then, like the analytic solver, it computes the mean, variance, probability density function and probability distribution function for each desired reward variable from known steady state-occupancy probabilities.

CHAPTER 3

ANALYSIS OF ASSOCIATIVE MEMORIES

3.1 Issues of Associative Memories

In chapter 2, Hopfield's Binary Associative Memory (BAM) and Continuous Associative Memory (CAM) are introduced. For information storing, the weight matrix with zero diagonal weights is formed by the Exterior Product Formulation (EPF). Hopfield proved that the convergence of the network is guaranteed if 1) the weight matrix is symmetric and with zero diagonal terms and 2) the activation function is monotonically increasing. However, Hopfield's model has three major limitations in performance: spurious memories, destabilized exemplars and limited capacity.

Much research has been done to improve the performance of BAMs and CAMs by modifying the model or changing the learning algorithm. Meany and Pimmel (1989, 1992) have proved that a weight matrix with non-negative diagonal weights still guarantees the convergence of a BAM and will stabilize otherwise unstable exemplars. They also use biases to destabilize the exemplar complements. (Complements are vectors with all signs reversed relative to the exemplars.)

Sudharsanan and Sundareshan (1990, 1991) modified the CAM model and applied a Backwards Error Propagation (BEP) algorithm to stabilize continuous patterns in their net. In their approach, there is no constraint on the choice of weight matrix. Recent research (Yoshizawa, Morita and Amari 1992) has also shown that a non-monotonic activation function may increase the capacity of a CAM. However, convergence may not be guaranteed.

However, during the development of associative memory, there are some remaining questions. Regarding the issue of spurious memories and unstable exemplars, the questions are: 1) under what conditions, are exemplars stabilized and 2) where, or more precisely, in which hyper quadrants do spurious memories exist? Regarding choosing non-zero diagonal weights and biases, the question is: how to choose those parameters that can stabilize all the exemplars and, at the same time, most effectively eliminate spurious memories? Regarding the generalization and modification of Hopfield's model, the question is: what is the convergence property when Hopfield's convergence condition no longer holds? Regarding the convergence behavior, how do different initial states gradually evolve to their destines? Regarding the different learning algorithms, what is the effectiveness of EPF and BEP algorithms?

In this chapter, both the BAM and CAM models will be studied analytically so that those issues are addressed and questions are answered. In section 3.2, vector analysis is performed on BAMs and the first two issues will be addressed. In section 3.3, the convergence property and convergence condition of CAM will be discussed. In section 3.4, the two learning algorithms, EPF and BEP, will be studied and a so called Normalized Exterior Product Formulation (NEPF) will be proposed. Finally, section 3.5 summarizes the chapter.

3.2 Vector Analysis of Binary Associative Memory

3.2.2 Stability conditions of BAM

As we know, a BAM formed by EPF often has spurious memories, and even worse, it sometimes may have destabilized exemplars. Then two questions arise: Under what conditions, are exemplars stabilized, and where, or more precisely, in which hyper quadrants, do spurious memories exist?

Theorem 1 (original) answers those two questions by stating the stability condition of a binary pattern in a BAM formed by EPF. For detailed description of BAM and notations, see section 2.3.

Theorem 1: A binary vector $\underline{v}$ is stabilized in a BAM constructed by EPF, if and only if the following condition holds.

$$\underline{v} * (\underline{\tilde{v}} + \underline{b}) >> M\underline{J} - D_w \underline{J}$$

$$\underline{\tilde{v}} = N \sum_{i=1}^{M} \underline{e}_i \cos(\theta_i) \qquad [3.1]$$

where θ_i is the angle between $\underline{v}$ and $\underline{e_i}$.

Proof:

From EPF (equation 2.2), we get the following equation.

$$W\underline{v} = \left(\sum_{i=1}^{M} \underline{e}_i \underline{e}_i^T - MI + D_w\right)\underline{v} = N\sum_{i=1}^{M} \underline{e}_i \cos(\theta_i) - MI\underline{v} + D_w\underline{v} \qquad [3.2]$$

According to the property of the quadrant vector, $\underline{u}(k)$ and $\underline{v}(k)$ are always in the same quadrant. The quadrant vector $\underline{v}(k)=\underline{v}(k+1)$ if and only if $\underline{u}(k+1)=\underline{u}(\underline{v}(k))=W\underline{v}(k)+\underline{b}$ and $\underline{v}(k)$ are in the same quadrant.

Therefore, a binary vector $\underline{v}$ is stabilized in a BAM, if and only if the vector $\underline{u}(\underline{v})$ is in the same quadrant as $\underline{v}$, i.e. $\underline{u}(\underline{v})*\underline{v}>>\underline{\mathbf{0}}$. Thus, we get the following inequality for a stabilized pattern:

$$\begin{aligned}\underline{u}(\underline{v})*\underline{v} &= (W\underline{v}+\underline{b})*\underline{v} = (N\sum_{i=1}^{M}\underline{e}_i\cos(\theta_i) - MI\underline{v} + D_w\underline{v} + \underline{b})*\underline{v} \\ &= (\underline{\tilde{v}}+\underline{b})*\underline{v} - (M\underline{J} - D_w\underline{J}) >> \underline{0}\end{aligned} \qquad [3.3]$$

Moving ($M\underline{J}-D_w\underline{J}$) to the right hand side of the above inequality, we get theorem 1.

Theorem 1 provides the general stability condition of all the states. As we can see, the right hand side of inequality 3.1 is independent of the pattern $\underline{v}$. The larger are the value of diagonal weights, the smaller is the right hand side of inequality 3.1, so, more patterns can satisfy the stability condition.

According to theorem 1, a spurious memory exists if a binary pattern $\underline{v}$ is not an exemplar but satisfies the condition. Therefore, in order to eliminate spurious memories as much as possible, we should choose diagonal weights as small as possible. However, the values of diagonal weights should not be too small to stabilize exemplars.

Corollary 1 (original) provides the stability condition of an exemplar.

Corollary 1: The exemplar $\underline{e}_i$ is stabilized in a BAM constructed by EPF of M exemplars, if and only if the following inequality holds.

$$(\underline{\tilde{e}}_i + \underline{b}) * \underline{e}_i >> (M-N)\underline{J} - D_w \underline{J}$$
$$\underline{\tilde{e}}_i = N \sum_{j \neq i}^{M} \underline{e}_j \cos(\theta_{ij}) \qquad [3.4]$$

where θ_{ij} is the angle between $\underline{e}_i$ and $\underline{e}_j$.

Proof:

From theorem 1, we have

$$(\underline{\tilde{v}} + \underline{b}) * \underline{v} = \underline{e}_i * \left(N \sum_{j=1}^{M} \underline{e}_j \cos(\theta_{ij}) + \underline{b} \right) = N\underline{e}_i * \underline{e}_i + \underline{e}_i \left(N \sum_{j \neq i}^{M} \underline{e}_j \cos(\theta_{ij}) + \underline{b} \right) \qquad [3.5]$$
$$= N\underline{J} + (\underline{\tilde{e}}_i + \underline{b}) * \underline{e}_i >> M\underline{J} - D_w \underline{J}$$

Moving N$\underline{J}$ to the right hand side of the above equation, we get corollary 1.

As we can see, if the diagonal weights are too small, some exemplars may become unstable. The smaller value of diagonal weights imply a larger right hand side of inequality 3.4. In order to keep all exemplars stable, the right hand side must be upper bounded. Therefore, the diagonal weights must be lower bounded.

In conclusion, if we choose the diagonal weights slightly greater than their lower bound, the BAM shall have least spurious memories while all the exemplars are stabilized at the same time. Now, the question is: what is the lower bound of the diagonal weights? In the next section, we will discuses how to choose the diagonal weights and bias so that spurious memories can be minimized.

3.2.2 Diagonal Weights and Bias

In Meany and Pimmel's paper, bias is used to destabilize the complements of exemplars. Actually, the function of bias can be more than that. As we will see in this section, bias can also eliminate other spurious memories by decreasing the lower bound of diagonal weights.

Lemma 1 (original) shows the relation between the lower bound of diagonal weights and the bias. Theorem 2 provides the choice of bias that results in the lowest lower bound of diagonal weights.

Lemma 1 The lower bound of the diagonal weights that guarantee the stability of exemplar $\underline{e}_i$ is $M-N-(\tilde{e}_{ji}+b_j)e_{ji}$, where $\tilde{e}_{ji}$ is the jth element of $\underline{\tilde{e}}_i$ and e_{ij} is the jth element of $\underline{e}_i$.

Proof:

Recall D_w in corollary 1 is a diagonal matrix whose diagonal elements are the diagonal weights W_{jj}. Solving inequality [3.4] for W_{jj}, we get

$$W_{jj} > M - N - (\tilde{e}_{ji} + b_j)e_{ij} \qquad [3.6]$$

So the lower bound of diagonal weights is the right hand side of the above inequality.

The lower bound of diagonal weights that stabilizes all the exemplars must satisfy inequality 3.6 for all exemplars. Lemma 1 shows that the lower bound of the diagonal weights is a function of bias. In other words, different biases may result in a different lower bound of the diagonal weights. Smaller values of the diagonal weights means fewer spurious memories, so the bias that results in lowest lower bound of the diagonal weights is desirable.

Theorem 2: (original) In a BAM constructed by EPF, by choosing bias and diagonal weights as follows, the diagonal weights have their lowest lower bound W_{jj}^b that keeps all the exemplars stable.

$$\underline{b} = -\frac{\underline{e}^+ + \underline{e}^-}{2}$$

$$W_{jj}^b = M - N - \frac{e_j^+ - e_j^-}{2} \qquad [3.7]$$

where $e_j^+ = \min\{\tilde{e}_{ji} | e_{ji} = +1\}$

$e_j^- = \max\{\tilde{e}_{ji} | e_{ji} = -1\}$

Proof:

If $e_{ji}=1$ in inequality 3.6, we have

$$W_{jj} > (M - N) - \tilde{e}_{ji} - b_j \qquad [3.8]$$

Let e_j^+ be the minimum of such $\tilde{e}_{ji}$ that $e_{ji}=+1$, then the above inequality holds if the following inequality holds.

$$W_{jj} > (M-N) - e_j^+ - b_j \qquad [3.9]$$

For $e_{ji}=-1$, we have the following inequality

$$W_{jj} > (M-N) + \tilde{e}_{ji} + b_j \qquad [3.10]$$

Let e_j^- be the maximum of such $\tilde{e}_{ji}$ that $e_{ji}=-1$, then the above inequality holds if the following inequality holds.

$$W_{jj} > (M-N) + e_j^+ + b_j \qquad [3.11]$$

Solving inequality 3.9 and inequality 3.11 for minimum W_{jj}, we get

$$b_j = \frac{e_j^+ + e_j^-}{2} \qquad [3.12]$$

Substituting the above equation into inequality 3.11, we get the lowest lower bound W_{jj}^b of W_{jj} as follows

$$W_{jj}^b = M - N - \frac{e_j^+ - e_j^+}{2} \qquad [3.13]$$

Theorem 2 shows how diagonal weights and bias should be chosen so that spurious memories are minimized. Notice that theorem 2 does not guarantee that there are no spurious memories. Instead, it states that given the choice of these parameters, that is the best that can be done to eliminate spurious memories while all the exemplars are kept stable.

3.3 The Convergence Property of Continuous Associative Memories

3.3.1 The Energy Function and Convergence Theorem

As we know, Hopfield's model has three major limitations in performance: spurious memories, destabilized exemplars and limited capacity. In order to improve the performance of CAM, many modified models have been proposed. These modifications include lifting the constraints on the weight matrix and using non-monotonic activation functions. However, under these modifications, Hopfield's convergence condition no longer holds. An appropriate convergence theorem is required.

In this section, an energy function is introduced. The condition of the existence of a Lyapunov function is used as a sufficient condition for convergence of the CAM. For description of the CAM and Lyapunov function, see section 2.2.

In order to study the convergence property of a recurrent layer, I introduce an energy function *E* which is defined as the magnitude of the time derivative of the state vector $\underline{u}$, i.e., $E=|\underline{\dot{u}}|$. Obviously, $E\geq 0$, and $E=0$ only if $\underline{\dot{u}}=0$. From equation 2.6 and the definition of the magnitude of a vector, we have

$$E=\sqrt{<(-\underline{u}+Wg(\underline{u})+\underline{b})*(-\underline{u}+Wg(\underline{u})+\underline{b})>} \qquad [3.14]$$

Theorem 3: (original) The energy function is a Lyapunov function if the matrix WG-I is negative definite, where G=dg($\underline{u}$)/d$\underline{u}$.

Proof:

First, recall that E $\geq$ 0.

Secondly, find $dE/d\underline{u}$ from equation 3.14,

$$\frac{dE}{d\underline{u}} = \frac{1}{E}(WG - I)\dot{u} \qquad [3.15]$$

Thirdly, according to the chain rule,

$$\frac{dE}{dt} = \left\langle \left(\frac{dE}{d\underline{u}}\right)^T * \frac{d\underline{u}}{dt} \right\rangle = \frac{1}{E}\underline{\dot{u}}^T (WG - I)^T \underline{\dot{u}} \qquad [3.16]$$

From the definition of a negative definite matrix and the condition WG-I is negative definite, we get $dE/dt \geq 0$, and $dE/dt = 0$ only if $\underline{\dot{u}} = 0$.

Therefore, the energy function is a Lyapunov function.

As we know, if there is a Lyapunov function existing, the dynamic system is guaranteed to converge to its stable state. Therefore, theorem 3 actually states a sufficient condition for the convergence of a recurrent network.

Recall that in the Hopfield's convergence theorem, the convergent conditions are 1)the weight matrix must be symmetric and with zero diagonal weights and 2)the activation function must be monotonically increasing. But theorem 3 states the condition differently. It can be applied to either monotonic or non-monotonic activation functions. It states that there are not necessarily any limitations on either the weight matrix or the activation function to guarantee the convergence of the network, as long as the combination of these two, i.e., (WG-I), satisfies the constraint.

3.3.2 The Convergent Trajectory of the CAM

As we know, there are many minima (attractors) that attract the state of the network to evolve to those attractors. But in a CAM, its convergence behavior, i.e., which and how states will evolve to attractors is usually still unknown. Now, with the help of the above defined energy function, this mystery can be unwrapped.

Recall that in BAM, Hopfield used the quadrant operator $Q(\underline{u})$ as the activation function (see section 2.1, 2.2). If all elements of $\underline{u}$ are not equal to zero, i.e., $u_j \neq 0$ for all j, then $dQ/d\underline{u}=\underline{\mathbf{0}}$. If there are some $u_j=0$, then $\underline{u}$ falls on the a hyper-plane that separates the hyper-quadrants.

Quite often, in CAM, a continuous function which is approximate to the quadrant operator, for example, $g(\underline{u}) = \text{acrtan}(100\pi\underline{u})*2/\pi$, is used. This type of activation function has two properties: 1) $G=dg/d\underline{u}\cong\mathbf{0}$, when $\underline{u}$ is far from the axes, i.e. all its elements u_j are not close to zero, and 2) $g(\underline{u}_1)\cong g(\underline{u}_2)$, when $\underline{u}_1$ and $\underline{u}_2$ are in the same quadrant and neither of them have elements close to zero.

Theorem 4: (original) When the state of CAM is far away enough from the hyper-planes which separate hyper-quadrants so that G≅**0**, the state will evolve towards the direction of the steepest gradient descent of the energy function with rate E*|∇E|, i.e., as specified in the following equation:

$$\dot{\underline{u}} = -E\nabla E \qquad [3.17]$$

Proof:

Consider the gradient of the energy function as follows:

$$\nabla E = \frac{\partial E}{\partial \underline{u}} = \frac{1}{E}(WG - I)\dot{\underline{u}} \qquad [3.18]$$

Since G ≅ **0**, Multiply both side of the above equation by E, we get theorem 4.

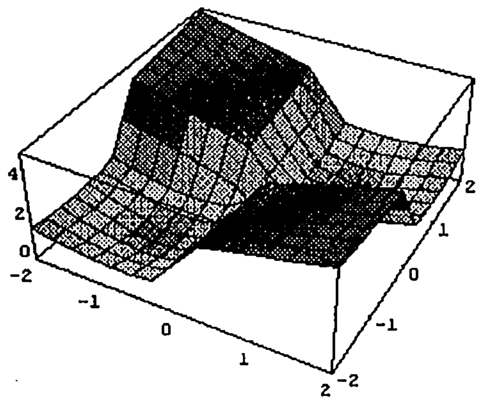
Theorem 4 shows that the energy space determines the convergence trajectory of a CAM. The minima of the energy function correspond to the stable states of the CAM. A plot of an energy space can show many important properties of the CAM. These properties are 1) the number of minima constructed in a recurrent layer, 2) the location of these minima, 3) the watershed (discriminant) of convergence, 4) the convergence direction, 5) the convergence destiny and 6) the probability of converging to one destiny. This fundamental study draws some important properties of the CAM and makes visualizing them possible.

Example: Now consider the net with weight matrix and bias as follows:

$$W = \begin{bmatrix} -0.8 & 1.6 \\ 0.75 & 0.25 \end{bmatrix}$$

$$\underline{b} = \{0.0 \quad 0.0\}^T$$

The energy function of the memory is plotted in the following figures.



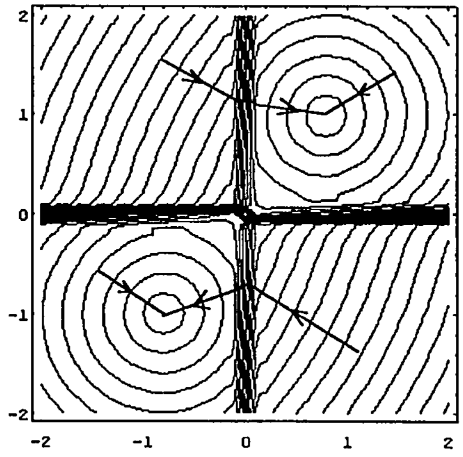


Figure 3.1
3-D Plot of E($\underline{u}$)
x axis is u_1 and y axis is u_2

Figure 3.2
Contour Plot of E($\underline{u}$)
x axis is u_1 and y axis is u_2

The energy function shows that there are two minima (attractors) located in the first quadrant and third quadrant respectively. The watersheds of convergence are close to the x axis. The convergence direction is perpendicular to the contour lines. The convergence destinies are (0.8, 1.0) and (-0.8, -1.0). The complement of the exemplar is also an attractor, due to the symmetry of the activitation function and to $\underline{b}$=0. Four convergence trajectories are drawn in the contour plot.

3.4 Analysis of Learning Algorithms

In this section, the CAM model proposed by Sudharsanan and Sundareshan is studied. The analysis shall focus on the effectiveness of different learning methods. For detailed description of the CAM model and BEP algorithms proposed by Sudharsanan and Sundareshan, see section 2.1.

3.4.1 Backwards Error Propagation Algorithm

The pattern storing procedure of the CAM model proposed by Sudharsanan and Sundareshan (see section 2.2) is based on the BEP algorithm. By minimizing the error, BEP drives the minima (attractors) towards the exemplars. The following example shows the effectiveness and limitation of the BEP algorithm.

Example: The problem of storing three vectors, $\underline{e}_1=\{1.7, -1.5, 1.7, -1.9, -1.5\}^T$, $\underline{e}_2=\{-1.7, -1.1, 1.3, 1.4, -1.2\}^T$ and $\underline{e}_3=\{1.4, 1.1, 1.8, -1.2, 1.7\}^T$ in a network comprising 5 PEs described by equation 2.6 is considered. The initial weight matrix and bias are selected to be $W=0.5I$, $\underline{b}=\{0.1, 0.1, ..., 0.1\}^T$, which are the same as those in Sudharsanan's dissertation. (1990)

The identical pattern storing (learning) procedure is used: The learning constant is selected to be 0.9, the same as that Sudharsanan used. After the net is trained 50 times by the BEP algorithm, the weight matrix and bias are converged to their converged values as follows and the error defined in equation 2.7 reduces to zero. Thus, the three exemplars are stabilized in the network.

$$W = \begin{bmatrix} 1.11 & -0.075 & -0.124 & -0.605 & -0.075 \\ -0.10 & 0.905 & -0.050 & 0.099 & 0.405 \\ 0.101 & 0.025 & 0.975 & -0.099 & 0.025 \\ -0.581 & 0.175 & -0.0002 & 1.08 & 0.175 \\ -0.075 & 0.555 & 0.075 & 0.075 & 1.06 \end{bmatrix}$$

$$\underline{b} = \{-0.025,\ 0.049,\ 0.582,\ 0.099,\ 0.175\}^T$$

However, if we check the existing attractors over the entire vector space, we find there are a total of 25 attractors, three of them are exemplars and the other 22 are spurious memories!

In conclusion, the simulation shows that the BEP algorithm stabilizes continuous exemplars effectively, but has no control of spurious memories. The BEP algorithm is not designed to eliminate spurious memories.

3.4.2 Exterior Product Formulation

There are not as many spurious memories in a BAM formed by EPF as those in a CAM trained by BEP algorithm. The EPF of pattern storing plays an important role.

The EPF storing mechanics is based on the orthogonality of exemplars. If the exemplars are nearly orthogonal to each other, i.e., $\cos(\theta_i) \approx 0$, the exemplars will be stabilized. Those stabilized exemplars become attractors which attract states in the adjacent area. In other words, EPF destabilizes states in the area adjacent to exemplars. As a consequence, it has the ability to eliminate spurious memories.

However, directly using EPF in CAM sometimes does not work very well. The problem arises from the variation in magnitude of patterns. Binary patterns are of the same magnitude, while the magnitudes of continuous patterns are usually quite different. The EPF has the tendency to stabilize the patterns of larger magnitudes. Exemplars of smaller magnitude are likely to be ignored. Our desired learning method should store continuous patterns without bias on their magnitudes.

3.4.3 Normalized Exterior-Product Formulation

The Normalized Exterior-Product Formulation (NEPF) of the weight matrix is inspired by the idea of taking advantage of the ability of EPF to eliminate spurious memories, and at the same time, overcoming the problem of its bias on the magnitude of patterns.

The NEPF of weight matrix W of dimension N to store M exemplars $\underline{e}_i$ is defined in the following equation.

$$W = \frac{1}{N}\sum_{i=1}^{M} \underline{e}_i g^T(\underline{e}_i) \qquad [3.19]$$

As we can see, the magnitude of $g(\underline{e}_i)/N$ is normalized to 1. Theorem 5 (original) shows where are the attractors in the CAM whose weight matrix is formed by the NEPF. Theorem 6 (original) estimates the error of the NEPF of the weight matrix.

Theorem 5: If the weight matrix of a CAM is initialized by the NEPF and $\underline{b}=0$, and the following condition holds, then there is an attractor $\underline{u}^s$ in the quadrant where $\underline{u}$ locates and the attractor is in the neighborhood of $\underline{\tilde{u}}$, i.e., $\underline{u}^s \cong \underline{\tilde{u}}$.

$\underline{u} * \underline{\tilde{u}} >> \underline{0}$
and $\underline{u}$, $\underline{\tilde{u}}$ are not close to the hyper planes
that separete the hyper quadrants. [3.20]

where $\underline{\tilde{u}} = \sum_{i=1}^{M} e_i \cos(\theta_i)$ and θ_i is the angle between $g(\underline{e}_i)$ and $g(\underline{u})$.

Proof:

Since $\underline{u}$ and $\underline{\tilde{u}}$ are in the same quadrant and not close to the hyper plane that separate the hyper-quadrant, according to the property of the activation function, we have

$$Wg(\underline{\tilde{u}}) \cong Wg(\underline{u}) = \frac{1}{N} \sum_{i=1}^{M} \underline{e}_i g^T(\underline{e}_i) g(\underline{u}) = \sum_{i=1}^{M} \underline{e}_i \cos(\theta_i) = \underline{\tilde{u}} \qquad [3.21]$$

Since $\underline{b}=\underline{\mathbf{0}}$, there must be an attractor $\underline{u}^s$, such that $\underline{u}^s = Wg(\underline{u}^s)$, in the neighborhood of $\underline{\tilde{u}}$.

Theorem 6: In the CAM whose matrix is initialized by the NEPF and $\underline{b}=\underline{\mathbf{0}}$, if there is an attractor $\underline{e}_i^s$ in the quadrant of an exemplar $\underline{e}_i$, then the error between them, $\underline{E}_i = \underline{e}_i^s - \underline{e}_i$ is

$$\underline{E}_i \cong \sum_{j \neq i}^{M} \underline{e}_j \cos(\theta_{ij}) \qquad [3.22]$$

where θ_{ij} is the angle between $g(\underline{e}_i)$ and $g(\underline{e}_j)$.

Proof:

From theorem 5, we have

$$\underline{e}_i^s \cong \sum_{j=1}^{M} \underline{e}_j \cos(\theta_{ij}) = \underline{e}_i + \sum_{j \neq i} \underline{e}_j \cos(\theta_{ij}) \qquad [3.23]$$

Moving $\underline{e}_i$ to the left hand side of the above equation, we get theorem 6.

3.4.4 Hybrid Learning Algorithm

In this learning algorithm, the learning procedure is separated into two steps. In the first step, the bias is initialized to be $\underline{\mathbf{0}}$, and the weight matrix is initialized by the NEPF to construct attractors near the exemplars and to destabilize states in other regions. By doing so, a large number of spurious memories are eliminated. Then, in the second step, the BEP algorithm is used to pull the attractors which are not too far away from exemplars towards the exemplars.

As a demonstrative example, the hybrid learning algorithm is used to handle the same problem as that described in section 3.4.1.

In the first step, the bias and the weight matrix are initialized as follows.

$$W = \begin{bmatrix} 0.954 & 0.277 & 0.279 & -0.953 & 0.278 \\ 0.139 & +0.734 & -0.298 & -0.138 & 0.735 \\ 0.437 & -0.239 & 0.954 & -0.437 & -0.238 \\ -0.894 & -0.138 & -0.339 & 0.894 & -0.139 \\ 0.278 & 0.873 & -0.198 & -0.277 & 0.874 \end{bmatrix}$$

After initialization, there are six attractors in the memory. Three of them are in the quadrant of the exemplars and the other three are their complements, due to the symmetry of the network. The attractors which are in the quadrant of the exemplars, and their errors are listed as follows. These results are consistent with theorem 5 and theorem 6.

$$e_1^s = \{1.623 \quad -1.480 \quad 2.291 \quad -1.840 \quad -1.380\}^T$$
$$E_i = \{-0.077 \quad -0.021 \quad -0.591 \quad -0.060 \quad -0.120\}^T$$
$$e_2^s = \{-2.175 \quad -2.030 \quad 0.541 \quad 1.722 \quad -2.485\}^T$$
$$E_2 = \{-0.475 \quad 0.930 \quad 0.759 \quad -0.321 \quad 1.285\}^T$$
$$e_3^s = \{2.728 \quad 1.441 \quad 1.344 \quad -2.392 \quad 2.093\}^T$$
$$E_3 = \{-1.328 \quad -0.341 \quad 0.456 \quad 1.192 \quad -0.393\}^T$$

Then the net was trained by the BEP algorithm. The same learning constant was used as before. After the net is trained 50 times, the error reduces to zero. After training, the weight matrix and bias are converged to:

$$W = \begin{bmatrix} 0.855 & -0.075 & 0.066 & -0.855 & -0.074 \\ -0.100 & 0.654 & -0.151 & 0.100 & 0.655 \\ 0.100 & 0.024 & 1.254 & -0.100 & 0.026 \\ -0.830 & 0.176 & -0.121 & 0.830 & 0.175 \\ -0.074 & 0.805 & 0.024 & 0.075 & 0.806 \end{bmatrix}$$
$$\underline{b} = \{-0.214 \quad 0.149 \quad 0.304 \quad 0.219 \quad 0.226\}$$

There are eight attractors in the net. Three of them are exemplars and the other five are spurious memories. Notice, as shown in section 3.4.1, that if the net is trained by BEP alone, according to Sundarsanan's method, there are 22 spurious memories. Although this hybrid learning algorithm does not eliminate all the spurious memories, the performance is much improved in terms of reducing spurious memories.

3.5 Summary

Through the above analysis in this chapter, many properties and behaviors of BAMs and CAMs have been more concisely explored. The analysis results are significant in terms of: 1) the stability condition of exemplars and spurious memories in BAMs, 2) the formula of choosing diagonal weights and bias that eliminates spurious memories most effectively in BAMs, 3) the convergence theory of CAMs that have non-zero diagonal weights and non-monotonically increasing activation functions, 4) the energy function that explores the convergence behavior of CAMs, and 5) the hybrid learning algorithm that reduces spurious memories effectively in CAMs.

CHAPTER 4

APPLICATION OF STOCHASTIC ACTIVITY NETWORKS TO THE EVALUATION OF SENSOR VALIDATION SYSTEMS

4.1 Sensor Validation Systems and their performability variables

4.1.1 Sensor Validation System

An advanced signal system usually has a sensor system, which provides primary measurements from a plant, and a validation system, which validates sensor measurements. There must be some degree of redundancy in these measurements, on which validation is based.

The status of a sensor can be normal or abnormal. The latter case implies a failure occurrence. The failure of a sensor can be classified into repairable and irreparable based on its repairability, or into recoverable and unrecoverable based on its recoverability, or into detectable or undetectable based on its detectability. If a failure is detectable, it can further be classified into isolable or unisolable based on its isolability.

A sensor validation system (SVS) takes the sensor measurements as its input, checks the consistency of redundant measurements, and provides the status of the sensor system and the best estimates of sensor measurements.

It has the functions of fault detection, isolation and accommodation(FDIA).

A robust SVS has redundant approaches for its functions. If one approach fails, either due to its intrinsic limitation or the failure of its physical devices, others still work, so that the validation function continues though its performance may degrade. The performance of the validation system depends on the status of not only the sensor system but also the sensor validation system itself. Therefore, recoverability, detectability, and isolability of a sensor usually depends on the configuration and the status of the whole sensor system and its validation system. Usually, the recoverability and the repairability also depend on the status of detection and isolation.

4.1.2 Performability of Sensor Validation System

The performability of a SVS quantifies the performance and effectiveness of the SVS in the presence of faults in the SVS. The performability of a SVS can be statistically measured by the following important performability variables.

1. Average detection time τ_d :

   the average time between a failure and its detection.

2. Average isolation time τ_i :

the average time between the detection and isolation of a failure.

3. Average validation time τ_v

   the average time between a failure and its isolation. $\tau_v = \tau_d + \tau_i$.

The above three performability variables are mainly dependent on the configuration of a SVS. Therefore, they represent the performability of a SVS most concisely.

4. Undetectability P_d:

   the probability that, at any given moment when a failure exists, it is not detected.

5. Unisolability P_i:

   the probability that, at any given moment when a failure exists, it is detected but not isolated.

6. Unvalidatability P_v:

   the probability that, at any given moment when a failure exists, it is not isolated. $P_v = P_d + P_i$.

7. Unrecoverability P_r :

   the probability that, at any given moment, the error of accommodation exceeds the allowed range.

The above four performability variables are dependent on the configuration of both the sensor system and the SVS. Therefore, they represent the performability of a SVS under the inferred failure frequency of the sensor system.

8. Average number of available sensors E_k.

9. Probability distribution of the number of available sensors P[k].

The above two performability variables mainly are dependent on the failure rate and repairing time of the sensor system and the validation time of the SVS. They might more closely relate to the performability of a sensor system. However, they also show the effectiveness of the SVS on sensor maintenance.

4.2 Sensor Validation System of a Pressurizer

4.2.1 Pressurizer

In a pressurized water reactor, the primary coolant is maintained at a pressure (around 2250 psia) greater than the saturation pressure corresponding to the maximum coolant temperature in the reactor. This avoids bulk boiling of the coolant and keeps it in the liquid phase throughout the loop. Because liquids are practically incompressible, small changes of volume, caused by changes in coolant temperature or by unforeseen expansions or contractions in the loop components, can cause severe or oscillatory pressure changes. If the pressure is too high, it may cause eruption of the reactor pressure relief valve. If the pressure is too low, it may cause flashing into steam and consequently melting of fuel elements. These changes may be quite unsafe.

It is necessary to provide a surge chamber that will accommodate coolant volume changes while maintaining pressure within acceptable limits. Such a device is called a pressurizer, which keeps the pressure within operating conditions.

4.2.2 Pressure Sensors in a Pressurizer

The sensor system in the pressurizer is important to control the pressure and to protect the reactor if the pressure exceeds the allowed range.

There are fourteen pressure sensors in a typical pressurizer, 2 of them for Pressure Control (PC), 4 for Pressure Low Trip (PLT), 4 for Pressure High Trip (PHT), and 4 for Supplemental Protection of Pressure High Trip (SPPHT).

The two PC sensors send signals to an automatic control system to control the pressure. Their deviation from each other is checked automatically. An alarm will turn on when the deviation exceeds the limit. If the operator identifies a failure of one of the PC sensors, he will set the control system to access the signal of the other sensor. If both sensors fail, an operator will be assigned to control the pressure manually, using safety sensors.

The four PLT sensors protect the reactor if pressure is less than the low pressure bound. Their operation rules are described as follows. If two of them indicate that the pressure is lower than the low bound at the same time, the control system will trip the reactor. If one sensor's reading is below the low pre-trip bound, its alarm will turn on. If the operator identifies a failure of one PLT sensor, he will set a bypass on it. If the operator identifies a failure of another PLT sensor, he has to trip one of the failed sensors, and then usually will shut the reactor down after six hours if neither failed sensor is repaired and passes a retest. If he identifies a failure of a third PLT sensor, he has no choice but to trip the sensor and as a consequence, trip the reactor.

The four PHT sensors protect the reactor pressure from exceeding the high pressure bound. The four SPPHT sensors are a redundant set of PHT sensors and do exactly the same job. These two groups of sensors have the same operation rules as the PLT sensors, except that their readings are compared to the high pressure bound instead of the low pressure bound if the reactor will be tripped, or compared to the high pre-trip bound if their alarm will turn on.

4.2.3 Validation and Maintenance

The automatic alarm system helps the operator to detect and to identify a failure. However, as we see above, it only works when the limit is exceeded. It may even malfunction by setting a false alarm or by a failure to alarm when it should alarm, due to the failure of itself. Therefore, the operators routinely check all the sensors visually to make sure that all sensors work properly. A failure can also be detected and identified by operator's visual checking, though it usually takes more time than the automatic alarm system.

The failure of a sensor can be either irreparable, which usually occurs in the containment building, or repairable, which usually occurs outside of the containment building. The failure of the alarm system is always repairable. Repair will only be activated when a failure is identified and is repairable.

The information in this section is documented from information provided by Robert L Simmons during a series of interviews. He is a senior engineer of Arizona Public Service Company, Palo Verde Nuclear Generating Station, Shift Engineering Group. He also recommended the following data which are reasonable for a nuclear power plant according to his knowledge. I appreciate his help on this project. The details of how these data are used will be explained in the model description section.

4.3 The SAN Model for Sensor Validation System of a Pressurizer

I partition the sensor system into a signal system and a validation system. The fourteen sensors are viewed as the signal system while the rest, the alarm system and operators, are viewed as the validation system. My objective is to evaluate the performability of the system by using SAN models. The SAN model can usually be divided into four submodels. They are signal submodel, validation submodel, validating submodel and resetting submodel. These submodels will be discussed in detail later.

In this project, I built two SAN models, one for the PC sensor system and the other for the PLT sensor system. Both of the models include validation systems. Since the PHT and SPPHT systems have the same operation rules as the PLT system, replication of the PLT system will model the three sensor systems (PLT, PHT and SPPHT).

4.3.1 SAN Model for PC Sensor Validation

The PC sensor system consists of two sensors, which are the signal system, an alarm and the operators, which are the validation system.

1) The signal submodel

The states of the pressure sensors are quite simple. They are normal, repairable failure and irreparable failure. 10% of failures are irreparable and 90% of failures are repairable. It takes a day in average to repair a sensor failure.

The signal submodel models the dynamic of the signal system, the two pressure sensors in this case, see detailed model in Figure 4.1. The place "SN", which initially has 2 tokens, stores the normal sensors. The time activity "s_fail" represents the failure of a normal sensor. The instantaneous activity "repairability" represents the selection of repairable or irreparable failures. The place "SR" stores the repairable failures and the place SIR stores irreparable failures. The time activity "s_repair" represents the repair of the repairable failures.

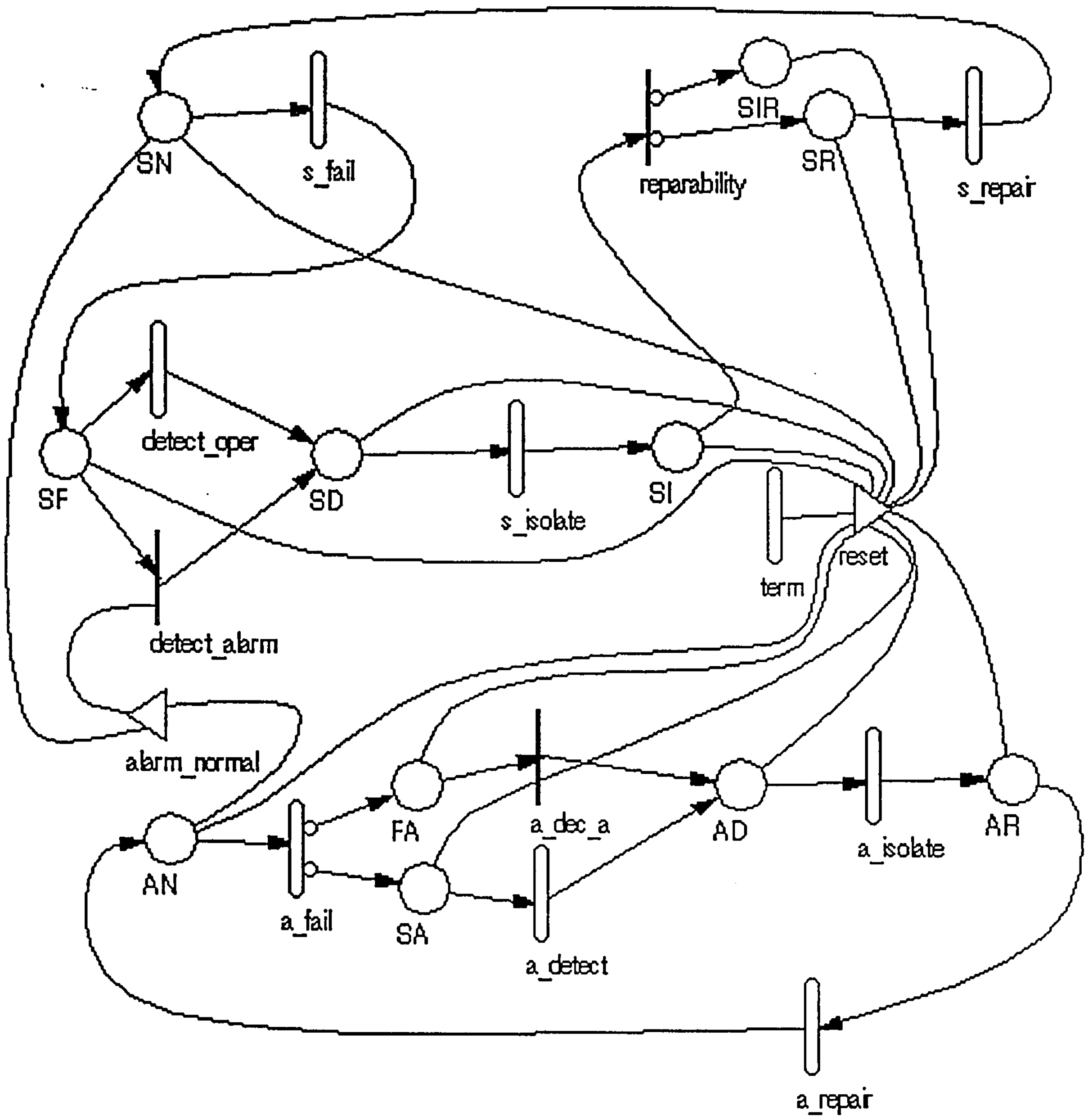


Figure 4.1 the SAN Model of the PC SVS

2) The validation submodel

In the validation system, operators are assumed to be always available and make routine checks. The alarm may be either normal or failed. If it is a false alarm, the alarm failure can be detected immediately. If the alarm 'fails to alarm', then it will be detected after about six hours. After it is detected, operators can isolate the failure in 5 minutes. Then it will be repaired and be ready to be use again after a day on average.

The validation submodel models the validation system, the operators and the alarm in this problem, see the detailed model in Figure 4.1. The place "AN", which initially has one token, stores the normal alarm. The time activity "a_fail" represents the failure of the normal alarm. The place "FA", which is connected to case 1 of activity "al_fail" and is followed by the instantaneous activity "a_dec_a", stores the failure of 'false alarm'. The place "SA", which is connected to case 1 of activity "al_fail" and is followed by the time activity "a_detect", stores the failure of 'failure to alarm'. The time activity "a_detect" represents the operators' detection of the alarm failure. The place AD stores the failures that have been detected but not yet isolated. The time activity "a_isolate" represents the isolation of the alarm failure. The place "AR" stores the isolated failure that is being repaired. The time activity "a_repair" represents the repair of the alarm failure.

3) The validating submodel

According to the status of validation, a failure of the sensors can be at the stage of undetected, of detected but unisolated or of isolated but unrepaired. If there is only one sensor that fails and the alarm is normal, then the failure can be detected immediately by the alarm system. Otherwise, it will be detected by the operator after 6 hours. A failure can be isolated 5 minutes after it is detected.

The validating submodel models the validating procedure of signal failures. The place "SF" stores undetected sensor failures. The time activity "detect_oper" represents operators' detection of failures. The instantaneous activity "detect_alarm" represents the alarm's detection of sensor failures. The input gate "alarm_normal" represents the condition that the alarm will work. The place "SD" stores the detected sensor failures which have not been isolated yet. The time activity "s_isolate" represents the isolation of the failure. The place "SI", which always has zero tokens, is a buffer between activity "s_isolate" and repairability.

4) The resetting submodel.

Since the irreparable failures are absorbing states, the SAN model does not have steady state solutions. Even worse, the server's rates are very stiff, and as a consequence, the transient solver does not work either. Consulting with Dr. Sanders in Department of Electric and Computer Engineering, the University of Arizona, I decided to use a stabilizing technique to solve this problem.

It is reasonable to assume that there are no failures in the system at the beginning of the period. We are only interested in the performance of the system during the operation period, about sixteen months on average. Based on these, I add a time-activity connected to an output gate, which resets the markings of all of the places back to their initial markings. By doing so, I make the SAN model irreducible so that the direct state solver and the iterative state solver work on this model.

4.3.2 SAN Model for PLT Sensor Validation

The PLT sensor system consists of four sensors, which are the signal system, and of operators and four alarms, which are its validation system. The SAN model is shown in Figure 4.2. This model is also applicable to the PHT sensor system and the SPPHT sensor system.

1) The signal submodel

The states of the pressure sensors here are more complicated. They are always detectable to the operators. However, to the alarms, 47% of failures can never be detected by their alarms, 33% of them will degrade on average one month before it is detected, and only 20% of them are detectable to their alarms as soon as they occur. Like the PC model, 10% of failures are irreparable and 90% of failures are repairable. It takes a day to repair a sensor failure.

In addition to the signal submodel of the PC sensor system, the signal submodel of the PLT sensor system has places "SF_aa", "SD_aa", "SD_ab" and "SD_a", instantaneous activities "a_det" and "abrupt" and time activity "degrade", see the detailed model in Figure 4.2. The place "SF_aa", which always has zero tokens, is a buffer between activities "s_fail" and "a_det". The instantaneous activity "a_det" represents the selections of different possible failures that may occur in a sensor. The place "SD_aa" stores the degrading failure and the place "SD_ab", which is followed by the instantaneous activity "abrupt", stores alarm-instantaneously-detectable failure. The time activity "degrade" represents the failure's degrading towards a failure that is detectable by its alarm.

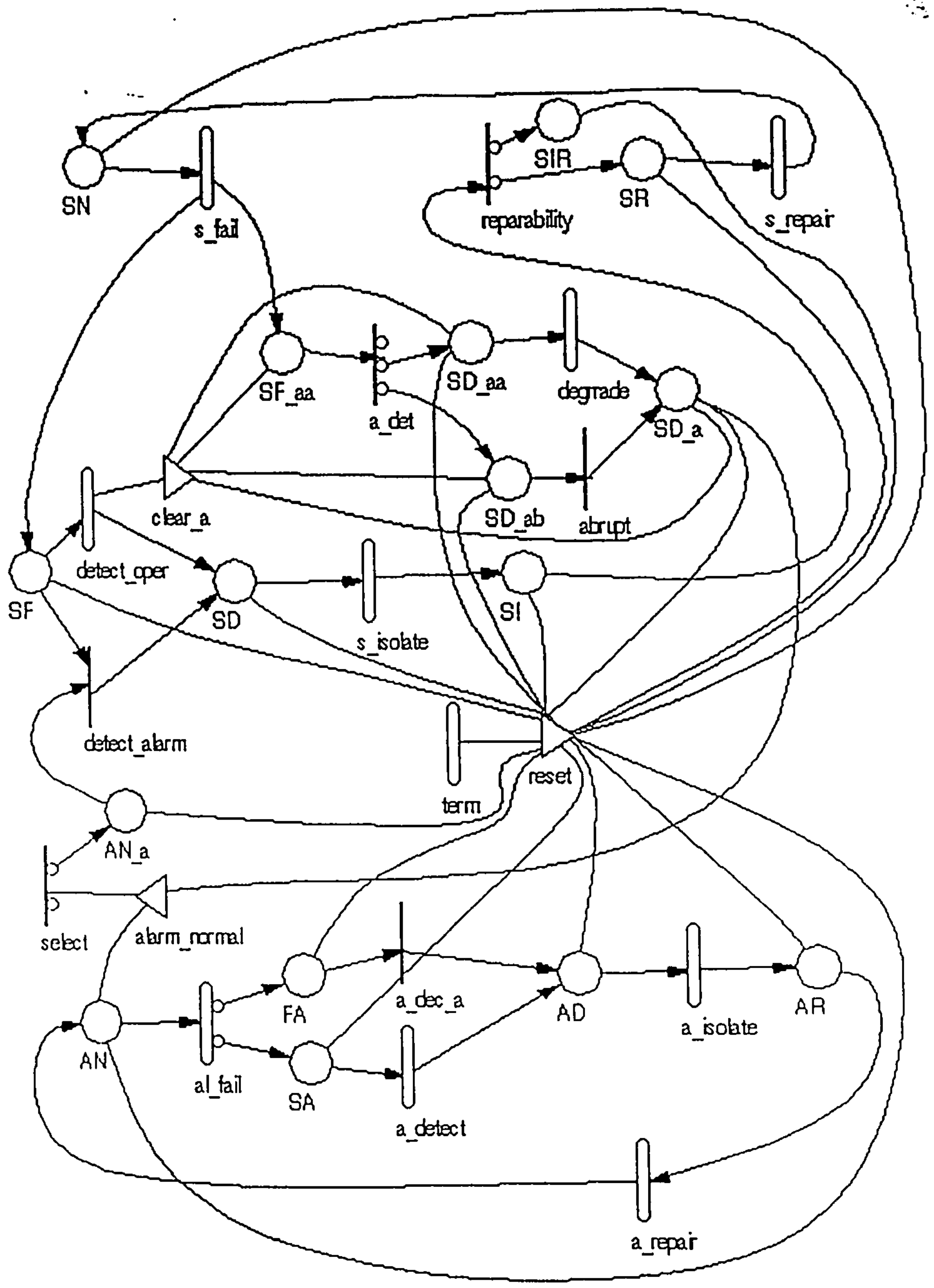


Figure 4.2 the SAN Model of the PLT SVS

2) The validation submodel

The validation system is the same as that of the PC sensor system, except that there are four alarms instead of one alarm. Therefore, the validation submodel is the same of that of the PC model, except the initial token number of place "AN" is four instead of one.

3) The validating submodel

The validating procedure is almost the same as that of the PC model. The difference is in the conditions under which a failure can be detected by its alarm. The possibility that the failure sensor's alarm fails is equal to the number of normal alarms divided by the total number of alarms, 4 in this case. Hence I designed the gate select to assign the possibility that a failed sensor's alarm is normal. The place "AN_a" is a buffer between the activities "select" and "detect_alarm". One more check is that when a degrading sensor is detected by a operator, there is no need for it to be detected by its alarm. Therefore, the gate "clear_a" sets the number of tokens of places "SF_aa", "SD_aa", "SD_ab" and "SD_a" equal to zero.

4) The resetting submodel.

When a reactor is restarted, all of its components are assumed to be in good condition. The reactor may be shut down at the end of its operation period, 16 months. It may also be shut down by the operator if it has been operated for six hours under the condition that there are two sensor failures being isolated. If the operator isolates three sensor failures, he has to trip the reactor.

Again, there is an absorbing state in this model. There is a need to reset the submodel to make the entire model irreducible. The output gate "reset" sets the markings of the places back to their initial markings. The time activity "term" not only represents the normal period, but also acts as the reactor trip and the reactor shut down due to sensor failures. Even better, this makes the subsystem a closed system so that it can be solved analytically.

4.4 Results and Analysis

4.4.1 Performability of Pressure Control Sensor System

In order to evaluate the performability of the PC's SVS, I defined the reward variables in the SAN model as shown in Table 4.1.

Table 4.1 Reward Variables of PC Sensor System

Variable	Predicator	Function	Mean Value
num_NS	1	MARK(SN)	1.753
num_SF	1	MARK(SF)	1.413
num_SR	1	MARK(SR)	4.498E-3
num_SIR	1	MARK(SIR)	2.423E-1
p1_an	MARK(AN)==1	1	0.994
p_SD_SF	MARK(SD)!=0 ||[1] MARK(SF)!=0	1	1.588E-4
p_SD_SF_SN0	(MARK(SF)!=0 || MARK(SD)!=0) &&[2] MARK(SN)==0	1	1.378E-4

From these reward variables, we can calculate some performability variables of interest and importance as follows.

1. The average number of available sensors, E[SN].

E[SN] = E[num_SN] = 1.753

2. The probability distribution of the number of available sensors P[k] is listed in Table 4.2.

P[k] = pdf[num_SN]

[1] In Unix operating system or C language, "||" represents logical "or".

[2] In Unix operating system or C language, "&&" represents logical "and"

Table 4.2
Probability Density Function
of Reward Variable "num_SN"

k	0	1	2
P[k]	0.026797	0.193399	0.779804

3. The true throughput of the system, λ = the number of sensors failures per hour.

$\lambda = E[SN] * \lambda_s = 1.753 * 1.2E-4 = 2.104e-4$

4. The average detection time of the validation system,

$\tau_d = E[num\_SF] / \lambda = 1.4125E-4 / 2.104E-4 = 0.6715$ (hour)

5. The average isolation time of the validation system, τ_i.

$\tau_i = E[num\_SD] / \lambda = 0.08333$ (hour) = 5 (minute)

6. The average validation time of the validation system, τ_v.

$\tau_v = \tau_d + \tau_i = 0.6715 + 0.08333 = 0.7548$ (hour)

7. The undetectability, Pd.

$P_d = 1 - P[num\_SF=0] = 1.413E-4$

8. The unisolability, Pi.

$P_i = 1 - P[num\_SD=0] = 1.7529E-5$

9. The unvalidatibility,

$P_v = P_d + P_i = 1.588E-4$.

10. The unrecoverability, probability that both failed sensors have not been validated, P_r.

$P_r = E[p\_SD\_SF\_SN0] = 1.378E-4$

11. The average repair time, τ_r.

τ_r = E[num_SR] / (0.9*λ) = 23.8 (hour)

12. The average recovery time, $\tau_{rc} = \tau_r + \tau_v$.

τ_{rc} = 23.8 + 0.7548 = 24.55 (hour)

From the above calculations, we find that

1) Some benchmarks, such as τ_i, τ_r are close to our intuitive expectation, therefore, the simulation results are reasonable;

2) The main portion of the recovery time comes from the repair time. The main portion of the validation time comes from the detection time.

However, there are two problems.

The first is that value of P_r is a little bit too high. Consider during a 16-month-operation period, there are 1.378E-4 * 11520 = 1.58 (hour) that the automatic control system is based on a wrong pressure signal!

I suppose that a reasonable operator will check the PC sensor more often, when he knows one of the PC sensors has failed. By modifying the service rate of time activity "detect_oper" to be marking dependent as follows, I calculated the detection time τ_d, unrecoverbility Pr, and time of a wrong signal being used τ_w, corresponding to the operator's checking period τ_c (check the signals every τ_c), when he knows that one sensor is failed.

```
if(MARK(SR)!=0 || MARK(SIR)!=0)
  return (checking_rate);
```

```
else
  return (0.17);
```

The results are listed in Table 4.3.

Table 4.3
The Effect of Increasing Checking rate

τ_c	Pr	τ_w (min)	τ_d (min)
12 (hour)	1.38E-4	95.2	40.3
6 (hour)	7.21E-5	49.8	21.5
4 (hour)	4.83E-5	33.4	14.7
2 (hour)	2.51E-5	17.4	8.1
1 (hour)	1.35E-5	9.4	4.8
30 (min)	7.74E-6	5.3	3.2
10 (min)	3.87E-6	2.7	2.1
5 (min)	2.90E-6	2.0	1.8
2 (min)	2.23E-6	1.6	1.6

The second problem is that the value of P[k=0] is too high. Considering a 16-month operation period, the operator has to control the pressurizer pressure manually for 11520 * 0.026797 = 308.7 (hour).

I think that this is because of irreparable failures. The modification for problem 1 does not change the probability distribution of the number of available sensors. Comparing the availability of the alarm system and the sensor system and checking the average queue length both suggest that irreparable failures play an important role in determining the sensor system's availability. In short, the system's availability will increase if the irreparable failures are reduced, that is, if the reliability of the components which are in the containment building is increased.

By adjusting the case probability to 1% irreparable failures and 99% repairable failures, I find the availability of the sensor system increases from 97.32% to 99.953%, that is, the unavailability decreases from 2.68% to 0.0467%. This changes the average time per cycle when manual control is needed to 4.38 hours.

Intuitively, we usually know how to improve a system's performance. However, with a detailed model, we can not only verify whether an approach works, but also calculate how much it will improve the system's performance. Thus modeling may result in facilitation of performance improvement.

4.4.2 Performability of PLT Sensor System

In order to evaluate the performability of the PLT sensor system, I defined the following performance variables in the SAN model. See Table 4.4

Table 4.4
Reward variables of PLT sensor system

Variable	Predicator	Function	Mean Value
num_NS	1	MARK(SN)	3.878
num_SF	1	MARK(SF)	2.184E-3
num_SR	1	MARK(SR)	9.166E-3
num_SIR	1	MARK(SIR)	1.109E-1
num_AN	1	MARK(AN)	0.994
p_udet	MARK(SF)!=0	1	2.183E-3
p_uiso_det	MARK(SD)!=0	1	3.876E-5
p_uiso	MARK(SD)!=0 || MARK(SF)!=0	1	2.221E-3
p_SN_1	MARK(SN)< 2	1	3.580E-7
p_SN_2	MARK(SN)==2 && MARK(SR)+MARK(SIR)<2	1	2.075E-4

From these performance variables, we can calculate some performance variables of interest and importance as follows.

1. The average number of available sensors, E[SN].

E[SN] = E[num_SN] = 3.878

2. The probability distribution of the number of available sensors, P[k]. See Table 4.5.

P[k] = pdf[num_SN]

Table 4.5
Probability Density Function
of Reward Variable "num_SN"

k	0	1	2	3	4
P[k]	0.0	0.0	0.000416	0.121485	0.8781

3. The true throughput of the system, λ = the number of sensor failures per hour.

$\lambda = E[SN] * \lambda_s = 3.878 * 1.2E-4 = 4.654e-4$

4. The average detection time of the validation system, τ_d.

$\tau_d = E[num\_SF] / \lambda = 2.184E-3 / 4.654E-4 = 4.691$ (hour)

5. The average isolation time of the validation system, τ_i.

$\tau_i = E[num\_SD] / \lambda = 0.08333$ (hour) = 5 (minute)

6. The average validation time of the validation system, τ_v.

$\tau_v = \tau_d + \tau_i = 4.691 + 0.08333 = 4.774$ (hour)

7. The undetectability, P_d.

$P_d = E[p\_udet] = 2.183E-3$

8. The unisolability, $P_i = E[p\_uiso\_det] = 3.876E-5$

9. The unvalidatibility,

$P_v = P_d + P_i = E[p\_uiso] = 2.221E-3$

10. The probability that the reactor should be tripped but is not tripped, P_v.

$P_v = E[p\_SN\_1] = 3.580E-7$

11. The average repairing time, τ_r.

τ_r = E[num_SR] / (0.9*λ) = 21.9 (hour)

12. The average recovery time, $\tau_{rc} = \tau_r + \tau_v$.

τ_{rc} = 21.9 + 4.774 = 26.67 (hour)

Although the above values are close to reasonable values, more analysis is necessary in order to understand this system thoroughly.

4.5 Conclusions

In conclusion, this project is a good demonstration of how to apply the SAN model to evaluate the performability of a signal validation system. The SAN model of the PC sensor system and the PLT sensor system have been built. The stabilizing technique is applied to transfer a reducible model to an irreducible model. Some important performance variables have been defined and evaluated. Discussion of the improvement of the performance of the system has been provided. I intended that this piece of work should explore the application possibilities of SAN for nuclear engineering areas and signal validation areas.

CHAPTER 5

CONCLUSIONS

In order to design an advanced sensor validation system (SVS) which is robust and fault-tolerant under faulty conditions, a promising technology which can be applied to SVS has been studied, and a novel approach has been explored and used to evaluate a SVS in a nuclear power plant. The promising technology studied here is an associative memory, one special type of neural network. The novel approach used here is the Stochastic Activity Network (SAN) model.

5.1 Advances in Associative Memories

The first objective of this work is to investigate the application of associative memories to fault-tolerant and robust sensor validation. An Associative memory provides fault-tolerant pattern recognition. Its robustness against disturbed patterns and against failures in the network itself makes it a promising information processing technology for sensor validation. However, this technology will not become mature for the applications to safety systems until significant advances that overcome its major limitations, especially its capacity and spurious memories, are made.

The studies of associative memories yield many results that help us understand them better so that they can be applied to the problems more effectively and appropriately. These results include

1) the stability condition of exemplars and spurious memories in BAMs,

2) the formula of choosing diagonal weights and bias that eliminates spurious memories most effectively in BAMs,

3) the convergence theory of CAMs that have non-zero diagonal weights and non-monotonically increasing activation functions,

4) the energy function that explores the convergence behavior of CAMs, and

5) the hybrid learning algorithm that reduces spurious memories effectively in CAMs.

In BAM, analysis concludes that the exemplars (patterns to be stored) can always be stabilized (correctly stored) in a BAM if non-zero diagonal weights are used. In this case, the capacity is not a problem in terms of destabilized exemplars. The actual problem is that the number of spurious memories may increase until unacceptable, when the number of exemplars increases. The formula for choosing the diagonal weights and bias which minimize spurious memories has been derived.

In CAM, the analysis discovered that the disadvantage of Backwards Error Propagation (BEP) learning is that no control of spurious memories is maintained. An appropriate weight initialization is found necessary. Both theoretical analysis and numerical simulation have shown the effectiveness of the Normalized Exterior Product Formulation (NEPF).

One of the major difficulties of directly applying associative memories to sensor validation systems is that the possible outputs of the sensors span a continuous space, which cannot be represented by a finite number of attractors in a memory. In other words, a memory is only able to remember a finite number of patterns, but not infinite numbers. If the redundancy of sensor signals can be transformed to a finite number of patterns, then the memory is able to store them and then recognize them fault-tolerantly.

5.2 Application of Stochastic Activity Networks

The second objective is to evaluate SVSs. The concept of performability, the ability of a system to perform in the presence of faults, has been introduced. A set of important performability variables have be introduced to substantiate that concept. A concrete example, evaluation of the pressurizer SVS of a PWR, has demonstrated how to apply the SAN model to evaluate the performability of a SVS.

SAN not only provides an effective evaluation tool, it can also be an effective analysis tool. By evaluating the system, it shows the deficiency of the system. Analyzing and testing different variations of the system, it explores the degree of improvement of different approaches.

With the speed and capacity of computers increasing rapidly, I believe that SANs will be able to model larger and larger complicated systems. However, some improvements, such as modulation and customization, are still necessary before its application and dissemination in the nuclear industry. I think that "object oriented" coding is a nice feature that SANs should have. This is needed to aid in the validation of safety critical software.

REFERENCES

Basseville, Michele. (1988) "Detecting Changes in Signals and Systems -- A Survey". Automatica, Vol 24, No 3, Pp 309-326.

Beard, R. V. (1971) "Failure Accommodation in Linear Systems Through Self-Reorganization". Dept. MVT-71-1, Man Vehicle Laboratory, Cambridge, MA.

Clark, R. N., Fosth, D. C. and Walton, V. M. (1975) "Detection Instrument Malfunctions in Control Systems". IEEE Trans. Aerospace Electron, Syst., AES-11, 465-473.

Couvillion J.A. et. al (1991) "Performability Modeling with UltraSAN" IEEE Software September

Desai, M. and Ray, A. (1981) "A Fault Detection and Isolation Methodology". Proc. 20th Conf. on Decision and Control, 1363-1369.

Eryrek E. and Updahaya B.R. (1992) "Sensor Validation in Power Plants Using Adaptive Backpropagation Neural Network". IEEE Nuclear Science Symposium, San Francisco, CA, 15-19.

Frank, P. M. (1988) "Fault Diagnosis on the Basis of Dynamic Process Models". Presented at 12th IMACS World Congress on Scientific Computation, Paris, 18-22 July.

Frank, Paul M. (1990) "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy -- A Survey and Some New Results". Automatica, Nol. 26, No. 3, 459-474.

Guo, T. -H. and Nurre, J. (1991) "Sensor Failure Detection and Recovery by Neural Networks". Report No. NASA TM-104484.

Hopfield, J. J. (1982) "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". Proc. Natl. Acad. Sci. U.S.A., vol. 79, pp 2554-2558.

Hopfield, J. J. (1984) "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-state Neurons" Biophysics, Vol. 81, pp. 3088-3092.

Isermann, R. (1984) "Process Fault Detection Based on Modeling and Estimation Methods -- a Survey". Automatica, 20, 387-404.

Jones, H. L. (1973) "Failure Detection in Linear Systems". Ph.D. Thesis, MIT, Cambridge, MA.

Kligene, N. I. and Telksnys, Tel'ksnys L.A. (1983) "Methods of Detecting Instants of Change of Random Process Properties". Automn Remote Control, 44, 1241-1283.

Korsah K., Damiano B., and Wood R.T. (1992) "Representation of Neutron Noise Data Using Neural Networks". Power Plant Dynamics, Control and Testing Symposium (8th), Knoxville, TN. 27-29 May.

Meany, J.J. (1989) "Improving Convergence in Neural Networks" Ph.D. dissertation, Dept. of Electrical and Computer Engineering, University of Missouri-Columbia.

Meany, J.J. and Pimmel, R.L. (1991) "Associaive Memory Networks with Bias and Nonzero Diagonal Terms". Proceedings of the Artificial Neural Networks in Engineering (ANNIE '91), St. Louis, Missouri, Nov.

Meyer J.E. (1980) "On Evaluating the Performability of Degradable Computing Systems". IEEE Trans. Computers, Aug. pp 720-731.

Mironovski, L. A. (1980) "Functional Diagnosis of Dynamic Systems -- a Survey". Automn Remote Control, 41, 1122-1143.

Mott-J.E. et. al, (1992) "Universal, Fault-tolerant, Non-linear Analytic Network for Modeling and Fault Detection", Power Plant Dynamics, control and Testing Symposium (8th), Knoxville, TN, 27-29 May.

Movaghar A. and Meyer J.E. (1984) "Performability Modeling with Stochastic Activity Networks" Proc. 1984 Real-Time Systems Symp., CS Press, Los Alamitos, Calif.,

Patton, R. J., Frank P. M. and Clark R. N. (Ed) (1989) "Fault Diagnosis in Dynsmic Systems, Theory and Applications". Prentice-Hall, Englewood Cliffs, NJ.

Potter, I. E. and M. C. Sumnam. (1977) "Thresholdless Redundancy Management with Arrays of Skewed Instruments." Integrity in Electronic Flight Control Systems, AGARDOGRAPH-224, 15-25.

Sanders W.H. and Meyer J.E. (1986) "METASAN: A Performability Evaluation Tool Based on Stochastic Activity Networks". Proc., ACM-IEEE Computer Soc. Fall Joint Computer Canf., CS Press, Los Alamitos, Calif., pp 807-816

Sudharsanan S.I. and Sundareshan M. K. (1991) "Training of a Three-Layer Dynamical Recurrent Neural Network for Nonlinear Input-Output Mapping". Proceedings of the 1991 International Joint Conference on Neural Networks (IJCNN-91), Seattle, Washington.

Sudharsanan. S.I. (1990) "Equilibrium Characterization for a Class of Dynamical Neural Networks With Applications to Learning and Synthesis". Ph.D. Dissertation, Dept. of Electrical & Computer Eng., The University of Arizona.

Whiteson R. and Howell J.A. (1992) "Anomaly Detection in an Automated Safeguards System Using Neural Networks". Institute of Nuclear Materials Management (INMM) annual Meeting, Drlando, FL, 19-22 July.

Willsky, A. S. (1976) "A Survey of Design Methods for Failure Detection in Dynamic Systems". Automatica, 12, 601-611.

Yoshizawa, S., Morita, M. and Amari S. (1993) "Capacity of Associative Memory Using a Nonmontonic Neuron Model". Neural Networks, Vol. 6, pp 167-176