Volume 2 ▪ Issue 2 ▪ May 2011

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS (IJAE)

ISSN : 2180-124X

Publication Frequency: 6 Issues / Year

CSC PUBLISHERS
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS (IJAE)

VOLUME 2, ISSUE 2, 2011

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 2180-124X

International Journal of Artificial Intelligence and Expert Systems (IJAE) is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJAE Journal is a part of CSC Publishers
Computer Science Journals
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS (IJAE)

Book: Volume 2, Issue 2, May 2011
Publishing Date: 31-05-2011
ISSN (Online): 2180-124X

IJAE Journal is a part of CSC Publishers
http://www.cscjournals.org

Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2011

EDITORIAL PREFACE

The International Journal of Artificial Intelligence and Expert Systems (IJAE) is an effective medium for interchange of high quality theoretical and applied research in Artificial Intelligence and Expert Systems domain from theoretical research to application development. This is the fourth issue of volume first of IJAE. The Journal is published bi-monthly, with papers being peer reviewed to high international standards. IJAE emphasizes on efficient and effective Artificial Intelligence, and provides a central for a deeper understanding in the discipline by encouraging the quantitative comparison and performance evaluation of the emerging components of Expert Systems. IJAE comprehensively cover the system, processing and application aspects of Artificial Intelligence. Some of the important topics are AI for Service Engineering and Automated Reasoning, Evolutionary and Swarm Algorithms and Expert System Development Stages, Fuzzy Sets and logic and Knowledge-Based Systems, Problem solving Methods Self-Healing and Autonomous Systems etc.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 2, 2011, IJAE appears in more focused issues. Besides normal publications, IJAE intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

IJAE give an opportunity to scientists, researchers, and vendors from different disciplines of Artificial Intelligence to share the ideas, identify problems, investigate relevant issues, share common interests, explore new approaches, and initiate possible collaborative research and system development. This journal is helpful for the researchers and R&D engineers, scientists all those persons who are involve in Artificial Intelligence and Expert Systems in any shape.

Highly professional scholars give their efforts, valuable time, expertise and motivation to IJAE as Editorial board members. All submissions are evaluated by the International Editorial Board. The International Editorial Board ensures that significant developments in image processing from around the world are reflected in the IJAE publications.

IJAE editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJAE. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJAE provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

Editorial Board Members
International Journal of Artificial Intelligence and Expert Systems (IJAE)

EDITORIAL BOARD

TABLE OF CONTENTS

Volume 2, Issue 2, May 2011

Pages

Towards Automated Intrusion Response: A PAMP - Based Approach

Rimiru Richard M *rimirurm@yahoo.com*
College of Information Science and Engineering
Central South University
Changsha, 410083, China

Guanzheng Tan *tgz_csu@yahoo.com.cn*
College of Information Science and Engineering
Central South University
Changsha, 410083, China

Njuki S. N. *wisenjuki@gmail.com*
College of Information Science and Engineering
Central South University
Changsha, 410083, China

Abstract

Most of the current Intrusion Detection Systems have mainly concentrated on detection of intrusions with no mechanisms incorporated to respond to such intrusions. The major problem in automating IDS responses has mainly been because currently IDS experience high false alarms which if automated would introduce denial of service or related problems. In this paper we propose a mechanism that allows for some level of automation of intrusions response. In particular we emphasize that patterns exclusively associated with intrusions should be used as the basis, thereby separating between the network connections that require further processing to establish as to whether they are anomalous. We base our argument on the Human Immune system immune system and as such some biological overview of the same is presented. Finally, we demonstrate that our proposed approach incorporates most of the desired features that have for long been considered advantageous from studies of the immune system.

Keywords: Intrusion Detection System, Artificial Immune Systems, Pathogen Associated Molecular Patterns, Human Immune Systems.

1. INTRODUCTION

As the use of computer systems continues to proliferate so are the threats and other concerns of security against them. Intrusion detection systems (IDS) have been employed to incrementally improve security based on the assumption that a system will not be secure, but that violations of security policy (intrusions) can be detected by monitoring and analyzing system behavior [1], [2].

Though many different ways have been proposed to classify IDS [3,4,5], the more popular classification method is based on the detection method or principles used by the IDS resulting in two basic classes of: Misuse-based IDS, aimed at examining the network and system activities for known intrusions (also known as signatures hence also referred to as signature detection method) and Anomaly-based IDS, which assumes the nature of intrusion, is unknown, but that intrusion will result in a significant deviation in behavior from that normally observed in the system, thus requires a profile of normal network and system behavior be constructed. Additionally, IDS can also be either Host-based (HIDS) or Network-based (NIDS) depending on the activities monitored. Much of the work done in IDS to date has concentrated on detection mechanisms with little efforts seen towards response mechanisms as such high number of alerts (both of true attacks - True Positives (TP) and false alarms – False Positives (FP)) are produced

and human intervention is normally required to deal with the alerts. Largely we can attribute this to the fact that most of the anomaly-based IDS have no mechanisms of associating an alert to the cause i.e. they just indicate that an intrusion may have occurred but in no way do they indicate its nature. In this paper we propose a mechanism akin to the one used by the Human Immune System (HIS) that can be used to allow for automated response thereby addressing the issue of high number of alerts produced by an IDS. An IDS with an automated response is also desirable for it can protect a system from an ongoing intrusion. Since automated response allows dealing with a large number of attacks early enough, the resulting system is lightweight in nature which makes it desirable if used in a real-time environment.

The remainder of this paper is organized as follows. Section 2 presents background on the immunological inspiration for our proposed method and discusses related work. Section 3 then discusses our proposed mechanism before concluding and discussing the way forward in section 4.

2. BACKGROUND

This section provides an overview of the immunological concepts that inspire our proposed mechanism. An overview of Artificial Immune Systems (AIS), algorithms inspired by the immune system, as applied to the problem of intrusion detection is also presented to help root for our proposed model as well as help highlight the trend so far in the related work section. It is then concluded with a discussion which allows us to relate the immune system mechanisms and the work done so far.

2.1 Overview of the Human Immune System

We in no way claim to give a comprehensive coverage of immunology, but try to give enough to allow a reader understand general concepts and terminologies of immunology used within our proposed mechanism and the related work presented thereafter. Most of our material on immunology is borrowed from [6] unless where specified otherwise.

The human body is an amazingly complex organism which can be viewed at different levels of abstraction, with cells as the most basic structural and functional units of biological organisms [7]. The body itself exists in a world which is full of microorganisms. It is susceptible to attacks from many of these microorganisms as they find the body a rich resource of energy and material. If left unchecked, they would inevitably lead to the destruction of the body and death would eventually occur. Damage to the body is called pathology, and the damaging agent, such as bacteria or virus, a pathogen. Functionally, the human immune system is able to detect and remove many of these pathogens from the body and maintain the body in a healthy state. The primary function of the immune system therefore is to fight infection [8].

The architecture of the immune system is multi-layered [2, 9, 10] with defenses on several levels. Most elementary is the skin whose epithelial surfaces form a physical barrier that is very impermeable to most infectious agents. Thus, the skin acts as our first layer of defense against invading organisms. Also included are the chemical and biological factors (physiological conditions), which provide inappropriate living conditions for foreign organisms [9, 10]. Once pathogens have entered the body, they are dealt with by the innate immune system and by the acquired or adaptive immune system. Both systems consist of a multitude of cells and molecules that interact in a complex manner to detect and eliminate pathogens and its these two systems that are mainly considered as comprising the immune system.

The purpose of the immune system is not only to protect the body from pathogens which may be either intracellular (inside or within a cell) mainly viruses, some bacteria and parasites or extracellular (found outside of a cell) which includes most bacteria, fungi and parasites, but also eliminate modified or altered “self” cells. Both the innate and the adaptive systems have the cellular and humoral components that aid in elimination of pathogens and/ or transformed “self” cells in distinct ways. The innate immune system is our first line of defense against invading organisms (as such both the skin and physiological conditions discussed earlier are considered

part of innate) while the adaptive immune system acts as a second line of defense and also affords protection against re-exposure to the same pathogen.

The innate immune system is characterized as having three roles: host defence in the early stages of infection through nonspecific recognition of a pathogen, induction of the adaptive immune response, and determination of the type of adaptive response. The main characteristics of adaptive immunity are specific recognition of pathogen (i.e. adaptive immune system are specific and reacts only with the organism that induced the response - antigen) leading to the generation of pathogen specific long-term memory [7].

The receptors of innate system cells are entirely germline-encoded. In other words their structure is determined by the genome of the cell and has a fixed, genetically-determined specificity [8]. They recognize a genetically-determined set of molecules under evolutionary pressure. One key group of innate receptors is the Pattern Recognition Receptor (PRR) superfamily which recognizes evolutionary-conserved Pathogen-Associate Molecular Patterns (PAMPs), with Toll-Like Receptors (TLRs) identified as the most important class [11]. PRRs do not recognize a specific feature of a specific pathogen as variable-region adaptive immune systems receptors do, but instead recognize common features or products of an entire class of pathogens as such innate immune system receptors are termed non-specific, while adaptive immune system receptors are termed specific [7].

So what happens after pathogens have penetrated through the tissues (overcoming skin and physiological conditions barrier)? Another innate defense mechanism comes into play, namely acute inflammation. Humoral factors play an important role in inflammation, which is characterized by edema (swelling as a result of excessive accumulation of serum in tissue spaces or a body cavity) and the recruitment of phagocytic cells (cells involved in phagocytosis). These humoral factors are found in serum or are formed at the site of infection. The complement system is the major humoral innate defense mechanism. Once activated (Complement activation pathways is beyond the scope of this work.) complement can lyse bacteria, lead to increased vascular permeability hence allowing a large number of circulating phagocytic cells to be recruited to the site of infection, as well as helps with opsonization of bacteria. Opsonization refers to the coating of bacteria with complement enabling the bacteria to be detected by macrophages. Coagulation System (process by which blood forms solid clots) is also considered part of the innate humoral mechanisms and tends to lyse bacteria, increase vascular permeability and act as chemotactic (cell movement) agents for phagocytic cells once activated.

As noted, part of the inflammatory response is the recruitment of PolyMorphoNuclear (PMN) cells and macrophages to sites of infection. These cells are the main line of defense in the non-specific immune system forming the cellular component of the innate system. They include neutrophils that phagocytose invading organisms and kill them intracellularly, tissue macrophages and newly recruited monocytes, which differentiate into macrophages, also function in phagocytosis and intracellular killing of microorganisms and eosonophils that have proteins in granules effective in killing certain parasites. In addition, macrophages are capable of extracellular killing of infected or altered self target cells. Also considered part of the innate cellular component are the Natural killer (NK) and lymphokine activated killer (LAK) cells - NK and LAK cells can nonspecifically kill virus infected and tumor cells. These cells are not part of the inflammatory response though, but they are important in nonspecific immunity to viral infections and tumor surveillance.

So how does the recruitment of phagocytic cells occur and how do they identify the invaders? Circulating PMNs and monocytes respond to danger signals generated at the site of an infection. Danger signals include N-formyl-methionine containing peptides released by bacteria, clotting system peptides, complement products and also cytokines released from tissue macrophages that have encountered bacteria in tissue. Some of the danger signals stimulate endothelial cells near the site of the infection to express cell adhesion molecules such as Inter-Cellular Adhesion Molecule 1 (ICAM-1) and selectins which bind to components on the surface of phagocytic cells and cause the phagocytes to adhere to the endothelium. Vasodilators produced at the site of

infection cause the junctions between endothelial cells to loosen and the phagocytes then cross the endothelial barrier by “squeezing” between the endothelial cells in a process called diapedesis allowing for increased permeability. Once in the tissue spaces some of the danger signals, chemokines, attract phagocytes to the infection site by chemotaxis (movement toward an increasing chemical gradient). The danger signals also activate the phagocytes, which results in increased phagocytosis and intracellular killing of the invading organisms. Once at the infection site phagocytic cells have a variety of receptors on their cell membranes through which infectious agents bind to the cells. These include; Complement receptors - Phagocytic cells have a receptor for the 3rd component of complement, C3b. Binding of C3b-coated bacteria to this receptor also results in enhanced phagocytosis and increased metabolic activity of phagocytes, Toll-like receptors - Phagocytes have a variety of Toll-like receptors (Pattern Recognition Receptors or PRRs) which recognize broad molecular patterns called PAMPs (pathogen associated molecular patterns) on infectious agents. Binding of infectious agents via Toll-like receptors results in phagocytosis and the release of inflammatory cytokines (IL-1, TNF-alpha and IL-6) by the phagocytes. These cytokines have the effect of inducing fever, activating other macrophages, recruitment of PMNs as well as activating T cells. In cases where bacteria may have had prior interaction with an antibody (components of adaptive immune system) then Fc receptors may also be used. Binding of antibody-coated bacteria to Fc receptors results in enhanced phagocytosis and activation of the metabolic activity of phagocytes.

How does the adaptive immune system come into play then? A specialized subset of cells called antigen presenting cells (APCs) are a heterogenous population of leukocytes that play an important role in innate immunity and also act as a link to the adaptive immune system by participating in the activation of helper T cells (Th cells), cellular components of the adaptive system. Antigen presentation involves processes that occur within a cell that result in fragmentation (proteolysis) of proteins, association of the fragments with the major histocompatibility complex (MHC) molecules, and expression of the peptide-MHC molecules at the cell surface of the cell where they can be recognized by the T cell receptor on a T cell. These cells include dendritic cells (DCs) and macrophages and are characterised by the expression of a cell surface molecule encoded by genes in the MHC, referred to as class II MHC molecules. B lymphocytes, the humoral component of the adaptive system, also express class II MHC molecules and so they also function as APCs.

Basically, MHC molecules display fragments of processed proteins (whether self or non self) on the cell surface. Generally two classes of MHC molecules exist: Class I and Class II. Class I molecules are expressed on all nucleated cells and present fragments from endogenous (intracellular) proteins whilst Class II are mostly found on APCs and present fragments from exogenous (extracellular) proteins.

Dendritic cells are considered the most effective APCs as they can present antigens to naive (virgin) T cells and have the ability to present antigens in association with either class I or class II MHC molecules with class II being the most common. On the other hand macrophages and B cells are considered effective in activating memory cells and present antigen associated with only class II MHC. Once activated, DCs are efficient stimulators of T cells (hence the adaptive immune system) through their presentation of MHC-peptides complexes.

As noted earlier the adaptive immune system has two major components with T cells and B cells constituting the cellular and humoral components respectively. Both types of cells originate from the lymphoid progenitor, with T cells migrating to the thymus and B cells to the bone marrow for maturation. Its while in the thymus where T cells undergo what is considered “thymic education”. First, their receptors undergo rearrangement and unproductive rearrangement leads to apoptosis (programmed cell death). Secondly, successful cells then undergo positive selection where those whose receptors recognize self MHC are selected while the rest undergo apoptosis. This is then followed by negative selection where those cells that react with self-peptides are eliminated. The outcome is naïve - T cells that are MHC- restricted, ensuring that they will recognize a peptide antigen only when it is bound to a particular MHC molecule (self - MHC) and naïve - T cells that

do not react with self-peptides, which would otherwise lead to autoimmune diseases. Depending on the MHC molecules they are exposed to more, the resulting T cells differentiate into either T lymphocytes capable of recognizing antigen presented with class I MHC molecules or that presented in class II MHC molecules context. They later mature when their antigen receptors bind with antigens presented by the DCs and they receive a costimulatory signal from the DC. These two conditions must happen to activate a naïve lymphocyte. Once activated, T lymphocytes first undergo a period of proliferation, known as clonal expansion, which results in a large population of T lymphocytes which all possess antigen receptors with the same specificity. The clones then differentiate into either memory T lymphocytes or effector T lymphocytes. Those capable of recognizing antigens presented in class I MHC context are referred to as cytotoxic T cells (CTLs) and the other group of class II MHC context being refered to as helper T cells (Th). It is these latter class that as we noted earlier that is primarily activated by the APCs during antigen presentation. Th cells further differentiate into Th1 and Th2 cells with DCs producing IL-12 priming Th cells to differentiate along the Th1 pathway while activated T cells and other cells produce IL-4 promoting Th2 pathway. Th1 cells produce IFN-γ and IL-2 to primarily mediate cellular immunity (CTLs) though they are also known to activate macrophages and help in differentiating NK cells to LAK cells. Th2 cells produce IL-4, IL-5, IL-6, IL-10 and IL-13 and mediate humoral immunity (B cells) in effect causing them produce antibodies. CTLs are responsible for the killing of intracellular pathogens in tissue cells by inducing apoptosis whilst B cells help in elimination of extracellular pathogens by neutralization, opsonization and/or complement activation.

When the pathogens have been eliminated mechanisms within the immune system have to help to contain any more inflammatory response. Regulatory T cells (T – reg) are known to help with these. Much of their details remain unclear but they are known to produce IL-10 and TGF – beta that inhibit DC and T cells activation respectively [12]. IL-10 is also produced by Th2 cells and inhibits production of IFN-γ by Th1 cells, which shifts immune responses toward a Th2 type. It also inhibits cytokine production by activated macrophages and the expression of class II MHC and costimulatory molecules on macrophages, resulting in a dampening of immune responses.

2.2 Related Work

Indeed so much literature exists of work that has applied immune system methods to problems in intrusion detection. Detailed reviews exist with different emphasis, for example work reported in [13, 14, 15] covers use of AIS – based algorithms to wide areas of application, that of [16, 17, 18] view AIS as one of the many approaches in soft computing. The approach used in [19] looks at both computer programs used to simulate the natural immune system and those inspired by natural immune system to solve practical engineering problems. However, work reported in [20] is more focused to research mainly in the use of AIS in IDS and can be taken to be an extension of the work previously reported in [21]. We thus just highlight some of the developments here.

Kim *et al* [20] classified the existing works of use if AIS in IDS into 3 major groups: Methods based on conventional algorithms, which were one of the earliest attempts at exploiting features of the Human Immune System (HIS), for example , a virus detection system developed by Kephart *et al* [22] at the IBM research centre. They identified some traits of the HIS that make it attractive for virus detection and implemented them using established algorithms. Dasgupta [23] proposed an alternative immunity-based IDS framework that applied a multi – agent architecture. This architecture followed the multi-level detection feature of the HIS. Other works in this category as reported in [20] include ADENOIDS that attempted to identify and understand useful processes of the HIS, but did not attempt to implement the processes using the mechanism of the HIS.

The second approach based its work on the negative selection paradigm of the adaptive immune system. Almost around the same time that Kephart *et al* were doing their work, Forrest *et al* [25] identified the possibility of using negative selection in the T-cell maturation process for virus detection or change detection. It was actually this work that lay foundation for most of the work done on AIS with relation to IDS by the Adaptive Computation Group at the University of New Mexico headed by Stephanie Forrest. They, Forrest *et al*. In [25] then made an attempt to define

"self" for a computer process where self was treated synonymously with normal behavior. The goal was to try and protect executing programs. It was this work that [1] later extended and also introduced some form of matching rules. This was then followed in [26] by suggestion of more features that could be borrowed from the HIS to construct robust computer systems. The HIS features identified by this work were: multilayered protection; highly distributed detectors, diversity of detection ability across individuals; inexact matching strategies, sensitivity to most new foreign patterns, disposability, automated response and self repair, no secure components, and dynamic coverage. It is these features that Somayaji *et al*. in [2] refers to as the organizing principles that should guide the design of computer security systems. Having successfully experimented with and implemented some Host-based mechanisms for intrusion detection, the Forrest group went on to design an AIS to protect computer networks based on immunological concepts. Normally occurring TCP/IP (Transmission Control Protocol over Internet Protocol) connections were considered as "self" and all the others formed the set of non self patterns [27, 19]. Based on most of the design aspects in [27], Hofmeyr and Forrest then proceeded to develop a general architecture for AIS in [28] which they called ARTIS. They indicated that some of the features that most of artificial systems lacked by then were; robustness, adaptability and autonomity. Its these architecture that they based their LYSIS system which they revisited in [29] to highlight need for the various concepts used. Due to the large set of non self patterns, more and more research was interested in development of detectors that could cover the non self space better. Hofmeyr's *et al*. had introduced permutation mask in [27] while Dasgupta *et al*. [30] introduced use of hypercubes. Balthrop *et al*. in [31] focused on generalization of detectors of LYSIS.

Besides the Forrest and Dasgupta groups, some others were also using negative selection methods to develop detectors; examples include Harmer *et al*. works reported in [32], who implemented a self-adaptive distributed agent-based defense immune system based on HIS concepts, within a hierarchical layered architecture used to provide system management aspects. In [33] though, using an evolutionary programming approach to create antibodies represented as Finite State Transducers (FST), the authors tried to extend the work reported in [32] by trying to detect modified or stealthy versions of existing attacks. They introduced the concept of "vaccination", which injected existing knowledge about an attack. In a way they were still concerned about how to generalize the detectors to be able to detect closely related attacks.

Other works found in the literature include that of Tao [34] who proposed the use of a dynamic evolution model of self that keeps updating self used for tolerization at time t by using the set of self introduced at t-1 that did not react to existing detectors. This was to try and introduce some adaptation to the set of detectors to the changing set of self. More recently Luther *et al*. [35] have developed a cooperative AIS framework for IDS where the concept of collaborative detection is used. A peer-to-peer (P2P) infrastructure is used to handle the tasks of look-up, maintenance and communication between detectors. Basically it requires that when a host detects an anomaly it updates its neighbors as well as sends them the actual detectors associated with the alarm. A server is used initially to setup a peer list.

Due to the problems associated with negative selection approach majorly scalability and generalization of detectors, different methods were being sought for intrusion detection which resulted in what is now called the "danger theory" approach. In their review Kim *et al* [20] note that the danger model had been considered for development for AIS-based IDS by Burgess as early as 1998. Burgess is reported to have developed a system called Cfengine based on the danger model concept. Burgess put the emphasis of AIS on an autonomous and distributed feedback and healing mechanism, triggered when a small amount of damage could be detected at an initial attacking stage. However, it's the work reported in [36] that Aickelin et al. presented the first in-depth discussion on the application of danger theory (which basically argues that the immune system actually does not use a self/nonself model to protect body but responds to danger signals produced by necrotic cells) to intrusion detection and the possibility of combining research from wet and computer laboratory results. As a result of this notion, the Danger Project [37] was proposed and subsequently instigated as an interdisciplinary research project, involving both a team of practical immunologists and biologically inspired computer scientists and this work

has been the source of most work reported on use of danger model in IDS to-date. As part of these work, they have developed design principles reported in [7, 38] and built a general system and API named *libtissue* within which a number of different artificial immune system algorithms can be implemented [7, 39]. Subsequently, they have implemented a range of Danger Theory inspired algorithms, for example, two cell algorithms which were "expanded" to TLR by implementing the aspects of compartmentalization and complex cell differential pathways has been reported as used for process anomaly detection in [40] and [41]. The most advanced ones [37] being the toll-like receptor or TLR algorithm [7] which is modeled around the interactions of DC and T cells and the DCA [42] which performs correlation of context, derived from the processing of a set of input signals, with antigen - the data to be correlated. This is based on the premise that 'suspects' in the form of antigen can be paired with 'evidence' in the form of signals to identify potential sources of anomaly or intrusion. They have had some success in cases where they have been experimented with as reported is [14, 42]. Kim et al have also reported work involving extensions of the DCA in [43, 44]. Other works have also been reported in [43] which model a variant of a DC - T cells interaction with response given in form of alerts.

2.3 Discussion

From the foregoing it is clear that the human immune system is able to protect the body from infections through an intricate interaction of both its innate and adaptive subsystems. The innate system plays a major role in the recognition of pathogens through the binding of PAMPs and PRRs. Once a pathogen is detected, the innate cells mount an immediate response trying to fight the invading pathogen. Arguably, the innate system has a very limited ability to down regulate itself as such may requires the adaptive system to help in the same. Th2 productions of IL-10 and T-reg cells have indeed been shown to help with that functionality. The innate system is also not known to have any memory of pathogens encountered in the past, a property displayed by the cells of the adaptive system in being able to mount a faster response for previously encountered pathogen using memory lymphocytes, what is referred to as secondary response, in contrast to a primary response mounted for an initial exposure that has some lag time. The cells of the adaptive system undergo high mutations (somatic hypermutation of B cells) and/or rearrangement to help keep pace with mutating viruses, something that the innate cells do not. So the two subsystems play complimentary roles to each other, though with some redundancy like in the role of Innate NK cells and CTLs of the adaptive, both used to eliminate infected cells.

Several researchers have studied the human immune system and identified several distinguishing features that provide important clues about how to build information processing systems. Some of these works are reported in [2, 32, 45, 46, 26, 27]. However, some AIS features (derived from HIS) that would be advantageous in the design and development of novel IDS and intuitively provide some good reference to any researcher working on IDS are summarized in [20] as:

- Distributed: a distributed IDS supports robustness, configurability, extendibility and scalability. It is robust since the failure of one local intrusion detection process does not cripple the overall IDS.
- Self-organized: A self-organizing ID provides adaptability and global analysis. Without external management or maintenance, a self-organizing IDS automatically detects intrusion signatures which are previously unknown and/or distributed, and eliminates and/or repairs compromised components.
- Lightweight: A lightweight IDS supports efficiency and dynamic features. A lightweight IDS does not impose a large overhead on a system or place a heavy burden on CPU and I/O. It places minimal work on each component of the IDS.
- Multi-layered: a multi-layered IDS increases robustness. The failure of one layer defence does not necessarily allow an entire system to be compromised. While a distributed IDS allocates intrusion detection processes across several hosts, a multi-layered IDS places different levels of sensors at one monitoring place.
- Diverse: A diverse IDS provides robustness. A variety of different intrusion detection processes spread across hosts will slow an attack that has successfully compromised one or more hosts.

- Disposable: A disposable IDS increases robustness, extendibility and configurability. A disposable IDS does not depend on any single component. Any component can be easily and automatically replaced with other components.

From our review it was clear that to-date LISYS remains the most advanced AIS-based network intrusion detection system and falls short of most of the requirements above. Glickman *et al*. [46] had seen the need to try and integrate it with an analog of the innate immune system to provide it with some signature-based capabilities. Similar suggestions had been aired earlier in the survey reported in [5], that there was lack of detectors in the signature/self-learning class, which arguably could combine the benefits of the two classes of; detection efficiency with automated "extraction" of signatures.

Most of the efforts in developing AIS-based IDS have been through the use of self-non self model of the immune system and as such generates detectors using negative selection. However, negative selection has been shown to have scaling and coverage problems as well outlined in [20]. Negative selection had also been criticized in [47] for its use for one class (self) for training and both classes (self and non-self) whilst testing which they claimed led to high false positives. Further, adoption of more sophisticated and realistic contemporary models as opposed to self/non self for AISs to prove successful at solving hard real world problems have been suggested in [48]. Similar sentiments are echoed in [49] where he notes that innate immune system has been largely ignored. Hart *et al*. [15] indicated that they suspected that the true value of the immune metaphor will be only revealed in systems which exploit the full richness of the natural immune system which is gained through the synergistic interaction between the innate and adaptive immune systems.

3. PROPOSED MECHANISM

Most arguments presented seem to point to the need to incorporate the aspects of the innate immune system into the development of effective IDS. The danger model achieved part of this in use of DCs to correlate signals (PAMPs, danger signal, safe signal and inflammation) to determine the context (normal or anomalous) of some given inputs into a system. They use the PAMPs as part of the signals correlated to determine the context, which promotes an anomalous context. We propose however, that PAMPs should infact be used to detect purely anomalous situations (attacks) which then should trigger an immediate automated (innate) response. If indeed a pattern is considered to be a PAMP, it signifies that normal occurring activities should never exhibit the same. Such patterns need no further processing as they are already known to be exclusively associated with attacks. This differentiates PAMPs from danger signals, which show potential of some attack taking place, but as to whether it is indeed an attack requires further processing. Safe signals should then comprise those inputs that have neither PAMPS nor danger signals as shown below:

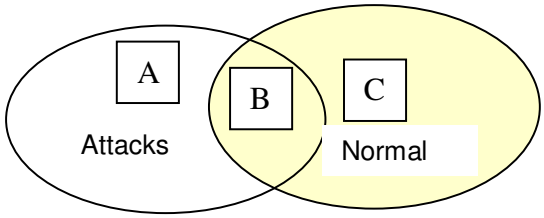


FIGURE 1: PAMP signals comprise that entire region A, safe signals are shown as C, and the danger signals are shown as B.

From a network intrusion detection perspective it thus means that connections that either fall within part A and C in figure 1 above need not be presented to the adaptive subsystem for further processing. Only those in B require being determined as to whether there occurrence is anomalous or not.

Thus the general mechanisms of the proposed model should look like shown in the figure 2 below:

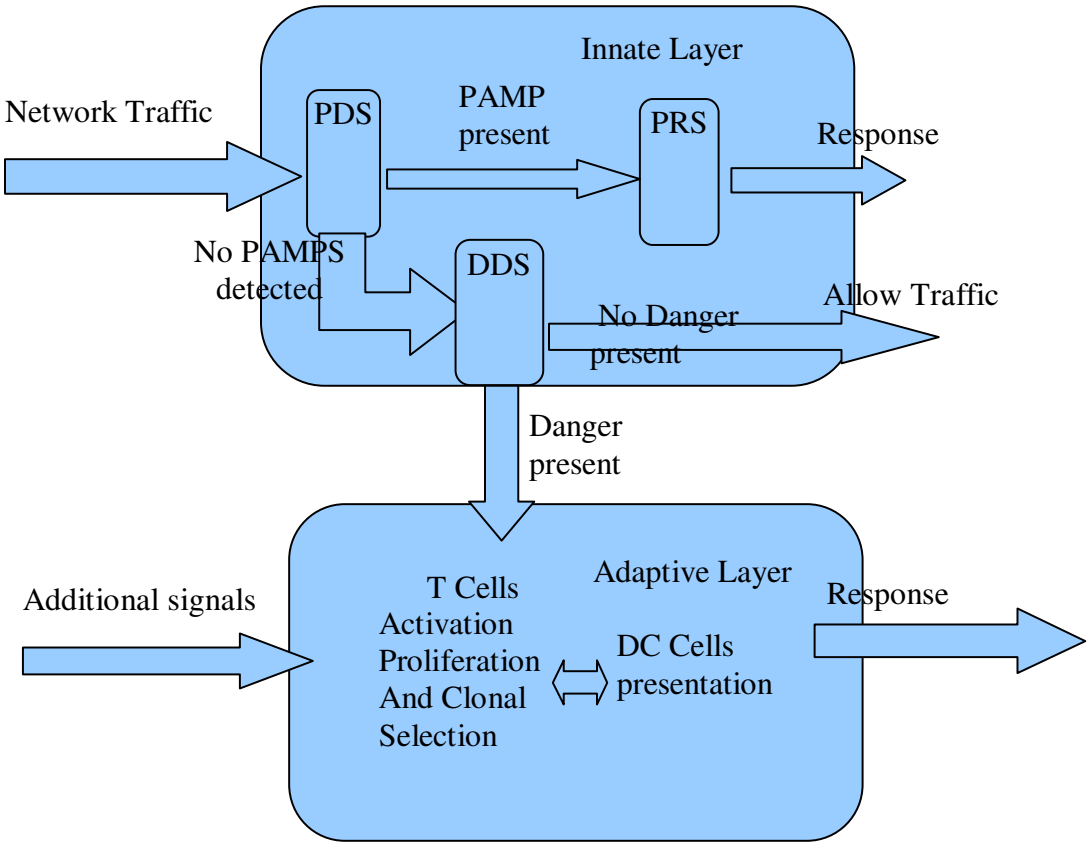


FIGURE 2: Proposed model overview

3.1 Innate Layer

Incoming network traffic stream is first presented to the PAMP-based Detection System (PDS) which if it does detect PAMPs it invokes the PAMP Response System (PRS) which immediately mounts a response. The response may be as simple as dropping a connection to initiating some recovery mechanisms. If no PAMPs are detected then the traffic goes through the Danger Detection System (DDS) where any signs considered to be danger signals are tested. If none is seen that traffic is passed as safe else the adaptive layer is invoked.

3.2 Adaptive Layer

We expect the adaptive layer mainly to differentiate the normal traffic from the anomalous where both have some danger signs present. More information may be needed to provide for further processing as such the need for additional signals. Algorithms like those developed by the danger project would be applicable in this area. So we expect to include such a variation in this layer.

3.3 Discussion

This simple modification has very different outcomes as opposed to the current implementations. Incorporating the innate concept of Pattern Recognition Receptors (PRRs) which recognize broad molecular patterns called PAMPs (pathogen associated molecular patterns) would help develop a broad (general) mechanism used for detection, and allow the antigen – antibody matching be used for more specified detection. We believe providing the two distinct layers will help in reducing the high False Positives currently evident in most IDS. It is indeed truly multi-layered

with responses provided at different levels thus increasing robustness of the resulting system. It is expected that even if an attack may not possess a particular PAMP, it will definitely have some danger signs. This is what is expected in disposability, such that a faulty PDS does not mean that the attack will necessarily go unspotted. Most of the currently implemented solutions have no automated response mechanisms incorporated. This is mainly because there is no direct way to relate detected intrusion to their cause. Most of the current detectors mainly show that an intrusion has been detected or is highly likely to have occurred but have no mechanisms of evading or preventing the same, where mechanisms have been incorporated, general responses have been adopted. Based on the PAMP detected it will be possible for us to tailor an appropriate response thus achieving self-organization.

We postulate that it should be possible to identify such PAMPs for different classes of attacks given that it is generally assumed that attacks will deviate in some way from normal behavior. Its actually more like identifying deviations in anomaly-based systems, however this deviations should be such that they are only possible with anomalous occurrence. PAMPs do not present a specific occurrence of an attack but instead a pattern associated with a class as such we expect the resulting system to have characteristics of both anomaly and misuse based IDS. Automated response will in a big way reduce the number of alerts produced by a system as compared to the current approach where alerts are generated and a human intervention is normally necessary. Immediate response will also shield the system being protected from adverse effects from a given attack.

4. CONCLUSION AND FUTURE WORK

In this paper we have proposed a mechanism that if incorporated into the current design of IDS and in particular network intrusion detection will provide an initial step to automating responses. This as shown will have a great impact, with the resulting system managing to incorporate most of the desired features of the immune system. The biology of the immune system presented indeed showed that the adaptive system is activated only in the presence of danger. Though it's not important to mirror the immune system, it gives us an appealing idea that we can use to reduce the amount of processing that takes place within the IDS thus making it possible to be deployed in a real time environment. We expect the resulting system to be highly portable and easy to maintain. Most importantly it's the introduction of distinct levels such that a fault in one level does not render the entire IDS unoperational.

We expect to embark on identification of the various PAMPs associated with the various classes of attacks as well as identifying what constitutes a danger sign(al). We hope to undertake a series of experiments to investigate the true worth of the proposed mechanism.

5. REFERENCES

[1] S. A. Hofmeyr, S.Forrest and A. Somayaji. "Intrusion Detection using Sequences of System Calls." *Journal of Computer Security*, vol.6**,** pp.151 – 180,1998.

[2] A.Somayaji, S.Hofmeyr and S.Forrest. "Principles of a Computer Immune System." In Proc. Of the New Security Paradigms Workshop, 1997, pp. 75 – 82.

[3] H. Debar, M. Dacier and A. Wespi. "Towards a taxonomy of intrusion detection systems." *Journal of Computer networks*, vol.31, pp.805 - 822,1999.

[4] H. Debar, M. Dacier and A. Wespi. "A Revised taxonomy of intrusion-detection systems." *Annales des Telecommunications,* vol. 55, pp. 83 – 100, 2000.

[5] S.Axelsson. "Intrusion Detection Systems: A Survey and Taxonomy." Technical Report 99 – 15, Department of Computer Engineering, Chalmers University of Technology, Mar 2000.

[6] G. Mayer. (2006). *Microbiology and Immunology,* On-Line Textbook, University of South Carolina, School of Medicine. Available: http://pathmicro.med.sc.edu/ghaffar/innate.html [January 10, 2010].

[7] J.Twycross, "Intergrated Innate and Adaptive Artificial Immune System applied to Process Anomaly Detection." PhD thesis, School of Computer Science, University of Nottingham, U.K. 2007.

[8] C.A. Janeway, Jr. "Presidential Address to the American Association of Immunologists: The Road Less Traveled by: The Role of Innate Immunity in the Adaptive Immune Response." *Journal of Immunology*, vol.161, pp. 539 – 544, 1998.

[9] S.A. Hofmeyr. "An Interpretative Introduction to the Immune System." in *Design Principles for the Immune Systems and other Distributed Autonomous Systems,* L.A. Segel and I.R. Cohen, Ed. New York: Oxford University Press, 2000.

[10] U. Aickelin and D. Dasgupta. "ARTIFICIAL IMMUNE SYSTEMS." In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Edmund K. Burke and Graham Kendall, Ed. 2005, pp. 375 – 399.

[11] S.G. Arancibia, C.J. Beltran, I.M. Aguirre, P. Silva, A.L. Peralta, F. Malinarich and M.A. Hermoso. "Toll-like Receptors are Key Participants in Innate Immune Responses." *Biol Res*, vol. 40, pp. 97 – 112, 2007.

[12] R. N. Germain. "An innately interesting decade of research in immunology." *Nature medicine*, vol.10 (12), pp.1307 – 1320, 2004.

[13] L. N. de Castro. "Artificial Immune Systems: Theory and Applications," presented at the Symposium on Neural Networks (SBRN 2000), Santos, Brazil, 2000.

[14] D. Dasgupta, Z. Ji, and F. Gonzalez. "Artificial Immune Systems (AIS) Research in the Last Five Years." in Proc. of the IEEE Congress on Evolutionary Computation Conference, 2003, vol.1, pp. 123–130.

[15] E. Hart and J. Timmis. "Application Areas of AIS: The past, the present and the future." *Applied Soft Computing*, vol.8, pp. 191 – 201, 2008.

[16] S. X. Wu and W. Banzhaf. "The use of computational intelligence in intrusion detection systems: A review." *Journal of Applied Soft Computing*, vol.10, pp.1–35, 2010.

[17] C. Langin and S. Rahimi. "Soft computing in intrusion detection: the state of the art." *Journal of Ambient Intell Human Comput*, vol.1, pp.133–145, 2010.

[18] G. Kumar, K. Kumar and M. Sachdeva. "The use of artificial intelligence based techniques for intrusion detection: a review." *Journal of Artif Intell Rev*, vol.34, pp. 369–387, 2010.

[19] S. Forrest and C. Beauchemin. "Computer Immunology." *Immunological reviews*, vol. 216(1), pp. 176-197, 2007.

[20] J.W. Kim, P. Bentley, U. Aickelin, J. Greensmith, G. Tedesco and J. Twycross, "Immune System Approaches to Intrusion Detection - A Review." *Natural Computing*, pp. 316 – 329, 2007.

[21] U. Aickelin, J. Greensmith and J. Twycross. “Immune System Approaches to Intrusion Detection - A Review.” in Proc. of the 3rd International Conference in AIS (ICARIS '04) LNCS 3239, 2004, pp. 316 - 329.

[22] J. O. Kephart, G.B.Sorkin, W. C. Arnold, D. M. Chess,G. J. Tesauro and S. R. White. “Biologically Inspired Defenses Against Computer Viruses.” in Proc. of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 985 – 996.

[23] D. Dasgupta. “Immunity - Based Intrusion Detection System: A General Framework.” in Proc. of the 22nd National Information Systems Security Conference (NISSC), 1999, pp. 147 – 160.

[24] S. Forrest, A.S. Perelson, L. Allen and R. Cherukuri. “Self - Nonself Discrimination in a Computer.” in Proc. of IEEE Symposium on Research in Security and Privacy, 1994, pp. 202-212.

[25] S. Forrest, S.A.Hofmeyr, A. Somayaji and T.A. Longstaff. “A Sense of Self for Unix Processes.” in Proc of the IEEE Symposium on Security and Privacy, 1996, pp. 120–128.

[26] S. Forrest, S.A. Hofmeyr and A. Somayaji. “Computer Immunology.” *Communications of the ACM,* vol. 40(10), pp. 88 – 96, 1997.

[27] S.A. Hofmeyr and S. Forrest. “Immunity by Design: An Artificial Immune System.” in Proc. of the 1st Annual Genetic and Evolutionary Computation Conference (GECCO). 1999. pp. 1289 - 1296.

[28] S.A. Hofmeyr and S. Forrest. “Architecture for an Artificial Immune System.” *Evolutionary Computation*, vol. 8(4), pp. 443 – 473, 2000.

[29] S. Forrest and S. Hofmeyr. “Engineering an Immune System.” *Graft*, vol. 4(5), pp. 5 – 9, 2001.

[30] D. Dasgupta and F. Gonzalez. “An Immunity-Based Technique to characterize Intrusions In Computer Networks.” *IEEE Transactions on Evolutionary Computation*, vol. 6(3), pp. 281 – 291, 2002.

[31] J. Balthrop, F. Esponda, S. Forrest and M. Glickman. “Coverage and Generalization in an Artificial Immune System.” in Proc. of Genetic and Evolutionary Computation Conference (GECCO), 2002, pp. 3 - 10.

[32] P.K. Harmer, P.D. Williams, G.H. Gunsh and G.B. Lamont. “An Artificial Immune System architecture for Computer Security Applications.” *IEEE Transactions on Evolutionary Computation*, vol. 6(3), pp. 252 – 280, 2002.

[33] K.P. Anchor, J.B. Zydallis, G.H.Gunsch and G.B. Lamont. “Extending the Computer Defense Immune System: Network Intrusion Detection with a Multi objective Evolutionary Programming approach.” in Proc. of the International Conference in Artificial Immune Systems (ICARIS), 2002.

[34] L. Tao. “An Immune-based dynamic intrusion detection model.” Chinese Science Bulletin, vol. 50(22), 2005.

[35] K. Luther, R. Bye, T. Alpcan, S. Albayrak, and A. Müller. “A Cooperative AIS Framework for Intrusion Detection.” in Proc. of the IEEE International Conference on Communications, (ICC), 2007, pp. 1409 – 1416.

[36] U. Aickelin, P. Bentley, S. Cayzer, J. Kim and J. McLeod. "The link between AIS and IDS?" in Proc. of the International Conference on Artificial Immune Systems (ICARIS), 2003, pp. 156 – 167.

[37] U. Aickelin and J. Greensmith. "Sensing Danger: Innate immunology for intrusion detection." Information Security Technical Reports, vol.12 (4), pp. 218 – 227, 2007.

[38] J. Twycross and U. Aickelin. "Towards a Conceptual Framework for Innate Immunity." in Proc. of the International Conference on Artificial Immune Systems (ICARIS), 2005, pp. 112-125.

[39] J. Twycross and U. Aickelin. "libtissue – Implementing Innate Immunity." in Proc. of the IEEE World Congress on Computational Intelligence, 2006, pp. 499-506.

[40] J. Twycross and U. Aickelin, "An Immune-Inspired Approach to Anomaly Detection." in *Handbook of Research on Information Assurance and Security,* S.Sharma and J. Gupta Ed. Miami: Idea Publishing Group, 2007, pp. 109-121.

[41]. J. Twycross and U. Aickelin, "Information Fusion in the Immune System." *Information Fusion*, vol. 11, pp. 35 – 44, 2010.

[42] J. Greensmith, U. Aickelin, and G. Tedesco. "Information fusion and anomaly detection with the dendritic cell algorithm." *Information Fusion*, vol.11(1), pp.21–34, 2010.

[43] J. Kim, W. Wilson, U. Aickelin and J. McLeod. "Cooperative Automated worm Response and Detection Immune Algorithm (CARDINAL) inspired by T-cell Immunity and Tolerance." in Proc. of the 4th National Conference on Artificial Immune Systems (ICARIS), 2005, vol. 3627, pp.168–181.

[44] J. Kim, J. Greensmith, J. Twycross and U. Aickelin. "Malicious code execution detection and response immune system inspired by the danger theory." in Proc. of the Adaptive and Resilient Computing Security Workshop (ARCS), 2005.

[45] S. Forrest and S.A. Hofmeyr. "Immunology as Information Processing." in *Design Principles for the Immune Systems and other Distributed Autonomous Systems,* L.A. Segel and I.R. Cohen, Ed. New York: Oxford University Press, 2001, pp. 361 – 387.

[46] M. Glickman, J. Balthrop and S. Forrest. "A Machine Learning Evaluation of an Artificial Immune System". *Evolutionary Computation*, vol.13(2), pp. 179–212, 2005.

[47] A.A. Freitas and J. Timmis. "Revisiting the Foundations of AIS: A problem-oriented perspective." in Proc. of the International Conference in Artificial Immune Systems (ICARIS), 2003, pp. 229 – 241.

[48] J. Twycross and U. Aickelin. "Biological inspiration for Artificial Immune Systems." in Proc. of the 6th International Conference on Artificial Immune Systems, 2007, pp. 300 – 311.

[49] S. M. Garrett. "How Do We Evaluate Artificial Immune Systems?". *Evolutionary Computation,* vol. 13(2), pp. 145 – 178, 2005.

Online Adaptive Control for Non Linear Processes Under Influence of External Disturbance

Nisha Jha *nishajha2010@gmail.com*
Department of Electronic Science
University of Delhi South Campus
New Delhi, 110021, India

Udaibir Singh *uday_mac2001@yahoo.co.in*
Department of Electronics
Acharya Narendra Dev College
University of Delhi
Govindpuri, Kalkaji, New Delhi, 110019, India

T.K. Saxena *tushyks@gmail.com*
National Physical Laboratory
Dr. K.S. Krishnan Road
New Delhi, 110 012, India

Avinashi Kapoor *avinashi_kapoor@yahoo.com*
Department of Electronic Science
University of Delhi South Campus
New Delhi, 110021, India

Abstract

In this paper a novel temperature controller, for non linear processes, under the influence of external disturbance, has been proposed. The control process has been carried out by Neural Network based Proportional, Integral and Derivative (NNPID). In this controller, two experiments have been conducted with respect to the setpoint changes and load disturbance. The first experiment considers the change in setpoint temperature in steps of 10°C from 50°C to 70°C for three different rates of flow of water. In the second experiment the load disturbance in terms of addition of 100ml/min of water at three different time intervals is introduced in the system. It has been shown that, in these situations, the proposed controller adjusts NN weights which are equivalent to PID parameters in both the cases to achieve better control than conventional PID. In the proposed controller, an error less than 0.08°C have been achieved under the effect of the load disturbance. Moreover, it is also seen that the present controller gives error less than 0.11°C, 0.12°C and 0.12°C, without overshoot for 50°C, 60°C and 70°C, respectively, for all three rate of flow of water.

Keywords: Neural Network Based PID (NNPID) Controller, Temperature Controller, Back-propagation Neural Network, Load Disturbance.

1. INTRODUCTION

Temperature control is an important factor in chemical, material and semiconductor manufacturing processes [1]-[3]. To design a general purpose temperature controller with good response time, smaller error and overshoot with load disturbance for the industrial implementation is still a challenge in the control research field. Over the past several years the on-off control and PID control schemes have been employed in commercial products with reasonable success.

A PID controller is the classical control algorithm in the field of process control. It still predominates in the process industries due to its robustness and effectiveness for a wide range

of operating conditions and partly to its functional simplicity [4]. For the existing controllers, there are three important parameters, namely, K_p, K_i and K_d which need to be evaluated [5]. The problem associated with the PID controller is to choose optimal value of these parameters so that the desired output is yielded for the appropriate process inputs. Usually, process engineers tune PID controller manually for an operation which, if done diligently, can take considerable time. Therefore, it is hard to establish an accurate dynamic model for a PID controller design. When the system has external disturbances, such as the variations of loads and changing process dynamics, then the transient response may go down. For this reason, free intelligent control schemes have gained the researcher attention.

In order to overcome the above disadvantages [4], [6], [7], researchers have proposed some adjusting rules for the self tuning controllers (STC) [8]-[19]. They have considerable potential for the process control problems since STCs provide a systematic and flexible approach for dealing with uncertainties, nonlinearities, and time varying parameters. A basic model structure for static nonlinearities is the back-propagation neural network (BPNN) [20]. The major advantages of BPNN over the traditional controller is that it can tune the three PID parameters on-line without requiring the prior knowledge of the mathematical model of different plants. Besides, the other advantages include its nonlinear mapping and self-learning abilities in various control processes, such as temperature control. It may be mentioned that the time varying and complex nonlinearity problems associated with PID controllers have been addressed by other researchers also using different algorithms [21], [22].

Neural Networks (NN) [23], which is the focus of the current work, is a better alternative to solve control engineering problems. It can be applied in two different ways: one is to use the NN to adjust the parameters of PID controller and the other is to use it as a direct controller. PID parameter values can also be adjusted by creating NN system based on the system output error signal [24]-[26], [27]-[30]. Prominent among them are the inverse model neuro-control approach by Widrow and Steams [29] and Psaltis, *et al*. [30] and further modified by other researchers [31]-[34].

In the present paper we have investigated two conditions viz the change in setpoint temperature and the load disturbance using Neural Network PID (NNPID) controller. In both the cases NN weights equivalent to PID parameters, are trained to achieve better control than existing conventional PID.

2. PROPOSED DESIGN APPROACH AND EXPERIMENTAL DETAILS

Fig.1 shows the block diagram of the proposed approach followed in the present work. According to this block diagram, the actuating error, T_{err}, can be expressed as

$$T_{err} = T_s - T_o \quad (1)$$

Where T_s and T_o are the setpoint temperature and observed temperature respectively and T_{err} is the error in terms of temperature.

The design of NNPID is shown in Fig. 2. It consists of three layers which are input layer, hidden layer and output layer. The input layer has two neurons represented by I_1 and I_2.The output layer

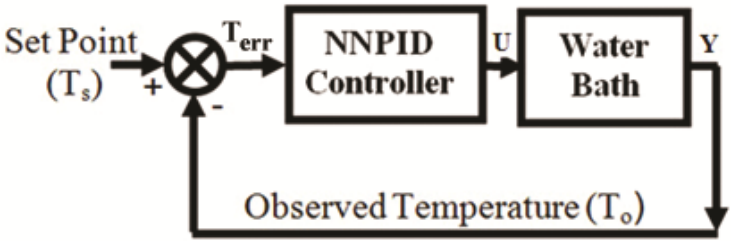


FIGURE 1: Block Diagram of the approach followed

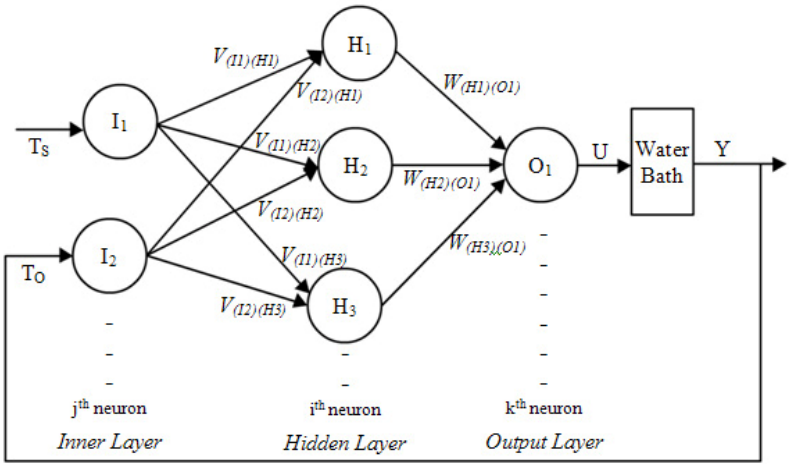


FIGURE 2: Neural Network tuning of PID Controller

has one neuron represented by O_1. The hidden layer has three neurons and they are symbolized as H1 (P-neuron), H2 (I-neuron) and H3 (D-neuron) respectively.

In the present case weights for the different layer combinations are taken as follows:
Weights between input layer and hidden layer are

$$V_{(I1)(Hi)} = +1, V_{(I2)(Hi)} = -1 \quad (2)$$

Weights between hidden layer and output layer are taken in terms of PID parameters as

$$W_{(H1)(O1)} = K_p, W_{(H2)(O1)} = K_i \text{ and } W_{(H3)(O1)} = K_d, \quad (3)$$

Then, input to hidden layer nodes are defined as

$$X_{(H1)} = V_{(I1)(H1)}I_1 + V_{(I2)(H1)}I_2 = T_S - T_o = T_{err} \quad (4)$$

$$X_{(H2)} = V_{(I1)(H2)}I_1 + V_{(I2)(H2)}I_2 = T_S - T_o = T_{err} \quad (5)$$

$$X_{(H3)} = V_{(I1)(H3)}I_1 + V_{(I2)(H3)}I_2 = T_S - T_o = T_{err} \quad (6)$$

where$X_{(H1)}$, $X_{(H2)}$ and $X_{(H3)}$are the inputs of the hidden layer nodes.

The outputs of the hidden layer nodes are equal to their inputs, which can be expressed as function of proportional, integral and derivative as mentioned below:

$$Y_{(H1)} = T_{err} \quad (7)$$

$$Y_{(H2)} = \int T_{err}dt \quad (8)$$

$$Y_{(H3)} = \frac{dT_{err}}{dt} \quad (9)$$

Then, input to output layer becomes

$$X_{(O1)} = W_{(H1)(O1)}Y_{(H1)} + W_{(H2)(O1)}Y_{(H2)} + W_{(H3)(O1)}Y_{(H3)} \quad (10)$$

$$= K_pT_{err} + K_i\int T_{err}dt + K_d\frac{dT_{err}}{dt} \quad (11)$$

where $Y_{(H1)}$, $Y_{(H2)}$ and $Y_{(H3)}$are output part of hidden layer nodes, and $X_{(O1)}$ is the input part of output layer.

Thus eq. (11) illustrates that PID parameters, which compared with weights as given in eq. (3), are tuned by using NNPID algorithm. It is well-known that most neural networks cannot be practically used in a controller because the initial connective weights of the neural networks are randomly selected. The randomized selection procedure imparts instability to the system. Therefore, it demands more experience to choose or tune PID parameters in order to ensure the stability. This can be achieved via training and learning capability of NNPID algorithm. The simple and prevalent algorithm which we have used in our work is BPNN algorithm [20] for weighting coefficients.

In the present controller, the main aim of the above algorithm is to minimize the error as given in eq. (12) in order to recover the system quickly from the effects of the external disturbance by tuning of PID parameters.

$$E_k = \frac{1}{2}[T_s - T_o]^2 \tag{12}$$

The weights of NNPID controller are adjusted by BPNN algorithm based on steepest descent on-line training process. It is done in terms of adjusted weights of hidden-to-output layer $[W_{(ik)}]$and input-to-hidden layer $[V_{(ij)}]$ [35]. The increments of weight in hidden-to-output connection are updated by using the gradient descent method as

$$\Delta W_{ik}(n) = -\eta \frac{\partial E_k}{\partial W_{ik}} + \alpha \Delta W_{ik}(n-1) \tag{13}$$

$$= -\eta \left[\frac{\partial E_k}{\partial Y_{(ok)}}\right]\left[\frac{\partial Y_{(ok)}}{\partial X_{(ok)}}\right]\left[\frac{\partial X_{(ok)}}{\partial W_{(ik)}}\right] + \alpha \Delta W_{ik}(n-1) \tag{14}$$

where η and α are learning coefficient and momentum, respectively. Here the values of these terms are taken to be η = 0.005 and α = 0.5.

Further eq. (14) can be rewritten as [35]

$$\Delta W_{ik}(n) = -\eta\beta\left[T_k - Y_{(Ok)}\right]Y_{(Ok)}\left[1 - Y_{(Ok)}\right]Y_{(Hi)} \quad + \alpha \Delta W_{ik}(n-1) \tag{15}$$

$$\Delta W_{ik}(n) = \eta \delta_k Y_{(Hi)} + \alpha \Delta W_{ik}(n-1) \tag{16}$$

Where we have defined

$$\delta_k = \beta\left[T_k - Y_{(Ok)}\right]Y_{(Ok)}\left[1 - Y_{(Ok)}\right] \tag{17}$$

Similarly, the incremental weights of input-to-hidden connection are updated as

$$\Delta V_{(ij)}(n) = -\eta \frac{\partial E_k}{\partial V_{(ij)}} + \alpha \Delta V_{(ij)}(n-1) \tag{18}$$

$$= -\eta \left[\frac{\partial E_k}{\partial Y_{(ok)}}\right]\left[\frac{\partial Y_{(Ok)}}{\partial X_{(Ok)}}\right]\left[\frac{\partial X_{(Ok)}}{\partial Y_{(Hi)}}\right]\left[\frac{\partial Y_{(Hi)}}{\partial X_{(Hi)}}\right]\left[\frac{\partial X_{(Hi)}}{\partial V_{(ij)}}\right] + \alpha \Delta V_{(ij)}(n-1) \tag{19}$$

Now, eq. (19) can be rewritten as [35]

$$\Delta V_{(ij)}(n) = \eta \gamma_i \beta Y_{(Hi)}\left[1 - Y_{(Hi)}\right]I_{ij} + \alpha \Delta V_{(ij)}(n-1) \tag{20}$$

$$\Delta V_{(ij)}(n) = \eta {\delta_k}^* I_{ij} + \alpha \Delta V_{(ij)}(n-1) \tag{21}$$

Where we can define

$$\gamma_i = W_{ik}\delta_k \tag{22}$$

$${\delta_k}^* = -\gamma_i \beta Y_{(Hi)}\left[1 - Y_{(Hi)}\right] \tag{23}$$

Now we use updated weights, $\Delta W_{(ik)}(n)$ and $\Delta V_{(ij)}(n)$ from eq. (16) and eq. (21) for finding new weights for hidden-to-output and input-to-hidden connections.

$$W_{ik}(n+1) = W_{ik}(n) + \eta \delta_k Y_{(Hi)} + \alpha \Delta W_{ik}(n-1) \tag{24}$$

$$V_{(ij)}(n+1) = V_{(ij)}(n) + \eta {\delta_k}^* I_{ij} + \alpha \Delta V_{(ij)}(n-1) \tag{25}$$

The new weights are adjusted by updated weights as per eq. (24) and eq. (25) with iterations till we get the minimum mean square error in terms of temperature. Now these updated weights are employed for the experiment discussed below.

The schematic diagram of the experimental setup of the water bath temperature controller is shown in Fig. 3.

The hardware for controlling the temperature of the bath has been designed and fabricated around the Atmel microcontroller 89C51. The temperature of the bath is acquired with the help of platinum resistance thermometer (PRT). When the PRT is excited with a constant current source of 1mA current, it gives the output in voltage form. This voltage is then fed to the 4½ digit analog

to digital converter (ADC). This digitized voltage is then sent to the personal computer (PC) by microcontroller 89C51through RS232C interface. The program in PC does the calculations using the NNPID algorithm. After doing the entire calculations microcontroller controls the TRIAC firing circuit and the firing angle for the required energy, through heater, to be given to the water bath. The NNPID program in PC continuously monitors the temperature and accordingly controls the same in the bath. In case it senses any change in the temperature, it automatically modifies the parameters of the temperature controller. The NNPID program in PC has been written in Visual BASIC-5.0 language. The program stores the data in the user defined file as well as plots the online data in the form of graph on the screen. A specially designed varying environment is created by continuous flow of fresh water in such a way that the level of the water inside the bath remains constant even if the hot water is removed at random outflow rates. Uniform heat distribution is maintained using the circulator, and the isolated system is used to minimize external disturbance. The cooling is achieved at a constant rate using the refrigeration system of the bath.

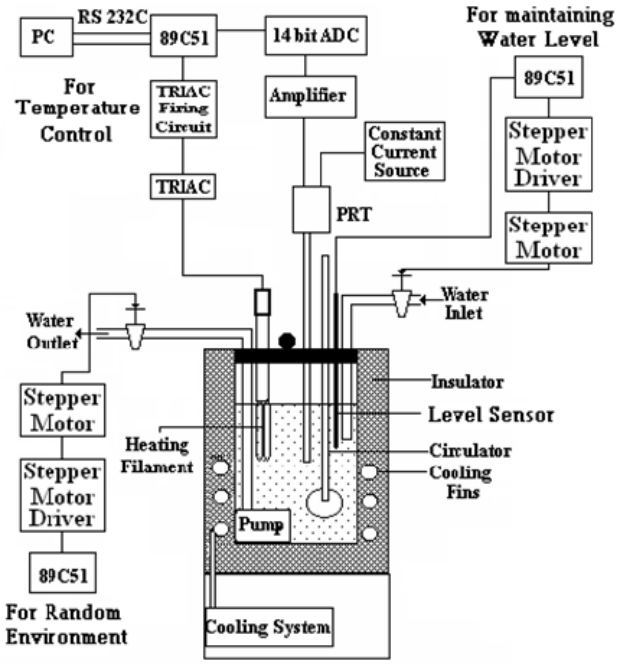


FIGURE 3: Block Diagram of the Experimental Setup

distribution is maintained using the circulator, and the isolated system is used to minimize external disturbance. The cooling is achieved at a constant rate using the refrigeration system of the bath.

3. EXPERIMENTAL AND SIMULATION RESULTS

In this paper two sets of experiments were conducted in the water bath. In the first set of experiments, the tracking performance of the two controllers i.e. NNPID controller and conventional PID controller with respect to setpoint changes are studied. In this system, further three set of experiments were conducted at three different flow of water i.e. at 100ml/min, 250ml/min and 500ml/min as shown in Figs. 4, 5 and 6 respectively. In these experiments the setpoint temperature of the water bath was increased in steps of 10^{o}C from 50^{o}C to 70^{o}C to investigate the effect of flow of water on temperature control at the different setpoint.

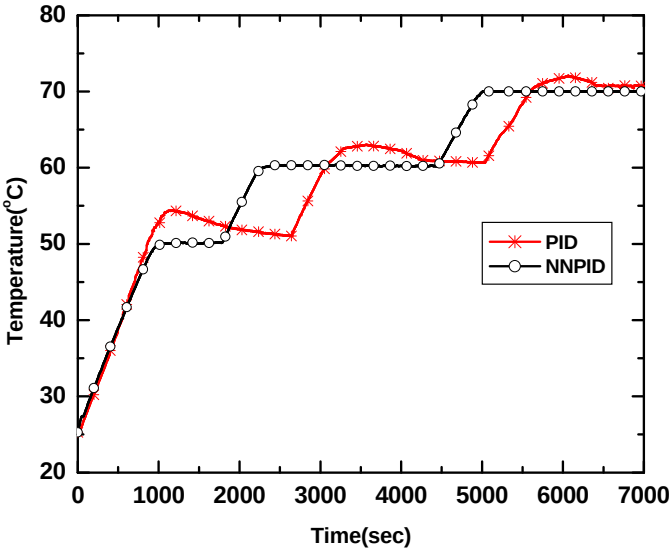


FIGURE 4: Showing the comparison of NNPID controller with the conventional PID controller of a water bath for 100 ml/min flow rate of water with respect to setpoint changes.

The simulation results subjected to the changes in setpoint for different flow rate of water are shown in Figs. (4-6). The three systems are categorized in terms of change in flow rate of water are shown in Table I. The settling time taken by NNPID and PID controllers to achieve target temperatures of 50°C, 60°C and 70°C for different flow rates of water are given in Table II. According to this table, when we refer Figs. (4-6), we infer that NNPID controller gives better performance in respect of less settling time as compared to the conventional PID controller in achieving change in setpoint temperature. Hence the experimental and simulation results of these systems show the simplicity, reliability and robustness of NNPID over conventional PID.

To compare the results of the NNPID controller with the results of the conventional PID controller, the parameters of the PID controller were tuned for initial gain setting of NNPID controller by its best fit values as proportional gain, K_p=2.5, integral gain, K_i=100 and derivative gain, K_d=10. The neural network fine tunes the system iteratively based on the performance of the closed loop

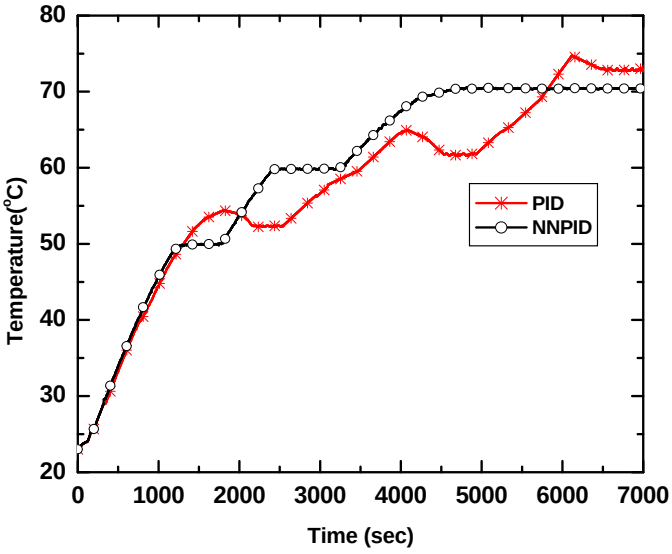


FIGURE 5: Showing the comparison of NNPID controller with the conventional PID controller of a water bath for 250 ml/min flow rate of water with respect to setpoint changes.

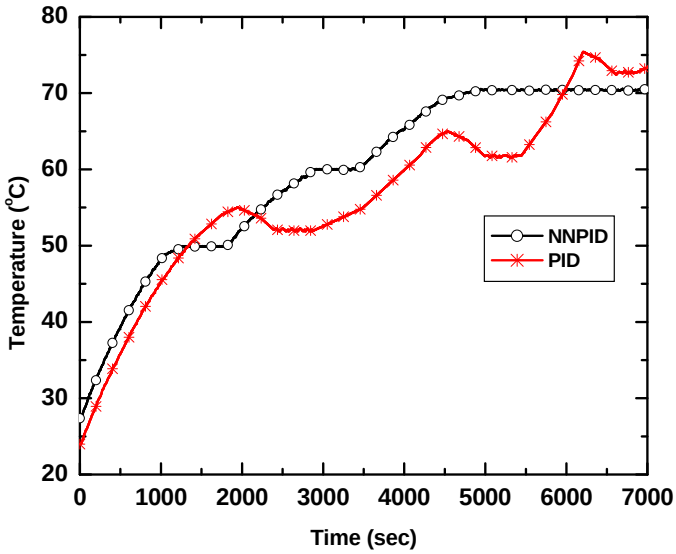


FIGURE 6: Showing the comparison of NNPID controller with the conventional PID controller of a water bath for 500 ml/min flow rate of water with respect to setpoint changes

K_p	2.5
K_i	100
K_d	10
Power of Heater	1500 Watt
Volume of water	15 liter
Voltage	5volts
Initial and Final Set point temperature	50°C and 70°C
Temperature change	+10°C
Flow rate of water	100ml/min, 250 ml/min, 500 ml/min
Load disturbance	100ml/min water

TABLE 1: Different Values of System Parameters

system. The temperature response of a water bath having 15 liter volume and heated with a power of 1.5KW for 100ml/min flow rate of water using NNPID and conventional PID are shown simultaneously for comparison in Fig.5. Similarly NNPID and conventional PID results for 250ml/min and 500ml/min flow rate of water are shown in Fig.5 and Fig.6 respectively. It is clear from these figures that there is always overshoot for conventional PID at initial settling time for each set temperature as 50°C, 60°C and 70°C of the system. This is shown in Table III. This table also indicates that NNPID controller gives error less than 0.11°C, 0.12°C and 0.12°C without overshoot for 50°C, 60°C and 70°C respectively for all the three flow rate of water. These errors are comparatively less than conventional PID controller. In addition, the neural network achieves setpoint fast as compared to the conventional PID controller as shown in Figs. (4-6). One can possibly say that the neural network controller tracked well all the three setpoint and has good generalization capability even with a small number of training patterns.

	NNPID Controller		PID Controller	
	Settling Time		Settling Time	
Temperature range	50°C-60°C	60°C-70°C	50°C-60°C	60°C-70°C
100 ml/min	7min	9min 30sec	23min	23min 30sec
250 ml/min	11min	18min 30sec	31min	31min
500 ml/min	17min	27min	35min	35min

TABLE 2: Settling Time of NNPID and PID Controllers For Three Flow Of Water

	NNPID Controller			Conventional PID Controller					
	Error without Overshoot			Error with Overshoot					
Set Temperature	50°C	60°C	70°C	50°C		60°C		70°C	
				Error	Over shoot	Error	Over shoot	Error	Over shoot
100 ml/min flow	0.09 °C	0.10 °C	0.10 °C	1.38 °C	4.49 °C	1.0 °C	3.03 °C	1.0 °C	2.01 °C
250 ml/min flow	0.10 °C	0.11 °C	0.12 °C	2.32 °C	4.35 °C	1.87 °C	4.9 °C	2.73 °C	4.47 °C
500 ml/min flow	0.11 °C	0.12 °C	0.11°C	2.54 °C	4.93 °C	1.90 °C	4.77 °C	2.88 °C	5.48 °C

TABLE 3: Error and Overshoot of NNPID and Conventional PID controller for three rate of flow of water

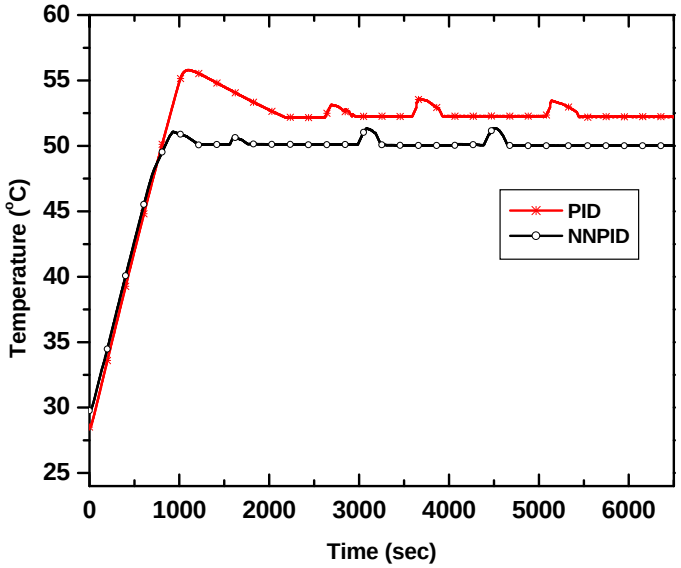


FIGURE 7: Showing the comparison of NNPID controller with the conventional PID controller of a water bath under the effect of load disturbances.

In second set of experiments, the load disturbances in terms of addition of 100ml/min water were introduced in the process of system for studying the ability of the two controllers when the external disturbance was imposed. These external disturbances were made in three steps at different interval of time. These three disturbances were added to the output at 43min, 59min and 84min respectively for PID controller and for NNPID controller at 25min, 49min and 72min as shown in Fig. 7. It could be observed from this figure that when we introduce external disturbance of 100ml/min of water during three steps in the system for set temperature of 50^{o}C, the NNPID controller takes much less settling time and overshoot as compared to conventional PID controller. So it is appropriate to say that neural network controller recovered fast with error less than 0.08 ^{o}C with less overshoot under the effect of these load disturbances. So we are able to say that NNPID controller has ability to adapt quickly to changes at its input. On the other hand the conventional PID controller has poor rate of recovery which deteriorate the system. Additionally, it has error greater than 0.2^{o}C. Our experimental setup gives better settling time, less overshoot and minimum deviation in setpoint.

4. CONCLUSION

In conclusion, the present work shows the new approach of controlling the temperature of the dynamic system. This particular system designed and developed around Atmel's 89C51 microcontroller employed on a water bath. The temperature control of the system has been analyzed by conducting two experiments in respect of setpoint changes and load disturbances. The first experiment considers change in setpoint temperature in step of 10^{o}C from 50^{o}C to 70^{o}C for three different rate of flow of water. It is observed that NNPID controller gives error less than 0.11^{o}C, 0.12^{o}C and 0.12^{o}C without overshoot for 50^{o}C, 60^{o}C and 70^{o}C respectively for all three flow rate of water. In second experiment, the load disturbance in terms of addition of 100ml/min water at three different intervals of time is introduced. It gives error less than 0.08 ^{o}C with less overshoot under the effect of the load disturbance. In both the cases NN weights corresponding to PID parameters, are trained, to achieve better control than existing conventional PID. This paper has shown that inexpensive neural hardware may become an important technology for many modern industrial control applications.

5. REFERENCES

[1] M. Khalid, S. Omatu and R. Yusof, "MIMO furnace control with neural networks," *IEEE Trans. Contr. Syst. Technol.*, vol. 1, pp. 238–245, 1993.

[2] J. Tanomaru, S. Omatu, "Process Control by On-line Trained Neural Controllers," *IEEE Transactions on Industrial Electronics*, vol. 39,pp. 511-521, 1992.

[3] M. Khalid and S. Omatu, "A neural network controller for a temperature control system," *IEEE Contr. Syst.*, vol. 12, pp. 58–64, June 1992.

[4] W. Wu, J. Yuan and L. Cheng, "Self-tuning sub-optimal control of time-invariant systems with bounded disturbance," in *Proc. of the 2005 American Control Conference.*, vol. 2, 2005, pp. 876–882.

[5] C. Y. Guo, Q. Song, and W. J. Cai, "Supply Air Temperature Control of AHU with a Cascade Control Strategy and a SPSA Based Neural Controller," in *Proceedings of the 2005 International Joint Conference on Neural Networks*, vol. 4, 2005, pp. 2243-2248.

[6] S. Omatu, T. Iwasa, M. Yoshioka, "Skill-based PID Control by Using Neural Networks," in *Proceedings of the 1998 IEEE International Conference on System Man and Cybernetics*, vol. 2, 1998, pp. 1972-1977.

[7] Q. H. Hu, A. T. P. So, W. L. Tes and A. Dong, "Use of Adaline PID Control for a Real MVAC System," *Proceedings of the 2005 International Conference on Wireless Communications, Networking and Mobile Computing,* vol. 2, 2005, pp. 1374 – 1378.

[8] K. J. Astrom and T. Hagglund, "Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins," *Automatica*, vol. 20, pp. 645-651, 1984.

[9] C. C. Hang, K.J. Astrom and W.K. Ho, "Refinements of the Ziegle-Nichols tuning formula," in 1991 *IEE proceedings Pt. D, Control theory & Applications*, vol.138, no. 2, 1991, pp. 111-118.

[10] W. K. Ho, C. C. Hang and L. S. Cao, "Tuning of PID Controllers Based on Gain and Phase Margin Specifications, *Automatica*, vol.31, no. 3, pp. 497-502, 1995.

[11] K. J. Astrom, C. C. Hang, P. Persson and W. K. Ho, "Toward Intelligent PID Control," *Automatica*, vol.28., no. 1, pp.1-9, 1992.

[12] W. K. Ho, O. P. Gan, E. B. Tay and E. L. Ang, "Performance and Gain and Phase Margins of Well Known PID Tuning Formulas," *IEEE Trans. On Control Systems Technology*, vol.4, pp.473-477, 1996.

[13] W. K. Ho, C. C. Hang and J. H. Zhou, "Performance and Gain and Phase Margins of Well-Known PI Tuning Formula," *IEEE Trans. On Control Systems Technology*, vol.3, no. 2, pp.245-248, 1995.

[14] F. Cameron and D.E. Seborg, "A self-tuning controller with a PID structure," *Int. J. Control* vol. 30, pp. 401-417, 1983.

[15] D.W. Clark and P.J. Gawthrop, "Self-tuning control," in *Proc. IEE, Pt-D*, vol. 126, 1979, pp. 633-640.

[16] R. Ortega and R. Kelly, "PID self-tuners: Some theoretical and practice aspects," *IEEE Trans. Ind. Electron*, vol. 31, pp. 312, 1984.

[17] C.G. Proudfoot, P.J. Gawthrop and O.L.R. Jacobs, "Self-tuning PI control of a pH neutralization process," in *Proc. IEE, Pt-D*, vol. 130, 1983, pp. 267-272.

[18] F. Radke and R. Isermann, "A parameter-adaptive PID controller with stepwise parameter optimization," *Automatic*, vol. 23, pp. 449-457, 1987.

[19] B. Wittenmark, "Self-tuning PID Controllers Based on Pole Placement," *Lund Institute Technical Report*, TFRT-7179, 1979.

[20] D. E. Rumelhart and J. L. McClelland, "Parallel Distributed Processing," vol. 1, MIT Press, Cambridge, MA, 1986.

[21] J. H. Taylor and K. J. Astrom, "A non-linear PID auto tuning algorithm", American Automatic control conference, Seattle, W.A., 1986, pp. 1-6.

[22] M. A. Unar, D. J. Murray-Smith and S. F. Ali Shah, "Design and tuning for fixed structure PID controllers—A survey", report CSC-96016, *Centre for systems and control & department of mechanical Engineering, university of Glaslow*, 1996.

[23] A. E. B. Ruano, P. J. Fleming and D. I. Jones, "Connectionist approach to PID autotuning," in *IEE proceedings-D*, vol. 139 (3), 1992, pp. 279-285.

[24] K. C. Chan, S. S. Leong and G. C. I. Lin, "A neural network PI controller tuner," *Artificial Intelligence in Engineering*, vol. 9, pp. 167-176, 1995.

[25] C. L. Chen and F. Y. Chang, "Design and analysis of neural/fuzzy variable structural PID control systems," in *IEE Proceedings Control Theory Application*, vol. 143 (2), 1996, pp. 200-208.

[26] V. VanDoren, "Model free adaptive control", *Control engineering, Europe*, pp. 25-31, 2001.

[27] M. Khalid, S. Omatu, "A neural network controller for a temperature control system," *IEEE Contr. Syst. Mag.*, vol. 12, pp. 58-64, 1990.

[28] A. G. Barto, "Connectionist learning for control," in W. T. Miller, 111, R. S. Sutton, P. J. Werbos, eds., Neural Networks for Control. Cambridge, MA: MI, 1990.

[29] B. Widrow, S. D. Steams, "Adaptive Signal Processing," Englewood Cliffs, NJ: Prentice Hall, 1985.

[30] D. Psaltis, A. Sideris, A. Yamamura, "A multilayered neural network controller," *IEEE Control Syst. Mag.*, vol. 10, pp. 44-48, 1988.

[31] K. S. Narendra, K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans.Neura1 Networks*, vol. 1, pp. 4-27, 1990.

[32] P. J. Werbos, "Backpropagation through time: What it does and how to do it?," in *Proc. IEEE. 78*, 1990, pp. 1550-1560.

[33] D. H. Nguyen and B. Widrow, "Neural networks for self-leaming control systems," *IEEE Control Syst. Mag.*, vol. 10, pp. 18-23, 1990.

[34] M. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive Science.*, vol. 16.pp. 307-354, 1992.

[35] A. N. Ponce, A. A. Behar, A. O. Hernandez and V. R. Sitar, "Neural Network for Self-tuning Control Systems", *Acta Polytechnica*, vol. 44, pp.49-52, 2004.

Fuzzy Logic and Neuro-fuzzy Systems: A Systematic Introduction

Yue Wu *wuyue@enjoyor.cc*
Enjoyor Inc
Hangzhou, 310030, China

Biaobiao Zhang *zhangbb@enjoyor.net*
Enjoyor Inc
Hangzhou, 310030, China

Jiabin Lu *lujiabin@gdut.edu.cn*
Faculty of Electromechanical Engineering
Guangdong University of Technology
Guangzhou, 510006, China

K. -L. Du *kldu@ece.concordia.ca*
Department of Electrical and Computer Engineering
Concordia University
Montreal, H3G 1M8, Canada
and
Enjoyor Inc
Hangzhou, 310030, China

Abstract

Fuzzy logic is a rigorous mathematical field, and it provides an effective vehicle for modeling the uncertainty in human reasoning. In fuzzy logic, the knowledge of experts is modeled by linguistic rules represented in the form of IF-THEN logic. Like neural network models such as the multilayer perceptron (MLP) and the radial basis function network (RBFN), some fuzzy inference systems (FISs) have the capability of universal approximation. Fuzzy logic can be used in most areas where neural networks are applicable. In this paper, we first give an introduction to fuzzy sets and logic. We then make a comparison between FISs and some neural network models. Rule extraction from trained neural networks or numerical data is then described. We finally introduce the synergy of neural and fuzzy systems, and describe some neuro-fuzzy models as well. Some circuits implementations of neuro-fuzzy systems are also introduced. Examples are given to illustrate the concepts of neuro-fuzzy systems.

Keywords: Fuzzy Set, Fuzzy Logic, Fuzzy Inference System, Neuro-fuzzy System, Neural Network, Mamdani Model, Takagi-Sugeno-Kang Model.

1. INTRODUCTION

Fuzzy set, a concept first proposed by Zadeh [123], is a method for modeling the uncertainty in human reasoning. Fuzzy logic is suitable for the representation of vague data and concepts on an intuitive basis, such as human linguistic description, e.g. the expressions *approximately*, *large*, *young*. The conventional set, also called the crisp set, can be treated as a special form of fuzzy set. Unlike the binary logic, fuzzy logic uses the notion of membership. A fuzzy set is uniquely determined by its membership function (MF), and it is also associated with a linguistically meaningful term.

Fuzzy logic provides a systematic tool to incorporate human experience. It is based on three core concepts, namely, fuzzy sets, linguistic variables, and possibility distributions. Fuzzy set is used to characterize linguistic variables whose values can be described qualitatively using a linguistic expression and quantitatively using an MF [124]. Linguistic expressions are useful for communicating concepts and knowledge with human beings, whereas MFs are useful for processing numeric input data. When a fuzzy set is assigned to a linguistic variable, it imposes an elastic constraint, called a possibility distribution, on the possible values of the variable.

Fuzzy logic is a rigorous mathematical discipline. Fuzzy reasoning is a straightforward formalism for encoding human knowledge or common sense in a numerical framework, and FISs can approximate arbitrarily well any continuous function on a compact domain [55], [113]. FISs and feedforward neural networks (FNNs) can approximate each other to any degree of accuracy [13]. Fuzzy logic first found popular applications in control systems, where an FIS is built up by codifying human knowledge as linguistic IF-THEN rules. Since its first reported industrial application in 1982 [41], it has aroused global interest in the industrial and scientific community, and fuzzy logic has also been widely applied in data analysis, regression and prediction, as well as signal and image processing. Many application-specific integrated circuits (ASICs) has also been designed for fuzzy logic [31].

In this paper, we give a systematic introduction to fuzzy logic and neuro-fuzzy systems. The paper is organized as follows. In Section 2, we provide a short tutorial on fuzzy logic. Section 3 compares fuzzy logic and neural network paradigms. Section 4 compares the relation between fuzzy logic and MLP/RBFN, and rule generation from trained neural networks is introduced in this section. Rule extraction from numerical data is introduced in Section 5. The paradigm of neuro-fuzzy systems is described in Section 6. Some neuro-fuzzy models are introduced in Section 7. In Section 8, we describe some fuzzy neural circuits. An illustration of using neuro-fuzzy systems is given in Section 9. We summarize this paper in Section 10.

2. FUNDAMENTALS OF FUZZY LOGIC

2.1 Definitions

We list below some definitions and terminologies used in the fuzzy logic literature.

2.1.1 Universe of Discourse

The universal set X: $X \rightarrow [0,1]$ is called the universe of discourse, or simply the universe. The implication $X \rightarrow [0,1]$ is the abbreviation for the IF-THEN rule: "IF x is in X, THEN its MF $\mu_X(x)$ is in [0,1].", where $\mu_X(x)$ is the MF of x. The universe X may contain either discrete or continuous values.

2.1.2 Linguistic Variable

A linguistic variable is a variable whose values are linguistic terms in a natural or artificial language. For example, the size of an object is a linguistic variable, whose value can be *small*, *medium*, and *big*.

2.1.3 Fuzzy Set

A fuzzy set A in X is defined by

$$A = \{x, \mu_A(x) | x \in X\}, \tag{1}$$

where $\mu_A(x) \in [0,1]$ is the MF of x in A. For $\mu_A(x)$, the value 1 stands for complete membership of the set A, while 0 represents that x does not belong to the set at all. A fuzzy set can also be syntactically represented by

$$A = \begin{cases} \sum_{x_i \in X} \frac{\mu_A(x_i)}{x_i}, & \text{if } X \text{ is discrete} \\ \int_X \frac{\mu_A(x)}{x}, & \text{if } X \text{ is continuous} \end{cases}. \quad (2)$$

2.1.4 Support

The elements on fuzzy set A whose membership is larger than zero are called the support of A

$$\text{sp}(A) = \{x \in A | \mu_A(x) > 0\}. \quad (3)$$

2.1.5 Height

The height of a fuzzy set A is defined by

$$\text{hgt}(A) = \sup\{\mu_A(x) | x \in X\}. \quad (4)$$

2.1.6 Normal Fuzzy Set and Non-normal Fuzzy Set

A fuzzy set A is said to be *normal* if $\text{hgt}(A) = 1$. If $0 < \text{hgt}(A) < 1$, the fuzzy set A is said to be *non-normal*. The non-normal fuzzy set can be normalized by dividing the height of A, i.e., $\bar{\mu}_A(x) = \frac{\mu_A(x)}{\text{hgt}(A)}$.

2.1.7 Fuzzy Subset

A fuzzy set $A = \{(x, \mu_A(x)) | x \in X\}$ is said to be a fuzzy subset of $B = \{(x, \mu_B(x)) | x \in X\}$ if $\mu_A(x) \leq \mu_B(x)$, denoted by $A \subseteq B$.

2.1.8 Fuzzy Partition

For a linguistic variable, a number of fuzzy subsets are enumerated as the value of the variable. This collection of fuzzy subsets is called a *fuzzy partition*. Each fuzzy subset has a MF. For a finite fuzzy partition $\{A_1, A_2, \cdots, A_n\}$ of a set A, the MF for each $x \in A$ satisfies

$$\sum_{i=1}^{n} \mu_{A_i}(x) = 1, \quad (5)$$

and A_i is normal. A fuzzy partition is illustrated in Fig. 1.

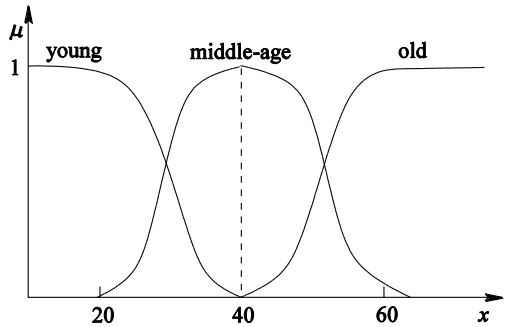


FIGURE 1: A fuzzy partition of human age. The fuzzy set for representing the linguistic variable *human age* is partitioned into three fuzzy subsets, namely, *young*, *middle-age*, *old*.
Each fuzzy subset is characterized by an MF.

2.1.9 Empty Set

The subset of X having no element is called the *empty set*, denoted by $\emptyset$.

2.1.10 Complement

The complement of A, written $\bar{A}$, $\neg A$ or NOT A, is defined as $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$. Thus, $\bar{X} = \emptyset$ and $\bar{\emptyset} = X$.

2.1.11 α-cut

The α-*cut* or α-*level set* of a fuzzy set A, written $\mu_A[\alpha]$, is defined as

$$\mu_A[\alpha] = \{x \in A | \mu_A(x) \geq \alpha\}, \tag{6}$$

where $\alpha \in [0,1]$. For continuous sets, $\mu_A\,[\alpha]$ can be characterized by an interval or a union of intervals.

2.1.12 Kernel or Core

All the elements in a fuzzy set A with membership degree 1 constitute a subset called the kernel or core of the fuzzy set, written as $\mathrm{co}(A) = \mu_A\,[1]$.

2.1.13 Convex Fuzzy Set

A fuzzy set A is said to be convex if and only if

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \mu_A(x_1) \wedge \mu_A(x_2) \tag{7}$$

for $\lambda \in [0,1]$, and $x_1, x_2 \in X$, where $\wedge$ denotes the minimum operation. Any α-cut set of a convex fuzzy set is a closed interval.

2.1.14 Concave Fuzzy Set

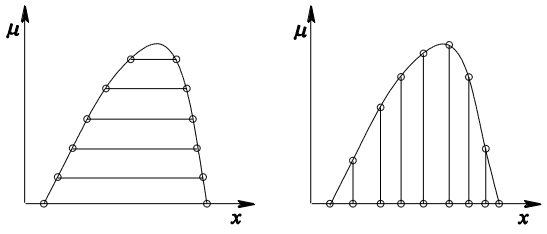
A fuzzy set A is said to be concave if and only if

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \leq \mu_A(x_1) \vee \mu_A(x_2). \tag{8}$$

For $\lambda \in [0,1]$, and $x_1, x_2 \in X$, where $\vee$ denotes the maximum operation.

2.1.15 Fuzzy Number

A fuzzy number A is a fuzzy set of the real line with a normal, convex and continuous MF of bounded support. A fuzzy number is usually represented by a family of α-level sets or by a discretized MF, as illustrated in Fig. 2.



(a) (b)

FIGURE 2: Representations of a fuzzy number. (a) α-level sets. (b) Discretized MF.

2.1.16 Fuzzy Singleton

A fuzzy set $A = \{(x, \mu_A(x)) | x \in X\}$ is said to be a *fuzzy singleton* if $\mu_A(x) = 1$ for $x \in X$ and $\mu_A(\mathrm{x}') = 0$ for all $\mathrm{x}' \in X$ with $\mathrm{x}' \neq x$.

2.1.17 Hedge

A hedge transforms a fuzzy set into a new fuzzy set. A hedge operator is comparable to an adverb in English. Hedges are used to intensify or dilute the characteristic of a fuzzy set such as *very* and *quite*, or to approximate a fuzzy set or convert a scalar to a fuzzy set such as *roughly*. For example, for a fuzzy set *strong* with membership degree $\mu_A(x)$, *very strong* can be described using the membership degree $\mu_A^2(x)$, while *quite strong* can be described using the membership degree $\mu_A^{\frac{1}{2}}(x)$.

2.1.18 Extension Principle

Given mapping $f: X \to Y$ and a fuzzy set $A = \{ (x, \mu_A(x)) | x \in X \}$, the extension principle is given by

$$f(A) = \{ (f(x), \mu_A(x)) | x \in X \}. \quad (9)$$

2.1.19 Cartesian Product

If X and Y are two universal sets, then $X \times Y$ is the set of all ordered pairs (x, y) for $x \in X$ and $y \in Y$. Let A be a fuzzy set of X and B a fuzzy set of Y. The Cartesian product is defined as

$$A \times B = \{ (z, \mu_{A \times B}(z)) | z = (x, y) \in Z, Z = X \times Y \}, \quad (10)$$

where $\mu_{A \times B}(z) = \mu_A(x) \wedge \mu_B(x)$, $\wedge$ denoting the t-norm operation.

2.1.20 Fuzzy Relation

Fuzzy relation is used to describe the association between two things. If R is a subset of $X \times Y$, then R is said to be a relation between X and Y, or a relation on $X \times Y$. Mathematically,

$$R(x, y) = \{ ((x, y), \mu_R(x, y)) | (x, y) \in X \times Y, \mu_R(x, y) \in [0,1] \}, \quad (11)$$

where $\mu_R(x, y)$ is the degree of membership for association between x and y. A fuzzy relation is also a fuzzy set.

2.1.21 Fuzzy Matrix and Fuzzy Graph

Given finite, discrete fuzzy sets $X = \{ x_1, x_2, \cdots, x_m \}$ and $Y = \{ y_1, \ldots, y_n \}$, a fuzzy relation on $X \times Y$ can be represented by an $m \times n$ matrix $\mathbf{R} = [R_{ij}] = [\mu_R (x_i, y_j)]$. This matrix is called a *fuzzy matrix*. The fuzzy relation R can be represented by a fuzzy graph. In a fuzzy graph, all x_i and y_j are vertices, and the grade $\mu_R(x_i, y_j)$ is added to the connection from x_i and y_j.

2.1.22 *t*-norm

A mapping $T: [0,1] \times [0,1] \to [0,1]$ with the following four properties is called *t-norm*. For all $x, y, z \in [0,1]$,

- Commutativity: $T(x, y) = T(y, x)$;
- Monotonicity: $T(x, y) \leq T(x, z)$, if $y \leq z$;
- Associativity: $T(x, T(y, z)) = T(T(x, y), z)$;
- Linearity: $T(x, 1) = x$.

2.1.23 *t*-conorm

A mapping $C: [0,1] \times [0,1] \to [0,1]$ having the following four properties is called *t-conorm*. For all $x, y, z \in [0,1]$,

- Commutativity: $C(x, y) = C(y, x)$;
- Monotonicity: $C(x, y) \leq C(x, z)$, if $y \leq z$;

- Associativity: $C(x, C(y, z)) = C(C(x, y), z)$;
- Linearity: $C(x, 0) = x$.

2.2 Membership Function

A fuzzy set A over the universe of discourse X, $A \subseteq X \to [0,1]$, is described by the degree of membership $\mu_A(x) \in [0,1]$ for each $x \in X$. Unimodality and normality are two important aspects of the MFs [23]. Piecewise-linear functions such as triangles and trapezoids are popular MFs. The triangular MFs can be defined by

$$\mu(x; a, b, c) = \begin{cases} \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b < x \le c \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$

where the shape parameters satisfies $a \le b \le c$, and $b \in X$. Triangular MFs are useful for modeling fuzzy numbers or linguistic terms such as "The temperature is about $20°$C". The trapezoid MFs have flat top with constant value 1. Trapezoid MFs are suitable for modeling such linguistic terms as "He looks like a teenager".

The Gaussian and bell-shaped functions have continuous derivatives, and are usually used to replace the triangular MF when shape parameters are adapted using the gradient-descent procedure. Another popular MF is a sigmoidal functions of the form

$$\mu(x; c, \beta) = \frac{1}{1 + \mathrm{e}^{-\beta(x-c)}}, \tag{13}$$

where c shifts the function to the left or to the right, and β controls the shape of the function. When $\beta > 1$ it is an S-shaped function, and when $\beta < -1$ it is a Z-shaped function. By multiplying an S-shaped function by a Z-shaped function, a π-shaped function is obtained [29]. π-shaped MFs can be used in situations similar to that where trapezoid MFs are used.

2.3 Intersection and Union

The set operations intersection and union correspond to the logic operations conjunction (*AND*) and disjunction (*OR*), respectively. Intersection is described by the so-called triangular norm (t-norm), denoted by $T(x, y)$, whereas union is described by the so-called triangular conorm (t-conorm), denoted by $C(x, y)$.

If A and B are fuzzy subsets of X, then intersection $I = A \cap B$ is defined by

$$\mu_I(x) = T\big(\mu_A(x), \mu_B(x)\big). \tag{14}$$

Basic t-norms are given as the standard intersection, the bound sum, the algebraic product and the drastic intersection [14]. The popular standard intersection and algebraic product are respectively defined by

$$T_{\mathrm{m}}(x, y) = \min(x, y), \tag{15}$$

$$T_{\mathrm{p}}(x, y) = xy. \tag{16}$$

Similarly, union $U = A \cup B$ is defined by

$$\mu_U(x) = C\big(\mu_A(x), \mu_B(x)\big). \tag{17}$$

The corresponding basic t-conorms are given as the standard union, the bounded sum, the algebraic sum, and the drastic union [14]. Corresponding to the standard intersection and algebraic product, the two popular t-conorms are respectively the standard union and the algebraic sum

$$C_{\mathrm{m}}(x,y) = \max(x,y), \tag{18}$$

$$C_{\mathrm{p}}(x,y) = x + y - xy. \tag{19}$$

When the t-norm and the t-conorm satisfy $1 - T(x,y) = C(1-x, 1-y)$, they are said to be *dual*. This makes De Morgan's laws $\overline{\mathrm{A} \cap \mathrm{B}} = \overline{\mathrm{A}} \cup \overline{\mathrm{B}}$ and $\overline{\mathrm{A} \cup \mathrm{B}} = \overline{\mathrm{A}} \cap \overline{\mathrm{B}}$ to still hold in fuzzy set theory. The above t-norms and t-conorms with the same subscripts are dual. To satisfy the principle of duality, they are usually used in pairs.

2.4 Aggregation, Fuzzy Implication, and Fuzzy Reasoning

Aggregation or composition operations on fuzzy sets provide a means for combining several sets in order to produce a single fuzzy set. T-conorms are usually used as aggregation operators. Consider the relations

$$R_1(x,y) = \left\{ \left((x,y), \mu_{R_1}(x,y) \right) \middle| (x,y) \in X \times Y, \mu_{R_1}(x,y) \in [0,1] \right\}, \tag{20}$$

$$R_2(y,z) = \left\{ \left((y,z), \mu_{R_2}(y,z) \right) \middle| (y,z) \in Y \times Z, \mu_{R_2}(y,z) \in [0,1] \right\}. \tag{21}$$

The max-min composition, denoted by $R_1 \circ R_2$ with MF $\mu_{R_1 \circ R_2}$, is defined by

$$R_1 \circ R_2 = \left\{ \left((x,z), \max_y \left\{ \min\left(\mu_{R_1}(x,y), \mu_{R_2}(y,z) \right) \right\} \right) \middle| (x,z) \in X \times Z, y \in Y \right\}. \tag{22}$$

There are some other composition operations, such as the min-max composition, denoted by $R_1 \diamond R_2$ with the difference that the role of max and min are interchanged. The two compositions are related by $\overline{R_1 \diamond R_2} = \overline{R_1} \circ \overline{R_2}$.

Fuzzy implication is used to represent fuzzy rules. It is a mapping $f: A \to B$ according to the fuzzy relation R on $A \times B$

$$\left(y, \mu_B(y)\right) = f\left(\left(x, \mu_A(x)\right) \right). \tag{23}$$

Denote p as "x is A" and q as "y is B", then (23) can be stated as $p \to q$ (if p then q). For a fuzzy rule expressed as a fuzzy implication using the defined fuzzy relation R, the output linguistic variable B is denoted by $B = A \circ R$, which is characterized by $\mu_B(y) = \vee_x \left(\mu_A(x) \wedge \mu_R(x,y) \right)$.

Fuzzy reasoning, also called approximate reasoning, is an inference procedure for deriving conclusions from a set of fuzzy rules and one or more conditions [51]. The compositional rule of inference is the essential rational behind fuzzy reasoning. A simple example of fuzzy reasoning is described here. Consider the fuzzy set $A = \left\{ \left(x, \mu_A(x) \right) \middle| x \in X \right\}$ and the fuzzy relation R on $A \times B$, given by $R(x,y) = \left\{ \left((x,y), \mu_R(x,y) \right) \middle| (x,y) \in X \times Y \right\}$. Fuzzy set B can be inferred from fuzzy set A and their fuzzy relation $R(x,y)$ by the max-min composition

$$B = A \circ R = \left\{ \left(y, \max_x \left\{ \min\left(\mu_A(x), \mu_R(x,y) \right) \right\} \right) \middle| x \in X, y \in Y \right\}. \tag{24}$$

2.5 Fuzzy Inference Systems

In control systems, the inputs to the systems are the error and the change in the error of the feedback loop, while the output is the control action. Fuzzy logic-based controllers are popular control systems. The general architecture of a fuzzy controller is depicted in Fig. 3. The core of a

fuzzy controller is an FIS, in which the data flow involves fuzzification, knowledge base evaluation, and defuzzification. In an FIS, sometimes termed a fuzzy system or a fuzzy model, the knowledge base is comprised of the fuzzy rule base and the database. The database contains the linguistic term sets considered in the linguistic rules and the MFs defining the semantics of the linguistic variables, and information about domains. The rule base contains a collection of linguistic rules that are joined by the ALSO operator. Expert provides his knowledge in the form of linguistic rules. The fuzzification process collects the inputs and then converts them into linguistic values or fuzzy sets. The decision logic, called fuzzy inference engine, generates output from the input, and finally the defuzzification process produces a crisp output for control action.

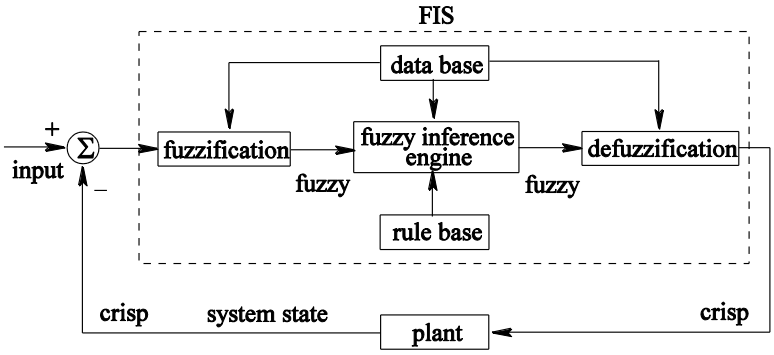


FIGURE 3: The architecture of a fuzzy controller. The core of the fuzzy controller is an FIS.

Interpretations of a certain rule or the rule base depends on the FIS model. The Mamdani [69] and the TSK [103] models are two popular FISs. The Mamdani model is a nonadditive fuzzy model that aggregates the output of fuzzy rules using the maximum operator, while the TSK model is an additive fuzzy model that aggregates the output of rules using the addition operator. Kosko's standard additive model (SAM) [56] is another additive fuzzy model. All these models can be derived from fuzzy graph [122], and are universal approximators [55], [113], [13], [15], [75]. When approximating an unknown control function, neural networks achieve a solution using the learning process, while FISs apply a vague interpolation technique. Unlike neural networks and other numerical models, fuzzy models operate at a level of information granules—fuzzy sets.

2.6 Fuzzy Rules and Fuzzy Interference

Fuzzy mapping rules and fuzzy implication rules are the two types of fuzzy rules [122]. A fuzzy mapping rule describes a functional mapping relationship between inputs and an output using linguistic terms, while a fuzzy implication rule describes a generalized logic implication relationship between two logic formulas involving linguistic variables. Fuzzy implication rules generalize set-to-set implications, whereas fuzzy mapping rules generalize set-to-set associations. The former was motivated to allow intelligent systems to draw plausible conclusions in a way similar to human reasoning, while the latter was motivated to approximate complex relationships such as nonlinear functions in a cost-effective and easily comprehensible way. The foundation of fuzzy mapping rule is fuzzy graph, while the foundation of fuzzy implication rule is a generalization to two-valued logic.

A rule base consists of a number of rules given in the form "IF $condition$, THEN $action$". The condition, also called premise, is made up of a number of antecedents that are negated or combined by different operators such as AND or OR computed with t-norms or t-conorms. In a fuzzy rule system, MFs for fuzzy subsets can be selected according to human intuition, or by learning from training data.

A fuzzy inference is made up of several rules with the same output variables. Given a set of fuzzy rules, the inference result is a combination of the fuzzy values of the conditions and the corresponding actions. For example, we have a set of N_{r} rules

$$\mathrm{R}_i\text{: IF } (condition = C_i) \text{ THEN } (action = A_i)$$

for $i = 1, \dots, N_{\mathrm{r}}$, where C_i is a fuzzy set. Assuming that a condition has a membership degree of μ_i associated with the set C_i. The condition is first converted into a fuzzy category using a syntactical representation, $condition = \sum_i^{N_{\mathrm{r}}} \frac{C_i}{\mu_i}$. We can see each rule is valid to a certain extent. A fuzzy inference is the combination of all the possible consequences. The action coming from a fuzzy inference is also a fuzzy category, with a syntactical representation

$$action = \frac{A_1}{\mu_1} + \frac{A_2}{\mu_2} + \cdots + \frac{A_{N_{\mathrm{r}}}}{\mu_{N_{\mathrm{r}}}}. \tag{25}$$

The inference procedure depends on fuzzy reasoning. This result can be further processed or transformed into a crisp value.

2.7 Fuzzification and Defuzzification

Fuzzification is to transform crisp inputs into fuzzy subsets. Given crisp inputs x_i, $i = 1, \dots, n$, fuzzification is to construct the same number of fuzzy sets A^i,

$$A^i = \mathrm{fuzz}(x_i), \tag{26}$$

where $\mathrm{fuzz}(\cdot)$ is a fuzzification operator. Fuzzification is determined according to the defined MFs.

Defuzzification is to map fuzzy subsets of real numbers into real numbers. In an FIS, defuzzification is applied after aggregation. Popular defuzzification methods include the centroid defuzzifier [69], and the mean-of-maxima defuzzifier [69]. The centroid defuzzifier is the best-known method, which is to find the centroid of the area surrounded by the MF and the horizontal axis [52]. Aggregation and defuzzification can be combined into a single phase, such as the weighted-mean method [36]

$$\mathrm{defuzz}(B) = \frac{\sum_{i=1}^{N_{\mathrm{r}}} \mu_i b_i}{\sum_{i=1}^{N_{\mathrm{r}}} \mu_i}, \tag{27}$$

where N_{r} is the number of rules, μ_i is the degree of activation of the ith rule, and b_i is a numeric value associated with the consequent of the ith rule, B_i. The parameter b_i can be selected as the mean value of the α-level set when α is equal to μ_i [36].

2.8 Mamdani Model

Given a set of N examples $\{(\mathbf{x}_p, \mathbf{y}_p) | \mathbf{x}_p \in R^n, \mathbf{y}_p \in R^m\}$, the underlying system can be identified by using the Mamdani or the TSK model.

For the Mamdani model with N_{r} rules, the ith rule is given by

$$\mathrm{R}_i\text{: IF } \mathbf{x} \text{ is } A_i, \text{ THEN } \mathbf{y} \text{ is } B_i$$

for $i = 1, \dots, N_{\mathrm{r}}$, where $A_i = \{A_i^1, A_i^2, \dots, A_i^n\}$, $B_i = \{B_i^1, B_i^2, \dots, B_i^m\}$, and A_i^j and B_i^k are respectively fuzzy sets that define an input and output space partitioning.

For an n-tuple input in the form of "$\mathbf{x}$ is A'", the system output "$\mathbf{y}$ is B'" is characterized by combining the rules according to

$$\mu_{B'}(\mathbf{y}) = \bigvee_{i=1}^{N_\mathrm{r}} \left(\mu_{A_i'}(\mathbf{x}) \wedge \mu_{B_i}(\mathbf{y}) \right), \tag{28}$$

where the fuzzy partitioning $A' = \{A'^1, A'^2, \ldots, A'^n\}$ and $B' = \{B'^1, B'^2, \ldots, B'^m\}$,

$$\mu_{A_i'}(\mathbf{x}) = \mu_{A'}(\mathbf{x}) \wedge \mu_{A_i}(\mathbf{x}) = \bigwedge_{i=1}^{n} \left(\mu_{A'^j} \wedge \mu_{A_i^j} \right). \tag{29}$$

$\mu_{A'}(\mathbf{x}) = \wedge_{j=1}^{n} \mu_{A'^j}$ and $\mu_{A_i}(\mathbf{x}) = \wedge_{j=1}^{n} \mu_{A_i^j}$ being respectively the membership degrees of $\mathbf{x}$ to the fuzzy sets A' and A_i, $\mu_{B_i}(\mathbf{y}) = \wedge_{k=1}^{m} \mu_{B_i^k}$ is the membership degree of $\mathbf{y}$ to the fuzzy set B_i, $\mu_{A'^j_i}$ is the association between the jth input of A' and the ith rule, $\mu_{B_i^k}$ is the association between the kth input of B and the ith rule, $\wedge$ is the intersection operator, and $\vee$ is the union operator.

When minimum and maximum are respectively used as the intersection and union operators, the Mamdani model is called a max-min model. We now illustrate the inference procedure for the Mamdani model. Assume that we have a two-rule Mamdani FIS with the rules of the form

R_i: IF x_1 is A_i and x_2 is B_i, THEN y is C_i

for $i = 1,2$. When the max-min composition is employed, for the inputs "x_1 is A'" and "x_2 is B'", the fuzzy reasoning procedure for the output y is illustrated in Fig. 4. A defuzzification strategy is needed to get crisp output value.

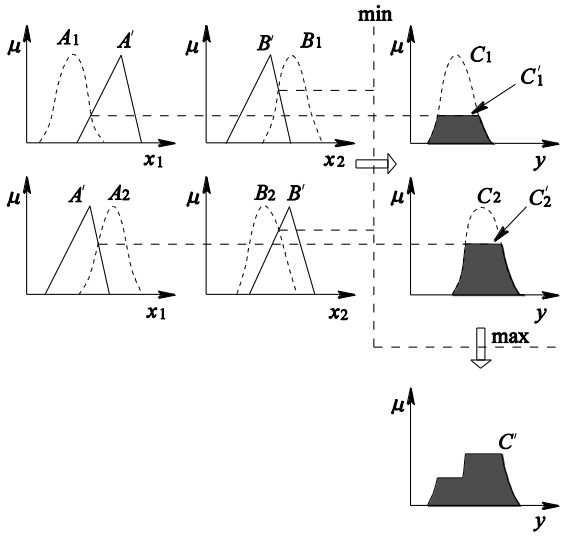


FIGURE 4: The inference procedure of the Mamdani model with the min and max operators and fuzzy inputs.

The Mamdani model offers a high semantic level and a good generalization capability. It contains fuzzy rules built from expert knowledge. However, FISs based only on expert knowledge may result in insufficient accuracy. For accurate numerical approximation, the TSK model can usually generate a better performance.

2.9 Takagi-Sugeno-Kang Model

In the TSK model [103], for the same set of examples $\{(\mathbf{x}_\mathrm{p}, \mathbf{y}_\mathrm{p})\}$, fuzzy rules are given in the form

$$\mathrm{R}_i\text{: IF } \mathbf{x} \text{ is } A_i\text{, THEN } \mathbf{y} = \mathbf{f}_i(\mathbf{x})$$

for $i = 1,2,\dots,N_\mathrm{r}$, where $\mathbf{f}_i(\mathbf{x}) = \left(f_i^1(\mathbf{x}),\dots,f_i^m(\mathbf{x})\right)^T$ is a crisp vector function of $\mathbf{x}$; usually $f_i^j(\mathbf{x})$ is selected as a linear relation with $f_i^j(\mathbf{x}) = \left(\mathbf{a}_i^j\right)^T\mathbf{x} + b_i^j$, where $\mathbf{a}_i^j$ and b_i^j are adjustable parameters.

For an n-tuple input in the form of "$\mathbf{x}$ is A'", the output $\mathbf{y}'$ is obtained by combining the rules according to

$$\mathbf{y}' = \frac{\sum_{i=1}^{N_\mathrm{r}} \mu_{A_i'}(\mathbf{x})\mathbf{f}_i(\mathbf{x})}{\sum_{i=1}^{N_\mathrm{r}} \mu_{A_i'}(\mathbf{x})}, \tag{30}$$

where $\mu_{A_i'}(\mathbf{x})$ is defined by (29), and can be derived by the procedure shown in the left part of Fig. 4. This model produces a real-valued function, and it is essentially a model-based fuzzy control method. The stability analysis of the TSK model is given in [104]. The TSK model typically selects $f_i^j(\cdot)$ as first-order polynomials, hence the model termed the first-order TSK model. When $f_i^j(\cdot)$ are selected as constants, it is called the zero-order TSK model and can be regarded as a special case of the Mamdani model.

In comparison with the Mamdani model, the TSK model, which is based on automatic learning from the data, can accurately approximate a function using fewer rules. It has a stronger and more flexible representation capability than the Mamdani mode. In the TSK model, rules are extracted from the data, but the generated rules may have no meaning for experts. The TSK model has found more successful applications in building fuzzy systems.

2.10 Complex Fuzzy Logic

Complex fuzzy sets and logic are mathematical extensions of fuzzy sets and logic from the real domain to the complex domain [87], [86]. A complex fuzzy set S is characterized by a complex-valued MF, and membership of any element x in S is given by a complex-valued membership degree of the form

$$\mu_S(x) = r_S(x)\mathrm{e}^{\mathrm{j}\varphi_S(x)}, \tag{31}$$

where the amplitude $r_S(x) \in [0,1]$, and φ_S is the phase. Thus, $\mu_S(x)$ is within a unit circle in the complex plane.

In [87], [86], basic set operators for fuzzy logic have been extended for the complex fuzzy logic, and some additional operators such as the vector aggregation, set rotation and set reflection, are also defined. The operations of intersection, union and complement for complex fuzzy sets are defined only on the modulus of the complex membership degree. In [27], the complex fuzzy logic is extended to a logic of vectors in the plane, rather than scalar quantities. In [74], a complex fuzzy set is defined as an MF mapping the complex plane into $[0,1] \times [0,1]$.

Complex fuzzy sets are superior to the Cartesian products of two fuzzy sets. Complex fuzzy logic maintains both the advantages of the fuzzy logic and the properties of complex fuzzy sets. In complex fuzzy logic, rules constructed are strongly related and a relation manifested in the phase term is associated with complex fuzzy implications. In a complex FIS, the output of each rule is a complex fuzzy set, and phase terms are necessary when combining multiple rules so as to generate the final output. Complex FISs are useful for solving some hard problems for traditional fuzzy methods, in which rules are related to one another with the nature of the relation varying as a function of the input to the system [86].

The fuzzy complex number [11], introduced by incorporating the complex number into the support of the fuzzy set, is a different concept from the complex fuzzy set [87]. A fuzzy complex number is

a fuzzy set of complex numbers, which have real-valued membership degree in the range $[0,1]$. An α-cut of a fuzzy complex number is based on the modulus of the complex numbers in the fuzzy set. A fuzzy complex number is a fuzzy set in one dimension, while a complex fuzzy set or number is a fuzzy set in two dimensions.

3. FUZZY LOGIC VS. NEURAL NETWORKS

Like FNNs, many fuzzy systems are proved to be universal approximators [63], [50], [13], [35], [57], [118]. In [63], the Mamdani model and FNNs are shown to be able to approximate each other to an arbitrary accuracy. The equivalence between the TSK model and the RBFN under certain conditions has been established in [50], [43] and the equivalence between fuzzy expert systems and neural networks has been proved in [13]. Gaussian-based Mamdani systems have the ability of approximating any sufficiently smooth function and reproducing its derivatives up to any order [35]. In [57], fuzzy systems with Gaussian MFs have been proved to be universal approximators for a smooth function and its derivatives.

From the viewpoint of an expert system, fuzzy systems and neural networks are quite similar as inference systems. An inference system involves knowledge representation, reasoning, and knowledge acquisition: (1) A trained neural network represents knowledge using connection weights and neurons in a distributed manner, while in a fuzzy system knowledge is represented using IF-THEN rules; (2) For each input, the trained neural network generates an output and this pure numerical procedure can be treated as a reasoning process, while reasoning in a fuzzy system is logic-based; (3) Knowledge acquisition is via learning in a neural network, while for a fuzzy system knowledge is encoded by a human expert. Both neural networks and fuzzy systems are dynamic, parallel distributed processing systems that estimate functions without any mathematical model and learn from experience with sample data.

Fuzzy systems can be applied to problems with knowledge represented in the form of IF-THEN rules. Problem-specific a priori knowledge can be integrated into the systems. Training pattern set and system modeling are not needed, and only heuristics are used. During the tuning process, one needs to add, remove, or change a rule, or even change the weight of a rule. This process, however, requires the knowledge of experts. On the other hand, neural networks are useful when we have training pattern set. We do not need any knowledge of the modeling of the problem. A trained neural network is a black box that represents knowledge in its distributed structure. However, any prior knowledge of the problem cannot be incorporated into the learning process. It is difficult for human beings to understand the internal logic of the system. Nevertheless, by extracting rules from neural networks, users can understand what neural networks have learned and how neural networks predict.

4. FUZZY INFERENCE SYSTEMS AND NEURAL NETWORKS

4.1 Fuzzy Inference Systems and Multilayer Perceptrons

For a three-layer (J_1-J_2-J_3) MLP, if the activation function in the hidden layer $\phi^{(1)}(\cdot)$ is selected as the logistic function $\phi^{(1)}(x) = \frac{1}{1+e^{-x}}$ and the activation function in the output layer $\phi^{(2)}(\cdot)$ is selected as the linear function $\phi^{(2)}(x) = x$, there always exists a fuzzy additive system that calculates the same function as the network does [7]. In [7], a fuzzy logic operator, called interactive-or (i-or), is defined by applying the concept of f-duality to the logistic function. The use of the i-or operator explains clearly the acquired knowledge of a trained MLP. The i-or operator is defined by [7]

$$a \otimes b = \frac{a \cdot b}{(1-a) \cdot (1-b) + a \cdot b}. \quad (32)$$

The i-or operator works on $(0,1)$. It is a hybrid between both a t-norm and a t-conorm. Based on the i-or operator, the equality between MLPs and FISs is thus established [7]. The equality proof also yields an automated procedure for knowledge acquisition. An extension of the method has been presented in [16], where the fuzzy rules obtained are in agreement with the domain of the input variables and a new logical operator, similar to, but with a higher representational power than the i-or, is defined.

In [32], relations between input uncertainties and fuzzy rules have been established. Sets of crisp logic rules applied to uncertain inputs are shown to be equivalent to fuzzy rules with sigmoidal MFs applied to crisp inputs. Integration of a reasonable uncertainty distribution for a fixed rule threshold or interval gives a sigmoidal MF. Crisp logic and fuzzy rule systems are shown to be respectively equivalent to the logical network and the three-layer MLP. Keeping fuzziness on the input side enables easier understanding of the networks or the rule systems. In [17], [100], MLPs are interpreted by fuzzy rules in such a way that the sigmoidal activation function is decomposed into three partitions, and represented by three TSK fuzzy rules with one TSK fuzzy rule for each partition. Each partition has its own MF. Accordingly, the value of the activation function at a point can be derived by the TSK model.

A fuzzy set is usually represented by a finite number of its supports. In comparison with conventional MF based FISs, α-cut based FISs [109] have a number of advantages. They can considerably reduce the required memory and time complexity, since they depend on the number of membership-grade levels, and not on the number of elements in the universes of discourse. Secondly, the inference operations can be performed for each α-cut set independently, and this enables parallel implementation. An α-cut based FIS can also easily interface with two-valued logic since the α-level sets themselves are crisp sets. In addition, fuzzy set operations based on the extension principle can be performed efficiently using α-level sets [109], [64]. For α-cut based FISs, each fuzzy rules can be represented as a pattern pair of degrees of membership at those points of the MFs obtained by dividing the intervals of the fuzzy sets linearly or by α-cut can be implemented by an MLP with the backpropagation (BP) rule. This is a learning problem of N_{r} samples with n inputs and m outputs.

4.2 Fuzzy Inference Systems and Radial Basis Function Networks

When the t-norm in the TSK model is selected as multiplication and the MFs are selected the same as RBFs in the normalized RBFN model, the two models are mathematically equivalent [50], [48]. Note that each hidden unit corresponds to a fuzzy rule. Normalized RBFNs provide a localized solution that is amenable to rule extraction. The receptive fields of some RBFs should overlap to prevent incompleteness of fuzzy partitions. To have a perfect match between the RBFs $\phi\left(\|\mathbf{x}-\mathbf{c}_i\|\right)$ and $\mu_{A_i'}(\mathbf{x})$ in (30), $\phi\left(\|\mathbf{x}-\mathbf{c}_i\|\right)$ should be factorizable in each dimension such that each component $\phi\left(|x_j - c_{i,j}|\right)$ corresponds to an MF $\mu_{\boldsymbol{A'^j}}$. The Gaussian RBF is the only strictly factorizable function.

In the normalized RBFN, w_{ij} 's typically take constant values and the normalized RBFN corresponds to the zero-order TSK model. When the RBF weights are linear regression functions of the input variables [59], [91], the model is functionally equivalent to the first-order TSK model.

When implementing the TSK model, one can select some $\mu_{A'^j_i} = 1$ or some $\mu_{A'^j_i} = \mu_{A'^j_k}$ in order to

increase the distinguishability of the fuzzy partitions. Correspondingly, one should share some component RBFs or set some component RBFs to unity [52]. This considerably reduces the effective number of free parameters in the RBFN. A distance measure like the Euclidean distance is used to describe the similarity between two component RBFs. After applying a clustering technique to locate prototypes and adding a regularization term describing the total similarity between all the RBFs and the shared RBF to the MSE function, a gradient-descent procedure is conducted so as to extract interpretable fuzzy rules from a trained RBFN [52]. The method can be applied to RBFNs with constant or linear regression weights. A fuzzy system can be first constructed according to heuristic knowledge and existing data, and then converted into an RBFN. This is followed by a refinement of the RBFN using a learning algorithm. Due to this learning procedure, the interpretability of the original fuzzy system may be lost. The RBFN is then again converted into interpretable fuzzy system, and knowledge is extracted from the network. This process refines the original fuzzy system design. The algorithm for rule extraction from the RBFN is given in [52].

In [107], normalized Gaussian RBFNs can be generated from simple probabilistic rules and probabilistic rules can also be extracted from trained RBFNs. Methods for reducing network complexity have been presented in order to obtain concise and meaningful rules. Two algorithms for rule extraction from RBFNs, which respectively generate a single rule describing each class and a single rule from each hidden unit, are given in [70]. Existing domain knowledge in rule format can be inserted into an RBFN as an initialization of optimal network training.

4.3 Rule Generation from Trained Neural Networks

In addition to rule generation from trained MLPs and RBFNs, rule generation can also be performed on other trained neural networks [46], [106]). Rule generation involves rule extraction and rule refinement. Rule extraction is to extract knowledge from trained neural networks, while rule refinement is to refine the rules that are extracted from neural networks and initialized with crude domain knowledge.

Recurrent neural networks (RNNs) have the ability to store information over indefinite periods of time, develop hidden states through learning, and thus conveniently represent recursive linguistic rules [72]. They are particularly well-suited for problem domains, where incomplete or contradictory prior knowledge is available. In such cases, knowledge revision or refinement is also possible. Discrete-time RNNs can correctly classify strings of a regular language [80]. Rules defining the learned grammar can be extracted in the form of deterministic finite-state automata (DFAs) by applying clustering algorithms [29] in the output space of neurons. Starting from an initial network state, the algorithm searches the equally partitioned output space of N state neurons in a breadth-first manner. A heuristic is used to choose among the consistent DFAs that model, which best approximates the learned regular grammar. The extracted rules demonstrate high accuracy and fidelity and the algorithm is portable. Based on [80], an augmented RNN that encodes fuzzy finite-state automata (FFAs) and recognizes a given fuzzy regular language with an arbitrary accuracy has been constructed in [81]. FFAs are transformed into equivalent DFAs by using an algorithm that computes fuzzy string membership. FFAs can model dynamical processes whose current state depends on the current input and previous states. The granularity within both extraction techniques is at the level of ensemble of neurons, and thus, the approaches are not strictly decompositional.

RNNs are suitable for crisp/fuzzy grammatical inference. A method that uses a SOM for extracting knowledge from an RNN [9] is able to infer a crisp/fuzzy regular language. Rule extraction is also carried out upon Kohonen networks [110]. A comprehensive survey on rule generation from trained neural networks is given from a softcomputing perspective in [72], where the optimization capability of evolutionary algorithms (EAs) are emphasized for rule refinement.

Rule extraction from RNNs aims to find models of an RNN, typically in the form of finite state machines. A recent overview of rule extraction from RNNs is given in [47].

4.4 Extracting Rules from Numerical Data

FISs can be designed directly from expert knowledge and data. The design process is usually decomposed into two phases, namely, rule generation and system optimization [39]. Rule generation leads to a basic system with a given space partitioning and the corresponding set of rules, while system optimization gives the optimal membership parameters and rule base. Design of fuzzy rules can be performed in one of three ways, namely, all the possible combinations of fuzzy partitions, one rule for each data pair, or dynamically choosing the number of fuzzy sets.

For good interpretability, a suitable selection of variables and the reduction of the rule base are necessary. During the system optimization phase, merging techniques such as cluster merging and fuzzy set merging are usually used for interpretability purposes. Fuzzy set merging leads to a higher interpretability than cluster merging. The reduction of a set of rules results in a loss of numerical performance on the training data set, but a more compact rule base has a better generalization capability and is also easier for human understanding. EAs [93] or learning [50] are also used for extracting fuzzy rules and optimizing MFs and rule base. Methods for designing FISs from data are analyzed and surveyed in [39]. They are grouped into several families and compared based on rule interpretability.

4.5 Rule Generation Based on Fuzzy Partitioning

Rule generation can be based on a partitioning of the multidimensional space. Fuzzy partitioning corresponds to structure identification for FISs, followed by parameter identification using a learning algorithm. There are usually three methods for partitioning the input space, namely, grid partitioning, tree partitioning, and scatter partitioning. These partitioning methods in the two-dimensional input space are illustrated in Fig. 5.

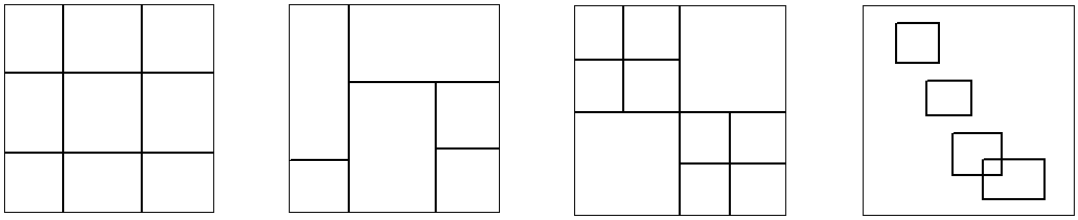


FIGURE 5: Partitioning of the two-dimensional input space. (a) Grid partitioning. (b) k-d tree partitioning. (c) Multilevel grid partitioning. (d) Scatter partitioning.

4.6 Grid Partitioning

The grid structure has easy interpretability and is most widely used for generating fuzzy rules. Fuzzy sets of each variable are shared by all the rules. However, the number of fuzzy rules grows exponentially with input dimension, namely, the curse-of-dimensionality problem. For n input variables, each being partitioned into m_i fuzzy sets, a total of $\prod_{i=1}^{n} m_i$ rules are needed to cover the whole input space. Since each rule has a few parameters to adjust, there are too many parameters to adapt during the learning process. Too many fuzzy rules also harm the interpretability of the fuzzy system. Thus, the method is appropriate for a small dimensional data set with a good coverage. A training procedure can be applied to optimize the grid structure and the rule consequences [50]. The grid structure is illustrated in Fig. 5 (a).

4.7 Tree Partitioning

k-d tree and multilevel grid structures are two hierarchical partitioning techniques that effectively relieve the problem of rule explosion [101]. The input space is first partitioned roughly, and a subspace is recursively divided until a desired approximation performance is achieved. The k-d tree results from a series of guillotine cuts. A guillotine cut is a cut that is entirely across the subspace to be partitioned. After the ith guillotine cut, the entire space is partitioned into $i + 1$ regions. Heuristics based on the distribution of training examples or parameter identification methods can usually be employed to find a proper k-d tree structure [101]. For the multilevel grid structure [101], the top-level grid coarsely partitions the whole space into equal-sized and evenly spaced fuzzy boxes, which are recursively partitioned into finer grids until a criterion is met. Hence, a multilevel grid structure is also called a box tree. The criterion can be that the resulting boxes have similar number of training examples or that an application-specific evaluation in each grid is below a threshold. A k-d tree partitioning and a multilevel grid partitioning are respectively illustrated in Fig. 5 (b) and (c). A multilevel grid in the two-dimensional space is called a quad tree. Tree partitioning needs some heuristics to extract rules and its application to high-dimensional problems faces practical difficulties.

4.8 Scatter Partitioning

Scatter partitioning usually generates fewer fuzzy regions than the grid and tree partitioning techniques owing to the natural clustering property of training patterns. Fuzzy clustering algorithms form a family of rule generation techniques. The training examples are gathered into homogeneous groups and a rule is associated to each group. The fuzzy sets are not shared by the rules, but each of them is tailored for one particular rule. Thus, the resulting fuzzy sets are usually difficult to interpret [39]. Clustering is well adapted for large work spaces with a small amount of training examples. However, scatter partitioning of high-dimensional feature spaces is difficult, and some learning or evolutionary procedures may be necessary. Clustering algorithms [29] can be applied for scatter partitioning. A scatter partitioning is illustrated in Fig. 5 (d). The curse of dimensionality can also be alleviated by reducing the input dimensions by discarding some irrelevant inputs or compressing the input space using feature selection or feature extraction techniques. Some clustering-based methods for extracting fuzzy rule for function approximation are proposed in [121], [20], [21], [4]. These methods are based on the TSK model. Clustering can be used for identification of the antecedent part of the model such as determination of the number of rules and initial rule parameters. The consequent part of the model can be estimated by the linear LS method. In [21], the combination of the subtractive clustering with the linear LS method provides an extremely fast and accurate method for fuzzy system identification, which is better than the adaptive-network-based FIS (ANFIS) [48]. Based on the Mamdani model, a clustering-based method for nonlinear regression is also given in [117].

4.9 Hierarchical Rule Generation

Hierarchical structure for fuzzy rule systems can also effectively solve the rule explosion problem [85], [114], [68]. A hierarchical fuzzy system is comprised of a number of low-dimensional fuzzy systems such as TSK systems connected in a hierarchical fashion. The total number of rules increases only linearly with the number of input variables. For example, for a hierarchical fuzzy system shown in Fig. 6, if there are n variables each of which is partitioned into m_i fuzzy subsets, the total number of rule is only $\sum_{i=1}^{n-1} m_i\, m_{i+1}$. Hierarchical TSK systems [114] and generalized hierarchical TSK systems [68] are universal approximators of any continuous function defined on a compact set.

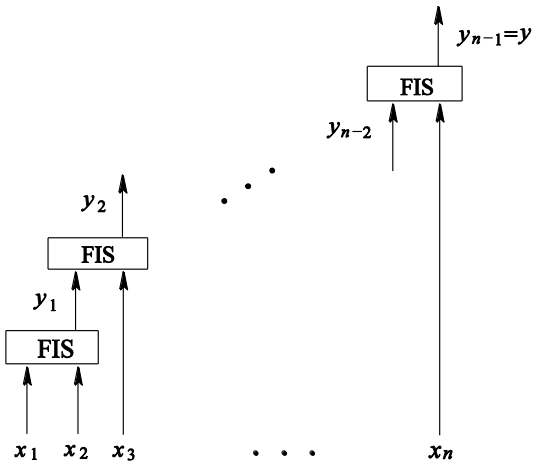


FIGURE 6: Example of a hierarchical fuzzy system with n inputs and one output. The system is comprised of $n-1$ two-input TSK systems. The n input variables are x_i, $i = 1, \ldots, n$, the output is denoted by y, and y_i is the output of the ith TSK system.

In Fig. 6, the n input variables are x_i, $i = 1, \ldots, n$, and the output is denoted by y. There exist relations

$$y_i = f_i(y_{i-1}, x_{i+1}) \tag{33}$$

for $i = 1, \ldots, n-1$, where f_i is the nonlinear relation described by the ith TSK system, y_i is the output of the ith TSK system, and $y_0 = x_1$. The final output is $y = y_{n-1}$. The output y is easily obtained by a recursive procedure. Thus, the inference in the hierarchical fuzzy system is in a recursive manner.

The hierarchical fuzzy system reduces the number of rules, however, the curse of dimensionality is inherent in the system. In the standard fuzzy system, the degree of freedom is unevenly distributed over the IF and THEN parts of the rules, with a comprehensive IF part to cover the whole domain and a simple THEN part. The hierarchical fuzzy system, on the other hand, provides with an incomplete IF part but a more complex THEN part. The gradient-descent method can be applied to parameter learning of these systems. Generally, conventional fuzzy systems achieve universal approximation using piecewise-linear functions, while the hierarchical fuzzy system achieves it through piecewise-polynomial functions [114], [68].

4.10 Rule Generation Based on Look-up Table

Designing fuzzy systems from pattern pairs is a nonlinear regression problem. In the simple look-up table (LUT) technique [115], [117], each pattern pair generates one fuzzy rule and then a selection process determines the important rules, which are used to construct the final fuzzy system. In the LUT technique, the input MFs do not change with the sampling data, thus the designed fuzzy system uniformly covers the domain of interest.

In the LUT technique, the input and output spaces are first divided into fuzzy regions, then a fuzzy rule is generated from a given pattern pair, and finally a degree is assigned to each rule to resolve rule conflicts and reduce the number of rules. When the number of examples is large, there is a high probability of conflicting rules, i.e., rules with the same IF parts but different THEN parts. Each rule is assigned a degree of fulfillment. For a group of conflicting rules, only the rule with the maximum degree is retained. When a new pattern pair becomes available, a rule is created for this pattern pair and the fuzzy rule base is updated. The generated rules as well as human expert's knowledge in the form of linguistic rules can be combined so as to produce a

fuzzy rule base. Finally a fuzzy system is built. The LUT technique is implemented in five steps given in [29], [115], [117].

The fuzzy system thus constructed is proved to be a universal approximator by using the Stone-Weierstrass theorem [115]. The approach has the advantage that modification of the rule base is very easy as new examples are available. It is a simple and fast one-pass procedure, since no iterative training is required. Naturally, this algorithm produces an enormous number of rules, when the total input data is considerable. There also arises the problem of contradictory rules, and noisy data in the training examples will affect the consequence of a rule. A similar grid partitioning-based method in which each datum generates one rule has also been derived in [1].

4.11 Other Methods

Many other general methods can be used to automatically extract fuzzy rules from a set of numerical examples and to build a fuzzy system for function approximation; some of these are heuristics-based approaches [42], [92], [28], [105], and hybrid neural-fuzzy approaches such as the ANFIS [48]. In [42], a framework for quickly prototyping an expert system from a set of numerical examples is established. In [92], the fuzzy system can be built in a constructive way. Starting from an initially simple system, the number of MFs in the input domain and the number of rules are adapted in order to reduce the approximation error. A function approximation problem can also be first converted into a pattern classification problem, and then solved by using a fuzzy system [28], [105].

5. FUZZY AND NEURAL: A SYNERGY

While neural networks have strong learning capabilities at the numerical level, it is difficult for the users to understand them at the logic level. Fuzzy logic, on the other hand, has a good capability of interpretability and can also integrate expert's knowledge. The hybridization of both the paradigms yields the capabilities of learning, good interpretation and incorporating prior knowledge. The combination can be in different forms. The simplest form may be the concurrent neuro-fuzzy model, where a fuzzy system and a neural network work separately. The output of one system can be fed as the input of the other system. The cooperative neuro-fuzzy model corresponds to the case that one system is used to adapt the parameters of the other system [38], [94]. The hybrid neural-fuzzy model is the true synergy that captures the merits of both the systems. It takes the form of either a fuzzy neural network or a neuro-fuzzy system. A hybrid neural-fuzzy system does not use multiplication, addition, or the sigmoidal function, but uses fuzzy logic operations such as t-norm and t-conorm.

A fuzzy neural network [84] is a neural network equipped with the capability of handling fuzzy information, where the input signals, activation functions, weights, and/or the operators are based on the fuzzy set theory. Thus, symbolic structure is incorporated. The network can be represented in an equivalent rule-based format, where the premise is the concatenation of fuzzy AND and OR logic, and the consequence is the network output. Two types of fuzzy neurons, namely AND neuron and OR neuron, are defined. The NOT logic is integrated into the weights. Weights always have values in the interval $[0,1]$, and negative weight is achieved by using the NOT operator. The weights of the fuzzy neural network can be interpreted as calibration factors of the conditions and rules. A neuro-fuzzy system is a fuzzy system, whose parameters are learned by a learning algorithm. It has a neural network architecture constructed from fuzzy reasoning, and can always be interpreted as a system of fuzzy rules. Learning is used to adaptively adjust the rules in the rule base, and to produce or optimize the MFs of a fuzzy system. Structured knowledge is codified as fuzzy rules. Expert knowledge can increase learning speed and estimation accuracy. Both fuzzy neural networks and neuro-fuzzy systems can be treated as neural networks, where the units employ the t-norm or t-conorm operator instead of an activation function. The hidden layers represent fuzzy rules. The line between the two hybrid models is blurred, and we call both types of synergisms as neuro-fuzzy systems.

Neuro-fuzzy systems can be obtained by representing some of the parameters of a neural network, such as the inputs, weights, outputs, and shift terms as continuous fuzzy numbers. When only the input is fuzzy, it is a Type I neuro-fuzzy system. When everything except the input is fuzzy, we get a Type II model. A type III model is defined as one where the inputs, weights, and shift terms are all fuzzy. The functions realizing the inference process, such as t-norm and t-conorm, are usually nondifferentiable. To utilize gradient-based algorithms, one has to select differential functions for the inference functions. For nondifferentiable inference functions, training can be performed by using EAs. The shape of the MFs, the number of fuzzy partitions, and rule base can all be evolved by using EAs. The neuro-fuzzy method is superior to the neural network method in terms of the convergence speed and compactness of the structure. Fundamentals in neuro-fuzzy synergism for modeling and control have been reviewed in [51].

5.1 Interpretability

Interpretability is one major reason for using fuzzy systems. Interpretability helps to check the plausibility of a system, leading to easy maintenance of the system. It can also be used to acquire knowledge from a problem characterized by numerical examples. An improvement in interpretability can enhance the performance of generalization when the data set is small. The interpretability of a rule base is usually related to continuity, consistency and completeness [39]. Continuity guarantees that small variations of the input do not induce large variations in the output. Consistency means that if two or more rules are simultaneously fired, their conclusions are coherent. Completeness means that for any possible input vector, at least one rule is fired and there is no inference breaking.

When neuro-fuzzy systems are used to model nonlinear functions described by training sets, the approximation accuracy can be optimized by the learning procedure. However, since learning is accuracy-oriented, it usually causes a reduction in the interpretability of the generated fuzzy system. The loss of interpretability can be due to incompleteness of fuzzy partitions, indistinguishability of fuzzy partitions, inconsistancy of fuzzy rules, too fuzzy or too crisp fuzzy subsets, or incompactness of the fuzzy system [52]. To improve the interpretability of neuro-fuzzy systems, one can add to the cost function, regularization terms that apply constraints on the parameters of fuzzy MFs. For example, the order of the L centers of the fuzzy subset $A^j(x)$, $j = 1, \ldots, L$, should be specified and remain unchanged during learning. Similar MFs should be merged to improve the distinguishability of fuzzy partitions and to reduce the number of fuzzy subsets [96]. One can also reduce the number of free parameters in defining fuzzy subsets. To increase the interpretability of the designed fuzzy system, the same linguistic term should be represented by the same MF. This results in weight sharing [75], [52]. For the TSK model, one practice for good interpretability is to keep the number of fuzzy subsets much smaller than N_r, the number of fuzzy rules, especially when N_r is large.

6. NEURO-FUZZY MODELS

A typical architecture of a neuro-fuzzy system includes an input layer, an output layer, and several hidden layers. The weights are fuzzy sets, and the neurons apply t-norm or t-conorm operations. The hidden layers are usually used as rule layers. The layers before the rule layers perform as premise layers, while those after perform as consequent layers. A well-known neuro-fuzzy model is the ANFIS model [48]. We describe the ANFIS model in this section and also give a brief survey of neuro-fuzzy models.

6.1 ANFIS Model

The ANFIS model [50], [48], [51], as shown in Fig. 7, has a five-layer (n-K-K-K-1) architecture, and is a graphical representation of the TSK model. The functions of the various layers are given below.

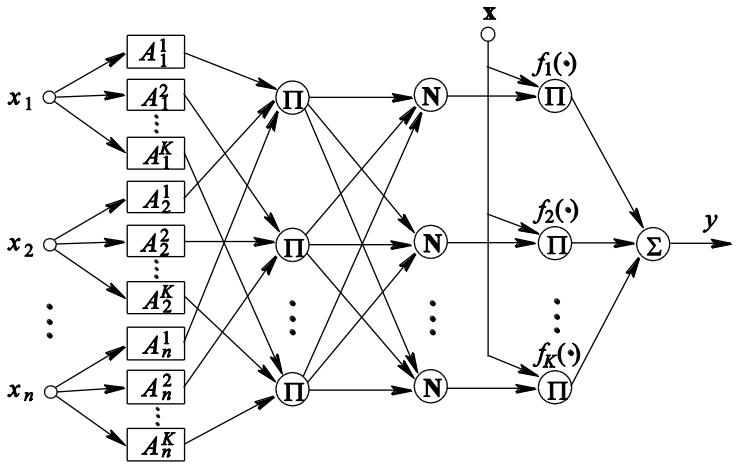


FIGURE 7: ANFIS: graphical representation of the TSK model. The symbol N in the circles denotes the normalization operator, and $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$.

Layer 1 is the input layer with n nodes. The weights between the first two layers, $w_{ij} = \mu_{A_j^i}(x_i)$, $i = 1, \dots, n$, $j = 1, \dots, K$, denotes membership values of the ith input (antecedent) of the jth rule, where A_j^i corresponds to a partition of the space of x_i, and $\mu_{A_j^i}(x_i)$ is typically selected as a generalized bell MF $\mu_{A_j^i}(x_i) = \mu(x_i; c_i^j, a_i^j, b_i^j)$, where c_i^j, a_i^j, and b_i^j are referred to as premise parameters. Layer 2 has K fuzzy neurons with the product t-norm as the aggregation operator. Each node corresponds to a rule, and the output of the jth neuron determines the degree of fulfillment of the jth rule

$$o_j^{(2)} = \prod_{i=1}^{n} \mu_{A_j^i}(x_i) \tag{34}$$

for $j = 1, \dots, K$. Each neuron in layer 3 performs normalization, and the outputs are called *normalized firing strengths*

$$o_j^{(3)} = \frac{o_j^{(2)}}{\sum_{k=1}^{K} o_k^{(2)}} \tag{35}$$

for $j = 1, \dots, K$. The output of each node in layer 4 is defined by

$$o_j^{(4)} = o_j^{(3)} f_j(\mathbf{x}) \tag{36}$$

for $j = 1, \dots, K$. Parameters in $f_j(\mathbf{x})$ are referred to as consequent parameters. The outputs of layer 4 are summed and the output of the network gives the TSK model (30)

$$o^{(5)} = \sum_{j=1}^{K} o_j^{(4)}. \tag{37}$$

In the ANFIS model, functions used at all the nodes are differentiable, thus the BP algorithm can be used to learn the premise parameters by using a sample set of size N, $\{(\mathbf{x}_t, y_t)\}$. The effectiveness of the model is dependent on the MFs used. The TSK fuzzy rules are employed in the ANFIS model

$$\mathrm{R}_i\text{: IF } \mathbf{x} \text{ is } A_i\text{, THEN } y = f_i(\mathbf{x}) = \sum_{j=1}^{n} a_{i,j}\, x_j + a_{i,0}$$

for $i = 1, \ldots, K$, where $A_i = \{A_i^1, A_i^2, \ldots, A_i^n\}$ are fuzzy sets and $a_{i,j}, j = 0, 1, \ldots, n$, are consequent parameters. The output of the network at time t is thus given by

$$\hat{y}_t = \frac{\sum_{i=1}^{K} \mu_{A_i}(\mathbf{x}_t) f_i(\mathbf{x}_t)}{\sum_{i=1}^{K} \mu_{A_i}(\mathbf{x}_t)}, \tag{38}$$

where $\mu_{A_i}(\mathbf{x}_t) = \prod_{j=1}^{n} \mu_{A_j^i}(x_{t,j})$. Accordingly, the error measure at time t is defined by $E_t = \frac{1}{2}(\hat{y}_t - y_t)^2$.

After the rule base is specified, the ANFIS adjusts only the MFs of the antecedents and the consequent parameters. The BP algorithm can be used to train both the premise and consequent parameters. A more efficient procedure is to learn the premise parameters by the BP, but to learn the linear consequent parameters by the RLS method [48]. The learning rate η can be adaptively adjusted by some heuristics. It is reported in [48] that this hybrid learning method provides better results than the MLP trained by the BP method and the cascade-correlation network [34]. In [49], the Levenberg-Marquardt (LM) method [29] is used for ANFIS training. Compared to the hybrid method, the LM method achieves a better precision, but the interpretability of the final MFs is quite weak. In [18], the RProp [89] and the RLS methods are used to learn the premise parameters and the consequent parameters, respectively. The ANFIS model has been generalized for classification by employing parameterized t-norms [101], where tree partitioning is used for structure identification and the Kalman filtering method for parameter learning.

The ANFIS is attractive for applications in view of its network structure and the standard learning algorithm. Training of the ANFIS follows the spirit of the minimal disturbance principle and is thus more efficient than the MLP [51]. However, the ANFIS is computationally expensive due to the curse-of-dimensionality problem arising from grid partitioning. Tree or scattering partitioning can resolve the curse of dimensionality, but leads to a reduction in the interpretability of the generated rules. Constraints on MFs and initialization using prior knowledge cannot be provided to the ANFIS model due to the learning procedure. The learning results may be difficult to interpret. Thus, the ANFIS model is suitable for applications, where performance is more important than interpretation. In order to preserve the plausibility of the ANFIS, one can add some regularization terms to the cost function so that some constraints on the interpretability are considered [51].

The ANFIS has been extended to the coactive ANFIS [73] and to the generalized ANFIS [5]. The coactive ANFIS [73] is a generalization of the ANFIS by introducing nonlinearity into the TSK rules. The generalized ANFIS [5] is based on a generalization of the TSK model and a generalized Gaussian RBFN. The generalized fuzzy model is trained by using the generalized RBFN model, based on the functional equivalence between the two models. The sigmoid-ANFIS [125] employs only sigmoidal MFs and adopts the interactive-or operator [7] as its fuzzy connectives. The gradient-descent algorithm can also be directly applied to the TSK model without representing it in a network structure [77]. The unfolding-in-time [119] is a method to transform an RNN into an FNN so that the BP algorithm can be used. The ANFIS-unfolded-in-time [99] is designed for prediction of time series data, and achieves much smaller error in the ANFIS-unfolded-in-time compared to that in the ANFIS.

6.2 Generic Fuzzy Perceptron

The generic fuzzy perceptron (GFP) [75] has a structure similar to that of the three-layer MLP. The network inputs and the weights are modeled as fuzzy sets, and t-norm or t-conorm is used as the activation function at each unit. The hidden layer acts as the rule layer. The output units usually use a defuzzufication function. The GFP can interpret its structure in the form of linguistic rules and the structure of the GFP can be treated as a linguistic rule base, where the weights between the input and hidden (rule) layers are called fuzzy antecedent weights and the weights

between the hidden (rule) and output layers fuzzy consequent weights. The GFP model is based on the Mamdani model.

The NEFCON [76], [75], [78], NEFCLASS [75], and NEFPROX [75] models are three neuro-fuzzy models based on the GFP model, which are used for control, classification and approximation, respectively. Due to the use of nondifferentiable t-norm and t-conorm, the gradient-descent method cannot be applied. A set of linguistic rules are used for describing the performance of the models. This knowledge-based fuzzy error is independent of the range of the output value. Learning algorithms for all these models are derived from the fuzzy error using simple heuristics. Initial fuzzy partitions are needed to be specified for each input variable. Some connections that have identical linguistic values are forced to have the same weights so as to keep the interpretability. Prior knowledge can be integrated in the form of fuzzy rules to initialize the neuro-fuzzy systems, and the remaining rules are obtained by learning.

The NEFCON has a single output node, and is used for control. A reinforcement learning algorithm is used for online learning. The NEFCLASS and the NEFPROX can learn rules by using supervised learning instead of reinforcement learning. Compared to neural networks, the NEFCLASS uses a much simple learning strategy, where no clustering is involved in finding the rules. The NEFCLASS does not use MFs in the rules' consequents. The NETPROX is similar to the NEFCON and the NEFCLASS, but is more general. If there is no prior knowledge, a NEFPROX system can be started with no hidden unit and rules can be incrementally learned. If the learning algorithm creates too many rules, only the best rules are kept by evaluating individual rule errors. Each rule represents a number of samples of the unknown function in the form of fuzzy sample. Parameter learning is used to compensate for the error caused by rule removing.

The NETPROX is more important for function approximation. An empirical performance comparison between the ANFIS and the NETPROX has been made in [75]. The NEFPROX is an order-of-magnitude faster than the ANFIS model of [48], but with a higher approximation error. Interpretation of the learning result is difficult for both the ANFIS and the NEFPROX: the ANFIS represents a TSK system, while the NEFPROX represents a Mamdani system with too many rules. To increase the interpretability of the NEFPROX, pruning strategies can be employed to reduce the number of rules.

6.3 Fuzzy Clustering

Fuzzy clustering is one of the most successful applications of neuro-fuzzy synergism, where fuzzy logic is incorporated into competitive learning-based clustering neural networks such as the Kohonen network and the ART models. In clustering analysis, the discreteness of each cluster endows conventional clustering algorithms with analytical and algorithmic intractabilities. Partitioning the dataset in a fuzzy manner helps to circumvent such difficulties. Each cluster is considered as a fuzzy set, and each feature vector may be assigned to multiple clusters with some degree of certainty measured by the membership function taking values in the interval $[0,1]$. Thus, fuzzy clustering helps to find natural vague boundaries in data. The most well-known fuzzy clustering algorithm is the fuzzy C-means algorithm [8]. Other fuzzy clustering algorithms can be based on the Kohonen network and learning vector quantization, on the ART or the ARTMAP models, or on the Hopfield model. A comprehensive survey on various clustering and fuzzy clustering algorithms is given in [29], [30].

6.4 Other Neuro-Fuzzy Models

Neuro-fuzzy systems can employ the topologies of the layered FNN architecture [40], [53], [23], [26], the RBFN model [2], [120], [79], the self-organizing map (SOM) model [111], and the RNN architecture [65], [66]. Neuro-fuzzy models are mainly used for function approximation. They typically have a layered FNN architecture, are based on TSK-type FISs, and are trained by using the gradient-descent method [82], [46], [40], [64], [54], [67], [108]. Gradient descent in this case is

sometimes termed as the fuzzy BP algorithm. Conjugate gradient (CG) algorithms are also used for training neuro-fuzzy systems [67]. Based on the fuzzification of the linear autoassociative neural networks, the fuzzy PCA [26] can extract a number of relevant features from high-dimensional fuzzy data.

Hybrid neural FIS (HyFIS) [54] is a five-layer neuro-fuzzy model based on the Mamdani FIS. Expert knowledge can be used for the initialization of these MFs. The HyFIS first extracts fuzzy rules from data by using the LUT technique [115]. The gradient-descent method is then applied to tune the MFs of input/output linguistic variables and the network weights by minimizing the error function. The HyFIS model is comparable in performance with the ANFIS [48].

Fuzzy min-max neural networks are a class of neuro-fuzzy models using min-max hyperboxes for clustering, classification, and regression [97], [98], [37], [102], [90]. The max-min fuzzy Hopfield network [66] is a fuzzy RNN for fuzzy associative memory (FAM). The manipulations of the hyperboxes involve mainly comparison, addition and subtraction operations, thus learning is extremely efficient.

Many neuro-fuzzy models employ the architecture of the RBFN [116], [60], [71], [22], [19]. These models use are based on the TSK model, and are a universal approximator. The FBFN can readily adopt various learning algorithms already developed for the RBFN.

Adaptive parsimonious neuro-fuzzy systems can be achieved by using constructive approach and a simultaneous adaptation of space partitioning and fuzzy rule parameters [22], [120]. The dynamic fuzzy neural network (DFNN) [120], [33] is an online implementation of the TSK system based on an extended RBFN and its learning algorithm. Similar to the ANFIS architecture, the self-organizing fuzzy neural network (SOFNN) [62] has a five-layer fuzzy neural network architecture. It is an online implementation of a TSK-type model. The SOFNN is based on neurons with an ellipsoidal basis function, and the neurons are added or pruned dynamically in the learning process. Similar MFs can be combined into one new MF. The SOFNN algorithm is superior to the DFNN in time complexity [120].

7. FUZZY NEURAL CIRCUITS

Fuzzy systems can be easily implemented in the digital form, which can be either general-purpose microcontrollers running fuzzy inference and defuzzification programs, or dedicated fuzzy coprocessors, or RISC processors with specialized fuzzy support, or fuzzy ASICs. The pros and cons of various digital fuzzy hardware implementation strategies are reviewed in [25].

A common approach to general-purpose fuzzy hardware is to use a software design tool such as the Motorola-Aptronix fuzzy inference development language and Togai InfraLogic's MicroFPL system to generate the program code for a target microcontroller [44]. This approach leads to rapid design and testing, but has a low performance. On the other hand, dedicated fuzzy processors and ASICs have physical and performance characteristics that are closely matched to an application, and its performance would be optimized to suit a given problem at the price of high design and test costs. Fuzzy coprocessors work in conjunction with a host processor. They are general-purpose hardware, and thus have a lower performance compared to a custom fuzzy hardware. A number of commercially available fuzzy coprocessors are listed in [95]. Some issues arising from the implementation of such coprocessors are discussed in [83]. RISC processors with specialized fuzzy support are also available [25], [95]. A fuzzy-specific extension to the instruction set is defined and implemented using hardware/software codesign techniques. In [44], the tool TROUT was created to automate fuzzy neural ASIC design. The TROUT produces a specification for small, application-specific circuits called smart parts. Each smart part is customized to a single function, and can be packaged in a variety of ways. The model library of the TROUT includes fuzzy or neural models for implementation as circuits. To synthesize a circuit,

the TROUT takes as its input an application data set, optionally augmented with user-supplied hints. It delivers, as output, technology-independent VHDL code for a circuit of the fuzzy or neural model.

There are also many analog [61], [24], [58], and mixed-signal [6], [10] fuzzy circuits. Analog circuits usually operate in the current mode and are fabricated using the CMOS technology, and this leads to the advantages of high speed, small-circuit area, high performance, and low power dissipation. A design methodology for fuzzy ASICs and general-purpose fuzzy processors is given in [58], based on the LR (left-right) fuzzy implication cells and the LR fuzzy arithmetic cells. In [6], [10], the fabrication of mixed-signal CMOS chips for fuzzy controllers is considered; in these circuits, the computing power is provided by the analog part while the digital part is used for programmability.

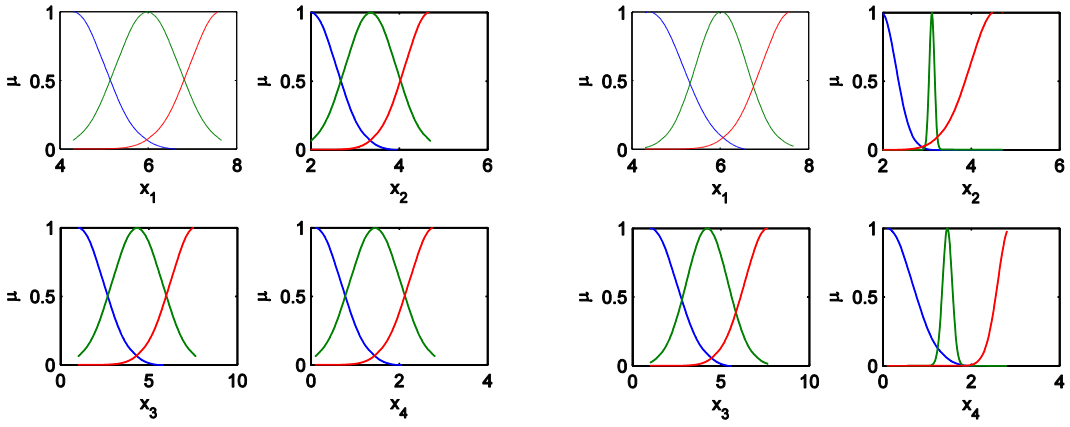
An overview of the existing hardware implementations of neural and fuzzy systems is made in [88], where limitations, advantages, and bottlenecks of analog, digital, pulse stream (spiking), and other techniques are discussed. Hardware/software codesign allows a fast design of complex systems with the highest performance-cost ratio by exploiting the best from both the hardware and software techniques. A survey of digital fuzzy logic controllers is given in [83].

8. COMPUTER SIMULATION: IRIS CLASSIFICATION

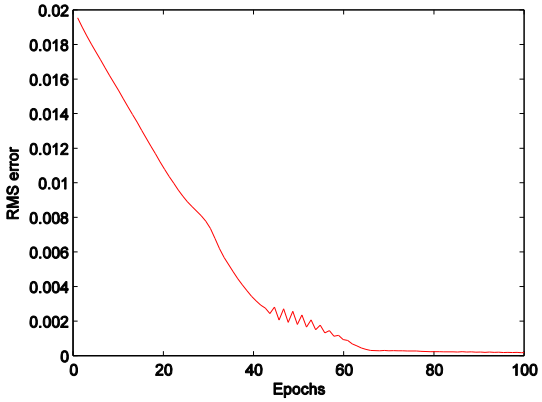
We now use the ANFIS model to solve the Iris classification problem. In the Iris data set, 150 patterns are classified into 3 classes. Each pattern has four numeric properties. The ANFIS model is available in the MATLAB Fuzzy toolbox. An initial TSK FIS is first generated by using grid partitioning. Since the ranges for x_1, x_2, and x_3 are very small, they each are partitioned into 2 subsets. The Gaussian MF is selected.

We use the ANFIS model to solve the IRIS classification problem. For the 120 patterns, the ranges of the input and output variables are $x_1 \in [4.3, 7.9]$, $x_2 \in [2.0, 4.4]$, $x_3 \in [1.0, 6.9]$, $x_4 \in [0.1, 2.5]$, $y \in [1, 3]$.

An initial TSK FIS is first generated by using grid partitioning. The variables each are partitioned into 3 subsets. The Gaussian MF is selected. The maximum number of epochs is 100. The fuzzy partitioning for the input space as well as the training error is illustrated in Fig. 8. The classification error rate is 0. The ANFIS model generates 193 nodes, 405 linear parameters, 24 nonlinear parameters, and 81 fuzzy rules. The training time is 53.70 s.



(a) (b)

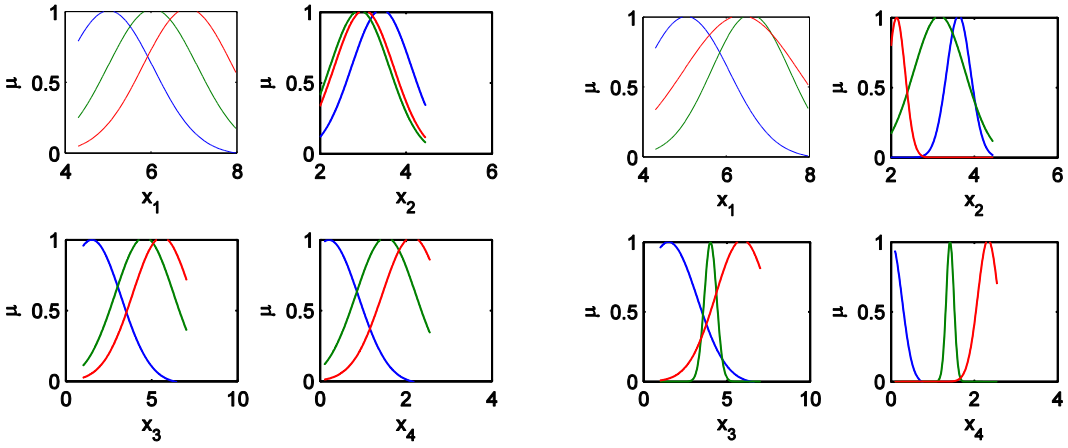


(c)

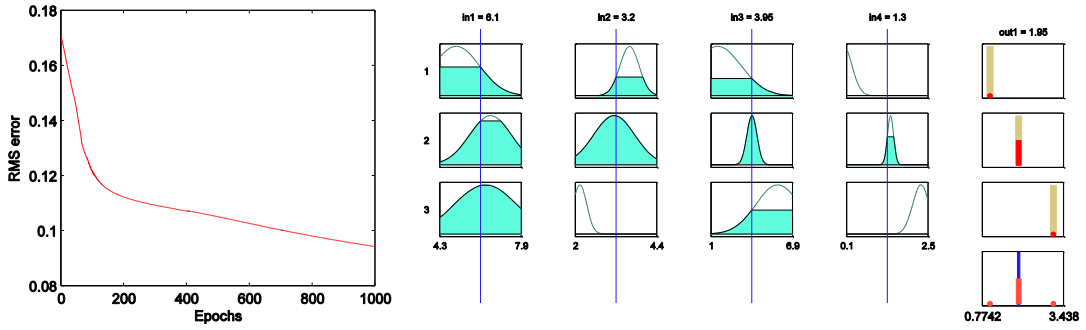
FIGURE 8: IRIS classification: grid partitioning of the input space. (a) The initialized MFs. (b) The learned MFs. (c) The training RMS error.

We further solve the IRIS problem using the ANFIS with scatter partitioning. Clustering the input space is a desired method for generating fuzzy rules. This can significantly reduce the total number of fuzzy rules, hence offer a better generalization capability. Subtractive clustering is used for rule extraction so as to find an initial FIS for ANFIS training. Radius r specifies the range of influence of the cluster center for each input and output dimension. The training error can be controlled by adjusting r, $r \in [0,1]$. Specifying a smaller cluster radius usually yields more, smaller clusters in the data, and hence more rules. The training runs for 200 epochs.

Since the range of the input space is very small when compared with that of the output space, we select $r = 0.8$ for all the input dimensions and the output space. The training time is 2.69 s. After training the RMS error is 0.1123. The ANFIS model has 37 nodes, 15 linear parameters, 24 nonlinear parameters, and 3 fuzzy rules. The classification error is 1.33%. The scatter partitioning is shown in Fig. 9a, b, and the training and testing errors are illustrated in Fig. 9c. The generated fuzzy rules are shown in Fig. 9d.



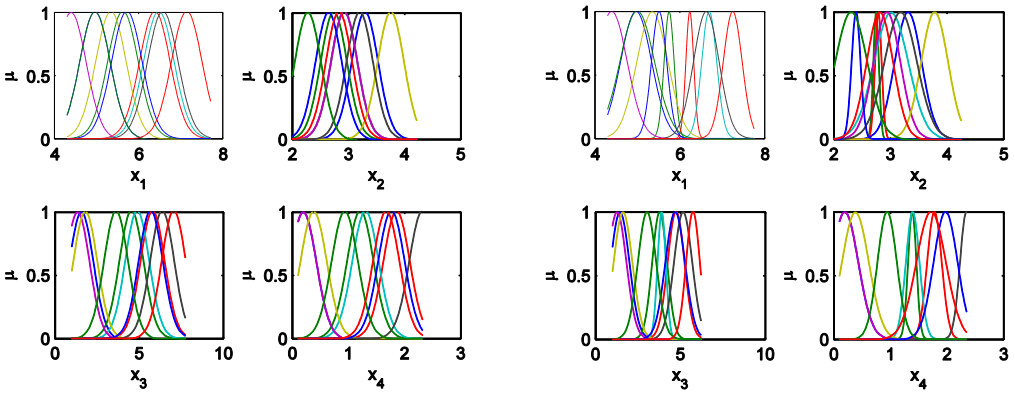
(a) (b)



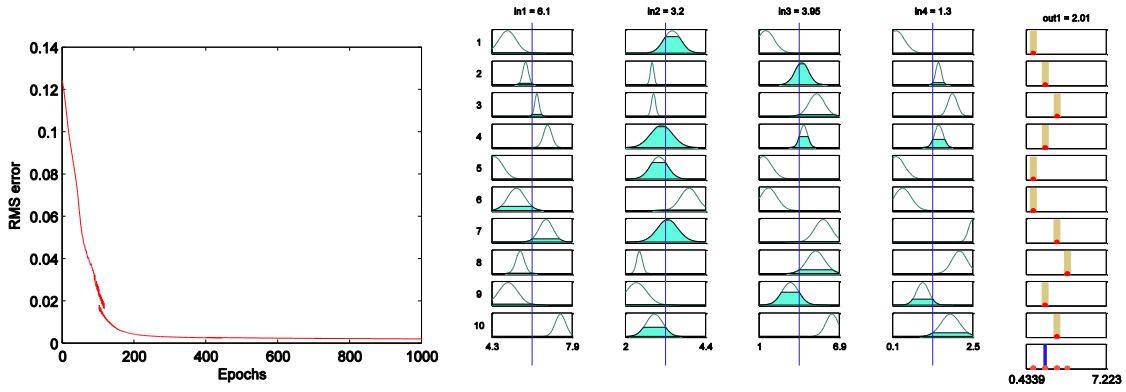
(c) (d)

FIGURE 9: IRIS classification: scatter partitioning of the input space. (a) The initialized MFs. (b) The learned MFs. (c) The training RMS error. (d) the generated fuzzy rules. Note that some MFs coincide in the figure. $\mathbf{r} = [0.8, 0.8, 0.8, 0.8, 0.8]$.

In order to further increase the training accuracy, we can select $r = 0.3$ for all the input dimensions and the output space to get a finer clustering. Then we can get more rules. The ANFIS model has 107 nodes, 50 linear parameters, 80 nonlinear parameters, and 19 fuzzy rules. The training time is 16.2624 s for 1000 epochs. The result is shown in Fig. 10.



(a) (b)



(c) (d)

FIGURE 10: IRIS classification: scatter partitioning of the input space. (a) The initialized MFs. (b) The learned MFs. (c) The training RMS error. (d) the generated fuzzy rules. Note that some MFs coincide in the figure. $\mathbf{r} = [0.9, 0.9, 0.9, 0.9, 0.1]$.

For the 10 rules generated, each rule has its own MF for each input variable. For example, the ith rule is given by

$$\mathrm{R}_i\text{: IF } x_1 \text{ is } \mu_{i,1} \text{ AND } x_2 \text{ is } \mu_{i,2} \text{ AND } x_3 \text{ is } \mu_{i,3} \text{ AND } x_4 \text{ is } \mu_{i,4} \text{ THEN } y \text{ is } \mu_{i,y}$$

where $\mu_{i,k}$, $k = 1, \ldots, 4$, and $\mu_{i,y}$ are MFs. The fuzzy rules for the DoA estimation using the ANFIS with scattering partitioning and the fuzzy-inference process from inputs to outputs. Each row of plots corresponds to one rule, and each column corresponds to either an input variable x_i or the output variable y.

9. SUMMARY

In this paper, we give a systematic introduction to concepts in fuzzy sets and fuzzy logic as well as neuro-fuzzy systems. Fuzzy logic provides an effective tools for modelling uncertainty in human reasoning. A fuzzy inference system represents knowledge in IF-THEN rules, and implement fuzzy reasoning. Like neural network models, some fuzzy inference systems have the universal approximation capability. Fuzzy logic is an alternative to neural networks for the purpose of classification and function approximation and for most applications where neural networks are applicable. Neuro-fuzzy systems combine the advantages of both computational paradigms, and are gaining more popularity.

10. REFERENCES

1. S. Abe, M.S. Lan. “Fuzzy rules extraction directly from numerical data for function approximation”. IEEE Trans. Syst. Man Cybern., 25(1), pp. 119–129, 1995.

2. D.F. Akhmetov, Y. Dote, S.J. Ovaska. “Fuzzy neural network with general parameter adaptation for modeling of nonlinear time-series”. IEEE Trans. Neural Netw., 12(1), pp. 148–152, 2001.

3. J.S. Albus. “A new approach to manipulator control: Cerebellar model articulation control (CMAC)”. Trans. ASME J. Dyna. Syst. Meas. Contr., 97, pp. 220–227, 1975.

4. P.P. Angelov, D.P. Filev. “An approach to online identification of Takagi-Sugeno fuzzy models”. IEEE Trans. Syst. Man Cybern. B, 34(1), 484–498, 2004.

5. M.F. Azeem, M. Hanmandlu, N. Ahmad. “Generalization of adaptive neuro-fuzzy inference systems”. IEEE Trans. Neural Netw., 11(6), pp. 1332–1346, 2000.

6. I. Baturone, S. Sanchez-Solano, A. Barriga, J.L. Huertas. “Implementation of CMOS fuzzy controllers as mixed-signal integrated circuits”. IEEE Trans. Fuzzy Syst., 5(1), pp. 1–19, 1997.

7. J.M. Benitez, J.L. Castro, I. Requena. “Are artificial neural networks black boxes?”. IEEE Trans. Neural Netw., 8(5), pp. 1156–1164, 1997.

8. J. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms, New York: Plenum Press, 1981.

9. A. Blanco, M. Delgado, M.C. Pegalajar. “Extracting rules from a (fuzzy/crisp) recurrent neural network using a self-organizing map”. Int. J. Intell. Syst., 15(7), pp. 595–621, 2000.

10. S. Bouras, M. Kotronakis, K. Suyama, Y. Tsividis. “Mixed analog-digital fuzzy logic controller with continuous-amplitude fuzzy inferences and defuzzification”. IEEE Trans. Fuzzy Syst., 6(2), pp. 205–215, 1998.

11. J.J. Buckley. “Fuzzy complex numbers”. Fuzzy Sets Syst., 33, pp. 333–345, 1989.

12. J.J. Buckley. “Sugeno type controllers are universal controllers”. Fuzzy Sets Syst., 53, pp. 299–304, 1993.

13. J.J. Buckley, Y. Hayashi, E. Czogala. “On the equivalence of neural nets and fuzzy expert systems”. Fuzzy Sets Syst., 53, pp. 129–134, 1993.

14. J.J. Buckley, E. Eslami. An Introduction to Fuzzy Logic and Fuzzy Sets, Heidelberg: Physica-Verlag, 2002.

15. J.L. Castro. “Fuzzy logic controllers are universal approximators”. IEEE Trans Syst Man Cybern., 25(4), pp. 629–635, 1995.

16. J.L. Castro, C.J. Mantas, J. Benitez. “Interpretation of artificial neural networks by means of fuzzy rules”. IEEE Trans. Neural Netw., 13(1), pp. 101–116, 2002.

17. A. Cechin, U. Epperlein, B. Koppenhoefer, W. Rosenstiel. “The extraction of Sugeno fuzzy rules from neural networks”, in Proc. Euro. Symp. Artif. Neural Netw., Bruges, Belgium, 1996, pp. 49–54.

18. M.S. Chen, R.J. Liou. “An efficient learning method of fuzzy inference system”, in Proc. IEEE Int. Fuzzy Syst. Conf., Seoul, Korea, 1999, pp. 634–638.

19. C.B. Cheng, E.S. Lee. “Fuzzy regression with radial basis function network”. Fuzzy Sets Syst., 119, pp. 291–301, 2001.

20. S. Chiu. “Fuzzy model identification based on cluster estimation”. J. Intell. & Fuzzy Syst., 2(3), pp. 267–278, 1994.

21. S.L. Chiu. “A cluster estimation method with extension to fuzzy model identification”, in Proc. IEEE Int. Conf. Fuzzy Syst., Orlando, FL, 2, 1994, pp. 1240–1245.

22. K.B. Cho, B.H. Wang. “Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction”. Fuzzy Sets Syst., 83, pp. 325–339, 1996.

23. M.Y. Chow, S. Altrug, H.J. Trussell. “Heuristic constraints enforcement for training of and knowledge extraction from a fuzzy/neural architecture—Part I: Foundations”. IEEE Trans. Fuzzy Syst., 7(2), pp. 143–150, 1999.

24. U. Cilingiroglu, B. Pamir, Z.S. Gunay, F. Dulger. “Sampled-analog implementation of application-specific fuzzy controllers”. IEEE Trans. Fuzzy Syst., 5(3), pp. 431–442, 1997.

25. A. Costa, A. De Gloria, P. Farabosch, A. Pagni, G. Rizzotto. “Hardware solutions of fuzzy control”. Proc. IEEE, 83(3), pp. 422–434, 1995.

26. T. Denoeux, M.H. Masson. “Principal component analysis of fuzzy data using autoassociative neural networks”. IEEE Trans. Fuzzy Syst., 12(3), pp. 336–349, 2004.

27. S. Dick. “Toward complex fuzzy logic”. IEEE Trans. Fuzzy Syst., 13(3), pp. 405–414, 2005.

28. J.A. Dickerson, B. Kosko. “Fuzzy function learning with covariance ellipsoids”, in Proc. IEEE Int. Conf. Neural Netw., San Francisco, 3, 1993, pp. 1162–1167.

29. K.-L. Du, M.N.S. Swamy. Neural Networks in a Softcomputing Framework, London: Springer, 2006.

30. K.-L. Du. “Clustering: a neural network approach”. Neural Netw., 23(1), pp. 89–107, 2010.

31. C. Dualibe, M. Verleysen, P.G.A. Jespers. Design of Analog Fuzzy Logic Controllers in CMOS Technology, Netherlands: Kluwer, 2003.

32. W. Duch. “Uncertainty of data, fuzzy membership functions, and multilayer perceptrons”. IEEE Trans. Neural Netw., 16(1), pp. 10–23, 2005.

33. M.J. Er, S. Wu. “A fast learning algorithm for parsimonious fuzzy neural systems”. Fuzzy Sets Syst., 126, pp. 337–351, 2002.

34. S.E. Fahlman, C. Lebiere. “The cascade-correlation learning architecture”, in Advances in Neural Information Processing Systems 2, D.S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 524–532.

35. G. Ferrari-Trecate, R. Rovatti. “Fuzzy systems with overlapping Gaussian concepts: Approximation properties in Sobolev norms”. Fuzzy Sets Syst., 130, pp. 137–145, 2002.

36. M. Figueiredo, F. Gomides, A. Rocha, R. Yager. “Comparison of Yager’s level set method for fuzzy logic control with Mamdani and Larsen methods”. IEEE Trans. Fuzzy Syst., 2, pp. 156–159, 1993.

37. B. Gabrays, A. Bargiela. “General fuzzy min-max neural networks for clustering and classification”. IEEE Trans. Neural Netw., 11(3), pp. 769–783, 2000.

38. S.I. Gallant. “Connectionist expert systems”. Commun. of ACM, 31(2), pp. 152–169, 1988.

39. S. Guillaume. “Designing fuzzy inference systems from data: An interpretability-oriented review”. IEEE Trans. Fuzzy Syst., 9(3), pp. 426–443, 2001.

40. Y. Hayashi, J.J. Buckley, E. Czogala. “Fuzzy neural network with fuzzy signals and weights”. Int. J. Intell. Syst., 8(4), pp. 527–537, 1993.

41. P. Holmblad, J. Ostergaard. “Control of a cement kiln by fuzzy logic”, in Fuzzy Information and Decision Processes, M.M. Gupta, E. Sanchez, Eds. Amsterdam: North-Holland, 1982, pp. 389–399.

42. T. Hong, C. Lee. “Induction of fuzzy rules and membership functions from training examples”. Fuzzy Sets Syst., 84, pp. 33–37, 1996.

43. K.J. Hunt, R. Haas, R. Murray-Smith. “Extending the functional equivalence of radial basis function networks and fuzzy inference systems”. IEEE Trans. Neural Netw., 7(3), pp. 776–781, 1996.

44. J.F. Hurdle. “The synthesis of compact fuzzy neural circuits”. IEEE Trans. Fuzzy Syst., 5(1), pp. 44–55, 1997.

45. M. Ishikawa. “Rule extraction by successive regularization”. Neural Netw., 13(10), pp. 1171–1183, 2000.

46. H. Ishibuchi, R. Fujioka, H. Tanaka. “Neural networks that learn from fuzzy IF-THEN rules”. IEEE Trans. Fuzzy Syst., 1, pp. 85–97, 1993.

47. H. Jacobsson. “Rule extraction from recurrent neural networks: A taxonomy and review”. Neural Comput., 17(6), pp. 1223–1263, 2005.

48. J.S.R. Jang. “ANFIS: Adaptive-network-based fuzzy inference systems”. IEEE Trans, Syst, Man Cybern., 23(3), pp. 665–685, 1993.

49. J.S.R. Jang, E. Mizutani. “Levenberg-Marquardt method for ANFIS learning”, in Proc. Biennial Conf. North Amer. Fuzzy Inf. Process. Soc. (NAFIPS), Berkeley, CA, 1996, pp. 87–91.

50. J.S.R. Jang, C.I. Sun. “Functional equivalence between radial basis function Networks and fuzzy inference systems”. IEEE Trans. Neural Netw., 4(1), pp. 156–159, 1993.

51. J.S.R. Jang, C.I. Sun. “Neuro-fuzzy modeling and control”. Proc. IEEE, 83(3), pp. 378–406, 1995.

52. Y. Jin. “Advanced fuzzy systems design and applications”. Heidelberg: Physica-Verlag, 2003.

53. C.F. Juang, C.T. Lin. “An on-line self-constructing neural fuzzy inference network and its application”. IEEE Trans. Fuzzy Syst., 6(1), pp. 12–32, 1998.

54. J. Kim, N. Kasabov. “HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems”. Neural Netw., 12, pp. 1301–1319, 1999.

55. B. Kosko. “Fuzzy system as universal approximators”, in Proc. IEEE Int. Conf. Fuzzy Syst., San Diego, CA, 1992, pp. 1153–1162.

56. B. Kosko. Fuzzy engineering. Prentice Hall, Englewood Cliffs, 1997.

57. V. Kreinovich, H.T. Nguyen, Y. Yam. “Fuzzy systems are universal approximators for a smooth function and its derivatives”. Int. J. Intell. Syst., 15, pp. 565–574, 2000.

58. Y.H. Kuo, C.L. Chen. “Generic LR fuzzy cells for fuzzy hardware synthesis”. IEEE Trans. Fuzzy syst., 6(2), pp. 266–285, 1998.

59. R. Langari, L. Wang, J. Yen. “Radial basis function networks, regression weights, and the expectation-maximization algorithm”. IEEE Trans. Syst. Man Cybern. A, 27(5), 613–623, 1997.

60. C.W. Lee, Y.C. Shin. “Construction of fuzzy systems using least-squares method and genetic algorithm”. Fuzzy Sets Syst., 137, pp. 297–323, 2003.

61. L. Lemaitre, M. Patyra, D. Mlynek. “Analysis and design of CMOS fuzzy logic controller in current mode”. IEEE J. Solid-State Circ., 29(3), pp. 317–322, 1994.

62. G. Leng, G. Prasad, T.M. McGinnity. “An on-line algorithm for creating self-organizing fuzzy neural networks”. Neural Netw., 17, pp. 1477–1493, 2004.

63. H.X. Li, C.L.P. Chen. “The equivalence between fuzzy logic systems and feedforward neural networks”. IEEE Trans. Neural Netw., 11(2), pp. 356–365, 2000.

64. C.T. Lin, Y.C. Lu. “A neural fuzzy system with fuzzy supervised learning”. IEEE Trans. Syst. Man Cybern. B, 26(5), pp. 744–763, 1996.

65. F.J. Lin, R.J. Wai. “Hybrid control using recurrent fuzzy neural network for linear-induction motor servo drive”. IEEE Trans. Fuzzy Syst., 9(1), pp. 102–115, 2001.

66. P. Liu. “Max-min fuzzy Hopfield neural networks and an efficient learning algorithm”. Fuzzy Sets Syst., 112, pp. 41–49, 2000.

67. P. Liu, H. Li. “Efficient learning algorithms for three-layer regular feedforward fuzzy neural networks”. IEEE Trans. Neural Netw., 15(3), 545–558, 2004.

68. P. Liu, H. Li. “Hierarchical TS fuzzy system and its universal approximation”. Inf. Sci., 169, pp. 279–303, 2005.

69. E.H. Mamdani. “Application of fuzzy algorithms for control of a simple dynamic plant”. Proc. IEEE, 12(1), pp. 1585–1588, 1974.

70. K.J. McGarry, J. MacIntyre. “Knowledge extraction and insertion from radial basis function networks”, in IEE Colloq. Applied Stat. Pattern Recogn, Birmingham, UK, 1999, pp. 15/1–15/6

71. S. Mitra, J. Basak. “FRBF: A fuzzy radial basis function network”. Neural Comput. & Appl., 10, pp. 244–252, 2001.

72. S. Mitra, Y. Hayashi. “Neuro-fuzzy rule generation: Survey in soft computing framework”. IEEE Trans. Neural Netw., 11(3), pp. 748–768, 2000.

73. E. Mizutani, J.S. Jang. “Coactive neural fuzzy modeling”, in Proc. IEEE Int. Conf. Neural Netw., Perth, Australia, 1995, vol. 2, pp. 760–765.

74. D. Moses, O. Degani, H.N. Teodorescu, M. Friedman, A. Kandel. “Linguistic coordinate transformations for complex fuzzy sets”, in Proc. IEEE Int. Conf. Fuzzy Syst., Seoul, Korea, 1999, 3, pp. 1340–1345.

75. D. Nauck, F. Klawonn, R. Kruse. Foundations of Neuro-fuzzy Systems. New York: Wiley, 1997.

76. D. Nauck, R. Kruse. “A neural fuzzy controller learning by fuzzy error propagation”, in Proc. Worksh. North Amer. Fuzzy Inf. Process. Soc. (NAFIPS92), Puerto Vallarta, Mexico, 1992, pp. 388–397.

77. H. Nomura, I. Hayashi, N. Wakami. “A learning method of fuzzy inference rules by descent method”, in Proc. IEEE Int. Conf. Fuzzy Syst., San Diego, CA, 1992, pp. 203–210.

78. A. Nurnberger, D. Nauck, R. Kruse. “Neuro-fuzzy control based on the NEFCON-model: Recent developments”. Soft Comput., 2, pp. 168–182, 1999.

79. S.K. Oh, W. Pedrycz, H.S. Park. “Multi-layer hybrid fuzzy polynomial neural networks: A design in the framework of computational intelligence”. Neurocomput., 64, pp. 397–431, 2005.

80. C.W. Omlin, C.L. Giles. “Extraction of rules from discrete-time recurrent neural networks”. Neural Netw., 9, pp. 41–52, 1996.

81. C.W. Omlin, K.K. Thornber, C.L. Giles. “Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks”. IEEE Trans. Fuzzy Syst., 6, pp. 76–89, 1998.

82. S.K. Pal, S. Mitra. “Multilayer perceptron, fuzzy sets, and classification”. IEEE Trans. Neural Netw., 3(5), pp. 683–697, 1992.

83. M.J. Patyra, J.L. Grantner, K. Koster. “Digital fuzzy logic controller: Design and implementation”. IEEE Trans. Fuzzy Syst., 4(4), pp. 439–459, 1996.

84. W. Pedrycz, A.F. Rocha. “Fuzzy-set based models of neurons and knowledge-based networks”. IEEE Trans. Fuzzy Syst., 1(4), pp. 254–266, 1993.

85. G.V.S. Raju, J. Zhou, R.A. Kisner. “Hierarchical fuzzy control”. Int. J. Contr., 54(5), pp. 1201–1216, 1991.

86. D. Ramot, M. Friedman, G. Langholz, A. Kandel. “Complex fuzzy logic”. IEEE Trans. Fuzzy Syst., 11(4), pp. 450–461, 2003.

87. D. Ramot, R. Milo, M. Friedman, A. Kandel. “Complex fuzzy sets”. IEEE Trans. Fuzzy Syst., 10(2), pp. 171–186, 2002.

88. L.M. Reyneri. “Implementation issues of neuro-fuzzy hardware: Going toward HW/SW codesign”. IEEE Trans. Neural Netw., 14(1), pp. 176–194, 2003.

89. M. Riedmiller, H. Braun. “A direct adaptive method for faster backpropagation learning: the RPROP algorithm”, in Proc. IEEE Int. Conf. Neural Netw., San Francisco, CA, 1993, pp. 586–591.

90. A. Rizzi, M. Panella, F.M.F. Mascioli. “Adaptive resolution min-max classifiers”. IEEE Trans. Neural Netw., 13(2), pp. 402–414, 2002.

91. I. Rojas, H. Pomares, J.L. Bernier, J. Ortega et al. “Time series analysis using normalized PG-RBF network with regression weights”. Neurocomput., 42, pp. 267–285, 2002.

92. I. Rojas, H. Pomares, J. Ortega, A. Prieto. “Self-organized fuzzy system generation from training examples”. IEEE Trans. Fuzzy Syst., 8(1), pp. 23–36, 2000.

93. M. Russo. “Fugenesys—A fuzzy genetic neural system for fuzzy modeling”. IEEE Trans. Fuzzy Syst., 6(3), pp. 373–388, 1998.

94. K. Saito, R. Nakano. “Rule extraction from facts and neural networks”, in Proc. Int. Neural Netw. Conf., Paris, France, pp. 379–382. Kluwer, Dordrecht, the Netherland, 1990.

95. V. Salapura. “A fuzzy RISC processor”. IEEE Trans. Fuzzy Syst., 8(6), pp. 781–790, 2000.

96. M. Setnes, R. Babuska, U. Kaymak, H.R. van Nauta Remke. “Similarity measures in fuzzy rule base simplification”. IEEE Trans. Syst. Man Cybern. B, 28(3), pp. 376–386, 1998.

97. P.K. Simpson. “Fuzzy min-max neural networks–Part I. classification”. IEEE Trans. Neural Netw., 3, pp. 776–786, 1992.

98. P.K. Simpson. “Fuzzy min-max neural networks–Part II: clustering”. IEEE Trans. Fuzzy Syst., 1(1), pp. 32–45, 1993.

99. N.A. Sisman-Yilmaz, F.N. Alpaslan, L. Jain. “ANFIS-unfolded-in-time for multivariate time series forecasting”. Neurocomput., 61, pp. 139–168, 2004.

100. E. Soria-Olivas, J.D. Martin-Guerrero, G. Camps-Valls, A.J. Serrano-Lopez, J. Calpe-Maravilla, L. Gomez-Chova. “A low-complexity fuzzy activation function for artificial neural networks”. IEEE Trans. Neural Netw., 14(6), pp. 1576–1579, 2003.

101. C.T. Sun. “Rule-base structure identification in an adaptive-network-based inference system”. IEEE Trans. Fuzzy Syst., 2(1), pp. 64–79, 1994.

102. R. Tagliaferri, A. Eleuteri, M. Meneganti, F. Barone. “Fuzzy min-max neural networks: From classification to regression”. Soft Comput., 5, pp. 69–76, 2001.

103. T. Takagi, M. Sugeno. “Fuzzy identification of systems and its applications to modelling and control”. IEEE Trans. Syst. Man Cybern., 15(1), pp. 116–132, 1985.

104. K. Tanaka, M. Sugeno. “Stability analysis and design of fuzzy control systems”. Fuzzy Sets Syst., 45, pp. 135–150, 1992.

105. R. Thawonmas, S. Abe. “Function approximation based on fuzzy rules extracted from partitioned numerical data”. IEEE Trans. Syst. Man Cybern. B, 29(4), pp. 525–534, 1999.

106. A. Tickle, R. Andrews, M. Golea, J. Diederich. “The truth will come to light: Direction and challenges in extracting the knowledge embedded within trained artificial neural networks”. IEEE Trans. Neural Netw., 9(6), pp. 1057–1068, 1998.

107. V. Tresp, J. Hollatz, S. Ahmad. “Representing probabilistic rules with networks of Gaussian basis functions”. Mach. Learn., 27, pp. 173–200, 1997.

108. G. Tsekouras, H. Sarimveis, E. Kavakli, G. Bafas. “A hierarchical fuzzy-clustering approach to fuzzy modeling”. Fuzzy Sets Syst., 150(2), pp. 245–266, 2004.

109. K. Uehara, M. Fujise. “Fuzzy inference based on families of α-level sets”. IEEE Trans. Fuzzy Syst., 1(2), pp. 111–124, 1993.

110. A. Ultsch, R. Mantyk, G. Halmans. “Connectionist knowledge acquisition tool: CONKAT”, in Artificial Intelligence Frontiers in Statistics: AI and Statistics III, : D.J. Hand, Ed. London: Chapman & Hall, 1993, pp. 256–263.

111. P. Vuorimaa. “Fuzzy self-organizing map”. Fuzzy Sets Syst., 66(2), pp. 223–231, 1994.

112. D. Wang, N.S. Chaudhari. “Binary neural network training algorithms based on linear sequential learning”. Int J. Neural Syst., 13(5), pp. 333–351, 2003.

113. L.X. Wang. “Fuzzy systems are universal approximators”, in Proc. IEEE Int. Conf. Fuzzy Syst., San Diego, CA, 1992, pp. 1163–1170.

114. L.X. Wang. “Analysis and design of hierarchical fuzzy systems”. IEEE Trans. Fuzzy Syst., 7(5), pp. 617–624, 1999.

115. L.X. Wang, J.M. Mendel. “Generating fuzzy rules by learning from examples”. IEEE Trans. Syst. Man Cybern., 22(6), pp. 1414–1427, 1992.

116. L.X. Wang, J.M. Mendel. “Fuzzy basis functions, universal approximation, and orthogonal least-squares learning”. IEEE Trans. Neural Netw., 3(5), pp. 807–814, 1992.

117. L.X. Wang, C. Wei. “Approximation accuracy of some neuro-fuzzy approaches”. IEEE Trans. Fuzzy Syst., 8(4), pp. 470–478, 2000.

118. S.Wang, H. Lu. “Fuzzy system and CMAC network with B-spline membership/basis functions are smooth approximators”. Soft Comput., 7, pp. 566–573, 2003.

119. P.J. Werbos. “Backpropagation through time: what it does and how to do it”. Proc. IEEE, 78(10), pp. 1550–1560, 1990.

120. S. Wu, M.J. Er. “Dynamic fuzzy neural networks—A novel approach to function approximation”. IEEE Trans. Syst. Man Cybern. B, 30(2), pp. 358–364, 2000.

121. R. Yager, D. Filev. “Generation of fuzzy rules by mountain clustering”. J. Intell. Fuzzy Syst., 2(3), pp. 209–219, 1994.

122. J. Yen. “Fuzzy logic—A modern perspective”. IEEE Trans. Knowl. Data Eng., 11(1), pp. 153–165, 1999.

123. L.A. Zadeh. “Fuzzy sets”. Inf. & Contr., 8, pp. 338–353, 1965.

124. L.A. Zadeh. “The concept of a linguistic variable and its application to approximate reasoning–I, II, III”. Inf. Sci., 8, pp. 199–249, pp. 301–357, 1975; 9, pp. 43–80, 1975.

125. D. Zhang, X.L. Bai, K.Y. Cai. “Extended neuro-fuzzy models of multilayer perceptrons”. Fuzzy Sets Syst., 142, pp. 221–242, 2004.

Faster Case Retrieval Using Hash Indexing Technique

Mohamad Farhan Mohamad Mohsin *farhan@uum.edu.my*
College of Arts & Sciences
Universiti Utara Malaysia
Kedah, 06010, Malaysia

Maznie Manaf *maznie@kelantan.uitm.edu.my*
Faculty of Computer Science & Mathematic
Universiti Teknologi Mara (Kelantan)
Kelantan, 18500, Malaysia

Norita Md Norwawi *norita@usim.edu.my*
Faculty of Science & Technology
Universiti Sains Islam Malaysia
71800, Nilai, Negeri Sembilan, Malaysia

Mohd Helmy Abd Wahab *helmy@uthm.edu.my*
Faculty of Electrical and Electronic Engineering
Universiti Tun Hussain Onn
Johor, 86400, Malaysia

Abstract

The main objective of case retrieval is to scan and to map the most similar old cases in case base with a new problem. Beside accurateness, the time taken to retrieve case is also important. With the increasing number of cases in case base, the retrieval task is becoming more challenging where faster retrieval time and good accuracy are the main aim. Traditionally, sequential indexing method has been applied to search for possible cases in case base. This technique worked fast when the number of cases is small but requires more time to retrieve when the number of data in case base grows. As an alternative, this paper presents the integration of hashing indexing technique in case retrieval to mine large cases and speed up the retrieval time. Hashing indexing searches a record by determining the index using only an entry's search key without traversing all records. To test the proposed method, real data namely Timah Tasoh Dam operational dataset, which is temporal in nature that represents the historical hydrological data of daily Timah Tasoh dam operation in Perlis, Malaysia ranging from year 1997-2005, was chosen as experiment. Then, the hashing indexing performance is compared with sequential method in term of retrieval time and accuracy. The finding indicates that hashing indexing is more accurate and faster than sequential approach in retrieving cases. Besides that, the combination of hashing search key x produces better result compared to single search key.

Keywords: Hashing Indexing, Sequential Indexing, Case Retrieval, Case Base Reasoning.

1. INTRODUCTION

Case-based reasoning (CBR) is a model of reasoning that mimics a human deal with unseen problem. It focuses on the human problem solving approach such as how people learn new skill and generates solution about new situations based on their past experience. Similar mechanism to human that intelligently adapts his experience for learning, CBR replicates the processes by considering experiences as set of old cases and problem to be solved as a new case. To derive to a conclusion, it executes four steps that are retrieve the most similar cases, reuse the retrieved cases to solve the problem, revise the reused solution, and finally retain the revised experience in case base for future decision making. Figure 1 illustrates the CBR decision making processes.

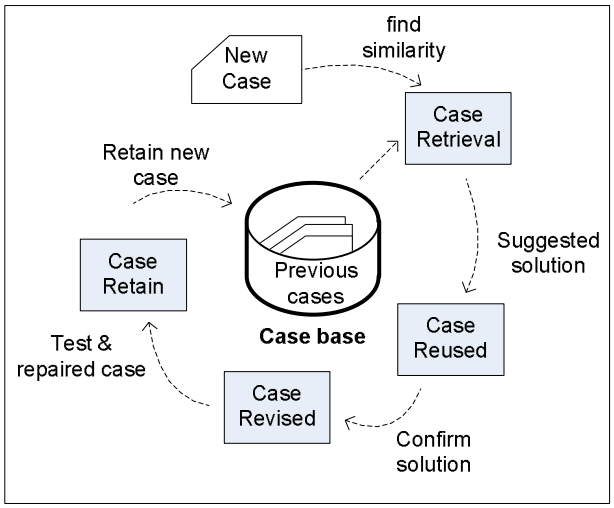


FIGURE 1: The CBR decision making processes [13]

Since it was introduced back in 1970, CBR has had a significant impact to many domains. For example, the technique is widespread across in biology [1], medical for diagnostic and therapeutic task [2], treatment [3], image retrieval [6, 12], project management and planning [7], education and tutoring [8]. The advantages of CBR such as flexibility in knowledge modeling that offers incremental case learning has made possible for CBR to be applied to extremely diverse application domains. Due to the complexity of problem, CBR also has been integrated with soft computing technique such as fuzzy logic [9], neural network [10], and genetic algorithm [11].

Theoretically, CBR maps the similarity between old and new case to derive conclusion. Therefore, the number of old cases is important to lead CBR in producing good decision [3]. It relies heavily on the quality of old cases but practically, to obtain a quality case is difficult to come by [4], [5]. Nowadays, CBR has capability to store million cases in case base due to the advance of data storage technology. With a parallel moving to that scenario, many researchers have undertaken study on case retrieval mainly on the case indexing technique for faster retrieval time. The selection of indexing type is important because it permits the system to match right case at the right time [13].

In general, there are two types of indexing structures which are sequential and non-sequential indexing. Sequential indexing- a conventional technique which has been applied to search for possible cases in case base. Through sequential technique, cases are retrieved case by case following a sequence until the most similar case is matched. It works fast when the number of cases is small but the problem arises when the number of cases contain in case base is huge which consume more time to retrieve.

In this study, a new approach for case indexing in CBR is proposed. This study researches the non-sequential indexing called hashing as an alternative to cater large cases and achieve faster retrieval time in CBR. Hashing indexing searches a record by determining the index using only an entry's search key without traveling to all records [14]. It utilizes small memory, faster retrieval time, and easier to code compared to other indexing technique like data structure [15]. This paper presents the review of the literature of both indexing methods and the integration of hashing indexing in case retrieval with the aim to improve the retrieval performance. To test the proposed method, a real data on Timah Tasoh Dam daily operation was chosen as an experiment. The dataset is a temporal data representing the historical hydrological data of daily Timah Tasoh dam operation in Perlis, Malaysia in the year 1997-2005. Then, the hashing indexing performance is compared with sequential method in term of retrieval time and accuracy.

This paper is organized as follows. Section 2 outlines the literature of case retrieval and hashing indexing. Then, the integration of hashing indexing technique in CBR is discussed in section 3. It will be followed by a discussion on the research design of the study in Section 4. Section 5 describes the experiment data used in this study. In Section 6, the finding and result of the study will be presented and final sections conclude this work.

2. CASE RETRIEVAL AND HASHING INDEXING

Decision making in CBR starts with the case retrieval. It involves the process of finding possible cases in case base that are closest to the new case. For a given new case $\hat{C}=\{x_1, x_2, \ldots, x_n, \theta\}$, where θ is the decision to be determined. The case retrieval is the process of finding old cases C that are close to $\hat{C}$. The mapping of C and $\hat{C}$ is represented as $(\hat{C}, C)$ where cases, $C = \{C_1, C_2, C_3, \ldots, C_n\}$ and $\hat{C}$ is a query case. The similarity between both cases will be determined $(\hat{C} \epsilon C)$ based on the similarity, $Sim = (\hat{C}|C)$.

Two important criterion need to be determined for a quality case retrieval. Firstly, the mechanism to control how the case base is searched and secondly, the suitable search key x to guide searching [13]. In reality, the case retrieval process is highly exploited computer memory and time consuming due to the searching process in huge case base. Therefore, the case indexing technique plays a very important role to determine the searching process either search for an entire case or portion of it. According to [16], indexing and database representation is a fundamental problem for efficient clustering, classification, and data retrieval. The main concern in case retrieval in CBR is how to assess the similarity between cases in order to retrieve appropriate precedents and how to adapt old solution to the case [17]. Beside the most similar, the minimum time consume during the process is also important.

One of the indexing technique uses in case retrieval is sequential indexing. It is a conventional approach applied in the early database technology which cases are retrieved case by case following a sequence until the most similar case is matched. Since it scans the case base following a sequence, this method is not efficient when the number of cases in case base is huge which consume more time to retrieve. As a solution, hashing indexing method with search key x is proposed in database technology.

2.1 **Hashing Indexing**

Hashing indexing is commonly used in database application during data retrieval. This technique has been developed to access large files residing on external storage, not just for accessing fixed size files but flies that grow over their time. The idea of hashing is to map each records to a hash key x using hash function; $H(x)$. $H(x) = i$ whereby i is an index to the hash table, HT and $HT = [1, \ldots, N]$. HT represents an array of size n. The $H(x)$ will take a search key and produces an integer type index representing each case in HT. After that, the case can be directly retrieved at respective address from the HT. The address or search key, x is generated from the function $H(x) = x \ mod \ n$ whereby x is the search key, n is the table size and mod is the modulo operator.

The efficiency of $H(x)$ can be seen in memory management. The approach requires substantial less amount of memory as well as easier to code compared to tree data structure in traditional indexing approach. It also works without requiring a complete file organization and rehashing [15].

In practice, $H(x)$ can map two or more addresses into HT. The $H(x)$ is capable to store more than two items in the same array location in HT or tends to open other address. This occurrence is called collision and the item involved are often called as synonyms [18]. An example of hashing indexing technique which is adopted from [19] is shown in Figure 2.

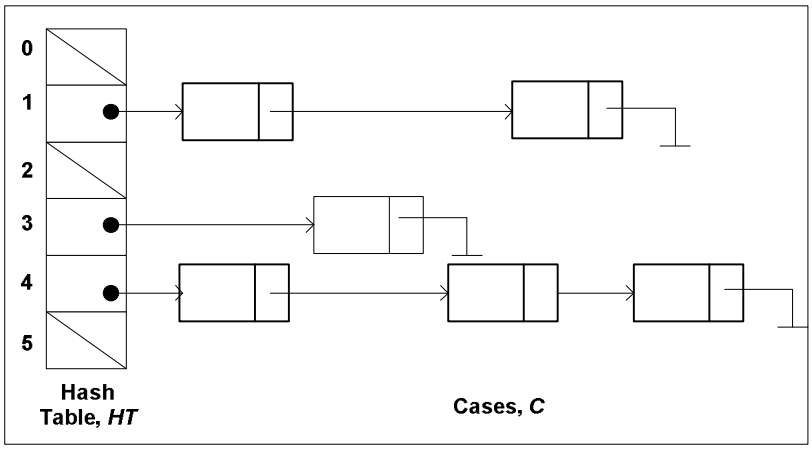


FIGURE 2: Hashing Indexing Technique [19]

One of the HT limitations is when the records become full. It will start working very badly unless separate chaining which is capable to handle collision is used. This is the reason why [18] suggested that HT should never be allowed to get full. To determine either HT is full, the ratio of the number entry located in HT need to be calculated. The ratio is known as load factor. Generally, HT size should be automatically increased and the records in the table should be rehashed when the ratio of table is reached 0.7 (70% full) or 0.8 (80% full) for open addressing [14, 18].

Recently, many applications utilized hashing mechanism to solve specific problem such as in programming that uses HT to keep track of declared variables in source code [14, 18, 19]. HT is an ideal application for this kind of problem because only two operations are performed: insert and find; identifiers are typically short, so the $H(x)$ can be computed quickly. In this application, most searches are successful.

Another common use of HT is in game programs. As the program search through different lines of play, it keeps track of positions that it has encountered by computing $H(x)$ based on the position (and storing its move for the position). If the same position recurs, usually by a simple transposition of moves, the program can avoid expensive recalculation.

3. THE MERGING OF HASHING INDEXING IN CASE RETRIVEVAL

The advantages of hashing indexing in data retrieval are faster retrieval time and minimize the usage of computer resources. This motivation has lead to the merging of hashing indexing in CBR since case retrieval requires a fast solution to retrieve case from case base. Figure 3(a) depicts the concept of this technique and Figure 3(b) is a sequential indexing method. Sequential indexing is a conventional technique practiced in CBR's case retrieval.

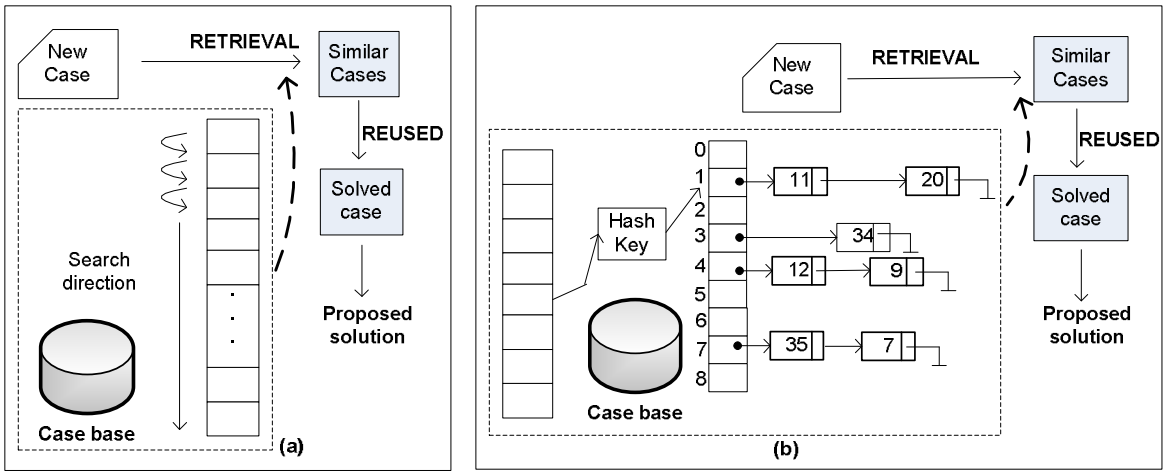


FIGURE 3 (a): The Sequential Indexing Method and **FIGURE 3 (b):** The Hashing Indexing Method

A good $H(x)$ is should be fast to compute, minimize collisions, and distribute entries uniformly through the HT. In the proposed hash model, the separate chaining or close addressing is chosen to resolve collisions. Through this method, specific location is allowed to store more than one value called bucket, b. A new x map or address can be simply placed into a particular location and associated value placed all the cases which have the related attribute in b. The Figure 4 summarizes the separated cases location in hash table, HT illustratively.

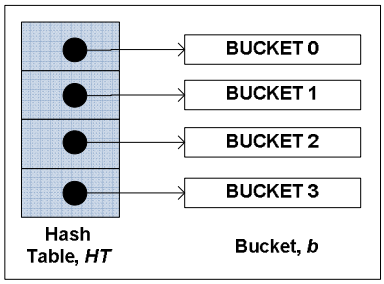


FIGURE 4: The Separated Cases Location in *HT*

The modified hashing indexing algorithm in CBR involves two main tasks that are storing new cases and retrieving a case. Process flow in Figure 5 represents the process of storing a new case in case base. It starts with calculation of $H(x)$ to determine b location at hash table and then store the current cases in b. The algorithm of storing a case into hash table is shown in Algorithm 1

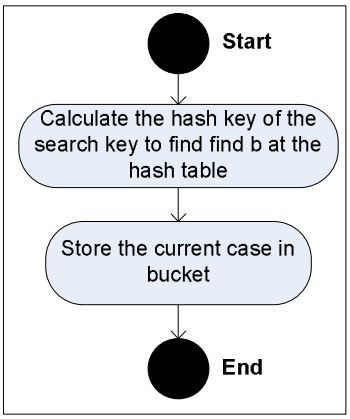


FIGURE 5: Storing a Case into Case Base

Algorithm 1: **Storing a case into hash table**

Input: Timah Tasoh Dam Dataset; search key x; size of data n; size of attribute m; the selection attribute to calculate $selectionColumn$; bucket quantity b, range of search key r

```
1 START
2    get data from dataset
3    store in array oldCases [n][m]
4    set int i = 0, j = 0, counter = 0;
5    WHILE (i = 0)
6        read oldCases[i][selectionColumn]
7        calculate x
8        WHILE (counter << b)
9           determine the range of r
11          store oldCases[i][j] in array bucket_counter [n][m];
12          counter ++
13       ENDWHILE
14      i ++
15   ENDWHILE
16 END
```

The case retrieving process based on CBR's hashing indexing is shown in Figure 6. It starts with calculating the hash key and map to the *HT*. The result is retrieved by finding the search key x after entering a new case. The $H(x)$ formula is used to find the address in HT. Finally, the similarity of cases in similar bucket is calculated to get the predicted result. The similarity of the cases is calculated based on the local similarity and global similarity. Equation 1 and 2 displays the calculation of both similarities.

$$sim(\hat{c}|c) = \frac{\hat{c}-c}{range} \quad (1)$$

where $sim(\hat{c}|c)$ is local similarity, $\hat{c}$is a new case and c is an old case

$$simG(\hat{C}|C) = \sum_{i=1}^{p} w_i sim_i(\hat{c_i}, c_i) \quad (2)$$

where $sim(\hat{C}|C)$ is global similarity, $\hat{C}$is a new case and C is an old case, p is total cases in case base, $sim_i(\hat{c_i}, c_i)$ the local similarity calculate the attribute i, and w_i is the weight of the attribute i

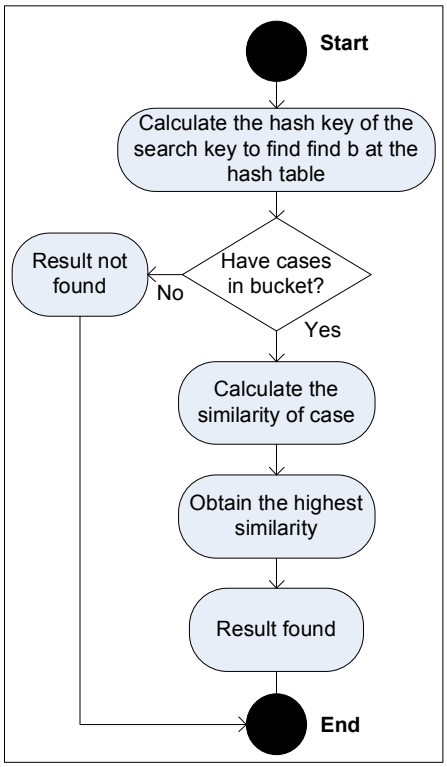


FIGURE 6: Retrieving a Case from *HT*

In this study, three search keys, x are defined. The x are mean of average rainfalls $(\overline{R_m})$, change water level (ΔWL), and combining mean average rainfall and change water level $(\overline{R_m} \wedge \Delta WL)$ which are considered as $H(x)$ as written in (2). Different x are used to determine which x will produces better result mainly in high accuracy and low time retrieval. The x represents the historical hydrological data of daily Timah Tasoh dam operation in Perlis, Malaysia in the year 1997-2005. Next section will describes this data set in detail.

$$H(x) = \begin{cases} \overline{R_m} \ mod \ n \\ \Delta WL \\ \overline{R_m} \wedge \Delta WL \end{cases} \quad (3)$$

Where n is the table size, Mod is the modulor operator, $\overline{R_m}$ refer to Equation 3, ΔWLrefer to Equation 4.

To calculate search key,

$$\overline{R_m} = \frac{\overline{R_{t-1}} + \overline{R}}{2} \quad (4)$$

Where $\overline{R}$is the average rainfall at time t , $\overline{R}$is the average rainfall at time *t -2*, *t* is the time index

and to calculate Δ search key

$$\Delta WL_t = \frac{WL_{t-1} + W.}{WL_{t-2}} \quad (5)$$

Where $WL$is the average rainfall at time $t-1, Wl$ is the average rainfall at time t , *t* is the time index

Every types of x will have different size of hash table or called bucket, b. The number of b will depends on the type of its x. For example, the change of water level (ΔWL) has three types of water level, which are Alert, Warning and Danger [15]. Therefore, ΔWL has three buckets. Table 1 shows the ΔWL key, the number of bucket, and the range of case ΔWL. From Table I, Figure 7 represents the bucket arrangement of ΔWL.

TABLE 1: *Type of ΔWL and The Number of b*

Search key: ΔWL		
b	**Type of water level**	**Range of / m**
0	Alert	$x \leq 0.0034$
1	Warning	$0.0034 < x < 0.0061$
2	Danger	$x \geq 0.0061$

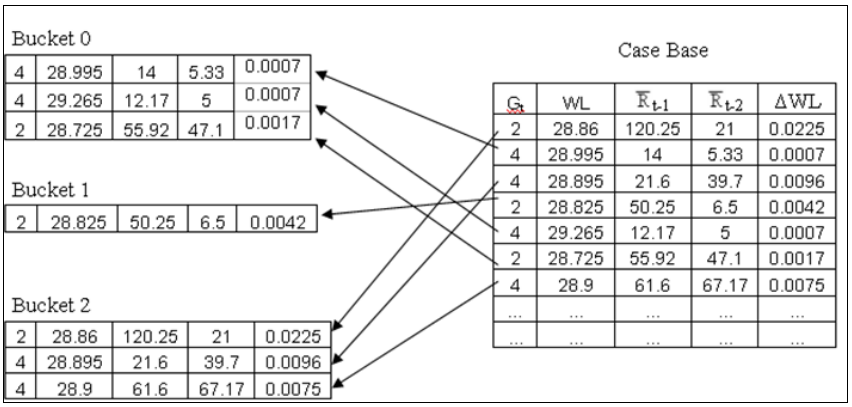


FIGURE 7: The b Arrangement Using ΔWL Key

For mean of average rainfall m key, it has four buckets which represent type of rainfall that are Light, Moderate, Heavy and Very Heavy. Table 2 elaborates the type of rainfall while Figure 8 illustratively represents the bucket arrangement of m key. The Figure 9 portrays the total number of b for the combination of m and ΔWL as thirds search key

TABLE 2: Type of Rainfall and The Number of b

Search key:		
b	**Type of Rainfall**	**Range of Rainfall / mm**
0	Light	$x \leq 11$
1	Moderate	$11 < x < 32$
2	Heavy	$32 < x < 62$
3	Very Heavy	$x \geq 62$

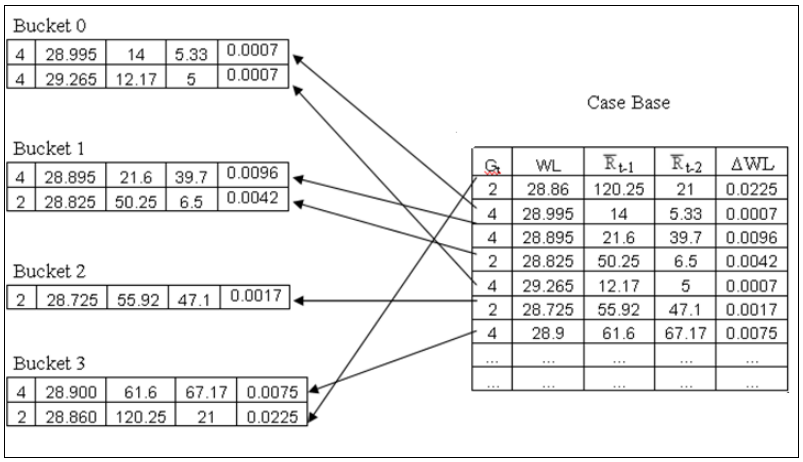


FIGURE 8: The b arrangement using $\overline{R_m}$ key

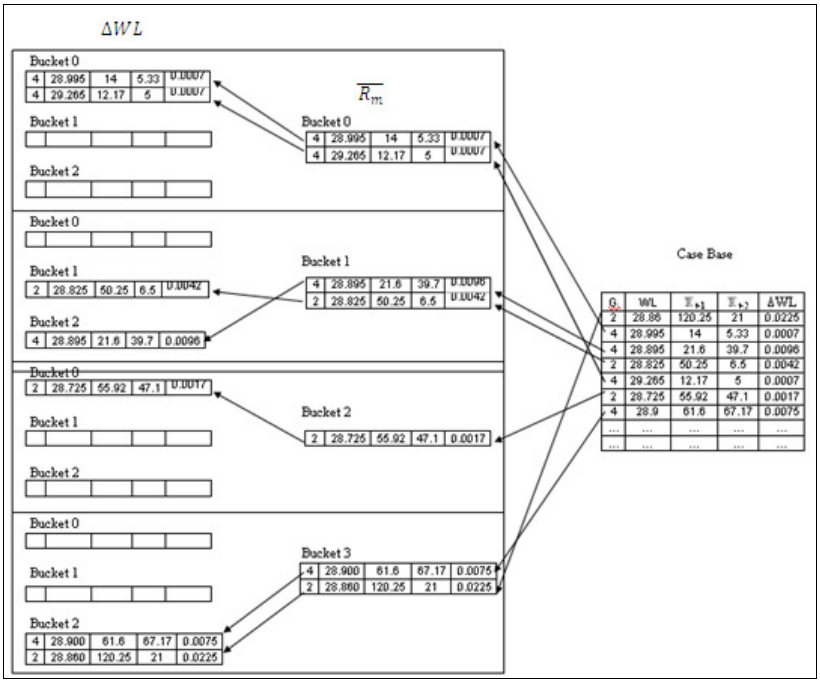


FIGURE 9: The b arrangement using $\overline{R_m}$^ ΔWL key

4. RESEARCH DESIGN

This section describes the research design used in this study which is illustrated in Figure 10. There are three phases which start with development, then preparing data for mining, and lastly is Case Mining.

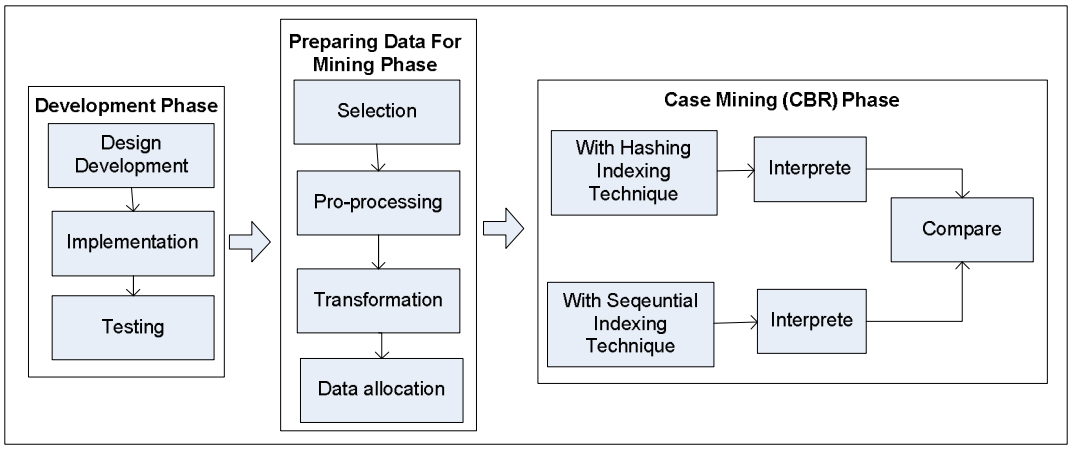


FIGURE 10: The Research Design

The development phase focuses on the algorithm modification. This phase covers three steps which are design development, implementation and testing. In the design development, two approaches: sequential indexing and hashing indexing technique are designed and integrated into CBR using Microsoft Visual C++. After that, the model will be tested. The aim of the testing is to check the accurateness of the hash table and the similarity calculation during mining.

The second phase is preparing data for mining which includes four activities – selection, pre-processing, transformation, and data partition. The aim of this process is to clean and prepare the Timah Tasoh Dam dataset before presenting into the CBR mining system. The selection, pre-processing, and data transformation process are explained in section 5. In data allocation, the experiment data is divided into five folds with different set of training and testing data allocation. The multiple folds are used for a variation set of result. The folds (training: testing) are 90:10, 80:20, 70:30, 60:40 and 50:50.

The last phase is case mining. It involves the mining of TImah Tasoh Dam data set with both indexing methods. During experiment, two measurement metrics are recorded that are accuracy and retrieval time. Then their results are compared. In order to measure the accuracy, the algorithm is tested using various data partition by taking cases in case based as a test set. The measurements are adopted from [15]. This is due to the fact that the real datasets consists of unbalanced data where the number of occurrences of event is lower as compared to non-event occurrence. The accuracy of the model is evaluated base on Equation 6.

$$acc\ (\%) = \frac{t_p + t_n}{t_n + f_n + t_p + f_p} \quad (6)$$

Where t_p is the number of event correctly predicted, f_p is the number of predicted event but in actual non-even, t_n is the number of non-event correctly predicted, and f_n the number of predicted non-even but in actual even

Second measurement is retrieval time which refers to time taken to search for the similarity case from case base. The time is tested by selecting one case from case base and the selected case will be measured for both hashing and sequential technique. The retrieval time will be recorded five times before calculate the average. A special loop is used to perform the task as shown in coding in Figure 11.

```
Double start = clock();

//dummy looping
for (i=0; i<900000; i++){
      for(i=0; i<column; i++){
            for (j=0; j<row; j++){
                  oldCases[i][j];
}}}

/****************************************
    CBR ENGINE (get the global similarity)
*/

Double end = clock();

//calculate retrieval time
cout << Time Taken :: " << (end-start)/60 << millisecond
```

FIGURE 11: Algorithm to Calculate Retrieval Time

5. TIMAH TASOH DAM DATASET

The experiment dataset set used in this study has 15 attributes of temporal data called Timah Tasoh Dam dataset. It comprises the historical hydrological data of daily Timah Tasoh dam operation in Perlis, Malaysia in the year 1997-2005. The preliminary observation on the raw dataset, found out that some attributes are not related to study and certain values were missing. Therefore, the dataset are pre-processed using temporal data series approach which was adopted from [14,15].

During data preprocessing, only relevant attributes were selected. Out of 15 attributes, 4 attributes were chosen that are current water level, average rainfall, current change water level, and current gate. Those attributes represents reservoir water level, rainfall measurement from 6 telemetry stations (Padang Besar, Tasoh, Lubuk Sireh, Kaki Bukit, Wang Kelian, and Guar Jentik) and the number of spillway gates. Spillway gate refers to a structure that makes it possible for the excess water to be released from the dam. Timah Tasoh has six gates and normally the water will be released using Gate 2, Gate 4, and Gate 6 depending on the situation. The selection is made using sliding window technique which is adopted from [14, 15]. After that, the data are re-scaled into a suitable representation to increase mining speed and minimize memory allocation.

Table 3 is a sample of clean data which is ready for mining using CBR's hashing indexing and CBR's sequential indexing model. Based on the table, the current water level (*WL*), average rainfall at $t-1$and $t-2$, current change water level (ΔWL) and current gate (*GT*) are the final input to be mined using CBR.

TABLE3: A Sample of Timah Tasoh Dam Dataset After Pre-process

GT	*WL*	*Average Rainfall* $t-1$	*Average Rainfall* $t-2$	ΔWL
2	29.275	7.33	5.375	0.0007
2	29.025	22.75	11	0.001
4	28.9	61.6	67.17	0.0075
4	28.895	21.6	39.7	0.0096
4	28.995	14	5.33	0.0007
2	29.32	17.5	32	0.0057

6. RESULT & FINDING

This section reports the finding of the integration of hashing indexing technique in case retrieval. The tested model was compared with case retrieval function embedded with sequential indexing technique. As elaborates in 4, the evaluation is conducted using two criteria that are accurateness of the model to obtain similar cases and how fast it takes to retrieve cases. The notation of the experiment is given as follows: The accuracy of the mining as **%**, and retrieval time in millisecond is ***Ms***, The result of the experiment is visually represented in Table 3.

TABLE 3: The Mining Result of Hashing and Sequential Indexing Technique in Ms and %

Data Partition	Sequential Indexing Technique		Hashing Indexing Technique (Search Key *x*)					
			$\overline{R}_m$		ΔWL		$\overline{R}_m \wedge \Delta WL$	
	Ms	%	Ms	%	Ms	%	Ms	%
90 : 10	15.27	50	15.09	75	14.36	50	13.96	75
80 : 20	12.03	38	11.68	38	11.09	50	10.41	57
70 : 30	10.31	46	10.26	38	10.02	46	9.95	42
60 : 40	9.85	47	9.74	35	9.02	41	8.96	50
50 : 50	8.69	38	8.64	38	8.49	52	8.02	61

The analysis starts with the retrieval time of both methods. The result indicates that hashing indexing method required less time for case retrieval in all experiments. For example, in the fold 60:40, sequential technique needs 9.85 *ms* to map all cases however the time taken are lesser in hashing indexing technique with different search key ($\overline{R_m}$= 9.74 *ms*, ΔWL= 9.02 *ms*, $\overline{R_m} \wedge \Delta WL$= 8.96 *ms*). Moreover, the finding also reveals the combination of hashing search key $\overline{R_m} \wedge \Delta WL$ is looked as the most efficient key to mine cases faster compared to single search key. The graph in figure 12 summarizes illustratively the retrieval time taken of both methods and figure 13 shows the retrieval time taken in 60:40 fold as discussed in this paragraph.

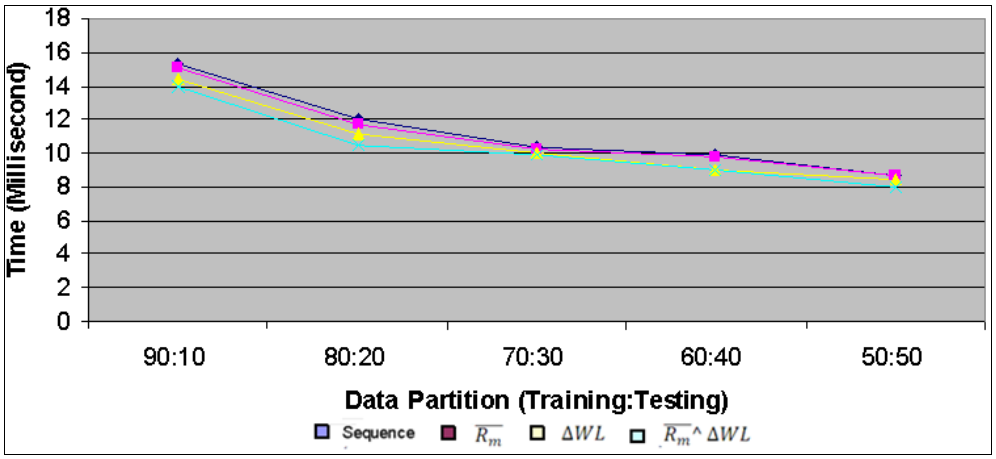


FIGURE 12: The Retrieval Time Taken Hashing and Sequential Indexing

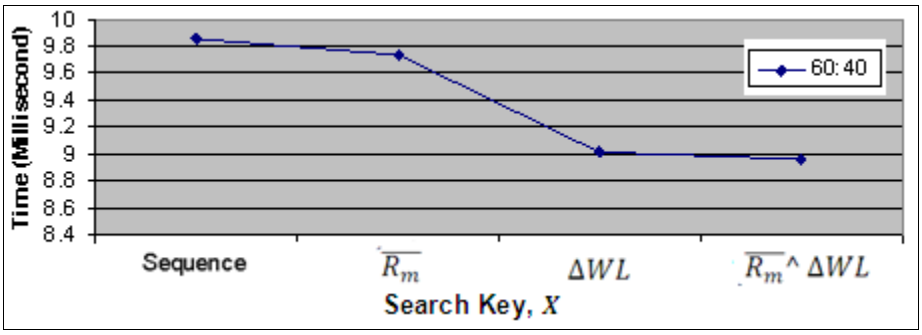


FIGURE 13: Retrieval Time Taken in 60:40 fold

Then, the accurateness of CBR to predict new case is evaluated. In this analysis, the CBR modeling with hashing indexing technique leads the high accuracy. The graph in figure 14 summarizes the accuracy of both methods. Similar in time retrieval evaluation, the $\overline{R_m}$^ ΔWL search key is out performed the single search key $\overline{R_m}$ and ΔWL. It consistently obtains high accuracy in all folds except in 70:30. Interestingly, the result also indicates that the sequential indexing technique also capable to obtain good accuracy when overcome the hash indexing in 70:30 with 46% accurate and left behind the $\overline{R_m}$ (38%) and $\overline{R_m}$^ ΔWL (42%).

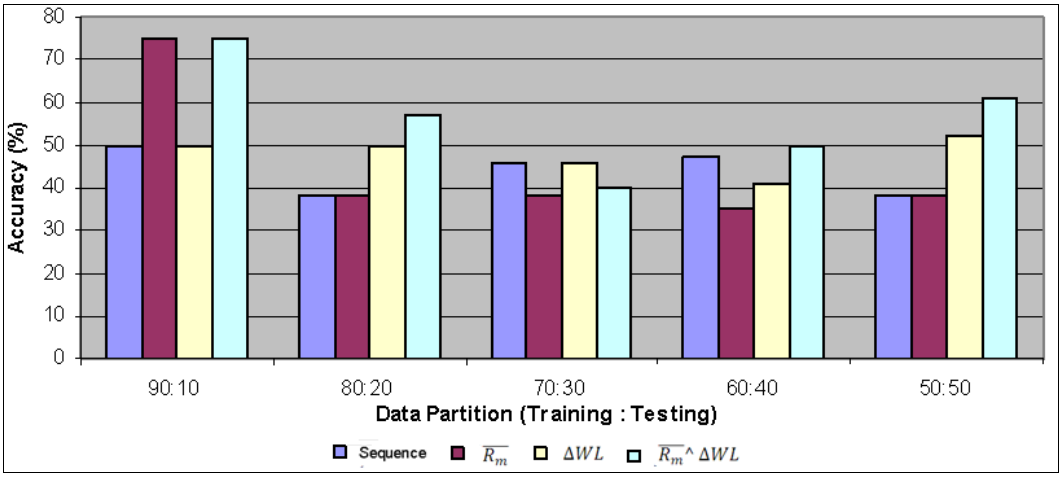


FIGURE 14: The Accuracy of Hashing and Sequential Indexing

Table 4 below summarizes the best technique of the whole experiments. The best technique is selected based on the highest accuracy and shortest time taken to mine Timah Tasoh Dam Dataset. From the table, it is clearly indicates that hashing indexing method has retrieved cases faster that sequential with the combination search key $\overline{R_m}$^ ΔWL as the best search key. In term of accuracy, hashing indexing has scored higher then sequential technique. Out of 5 folds, hashing indexing obtain better accuracies in 4 folds except in fold 70:30, the sequential indexing generates similar accuracy with ΔWL. Lastly, the combination search key $\overline{R_m}$^ ΔWL is chosen as the best search key due to is capability to generate high accuracy and retrieve case faster for Timah Tasoh Dam Dataset.

TABLE 4: The Summarization of the Best Technique based on Accuracy and Retrieval Time

Data Partition	Performance measurement metrics	
Setting	Accuracy	Case Retrieval Time
90 : 10	$\overline{R_m}$ and $\overline{R_m}$ ^ ΔWL	$\overline{R_m}$ ^ ΔWL
80 : 20	$\overline{R_m}$ ^ ΔWL	$\overline{R_m}$ ^ ΔWL
70 : 30	Sequence **and** ΔWL.	$\overline{R_m}$ ^ ΔWL
60 : 40	$\overline{R_m}$ ^ ΔWL	$\overline{R_m}$ ^ ΔWL
50 : 50	$\overline{R_m}$ ^ ΔWL	$\overline{R_m}$ ^ ΔWL

7. CONCLUSION

This research integrates the hashing indexing technique in case retrieval with the aim to cater large cases stored in case base and faster retrieval time. Its performance is compared with the sequential indexing technique using two criteria that are accuracy and retrieval time. From the experiment towards temporal dataset called Timah Tasoh Dam, the hashing indexing is more accurate and faster than sequential in retrieving cases. The finding of this study offers an alternative technique for case base representation and case retrieval. The finding also can assist future miner to mine cases faster, obtain better accuracy and minimize the computer resources usage. For future study, the case retrieval with hashing indexing approach will be tested with other type of data from various domains.

8. REFERENCES

[1] I. Jurisica, and J.I. Glasgow. “Applications of case-based reasoning in molecular biology”. AI Magazine, American Association for Artificial Intelligence, vol. 25(1), pp. 85-95, 2004.

[2] R. Schmid, and L. Gleri. “Case-based Reasoning for Medical Knowledge-based Systems”. International Journal of Medical Informatics, vol. 64, pp. 355, 2000.

[3] Yang, Z., Matsumura, Y., Kuwata, S., Kusuoka, H., and Takeda, H. “Similar Cases Retrieval From the Database of Laboratory Test Results”. Journal of medical systems (J. med. syst.), vol 27, pp. 271-282, 2003.

[4] E. Armengol, S. Ontanon, and E. Plaza. “Explaining Similarity in CBR”. Artificial Intelligence Review. Vol. 24, 2002

[5] P. Rong, Q.Yang, and J.P. Sinno**.** “Mining Competent Case Bases for Case-Based Reasoning”. Journal Artificial Intelligence, vol. 171, 2007.

[6] D.O. Sullivan, E. McLoughlin, B. Michela, and D.C. Wilson. *“*Capturing and reusing case-based context for image retrieval,*”* In Proc. of the 19th International Joint Conference on Artificial Intelligence, 2005.

[7] M.Emilia, N. Mosley, and C. Steve. “The Application of Case-Based Reasoning to Early Web Project Cost Estimation,” In Proc. of the 26 the Annual International Computer Software and Applications Conference (COMPSAC’02), 2002.

[8] K.S. Leen, and B. Todd. “Integrating Case-Based Reasoning and Meta-Learning for a Self-Improving Intelligent Tutoring System”. International Journal of Artificial Intelligence in Education table of contents archive, vol. 18(1):27-58, 2008.

[9] C.K.P. Wong. “Web access path prediction using fuzzy-cased based reasoning, Phd Thesis, Hong Kong Polytechnic University, Hong Kong, 2003.

[10] J.M. Corchodo and B. Lees. "Adaption of cases for case-base forcasting with neural network support," in Soft computing in case based reasoning, 1st ed., vol.1. S.K.Pal, S.D.Tharam, and D.S. Yeung, ed. London: Springer-Verlag, 2001, 293-320.

[11] K.S. Shin and I.Han. "Case-based reasoning supported by genetic algorithm for corporate bond rating". Expert system with application, vol. 1266, pg.1-12. 1997.

[12] H. Hamza, Y. Belaid, and A. Belaid. "A case-based reasoning approach for unknown class Invoice Processing," in Proc. of the IEEE International Conference on Image Processing, (ICIP), 2007, pp. 353-356.

[13] K.P. Sankar and K.S. Simon. Foundation of Soft Case-Based Reasoning, John Willey & Sons Inc, 2004, pp. 1-32.

[14] F. M. Carrano, and W. Savitch. Data Structures and Abstractions with Java. USA: Pearson Education, 2003.

[15] M. Griebel and G. Zumbusch. "Hash-Storage Techniques for Adaptive Multilevel Solvers and Their Domain Decomposition Parallelization". In Proc. of Domain Decomposition Methods 10 (DD10), 1998.

[16] X. He, D. Cai, H. Liu, and W. Ma. "Locality Preserving Indexing for Document Representation," in Proc. of the 27th conference on research and development in information retrieval, 2004.

[17] E. Armengol, S. Ontanon, and E. Plaza. "Explaining Similarity in CBR". Artificial Intelligence Review. vol. 24(2), 2004.

[18] W. D. Maurer, and T.G. Lewis. "Hash Table Methods". ACM Computing Surveys (CSUR), vol 1, pp. 5-19, 1975.

[19] N.M. Darus, Y. Yusof, H. Mohd, and F. Baharom. "Struktur data dan algoritma menggunakan java". Selangor, Malaysia: Pearson Prentice Hall, vol. 1, 2003.

INSTRUCTIONS TO CONTRIBUTORS

The main aim of International Journal of Artificial Intelligence and Expert Systems (IJAE) is to provide a platform to AI & Expert Systems (ES) scientists and professionals to share their research and report new advances in the field of AI and ES. IJAE is a refereed journal producing well-written original research articles and studies, high quality papers as well as state-of-the-art surveys related to AI and ES. By establishing an effective channel of communication between theoretical researchers and practitioners, IJAE provides necessary support to practitioners in the design and development of intelligent and expert systems, and the difficulties faced by the practitioners in using the theoretical results provide feedback to the theoreticians to revalidate their models. IJAE thus meets the demand of both theoretical and applied researchers in artificial intelligence, soft computing and expert systems.

IJAE is a broad journal covering all branches of Artificial Intelligence and Expert Systems and its application in the topics including but not limited to technology & computing, fuzzy logic, expert systems, neural networks, reasoning and evolution, automatic control, mechatronics, robotics, web intelligence applications, heuristic and AI planning strategies and tools, computational theories of learning, intelligent system architectures.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJAE.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 2, 2011, IJAE appears in more focused issues. Besides normal publications, IJAE intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

LIST OF TOPICS

The realm of International Journal of Artificial Intelligence and Expert Systems(IJAE) extends, but not limited, to the following:

- AI for Web Intelligence Applications
- AI Parallel Processing Tools
- AI Tools for Computer Vision and Speech Understand
- Application in VLSI Algorithms and Mobile Communic
- Case-based reasoning
- Derivative-free Optimisation Algorithms
- Evolutionary and Swarm Algorithms
- Expert Systems Components
- Fuzzy Sets and logic
- Hybridisation of Intelligent Models/algorithms
- Inference
- Intelligent Planning
- AI in Bioinformatics
- AI Tools for CAD and VLSI Analysis/Design/Testing
- AI Tools for Multimedia
- Automated Reasoning
- Data and Web Mining
- Emotional Intelligence
- Expert System Development Stages
- Expert-System Development Lifecycle
- Heuristic and AI Planning Strategies and Tools
- Image Understanding
- Integrated/Hybrid AI Approaches
- Intelligent Search

- Intelligent System Architectures
- Knowledge-Based Systems
- Logic Programming
- Multi-agent Systems
- Neural Networks for AI
- Parallel and Distributed Realisation of Intelligen
- Reasoning and Evolution of Knowledge Bases
- Rule-Based Systems
- Uncertainty
- Knowledge Acquisition
- Knowledge-Based/Expert Systems
- Machine learning
- Neural Computing
- Object-Oriented Programming for AI
- Problem solving Methods
- Rough Sets
- Self-Healing and Autonomous Systems
- Visual/linguistic Perception

CALL FOR PAPERS

Volume: 2 - **Issue:** 4 - July 2011

i. Paper Submission: July 31, 2011 **ii. Author Notification:** September 01, 2011

iii. Issue Publication: September / October 2011

CONTACT INFORMATION

Computer Science Journals Sdn BhD
M-3-19, Plaza Damas Sri Hartamas
50480, Kuala Lumpur MALAYSIA

Phone: 006 03 6207 1607
006 03 2782 6991

Fax: 006 03 6207 1697

Email: cscpress@cscjournals.org

CSC PUBLISHERS © 2011
COMPUTER SCIENCE JOURNALS SDN BHD
M-3-19, PLAZA DAMAS
SRI HARTAMAS
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6207 1607
006 03 2782 6991

FAX: 006 03 6207 1697
EMAIL: cscpress@cscjournals.org