MODELING AND OPTIMIZATION OF CHEMICAL MECHANICAL PLANARIZATION (CMP) USING NEURAL NETWORKS, ANFIS AND EVOLUTIONARY ALGORITHMS

By

WEN-CHEN LIH

Bachelor of Science in Mechanical Engineering
Chung Cheng Institute of Technology
Dasi, Taoyuan, Taiwan (R.O.C.)
July, 1993

Master of Science in Mechanical Engineering
The University of Michigan
Ann Arbor, Michigan
April, 1998

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 2006

MODELING AND OPTIMIZATION OF CHEMICAL MECHANICAL PLANARIZATION (CMP) USING NEURAL NETWORKS, ANFIS AND EVOLUTIONARY ALGORITHMS

Dissertation Approved:

Dr. Ranga Komanduri

Dissertation Adviser

Dr. Satish T.S. Bukkpatnam

Dr. Hongbing Lu

Dr. Prabhakar R. Pagilla

Dr. A. Gordon Emslie

Dean of the Graduate College

PREFACE

Higher density nano-devices and more metallization layers in microelectronic chips are unceasing goals to the present semiconductor industry. However, topological imperfections (higher non-uniformity) on the wafer surfaces and lower material removal rates (MRR) seriously hamper these pursuing motivations. Since'90, industry has been using chemical mechanical planarization/polishing (CMP) to overcome these obstacles for fabricating integrated circuits (IC) with interconnect geometries of < 0.18 μm.

Obviously, the much needed understanding of this new technique is derived basically on the ancient lapping process. Modeling and simulation are critical to transfer CMP from an engineering 'art' to an engineering 'science'. Many efforts in CMP modeling have been made in the last decade, but the available analytical MRR and surface uniformity models cannot precisely describe this highly complicated process, involving simultaneous chemical reactions (and etching), and mechanical abrasion.

In this investigation, neural networks (NN), adaptive-based-network fuzzy inference system (ANFIS), and evolutionary algorithms (EA) techniques were applied to successfully overcome the aforementioned modeling and simulation problems. In addition, fine-tuning techniques for re-modifying ANFIS models for sparse-data case using are developed. Furthermore, multi-objective evolutionary algorithms (MOEA) are firstly applied to search for the optimal input settings for CMP process to trade-off the higher MRR and lower non-Uniformity by using the previously constructed models. The results also show the simulation of MOEA optimization can certainly provide accurate guidance

to search the optimal input settings for CMP process to produce lower non-uniform wafer surfaces under higher MRR.

ACKNOWLEDGEMENTS

I wish to express my immense gratitude to my academic advisor, Professor Ranga Komanduri, for his guidance, instructing, encouragement, inspiration and continuous motivation for excellence in research. I feel fortune to him as a mentor and friend. My genuine appreciation to my co-advisor, Dr. S.T.S. Bukkapatnam for his patience, direction, reassurance and acquaintance. Sincere thanks are also due to my thesis committee members, Professor Hongbing Lu and Professor Pagilla, for their suggestions and encouragement. Special thanks are due to Professor Martin Hagan for his contributions to neural network design concepts. Also, special thanks to Professor Gary Yen for his instruction of computational intelligence course, which is so useful to my research. Thanks are due to Dr. Chandrasekaran of Micron Technology Inc. for technical advice and to the Micron Foundation for supporting our CMP project.

I am indebted to my teammates Mr. Prahalada K. Rao, Mr. Hui Yang, Mr. Milind Malshe, Ms. Yang Liu and all my friends and colleagues in Stillwater. Many thanks are due to all of them for their support and help. I would also like to thank my family – my Mom; my brother, Wen-Jone; and my sisters, Wen-Lan and Wen-Hwua, for encouraging me to overcome challenges all the time, most importantly, never to give up. Also, I sincerely thank my cute sons, Conway and Conrade for their smiles and love.

Finally, special honor and sincere appreciation give to my wife, Pei-Hsing. Without her understanding, love, constant help, happy smile, and especially her tolerance, I would never have the moral and spiritual strength necessary for the completion of this research and dissertation.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

LIST OF APPENDICES

NOMENCLATURE

CMP	Chemical Mechanical Planarization (or Polishing)
ULSI	Ultra Large Scale Integrated circuits
IC	Integrated Circuits
ILD	Inter-level Dielectrics
STI	Shallow Trench Isolation
MRR	Material Removal Rate
NU	Non-uniformity
NN	Neural Networks
FIS	Fuzzy Inference Systems
EA	Evolutionary Algorithms
ANFIS	Adaptive-based-Network Fuzzy Inference System Adaptive Neuro-Fuzzy Inference System
GP	Grid Partition
SC	Subtractive Clustering
UPM	Ultra Precision Machining
SOG	Spin-on Glasses
IMD	Inter-metal Dielectrics
RC	Resistance-Capacitance
RIE	Reactive Ion Etching
WPH	Wafers per Hour
WIWNU	Within Wafer Non-uniformity
CoO	Cost of Ownership

AFP	Abrasive-free Polishing
ECMP	Electrochemical Mechanical Polishing
MD	Molecular Dynamics
AI	Artificial Intelligence
ANOVA	Analysis of Variance
SOOP	Single-objective Optimization Problems
ES	Expert Systems
TSP	Traveling Salesman Problem
BPNN	Back-propagation Neural Network
TSK	Takagi-Sugeno-Kang
GA	Genetic Algorithms
MOOP	Multi-objective Optimization Problems
MOEA	Multi-objective Optimization Algorithms
NSGA-II	Elitist Non-dominated Sorting Genetic Algorithms

CHAPTER I

INTRODUCTION

1.1 Market Demands in the Semiconductor Industry

Undoubtedly, the electronics industry has become one of the fastest-growing industries in the past three decades. Currently, there have already been 10^8 or more nano-devices that can be fabricated on a chip in semiconductor wafers by the technique of Ultra Large Scale Integrated (ULSI) circuits to reduce the cost and to increase the performance of electronic products. According to the Semiconductor Industry Association (SIA) roadmap, the 0.25μm design rule (line width) of integrate circuit (IC) devices in production in 1997 has rapidly shrunk to ~ 75 nm in 2005. Substantial technical innovations in photo-lithography and interconnect technologies are among the efforts being made to reach lower than 20 nm ranges.

As the feature size of the IC chip keeps shrinking, the shorter wavelength of ultraviolet light is becoming a necessity, and the depth of focus of lithography tools is decreasing. Therefore, the topography of the wafer surface becomes a severe barrier in focusing for circuit pattern transfer. Furthermore, the increasing number of multi-layer films in IC chips brings numerous technological challenges in interconnects. The major challenges encountered in the interconnect technology are associated with the material changes from (SiO_2 and Al to low-k dielectrics and Cu) and the requirement of new process architectures (such as damascene process) [30]. On the other hand, the complexity of microchip design and fabrication has increased continuously with integration and miniaturization. Extremely high degree of repeatability and uniformity

are required in wafer fabrication for high production yield.

Since the beginning of ’80, Chemical Mechanical Planarization (CMP) process has gradually become one of the most widely used planarization techniques to overcome the crucial challenge of the global surface planarity for semiconductor wafers in the fabrication of interlevel dielectric (ILD) planarization, shallow trench isolation (STI), and metal damascene processes in the semiconductor industry. Additionally, new materials such as Cu, W, and low-k dielectric materials, are introduced in ULSI fabrication, requiring extensive use of CMP process to form inlaid interconnect structures.

1.2 Motivation and Objectives

A fundamental understanding of the CMP process is essential to improve process optimization and control, and to increase the process yield and throughput in the continuous integration and miniaturization in the semiconductor industry. In the past two decades, as a fundamental study on CMP, mechanical aspects of the material removal mechanism in CMP where investigated both analytically and experimentally. Among the many important variables, the role of consumables (polishing pad and abrasive particles in the slurry, for example) in CMP performance was evaluated, and tribological characteristic (lubrication, friction, and wear) features in CMP were also analyzed. To evaluate the role of slurry, the influence of chemistry on mechanical removal in material removal mechanism is examined. The mechanical and chemical contributions to material removal are studied to determine key mechanism of material removal in CMP. The whole theoretical development in CMP has to be advanced significantly if we were to advance the CMP technology.

Briefly, due to highly complicated interactions between mechanical abrasion, and chemical interactions (etching) during the polishing process, current analytical models and analyses cannot sufficiently provide more accurate process prediction for better quality control and higher throughput. With continuous shrinkage of feature size, the demand for more precise process models keeps increasing. In this thesis, three novel modeling techniques were applied for material removal rate (MRR) and within wafer non-uniformity (WIWNU) using neural networks (NN), adaptive-based-network fuzzy inference systems (ANFIS), and genetic algorithms (GA) are discussed in the following chapters. Also, the multi-objective evolutionary algorithms (MOEA) optimization technique is applied for searching the most suitable combination of input process variables to reach the optimal polishing process.

The overall goal of this research is to develop a new process modeling and optimization methodologies for highly non-linear and complex manufacturing processes, in particular CMP. Also, this methodology can be applied especially under sparse-data conditions. These newly-developed modeling methods can be applied not only to the current CMP practice or manufacturing processes but also to other fields, such as financial, risk analysis, behavior analysis in psychology, and recognition system.

1.3 Scope of the Dissertation

This dissertation is divided into the following chapters:

1. Fundamentals of CMP in Chapter II
2. Overview of CMP process in Chapter III

3. Fundamentals of Neural Networks, ANFIS, and Evolutionary Algorithms in Chapter IV.
4. Modeling and optimization of CMP process in Chapter V.
5. Results and Analysis in Chapter VI.
6. Conclusions and Future Work in Chapter VII.

Chapter II describes the general background and developments in CMP process. Chapter III presents material removal mechanisms of the CMP process. Some important process models introduced by the pioneers showing those relationships between significant process parameters and main performance variables are discussed. Chapter IV presents the applications of Neural Networks (NN) and Fuzzy Inference Systems (FIS) to the process modeling of the CMP process. The newly-developed models using Adaptive-based-Network Fuzzy Inference System (ANFIS) which is based on Subtractive Clustering (SC) method is emphasized, especially for the case of sparse available data. The simulation results are shown in Chapter VI. Chapter VII covers conclusions and future work, including process modeling and multi-objective optimization in CMP process by using genetic and evolutionary algorithms, and the later application of multisensor-fusion techniques to enhance the process models, can be expanded and employed to real-time process monitoring, such as endpoint detection in CMP, and chatter detection, prediction of surface finish in ultra precision machining (UPM), etc.

As initially discussed in Sections 2.2 and 2.3, CMP process involves highly dynamical and non-linear interactions between chemical reaction and mechanical abrasion. The potential of planarization capability and perspective market competition have motivated extensive research in industry and academia. Figure1.1 shows a research

methodology, including necessary steps for a deeper understanding and optimization of the various manufacturing processes. In this dissertation, steps involving performance variables, process parameters, analysis of results, process modeling and model testing are emphasized and discussed in detail in the following chapters. Certainly, the same ideas also can work on other manufacturing processes, such as Ultra Precision Machining (UPM).

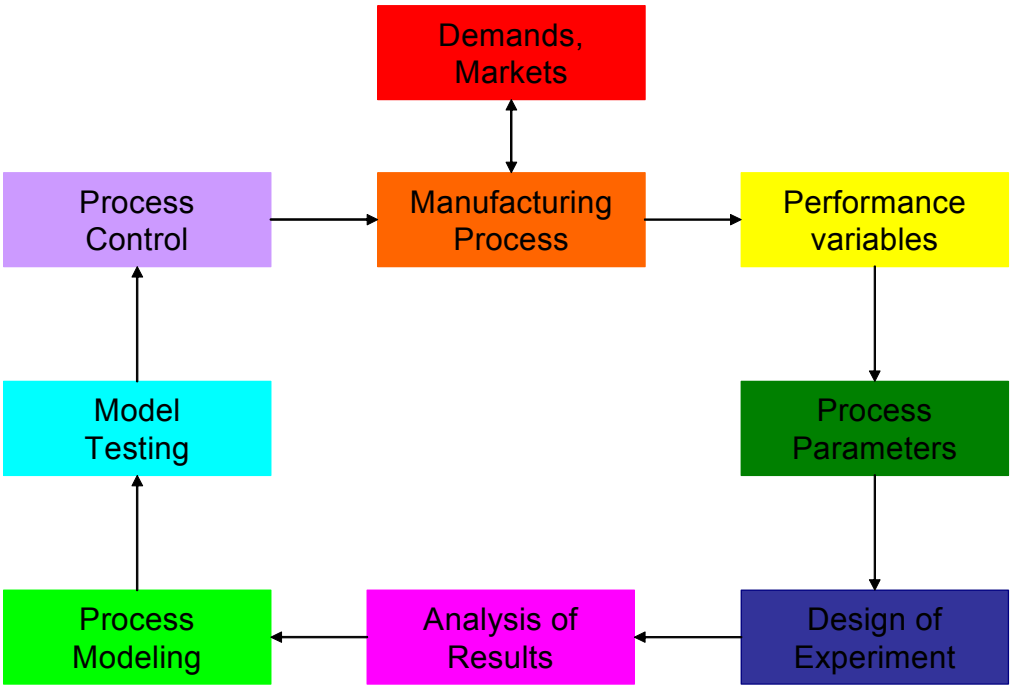


Figure1.1 Research methodology for optimizing manufacturing process

CHAPTER II

Fundamentals of the CMP Process

2.1 CMP in Semiconductor Manufacturing

2.1.1 Markets and Demands

Over a period of five-year from 2003, the research house expects the semiconductor market to grow at a compounded annual growth rate of 12.5 percent, with revenues seen rising to $282 billion in 2008 from $160 billion in 2003 (Bussiness Times Dec. 31, 2003). The CMP process is ready to make a positive impact on 30% of the global semiconductor market [25]. Simply speaking, CMP has become one of the major core technologies in the semiconductor manufacturing industry.

The relentless and never failing capability to fabricate integrated circuits (IC) on silicon wafers in a manner that continuously meets or exceeds Moore's law [31] is the physical basis for the hugely successful microelectronics industry. To drive that processing engine, critical advancements in IC fabrication processing must be available in a timely fashion. Further, as the demand for high density and high performance IC chips increases (Figure 2.1), multi-level wiring and metallization technologies, and the demands of smaller feature size (or pitch size) are widely used for new IC fabrication (Figure 2.2 and Table 2.1), a new planarization technique for lower non-uniformity and less defect density on wafer surface is simultaneously needed. Additionally, as low-k dielectrics and Cu are replacing with traditional dielectric materials and Al, respectively, more understanding and novel process modeling techniques will be absolutely necessary for next-generation planarization technologies

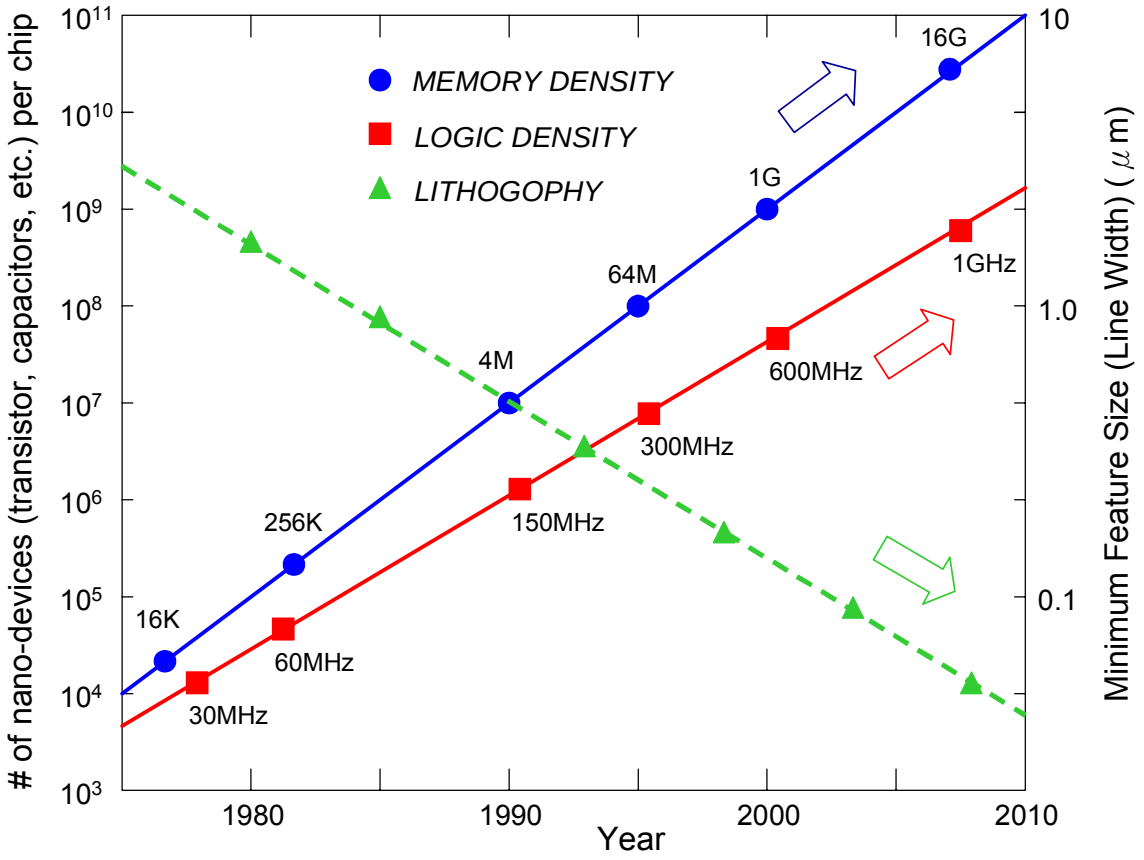


Figure 2.1 Trends in Logic, Memory Device and Lithography Technology [6]
(redrawn by Lih)

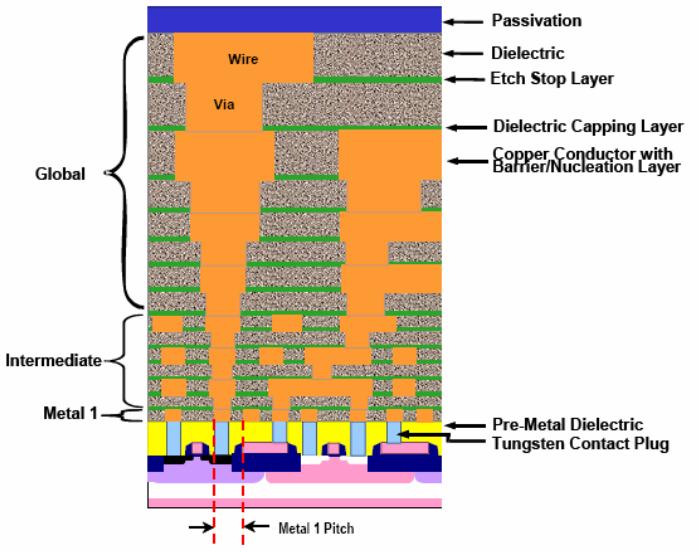


Figure 2.2 Cross-section of Hierarchical Scaling – Microprocessor (MPU) Device [6]

Table 2.1 MPU Interconnect Technology Requirements – Near-term Years [6]

Year of Production	*2005*	*2006*	*2007*	*2008*	*2009*	*2010*	*2011*	*2012*	*2013*
DRAM ½ Pitch (nm) (contacted)	*80*	*70*	*65*	*57*	*50*	*45*	*40*	*36*	*32*
MPU/ASIC Metal 1 ½ Pitch (nm)(contacted)	*90*	*78*	*68*	*59*	*52*	*45*	*40*	*36*	*32*
MPU Physical Gate Length (nm)	*32*	*28*	*25*	*22*	*20*	*18*	*16*	*14*	*13*
Number of metal levels	**11**	**11**	**11**	**12**	**12**	**12**	**12**	**12**	**13**
Number of optional levels – ground planes/capacitors	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**

2.1.2 Ultra Large Scale Integrated circuit (ULSI)

The new technology for Ultra Large Scale Integrated circuit (ULSI) was successfully developed in mid-90s. This technology can produce integrated circuits (Figure 2.3) with smaller feature size (e.g. width of microelectronic wire ~ 70 nm in 2005), more multi-layer metallization (> 9), and higher density (> 10^8) nano-devices (e.g. transistors, resistors, and capacitors) in a chip. However, the more stringent requirements for the ULSI, such as higher flatness or uniformity of wafer surface topography for providing sufficient depth-of-focus (or -field) ranges for sequential photo-lithography process (Figure 2.4) must be met, as depicted in Figure 2.5.

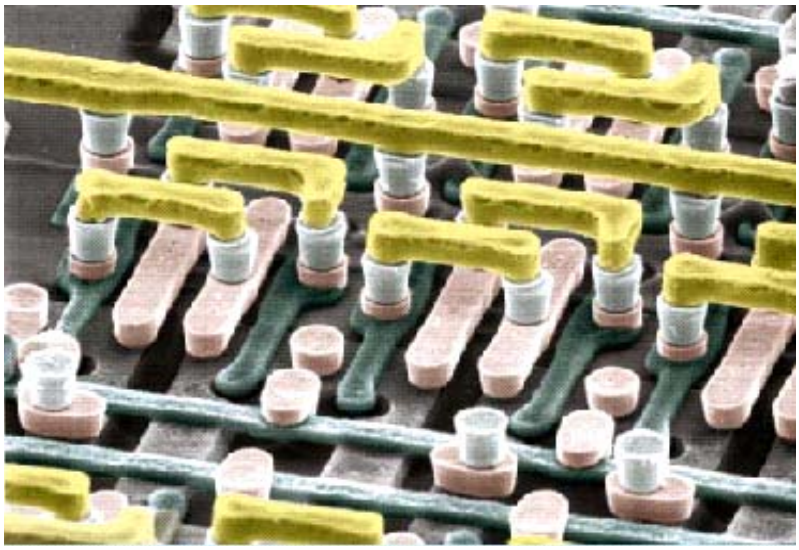


Figure 2.3 3D Structure of Integrated Circuits (IC) - different levels of metal interconnects without showing dielectric (www.me.gatech.edu/eml/cmp.htm)

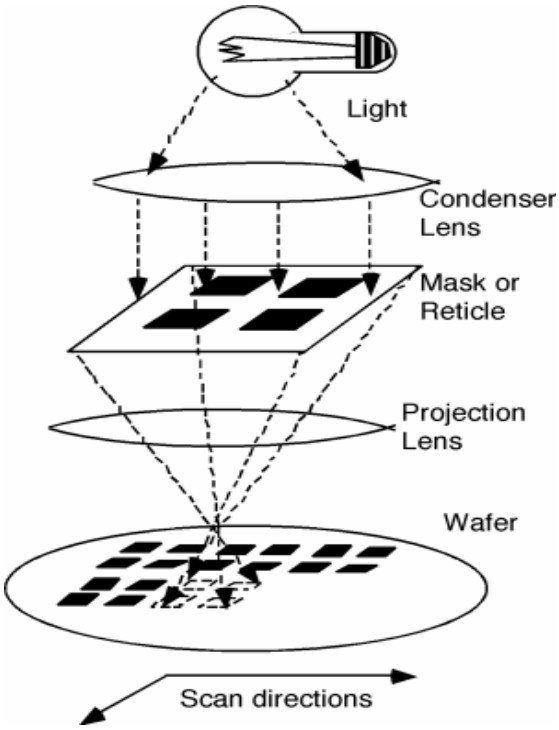


Figure 2.4 Schematic of Photo-lithography Process for IC Fabrication [9]

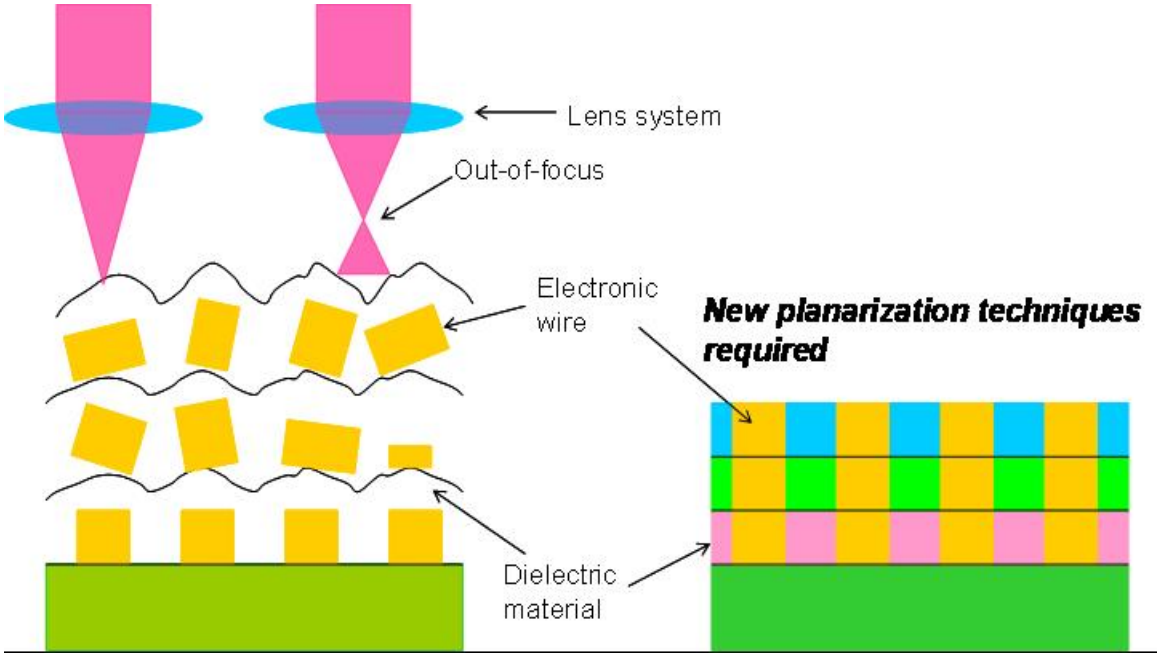


Figure 2.5 Schematics of Planarization and Non-planarization on Wafers [22] (redrawn by Lih)

2.1.3 Stringent Requirements of Lower Non-uniform Wafer Surface for ULSI

As circuits get more complex, we increasingly need to remove the effects of high topography. Routing challenges increase very rapidly with increasing numbers of transistors as mentioned in the previous paragraph. The simplest solution is to stack extra layers on top of each other to increase the interconnect routes available to the circuit. Each layer adds its own topography problems, compounded by the surface variations existing from the layers below. Within four or five layers, as shown in Figure 2.4, the problems become insurmountable, and planarization of the surface is required for reliability and yield.

Shrinking transistor size demands pattern definition tools that can print increasingly smaller features. However, increasing resolution for smaller features is obtained at the expense of the depth of focus. Depth of focus is the vertical range over which the image will be in acceptable focus (i.e. the image will adequately print). If the mask feature cannot be adequately imaged on the wafer surface (Figure 2.4), the image size will be distorted, and some of the circuit elements will have positions and spacing different from their design values. It will be up to the tolerances built into the circuit design whether the transistor element will work at all, or work with degraded performance.

Besides, many of the circuit elements are current-carrying conductors, and as such they need to have sufficient sectional area to transport the current efficiently around the circuit. As the surface dimensional features (e.g. width) shrink for greater circuit compaction, the only dimension left to support a sufficient cross-section is the vertical height of the current-carrying conductors. This means that while the circuit size shrinks from generation to generation, the height of the vertical steps continues to be significant.

The combination of these effects demands some form of planarization to reduce the circuit topography within the required depth of focus for accurate circuit imaging.

Using the Rayleigh criterion and depth of focus formula gives an expression for depth of focus, σ as shown in Equation (2.1). Table 2.2 gives the requirements for the depth of focus for smaller feature sizes. From Figure 2.5, reducing the increased cumulated surface non-planarity resulting from multi-level interconnect metallization in current ULSI is the major mission of the CMP process. Non-planarized surface topography also causes a hindrance in conformal coating of photo-resist and efficient pattern transfer with contact photo-lithography. Further, the irregular surface causes the

$$\sigma = 10.75 \cdot \frac{\boldsymbol{b}^2}{\lambda}$$
$$\boldsymbol{b} : \text{minimum feature size} \qquad (2.1)$$
$$\lambda : \text{wavelength of projection light}$$
$$\text{here } \lambda = 365 \text{ nm for Mercury}$$

Table 2.2 Minimum Feature Size and the Requirements of Depth of Focus [14]

Year	**Minimum Feature Size (nm)**	**Requirement of Depth of Focus (nm)**
2003	100	295
2004	90	239
2005	80	188
2006	70	144
2007	65	124
2008	57	96

variation of the thickness in fine wire widths, difficulties in Post-CMP cleaning process, and undesired erosion by residual chemical slurry or contamination. Obviously, this uncontrollable thickness leads to poor quality in microelectronic device.

2.1.4 Planarization Capacity of CMP

In addition to CMP, several other methods are known to enable achievement of higher level of planarization, such as laser reflow, coating with spin-on glasses (SOG), thermally reflowing material, and flowable oxide. However, CMP is the most economic process that meets the stringent requirements of local and global planarization in the semiconductor industry. By virtue of the ability of CMP to provide local and global flat surfaces of semiconductor wafers (as shown in Figure 2.6) [32], it has enabled chip designers to make use of advanced photo-lithographic patterning techniques, providing the continuous ability to shrink chips to smaller dimensions and seamlessly adding additional levels of wiring.

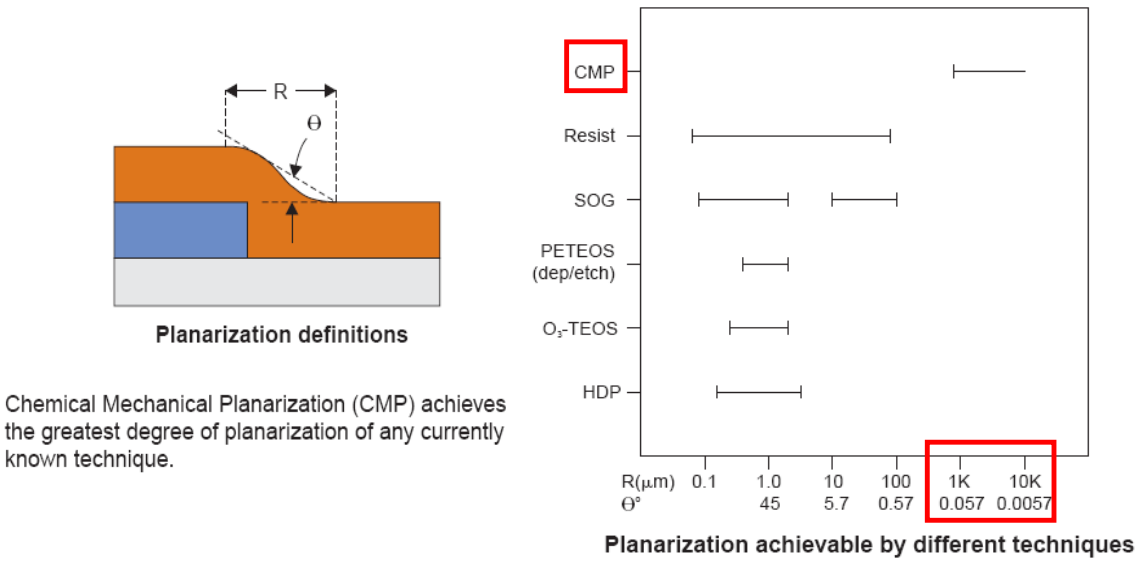


Figure 2.6 Planarization Achievable by Different Techniques (www.icknowledge.com)

2.1.5 CMP Applications in IC Fabrication

Conclusively, there are two major applications of CMP in ULSI manufacturing: (1) to smooth surface topography of inter-level dielectrics (ILD, usually silicon dioxide) and (2) to remove excess material to produce inlaid metal structure or isolation trenches. Currently, CMP is primarily applied in three areas of IC fabrications: (1) inter-level dielectric (ILD) and inter-metal dielectric (IMD) planarization, (2) Cu damascene process, and (3) shallow trench isolation (STI).

The ILD CMP is applied to conventional Al metallization, where Al is deposited on the oxide ILD layer, patterned, and etched to form interconnects. Another layer of oxide is then deposited to insulate the Al interconnects. Thus, three-dimensional microelectrical wiring is constructed. Device elements, such as resistors, capacitors, and transistors are connected to build up ICs. Figure 2.7 is a schematic of the CMP for the ILD. The desired process endpoint is determined based on the surface planarity and thickness of the ILD layer required for electrical isolation of the Al wire.

The CMP for copper damascene process (Figure 2.8) and STI (Figure 2.9) are

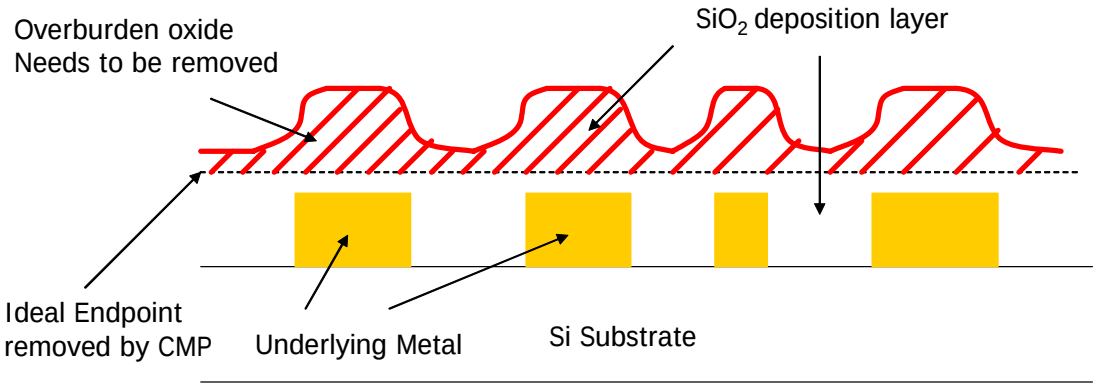


Figure 2.7 Schematic of Silicon Oxide ILD-CMP [9] (drawn by Lih)

employed to remove excess deposit covered on the trench. The underlying oxide (or other ILD material) layer is trenched by photo-lithography and etching. A thin Cu layer is deposited or electroplated to fill the trenches. The CMP process removes excess Cu and forms isolated Cu wirings. The process is stopped while the Cu layer and diffusion barrier layer (usually a thin layer of Ta, TaN, Ti, or TiN to prevent Cu diffusion into the oxide and "poisoning" the underlying devices) are completely polished through and the oxide is exposed.

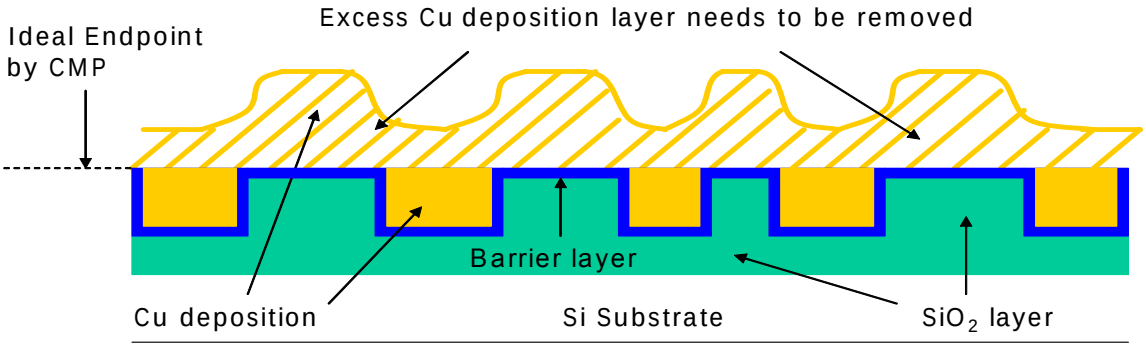


Figure 2.8 Schematic of Cu Damascene CMP [9] (drawn by Lih)

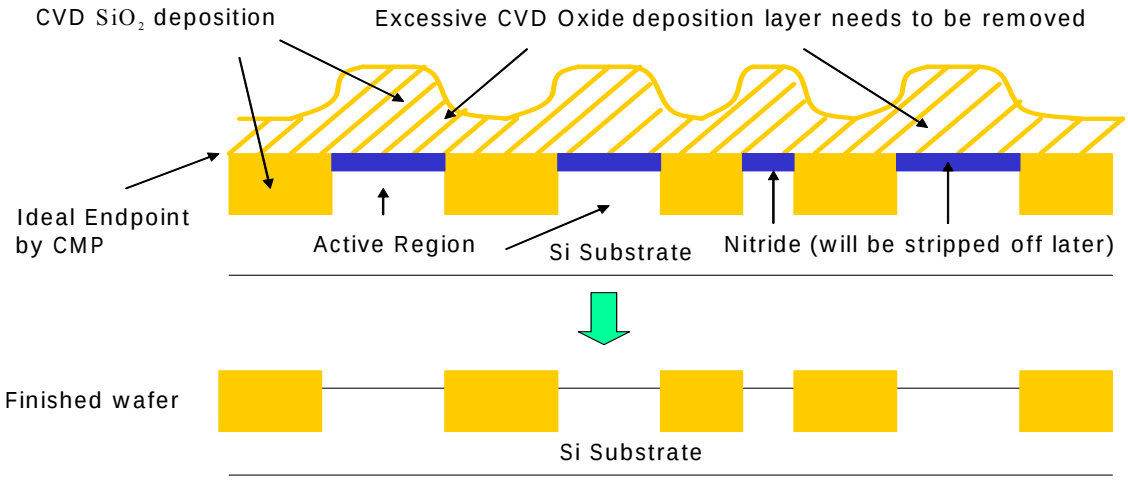


Figure 2.9 Schematic of Cu STI CMP [9] (drawn by Lih)

2.1.6 CMP in Cu Damascene and ULSI Manufacturing

Presently, Cu has begun to replace Al as the interconnect metal in IC in the new generation chips with interconnect critical dimensions below 0.25 μm. Not only has Cu the ability to reduce the resistance-capacitance (RC) delay due to its lower resistivity, but also increase the circuit reliability because of its higher electromigration resistance (electromigration is the phenomenon of metal atoms diffusing as electrical current carrying electrons "hit" them, which causing metal thinning) [33-35]. The conventional use of reactive ion etching (RIE) to pattern Cu is impractical because volatile Cu compounds form only at elevated temperatures [36]. The Cu damascene approach combined with extensive use of CMP provides a useful and economical solution for ULSI manufacturing [32, 37, 38]. In addition, Cu techniques require 20% to 30% fewer steps than conventional aluminum patterning due to new damascene approach and higher packing density of the smaller feature size [39].

For the above-mentioned reasons, major chipmakers, including IBM, Motorola, Texas Instruments separately announced in 1997 aggressive plans to put Cu into production in 1998. The main challenges, however, for Cu CMP is to control the uniformity of the surface topography while the interconnect layers increase to meet the more stringent die-level planarity requirements. Moreover, the non-uniformity must be reduced to prevent the retardation of signal transmission in interconnects, if the remaining copper wires and the variation of local surface topography due to *dishing* and *over-polishing* (or *erosion*). In addition, the difference between polishing rate of Cu and low-k materials, being replacing with the present oxide ILD layer (to reduce the

capacitance loss and increase the signal transmission rate of the circuits), is also another barrier for the new generation CMP technology.

2.1.7 Summary

- CMP has been one of the major core technologies in the semiconductor IC industry.
- ULSI demands higher flatness of wafer surface for providing sufficient depth-of-focus.
- CMP is the most economical process for attaining the stringent requirements of local and global planarization in the semiconductor industry.
- Smaller feature size of microelectronic circuits, smaller clear ranges of depth-of-focus, and lower non-uniformity on the wafer surface.
- CMP technology has been widely applied in planarizing Cu and low-k materials for deep-submicron (< 0.18 μm) IC fabrication.
- Major challenges in Cu CMP process include dishing, erosion, uniform planarization of low-k dielectric materials.

2.2 Schematic of the CMP Process and Equipment

2.2.1 Schematic of the CMP Process

Chemical Mechanical Planarization, also known as Chemical Mechanical Polishing (CMP) is an adaptation of the lapping technology used to polish plate glass. Differently, chemical reaction aids in the mechanical removal rate during polishing. As stated in Section 2.1, CMP is commonly used to polish-off high spots on the wafers or films deposited on wafers, flattening the film or wafer, referred to as planarization.

Figure 2.10 is a schematic of the CMP process. A wafer to be polished is mounted on a wafer carrier via back pressure or surface tension by wetting its back surface. This

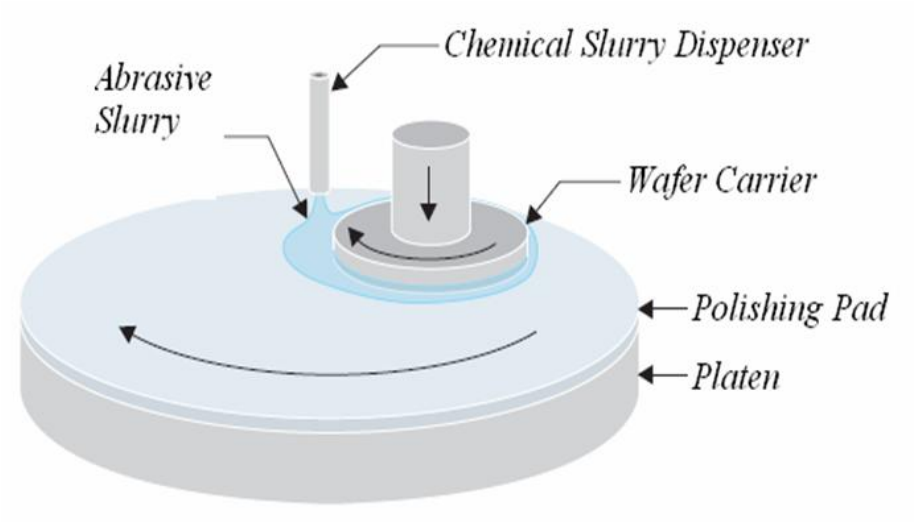


Figure 2.10 Schematic Diagram of rotary CMP Process (www.icknowledge.com)

wafer is pressed down against a rotating platen, which holds a compliant polishing pad. The wafer slides on the pad surface with a relative velocity generated by the rotation of the carrier and the platen. Concurrently, the abrasive slurry drips onto the platen surface and dispenses through the wafer and the pad contact interface. The chemical slurry and abrasive particles retained on the porous pad surface remove the material on the wafer surface as shown in Figure 2.11. The pressures (down pressure and back pressure) and velocity provide the mechanical force and velocity fields to push abrasive particles to abrade the chemically-reacted (or weakened) wafer surface by chemical slurry [16, 40].

During wafer polishing, the down pressure presses the wafer down onto the pad surface, back pressure provides uniform force distribution on the wafer surface. The abrasive slurry typically is dispensed onto the polishing pad upstream form the wafer carrier and centrifugal force causing abrasive slurry to move radially outward. As above-mentioned, the wafer is polished by the interaction of the wafer surface with the chemical

slurry, abrasive particles, and the polishing pad. During wafer polishing, the pad surface structure is also planarized. The protruding pad material is compresses by contact with the wafer and is abraded by the interaction with the abrasive slurry and the wafer surface. Residual pad material, abrasive particles, and re-deposited material from the wafer surface are deposited into pad pores, thus filling up the pores, causing a glazed appearance and diminished polishing performance. This is one of the key issues that results in the variation of material removal rate (MRR) and change in the material removal mechanisms.

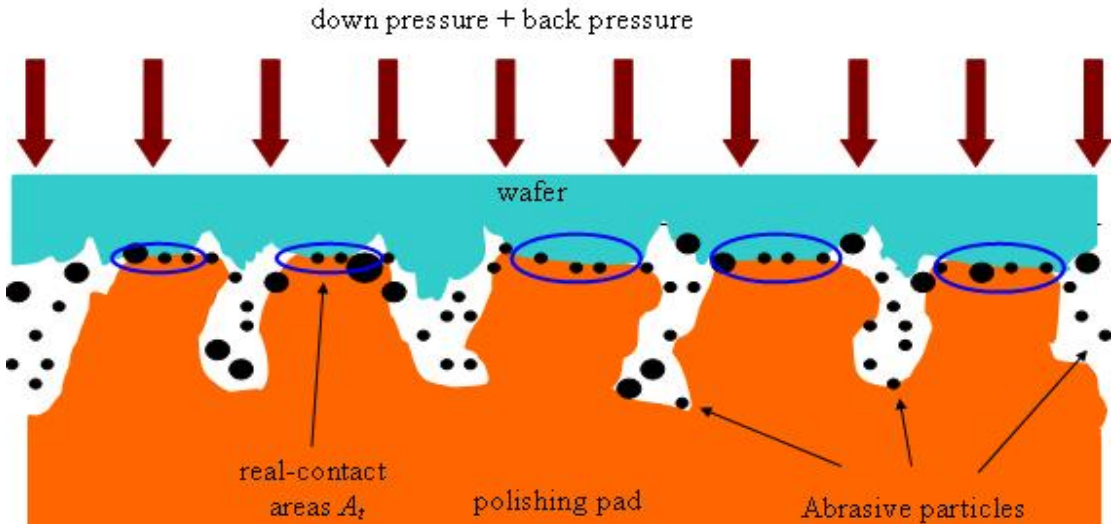


Figure 2.11 Contact Interfaces between Abrasive Particles and Porous Pad Surface [16] (drawn by Lih)

2.2.2 CMP Equipment

Figure 2.12 show the first-generation polisher, typically using a single robot to move wafers through the tool, and this robot also holds the wafer on the carrier head. The main system of the CMP polisher include platen, wafer carrier, slurry pump, down pressure control, back pressure control, spindle driving motors, temperature control, spindle speed control, and endpoint detection device. The removal rate on this first-generation polisher can be improved by simply increasing the rotational rate. However, the non-uniformity

can suffer if the platen run-out is not controlled. Platen run-out is a measure of a wobble which a platen undergoes during rotation and must be kept to a minimum to ensure good performance. Undoubtedly, this is one of the challenges in the development of CMP technology.

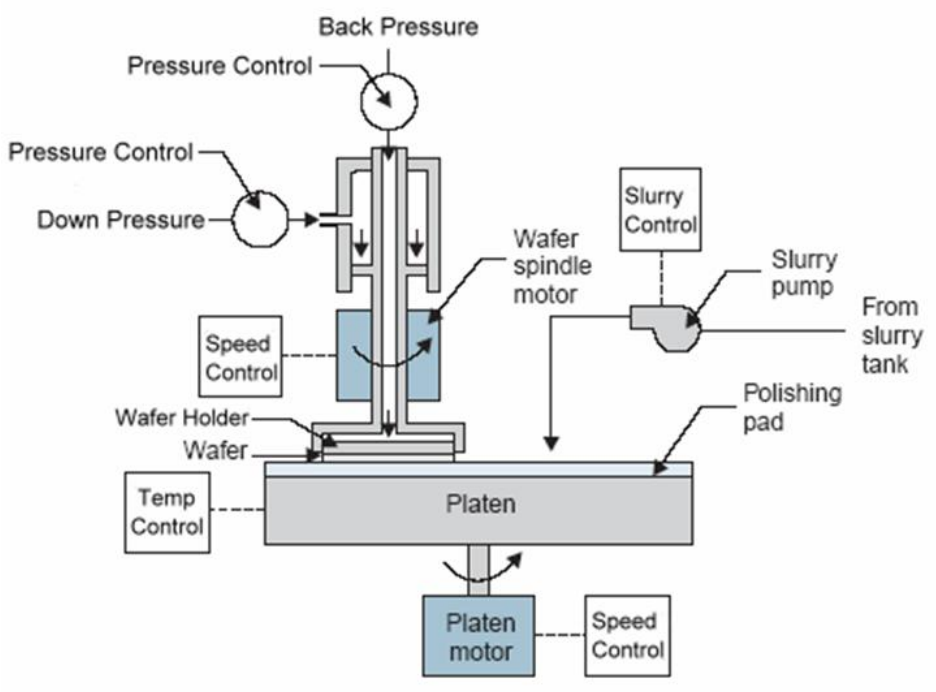


Figure 2.12 Schematic Diagram of a CMP Equipment (www.icknowledge.com) (redrawn by Lih)

2.2.3 Advanced Features in CMP Polisher

The advanced features involved in the newly-developed CMP polishers are listed as the following:

- Automatic robotic handling of the wafers for high precision control and processing stability.
- Multiple wafer-carrier spindles for higher throughput (> 70 wafers per hour or WPH)
- Built-in post CMP cleaning system for dry-in and dry-out features

- Advanced endpoint detection systems involving optical, change of motor current of spindles, eddy current, stop-layer coating, chemical potentiometer.
- Advanced wafer carriers with variables back-pressure distribution for reducing within wafer non-uniformity (WIWNU) by uniform pressure distribution.
- Sophisticated slurry delivery system for less slurry consumption, uniform slurry thickness
- Built-in temperature control system for effectively cooling processing temperature to prevent undesired thermal deformation on polisher and wafers.

2.2.4 Non-traditional CMP Polishers – Examples

1. Linear-type CMP Polisher (developed by LAM Research Corporation) as shown in Figure 2.13.
   - Reducing normal force – causing undesired pad deformation and leading to larger WIWNU.
   - Increasing lateral force – causing larger lateral shear force for planarizing the wafer surface and increasing the material removal without influencing the WIWNU.

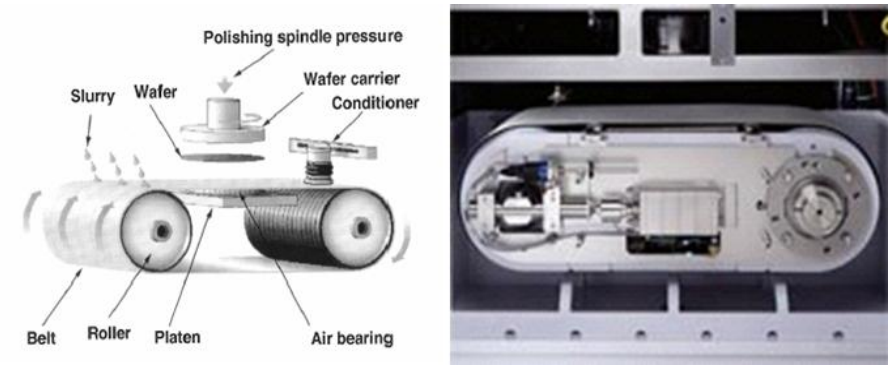


Figure 2.13 Linear-type CMP Polisher (www.lamrc.com)

2. Rotary Inverted CMP Polisher (developed by Nikon Inc.) as shown in Figure 2.14.
   - Face-up polishing using small pads applied at very low-pressure and high-speed rotation. Good for low-k materials and copper CMP.
   - Compact pad with light-weight and less pad deformation features. Good for lowering down WIWNU and increasing efficiency of slurry uniform distribution.
   - Enabling continuous optical or other non-contact endpoint detection techniques for more accurately monitoring the polishing process.

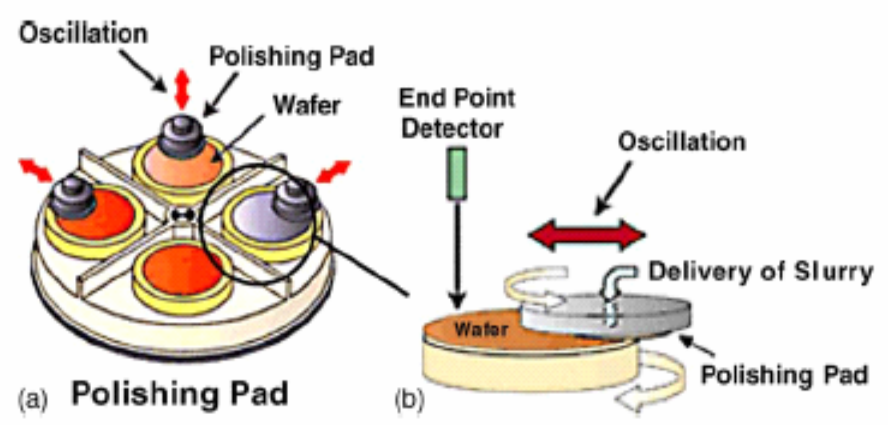


Figure 2.14 Rotary inverted CMP Polisher
(www.nikon.co.jp/main/eng/products/cmp_e.htm)

3. Pad-feed (or Web-type) CMP Polisher (developed by Applied Materials, Inc.) as shown in Figure 2.15
   - Roll-type polishing pad replacing the conventional circular-fixed pad is continuously fed onto platen.
   - No need to condition and no need to frequently shut down the machine for changing polishing pad which maximizes the equipment utilization time.

- Easy to combine with the new Fixed-Abrasive Polishing Pad (developed by 3M)

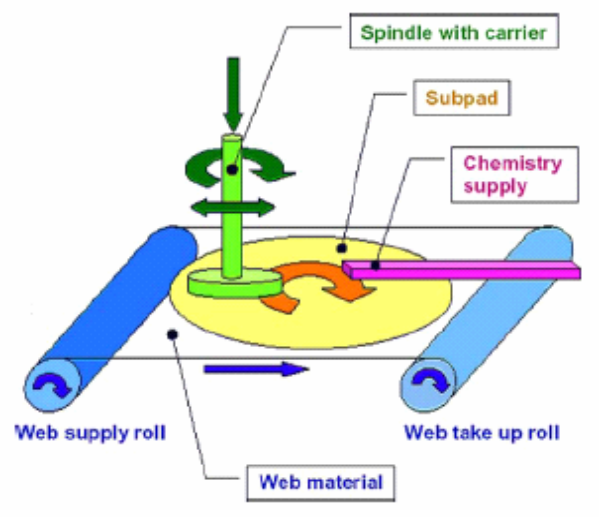


Figure 2.15 Pad-feed (Web-type) CMP Polisher (www.appliedmaterials.com)

2.2.5 Summary

- The pressures and relative velocity "push" abrasive particles to abrade the chemically-reacted wafer surface by chemical slurry.
- Advanced features in newly-developed CMP polisher provides higher precision in wafer handling, endpoint detection, slurry delivery, thermal control, and higher wafer throughput with fewer defects and more uniform surface topography.
- Non-traditional CMP polisher can planarize wafers under wider ranges of pressure and relative velocity for new materials (e.g. low-k dielectrics, Cu) in IC fabrication, and provide more advantageous options without declining output performance.

2.3 Challenges, Goals and Complexity of the CMP Process

2.3.1 Current Challenges

The year 2005 has marked over 20 years since the initial CMP patents were filed by IBM [41]. Opportunities for expanding the use of CMP in existing IC fabrication continue to flourish. In addition, the challenges (Table 2.3) ahead for CMP process to keep pace are formidable in this third wave of the evolution of the technology. Increasing concern about the challenges listed in Table 2.3 are all requirements for advanced microelectronic devices. Moreover, there is increasing attention to the overall cost of ownership (CoO) and logistics as the total number of CMP operations increases dramatically.

Table 2.3 Current Challenges of CMP Process [6] (arranged by Lih)

Challenges	Remarks
Smaller WIWNU and higher MRR	Smaller non-uniformity surface topography (< 3%) on larger size wafers (≧300 mm or 12 in)
Less defect density and contamination	Less dishing, erosion, over-polishing, under-polishing, chemical residual, micro/nano-scratch, residual stress, etc.
In-situ Process Monitoring System	Novel *in-situ* monitoring system for accurate endpoint detection, defect detection.
Real-time Process Control	To improve the rate of high quality and reduce the production costs.
Harmless abrasive slurry	To decrease overall costs for treating chemical waste, and potentially harmful to the environment and the ecological systems.
Accurate processing models	To effectively and accurately predict or capture the dynamic behavior during the polishing process, and improve the efficiency of process control.

2.3.2 Achieving Goals

Initially, except for the consideration of CoO, two more key issues affecting industry use of CMP semiconductor chip manufacturing are the lack of industry-wide CMP technology and integration knowledge, and the less than thorough understanding of the underlying science behind CMP. Presently, the leading CMP equipment suppliers not only sell CMP tools, but they also offer CMP process implementation and integration support as well. Five paramount short-term goals as listed in Table 2.4 are being quickly achieved. Obviously, CMP technology has been evolving and maturing to meet the above-mentioned challenges.

Table 2.4 Short-term Goals of CMP for ULSI in Semiconductor Manufacturing [25] (arranged by Lih)

Goals	Due to
Higher throughput and lower WIWNU	the new-generation polisher with multi-polishing heads (carriers), higher material removal rate (MRR), and novel polishing pads and polisher design.
Integration of the Film thickness metrology	the film thickness metrology has been successfully integrated into CMP tool, which greatly reduces or eliminates lost productivity due to test wafer queuing delays.
Advances in Endpoint stop and detection techniques	the integrated sensor systems for endpoint detection and development of stop-layer deposit accurately halt polishing process.
Integrated dry-in and dry-out configuration	the integration of Post-CMP cleaning for minimum defectivity and reduction of wafer transport time.
Superior Surface Finish	the less agglomeration of abrasive particles during polishing, novel finer abrasive particles and higher slurry selectivity.

Currently, new-generation polishers involving novel machine tool designs (i.e. high precision spindles, vibration-absorbing systems and automated robot systems), precision

control systems, advanced sensing systems, temperature control systems, and integrated configuration (i.e. combining post-CMP cleaning process), also known dry-in and dry-out capability are available. Apparently, these evolutions greatly improve the CMP process for higher throughput (i.e. production rate) and better quality control (e.g. lower WIWNU and fewer defects on the polished wafer).

Through incessant improvement, CMP has become a required semiconductor processing module used in sub-micron integrated circuit (IC) fabrication worldwide in the last fifteen years or so. As previously mentioned, CMP provides an extendable processing technology that enables chip makers to stay ahead of the depth-of-focus limitations in evolving optical exposure tools. Even this technique, derived from an age-old machining process, namely, glass lapping and metallographic finishing, has unexpectedly enabled optical lithography to work.

2.3.3 Complexity of the CMP Process

Inherently, chemical reactions and mechanical abrasion during CMP polishing lead to high non-linearity and complex material removal mechanisms. Their interactions at the interfaces between polishing pad, abrasive particles, slurry chemicals, and wafer materials bear complicated patterns of temperature, vibration, material distribution, forces, pressure, and fluid flow [20]. More than 30 variables and their coupled spatio-temporal dynamics are known to affect CMP process performance (Table 2.5). The current analytical CMP models with various assumptions can not sufficiently capture the incessant drift of complex behaviors, interactions, and relationships. At present, the statistical phenomenological models and experimental optimization are other approaches widely used for process design in industry [25]. However, these approaches, largely

rooted in linear statistics, are unable to provide satisfactory prediction or estimation. With the huge market potential, as above-mentioned, an effort to develop novel modeling methodology for polishing mechanisms and process control is essential to effectively keep improving the CMP technology.

Table 2.5 Various parameters and variables involved in the complex CMP process [9, 25] (arranged by Lih)

Process Parameters		State Variables	Performance Variables
Machine	Down Pressure, Back Pressure, Platen Speed, Wafer Carrier Speed, Oscillation Speed, Slurry Flow, Vibration, etc.	Stress Distribution, Velocity Distribution	**Endpoint Control** (Remaining Thickness Control)
Polishing pad	Stiffness (or Hardness), Macrostructure, Microstructure, Porosity, Topography, Pattern, etc.	Condition, Wet Hardness, Degradation, Temperature Distribution	**Material Removal Rate** (Å/min)
Slurry	Oxidizers, pH, pH Stabilizer, Complexing Agents, Dispersants, Selectivity ratio, Temperature	pH drifts, Concentration, Temperature Rise, Slurry Thickness	**Planarity** : Within Wafer Non-uniformity (WIWNU), Wafer to Wafer Non-uniformity, Within Die Non-uniformity (WIDNU)
Abrasive Particles	Size, Shape, Hardness, Chemistry, Density, Oversized Particles	Size Distribution, Aggregation, Agglomeration, Concentration, Debris	**Defects** : Dishing, Erosion, Micro-scratch, Pits, etc.
Wafer	Size, Curvature, Properties of Coating (E, ν, H), Initial Coating Thickness, Coating Thickness Variation, Pattern Geometry	Direct Contact, Semi- Contact, Hydroplaning	**Surface Finish** : Roughness, Waviness, Form Accuracy

2.3.4 Summary

- For deep-submicron (< 0.18 μm) IC fabrication, there still exist many challenges in CMP technology, such as machine tool design, defect density depression, slurry ingredients, accurate processing modeling.
- Due to high non-linearity and complexity of material removal mechanisms in CMP processes, novel and effective approaches for modeling the removal mechanisms are essential to improve the predictive capacity for the performance variables.

2.4 New Developments for CMP Process

In order to overcome the gigantic challenges ahead of the CMP process, a number of new techniques, for improving the polishing process, increasing the planarization quality, decreasing defect density, and reducing the harmfulness to the environment and the ecological systems, have been continuously proposed in the past several years. Four main categories are divided in the following:

Machine Tool Design

- New design concepts for the next-generation CMP polisher
- Wafer carrier design

Fix-abrasive Polishing

- Higher selectivity of step heights in feature-scale
- Lower WIWNU

Abrasive-free Polishing (AFP)

- Lower down pressure for Cu-CMP and low-k materials
- Fewer defects

Electrochemical Mechanical Polishing (ECMP)

- Fewer defects
- Lower WIWNU for 60 nm-process for lower

CHAPTER III

OVERVIEW OF THE CMP PROCESS

3.1 Introduction

Although CMP has been widely applied in the IC fabrication, it is still a process of trial and error. Understanding the basic mechanisms of the process has initiated research efforts from both industry and academia over a decade. As mentioned in Chapter I, CMP appears to contain two cooperating physical mechanisms [16, 40]. First, chemical interaction of the slurry with the material at the surface of the wafer weakens the surface to be polished. Second, the weakened surface is mechanically removed by a combination of abrasive particles, asperities of polishing pad, and hydrodynamic effects (Figure 3.1). The extent of each component is not well known. The individual contributions of the chemical and mechanical actions to material removal are dependent on the process parameters and consumables used.

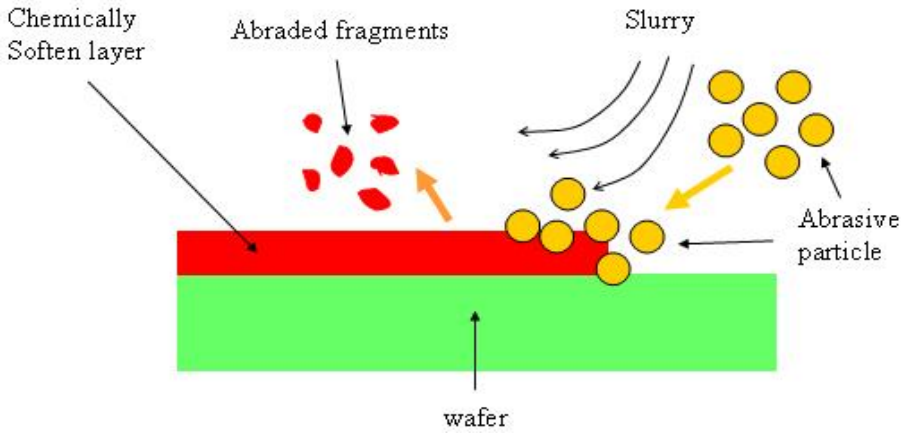


Figure 3.1 Schematic of Particle-scale Material Removal in CMP

The mechanical actions during CMP include *indentation*, *scratching*, *sliding*, and *rolling* of the abrasive particles that are held between the wafer surface and the polishing pad under the applied down-pressure (P) and relative velocity (V_r). The chemical influences include *surface passivation* and *etching* of the film materials by the slurry chemistry [40] and in some cases direct interaction between the abrasives and the surface that is required to be polished and planarized (Figure 3.2, particle-level schematic of the chemo-mechanical polishing [20]). Both chemical and mechanical actions are coupled together during the removal of the wafer surface material.

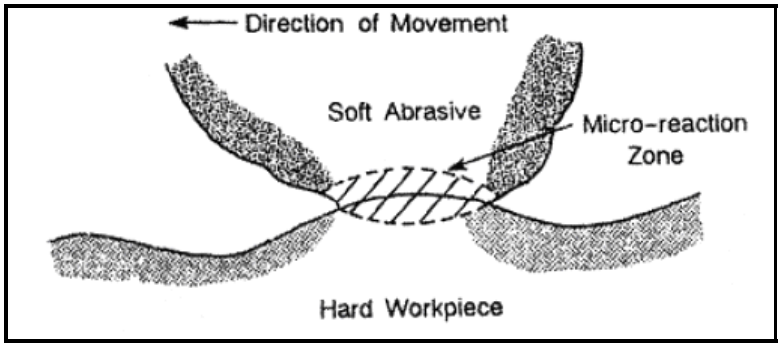


Figure 3.2 Particle-level Schematic of Chemo-mechanical Polishing [20]

Furthermore, the four most important elements in the CMP material removal process are the *abrasive particles*, *slurry chemicals*, *polishing pad* and *wafer materials*. Totally, there are six interactions among these four elements, namely, abrasive-wafer, abrasive-pad, pad-wafer, slurry-abrasive, slurry-pad, and slurry-wafer. These interactions individually contribute some extent to the material removal. Further, some interactions might be helpful to the mechanical action, but may reduce or slow down the effect of chemical actions. All of these questions will need more insightful analyses to get answers.

On the basis of performance variables in Table 3.1, CMP models are categorized into four types depending on the scale of observation as *particle*-, *feature*-, *die*-, and *wafer*-models, from several ten nanometer to 300 mm (or larger) ranges.

Table 3.1 CMP performance variables at different scales [15]
(arranged by Lih)

Model Scales	**Output Performance Variables**
Particle-scale	MRR at each single point, micro-scratch, surface roughness, micro-defects, etc.
Feature-scale	Step height reduction rate, dishing, erosion, delamination, etc.
Die-scale	Within die non-uniformity (WIDNU), etc.
Wafer-scale	Within wafer non-uniformity (WIWNU), MRR.

3.2 Preston's Polishing Model

In a typical modeling approach, the performance variables (e.g. MRR) are models as functions of easily controllable process parameters (e.g. down pressure, platen speed, etc.). The most basic model is one that predicts the bulk rate of material removal in a macroscopic fashion. An empirical model as depicted in Equation (3.1) developed by Preston [42] is widely accepted, in which the rate of material thickness reduction is proportional to the product of (1) the relative velocity between the wafer and the polishing pad, and (2) the applied pressure on the wafer surface:

$$\frac{dz}{dt} = K \cdot \frac{N}{A} \cdot \frac{ds}{dt} \tag{3.1}$$

where z is wafer thickness, t is time, $\frac{N}{A}$ is the pressure due to normal force N on the area A, and s is the distance some point on the wafer travels in contact with the pad. Preston's model in Equation (3.1) also conventionally appears as

$$MRR = K_p \cdot P \cdot V_r \tag{3.2}$$

where MRR is the material removal rate, K_p an all-purpose coefficient, P the down pressure and V_r the relative velocity between wafer and pad. Preston's model includes all effects of slurry chemicals, abrasive particles and polishing pads into the empirical constant K_p. It provides a linear dependence of MRR on the pressure and relative velocity. Its prediction of first order dependence on pressure and relative velocity is a good approximation and is the starting point for a majority of the mechanical CMP models proposed in literature. An equivalent model is the Archard's model [43] in area of wear. Even though Preston's model in Equation (3.1) is widely used in the CMP industry for a long time, because this model was created purely from mechanical actions, other actions such as chemical reactions were lumped into an all-purpose coefficient, K_p, also know as Preston coefficient. This model does not explicitly include the consumable and wafer parameters in this model.

Practically, experimental study at very low pressures and relative velocities appear to indicate a slightly nonlinear or different behavior (also known as non-Preston's effect) in these regimes. In addition, not all experimental MRR data in CMP, specially, in metal

CMP, supports the linear dependency. Maury *et al*. [44] introduces a fitting parameter $\boldsymbol{MRR}_0$ into Preston's model:

$$MRR = K_e \cdot P \cdot V_r + MRR_0 \tag{3.3}$$

Later, Wrschka *et al*. [45] proposed the nonlinear experimental equation to get a better fit of the experimental data.

$$MRR = Ke \cdot P^{\alpha} \cdot V_r^{\beta} \tag{3.4}$$

where α, β are two fitting parameters.

Current approach towards modeling CMP processes can be divided into three different starting fields, such as micro contact mechanics, thin-film fluid dynamics, and tribological lubrication. Some newly proposed models cover two of them at a time. In other words, it is obvious that the material removal mechanisms of CMP processes is complicated and can not be simply explained by only one of above-mentioned theories.

3.3 Microscopic Views of CMP Process

The proposed particle-scale models were reviewed in this section. As given in the Table 3.1, the two most important issues at the particle-scale are the material removal rate (MRR) and the surface quality, e.g., surface roughness and micro-scratches (Figure 3.3). The MRR determines the production rate of the process. Moreover, it is most sensible to process conditions, and therefore an output which is most of interest in process optimization. An understanding of the MRR first requires an understanding how the MRR varies with the process conditions.

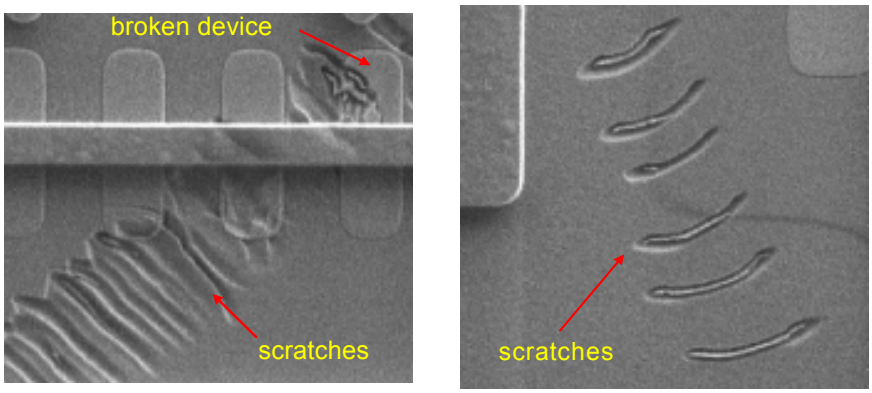


Figure 3.3 SEM photos of Micro-scratches in IC [11]

In addition, surface qualities determine the yield of the process. The micro-scratches diminish the reliability and reflectivity as well as the endurance of the surface. With the surface scratches, which are more or a concern in metal CMP than oxide CMP, the corrosion resistance in the thin film may be lowered and a short circuit was caused easily by *electromigration*. Therefore, an understanding of the formation mechanism of scratches and means for their reduction is needed. With low-k dielectrics replacing oxide dielectrics as the primary dielectric materials, the *delamination* (Figure 3.4) of the dielectrics and interconnect metals during CMP is becoming a critical surface quality issue as well.

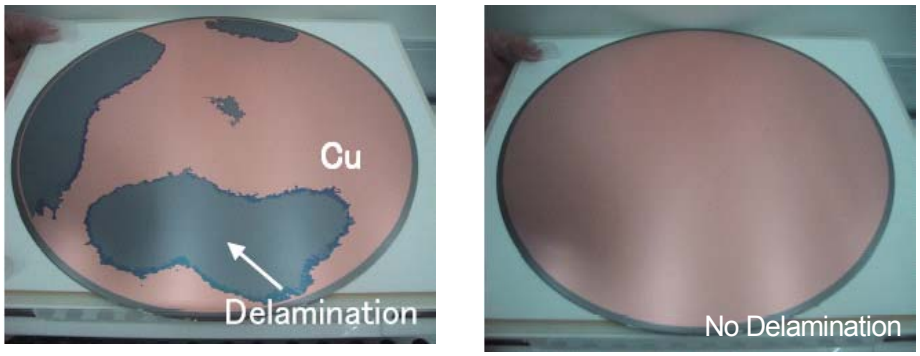


Figure 3.4 Delamination of low-k dielectrics on Cu wafer [23]

3.3.1 Interactions between Abrasive Particles and Wafer Materials

The interaction between an abrasive particle and the wafer surface is proposed as a Hertzian elastic penetration [46] of a spherical particle under uniform pressure $\boldsymbol{P}$ into the wafer surface, sliding along the surface with a relative velocity $\boldsymbol{V_r}$ and removing wafer volume proportional to the penetration (Figures 3.5 and 3.6). Cook [16] proposed a MRR formulation as:

$$MRR = (2E_w)^{-1} \cdot P \cdot V_r \tag{3.5}$$

where $\boldsymbol{E_w}$ is the Young's modulus of the wafer material. This model is taken as a theoretical verification of the Preston's model since it supports the linear dependency of $\boldsymbol{MRR}$ on pressure $\boldsymbol{P}$ and relative velocity $\boldsymbol{V_r}$..

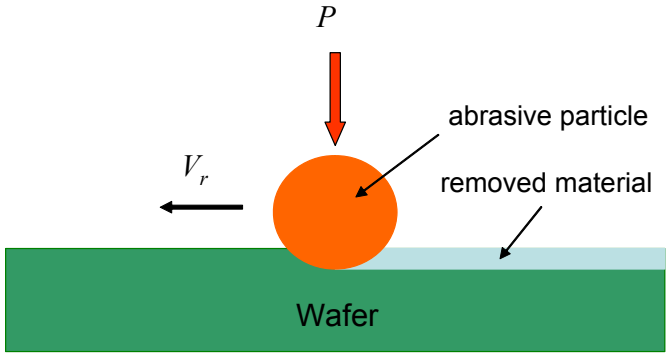


Figure 3.5 Mechanics of Abrasive Particle-Wafer Contact [16]

A similar model was developed by Liu *et al.* [47] based on the statistical method and Hertzian elastic penetration. Except for the wafer material parameters, including wafer hardness $\boldsymbol{H_w}$ and wafer Young's Modulus $\boldsymbol{E_w}$, this model also includes pad hardness $\boldsymbol{H_p}$ and Young's Modulus $\boldsymbol{E_a}$ of the abrasive particles:

$$MRR = C_e \cdot \left(\frac{H_w}{H_w + H_p} \right) \cdot \left(\frac{E_a + E_w}{E_a \cdot E_w} \right) \cdot P \cdot V_r \tag{3.6}$$

where $\boldsymbol{C_e}$ is a coefficient to account for the effect of slurry chemicals and other consumable parameters. This model, similar to Cook's model, suggests that material removal is proportional to the applied pressure and relative velocity.

The advantages of Cook's and Liu's models over Preston's model are that they provide insights into the role and interactions of the consumable parameters (i.e. hardnesses of abrasive particle and wafer, etc.). The mechanical removal by abrasive particles is the dominant mechanism in these two models.

Runnel *et al*. [48] assume that there exists a fluid film between the wafer and the pad interface, which affects the erosion/material removal rates at each single point through fluid stress tensors:

$$\boldsymbol{MRR} = \boldsymbol{C_e} \cdot \sigma_t \cdot \sigma_n \tag{3.7}$$

where C_e is an all-purpose coefficient , σ_t is the shear stress due to the slurry flow and σ_n the normal stress. They consider the material removal is due to mechanically enhanced erosion. Tseng and Wang [49] attributed the normal stress at the particle-wafer contact to the elastic indentation of the normal stress into wafer surface and calculated the normal stress over the wafer-particle interface as

$$\sigma_n = \frac{F}{\pi \cdot r_c}, \quad r_c = \left\{ \frac{3}{4} \cdot \frac{x}{2} \cdot F \cdot \left[\frac{\left(1 - \nu_w^2\right)}{E_w} + \frac{\left(1 - \nu_a^2\right)}{E_a} \right] \right\}^{1/3} \tag{3.8}$$

where $\boldsymbol{F}$ is force acting on the spherical particles, which is proportional to the pressure $\boldsymbol{P}$. $\boldsymbol{r_c}$ is the radius of wafer-particle contact, $\boldsymbol{x}$ is the diameter of abrasive particle, ν_w and ν_a the Poisson's ratios of wafer surface and abrasive particles, and $\boldsymbol{E_w}$ and $\boldsymbol{E_a}$ Young's moduli of the wafer and abrasive particles, respectively. The shear stress due to the slurry flow can be approximated as

$$\sigma_t = \boldsymbol{C_e}\sqrt{\mu \cdot \boldsymbol{V_r} \cdot \boldsymbol{P} \cdot \boldsymbol{A_0}} \tag{3.9}$$

where μ is the dynamic viscosity of the slurry and A_0 is the area of the wafer surface. Substitution of Equations (3.8) and (3.9) into Equation (3.7) yields:

$$MRR = K_e \cdot P^{5/6} \cdot V_r^{1/2} \tag{3.10}$$

where K_e is the parameter to account for material properties, slurry abrasive concentration and chemical processes.

Zhang and Busnaina [50] estimated the contact pressure between the particle and the contact surface and found that it is larger than the yield stress of the polished materials. Therefore, they proposed that plastic deformation is a more likely deformation mechanism of polishing surfaces. The contact pressure over the particle-wafer interfaces is suggested to be equal to the hardness H_w of the wafer materials. Replacing the normal stress in Equation (3.8) with the hardness H_w yields:

$$MRR = K_e \cdot (P \cdot V)^{1/2} \tag{3.11}$$

where $\boldsymbol{K_e}$ is the parameter to account for material properties, slurry abrasive concentration and chemical processes.

Moreover, it is noted that besides the external force applied on the abrasive particles from the pad, Zhang and Busnaina [50] also proposed that adhesion force, either van der Waals force and electrostatic force depending on the separation distance between the particle and the wafer, contributes to the indentation [51]. Ahmadi and Xia [52] also consider the adhesion wear of wafer in their model. In addition to the above analytic models, several particle-wafer interaction models based on molecular dynamics (MD) simulation have been developed for both silicon and Cu CMP.

The above models imply that abrasive particles are embedded into the pad and particles float in the slurry and impact the wafer surface from time to time (three-body abrasion). Models attributing the material removal to the erosion enhanced by the three-body abrasive impact, instead of a two-body indentation similar to that given by Tseng and Wang [49], may be developed (Figure 3.9 on page 43).

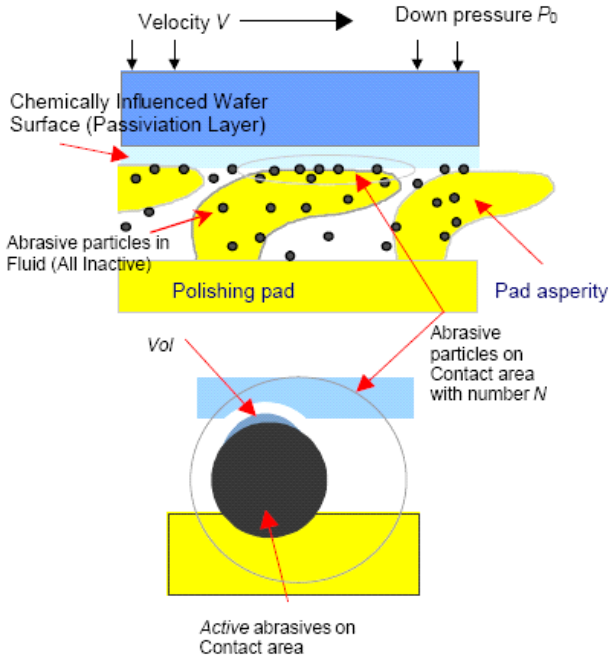


Figure 3.6 Schematic of Interactions between Abrasive Particles and Wafer [15]

3.3.2 Interactions of Polishing Pad and Wafer Materials

It has been observed that the pad topography and pad material play an important role in the material removal process. The MRR increases with the pad surface roughness [22] and softer pad yields a larger MRR [22]. Without conditioning of the polishing pad, the MRR decreases exponentially with polishing time [53, 54]. Li, *et al*. [54] reported MRR initially decreases almost linearly and then stays at a low value during polishing Si-Oxide without pad conditioning. They have suggested that the pad surface roughness is responsible for the imperfect planarization, especially for deep, narrow features. In contrast, they also proposed the planarization efficiency, defined as the step-height reduction per unit oxide removed at the "up feature" region, increases linearly with the polishing time. Steigerwald *et al*. [27] proposed that the MRR is proportional to the number of abrasive particles over the contact area as shown in Figure 3.6. In other words, the capacity of holding abrasive particles of the polishing pad is greatly decreased under the worn-out (or increasing of flattened portion) status (Figure 3.7), reducing the two-body mechanically abrasion actions.

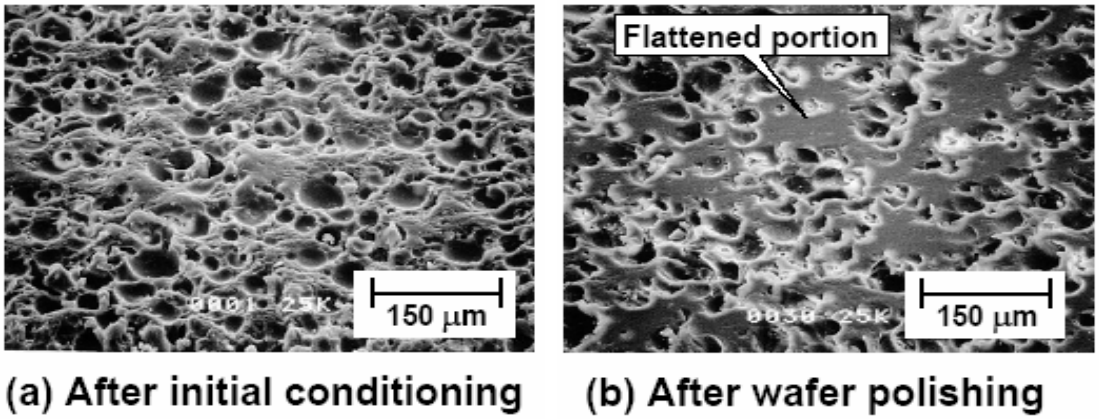


Figure 3.7 SEM Photos of Polishing Pad Surfaces [11]

Yu *et al*. [18] approximated the peaks on the pad surface by hemispherical asperities of constant radius and the asperity height was assumed to follow a Gaussian distribution (Figure 3.8). On the basis of this model, the real contact area is smaller than the nominal contact area and proportional to the down pressure. Combining Yu's model [18] with Steigerwald argument [27] yields a linear dependency of MRR on down pressure. This agrees with Preston's model.

Zhao and Shi [55] also proposed a model based on pad asperity-wafer contact without considering the Gaussian distribution of asperity heights. The contact area between an asperity and the wafer surface is given by $A \propto P^{2/3}$ based on Hertz elastic contact theory. Similarly, combining Steigerwald's argument [27], the MRR can be obtained as:

$$MRR = K_e(V_r) \cdot P^{2/3} \tag{3.12}$$

where $K_e(V_r)$ is a function of the relative velocity V_r and other CMP parameters. Further, Zhao and Shi [55] also considered that when the particles are rolling against the wafer surface, their contribution to MRR will be negligible. They argued that whether the particle is rolling or not is determined by the surface friction between the particles and the wafer, and only when the down pressure P is larger than a threshold down pressure P_{th}, pure rolling can be avoided. This leads to the following MRR formulation:

$$MRR = \begin{cases} K_e(V_r) \cdot (P^{2/3} - P_{th}^{2/3}) & P \geq P_{th} \\ 0 & P < P_{th} \end{cases} \tag{3.13}$$

Besides the above analytical models, the numerical models (by Bastawros *et al*. [56] and Seok *et al*.[57]) based on the finite element method have been developed and used to investigate the wafer –pad contact as well.

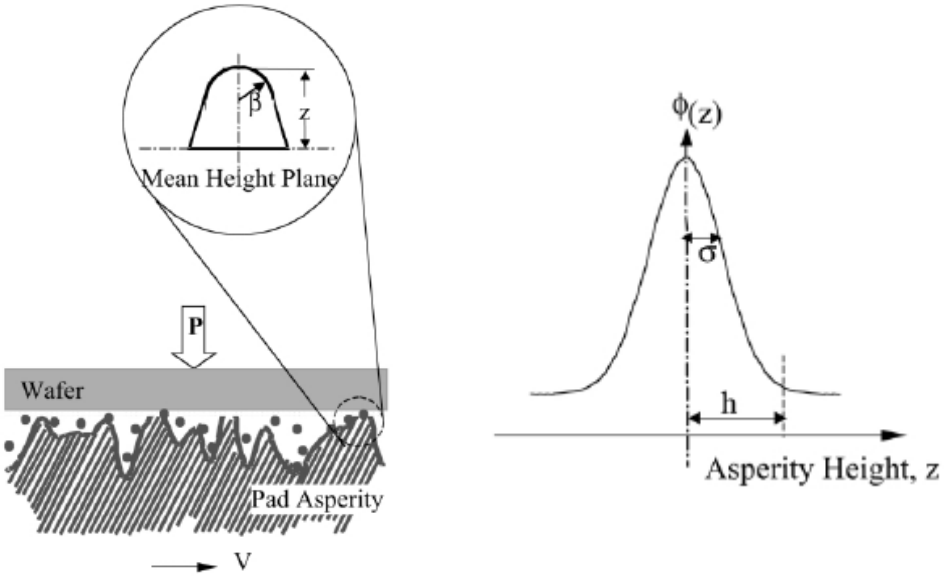


Figure 3.8 Wafer-pad contact model of Yu *et al*.[18]

3.3.3 Interactions of Slurry Chemicals and Wafer Materials

In the models developed initially, the contribution of slurry chemicals to the material removal is neglected or represented by an all-purposed coefficient. Cook [16] suggested a complete scenario of the chemical effects. He proposed that the surface removal during polishing should include the following five chemical processes: (i) slurry chemicals diffusion into the wafer surface; (ii) the subsequent wafer material dissolution under the load imposed by the abrasive particles; (iii) the absorption of the dissolution product onto the surface of the polishing grain; (iv) the re-deposition of the polishing materials back onto the wafer surface; and (v) the corrosion rate between particle impacts.

Due to complicated chemical reactions and difficulty in property's quantitative description, and a small contribution by chemical mechanism to the whole material removal processes, believed by most researchers, most early CMP models did not take this interaction into account.

In the Tungsten CMP model proposed by Kaufman *et al*. [40], they used the following formulation to describe the formation of a tungsten passivation layer in presence of ferricyanide:

$$W + 6Fe(CN)_6^3 + 3H_2O \rightarrow WO_3 + 6Fe(CN)_6^{-4} + 8H^+ \qquad (3.14)$$

They consider the pavissivation layer to be removed by the abrasive particles and fresh tungsten surfaces are exposed, which is subsequently passivated and removed. This mechanism of passivation-removal-repasivation can be used to explain aluminum, copper and other metal CMP as well. A similar mechanism of surface modification-removal-remodification is supposed to work on silicon, silicon oxide and low-k dielectric CMP. Paul [58] and Zhao *et al*. [59] have presented detailed surface kinetics models to connect the slurry chemical concentrations and fresh metal sites available to the formation rate of the surface layer.

Sundararajan *et al*. [60] proposed mass transfer model based on the lubrication theory to predict the polishing rate of copper CMP as a function of operating conditions and slurry composition. The dissolved copper species is convectively diffused and forms a concentration boundary layer in the slurry between the wafer and the polishing pad. This mass transfer model takes into account the slurry chemistry, the effect of abrasive particles and the hydrodynamics of the flow. Similarly, the model of Subramanian et al.

[61] accommodates transport phenomena coupled with chemical reactions may consider competing chemical processes, such as the formation of passivating films (Kaufman *et al*. [40]) and phenomena such as dishing and erosion. Besides, simple models of abrasion which are decoupled from the chemistry, as well as comprehensive models which couple abrasion with chemical weakening of the surface of the film, also can be accommodated into their transport model. Recently, Osseo-Asare [28] proposed a notice that considers the adsorption rate of the dissolution product onto the surface of the polishing abrasives.

In addition, Borst *et al*. [62] proposed a five-step model for CMP of SiLK dielectrics: (1) mass transport of the reactant from the slurry to the slurry/wafer interface; (2) adsorption of reactant to available surface site; (3) reaction between the absorbed reactant and specific wafer surface layer; (4) mechanical removal of altered wafer surface layer; and (5) mass transport of removed material to the bulk slurry. In their work, formulations to cover the steps (1), (2) and (3), which relate to the mass transportation, slurry chemical concentration and reaction rate to the formation of surface layer were included.

3.3.4 Interactions of Abrasive Particles and Polishing Pad

Several contact modes between the particles and pad exist [26]. The first model is that a slurry film is formed over the wafer-pad interface and therefore the particles are never embedded into the polishing pad but impact the pad body only. In this case, the pad contributes to the force through the slurry film. A detailed fluid mechanics model considering the topography and deformation of the pad is needed to evaluate the force

supporting the abrasive particles. Te models proposed by Runnel *et al*. [48] and Su [63] may be helpful in this respect.

The second possible model is that the abrasives are embedded into the polishing pad. This is the case of the "two-body" removal of materials (Figure 3.9). Cook's model [16] suggested a close packing of spherical abrasive particle into the pad. It assumes that the wafer and the pad are separated completely by the abrasive particles and no pad-wafer direct contact exists. The force applied on a single abrasive particle under these assumptions is given by

$$F = \frac{2\sqrt{3} \cdot P \cdot r^2}{k} \tag{3.15}$$

where P is the polishing pressure, r the abrasive size and k the particle fill fraction on the pad.

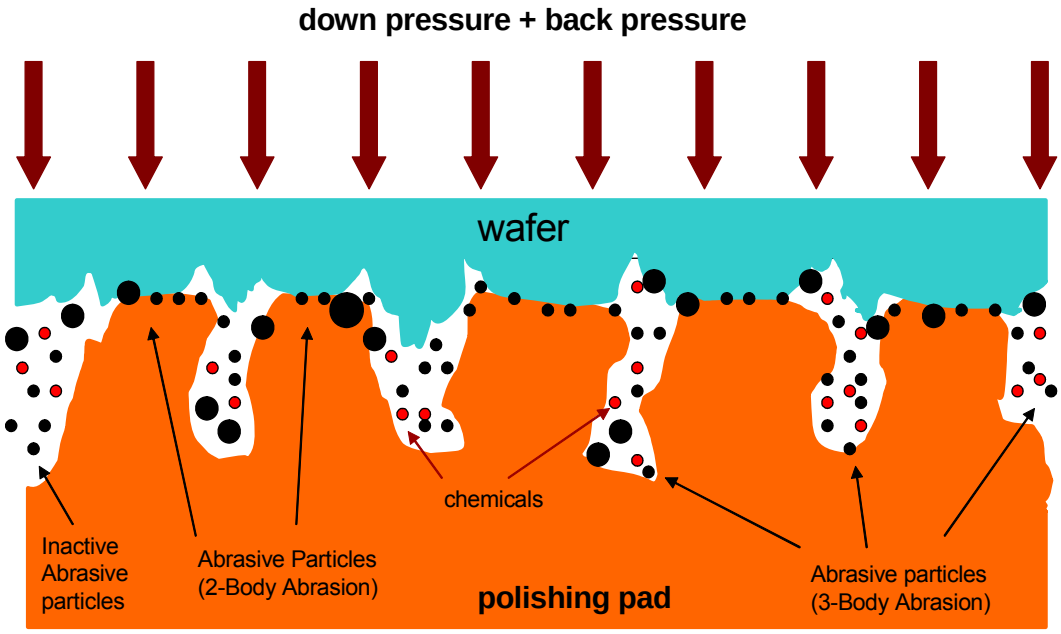


Figure 3.9 Schematic of interactions between wafer, abrasives, chemicals and pad [26, 27] (drawn by Lih)

This particle-pad interaction model has been integrated into the material removal model of Cook [16]. It is also used by Ahmadi and Xia [52] to evaluate the force on an abrasive particle, in their case a hard pad and larger concentration of abrasive particle.

Zhao and Shi [55] presented that when the pad is soft enough, the abrasive particles will be embedded into the pad deeply and the force from the wafer is supported by the pad and abrasive particles together. Luo and Dornfeld [26, 64, 65] and Fu *et al*. [29] applied the same ideas in their material removal model.

Luo and Dornfeld [26, 64, 65] suggested that this contact force as shown in Equation (3.15) is proportional to the contact pressure times the abrasive size, by assuming that the abrasive particles are closely packed to each other and these closely packed abrasive particles are enwrapped by the pads so that the effective contact area between wafer and pad is equal to that without abrasive particles. Moreover, the size of the abrasive particles that may be captured by the pad is a function of abrasive size distribution and pad properties (such as wet or dry hardness). Fu *et al*. [29] assumed that the abrasive particles are dispersed evenly over the pad surface and used a beam model to evaluate the wafer-pad direct contact between each abrasive particle. The force supported by a single abrasive particle can be obtained form the beam model and is a function of abrasive particle size, down pressure and pad material properties (Figure 3.10).

In addition to the contact forces between pad and the abrasive particles, the fractions of “two-body” abrasion, “three-body” abrasion and so on (as shown in Figure 3.9) are also important factors, which contribute different material removal mechanisms to the polishing process. Luo and Dornfeld [26, 64, 65] suggested that only a portion of

abrasive particles are involved in the material removal process, and this portion is also a function of the size of abrasive particles, pad topography and material properties.

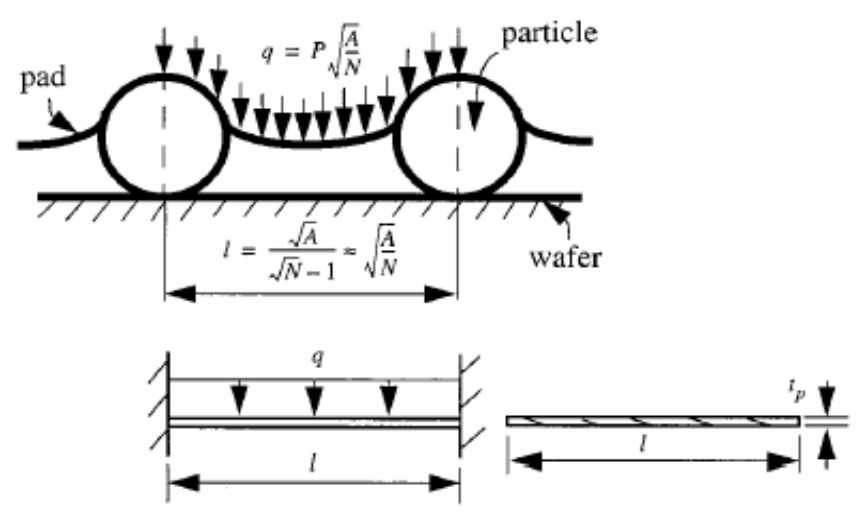


Figure 3.10 Equivalent beam model for a span of the pad pressure against two abrasive particles [26, 29]

where $\boldsymbol{P}$ is polishing pressure, $\boldsymbol{A}$ wafer surface area, $\boldsymbol{N}$ the numbers of abrasive particles on the wafer surface, $\boldsymbol{l}$ the distance between the neighboring particles, and t_p thickness of the pad.

3.3.5 Interactions between Slurry Chemicals, Abrasive Particles and Polishing Pad

The slurry chemicals affect not only the wafer but also the abrasive particles and the polishing pad, such as the alteration of the abrasive particle and pad properties (the abrasive shape, abrasive size (e.g., aggregation or agglomeration), and hardness (or Young's modulus) of the pad). Mazaheri and Ahmadi [66] proposed that the indentation of abrasives into the wafer surface is determined not only by the load from the polishing pad, but also by the double layer forces, which are a function of abrasive size and

abrasive zeta potential. They proposed experimental equations of abrasive zeta potential as a function of slurry pH values for three different abrasive materials, namely, tantalum pentoxide, alumina and slica. The zeta potential value can be substituted into the formulation of double layer forces to evaluate the material removed by a single abrasive particle by using the indentation model. Castillo-Mejia *et al*. [67] reported that a wafer can plasticize the polishing pad and reduce its elastic modulus. A formulation on the ratio of the Young's modulus of the wet pad to the dry pad is proposed. Also, this Young's modulus is then used in wafer-pad contact model to evaluate the material removal. Besides the above models, another slurry chemical and pad interaction model by treating the pad as a sort of catalyst has been proposed by Chen *et al*. [68].

3.4 Macroscopic Views of CMP Process

The macroscopic modeling of polishing effects in CMP begins with two key issues: *what are the process-related dependencies in the rate of removal of exposed surface material during polishing, and on what does the wafer-scale uniformity of that polish depend?* In this section, it will begin with the effects of pressure distribution, velocity distribution, and abrasive solid content (or abrasive particle concentration) that impact the material removal rate (MRR) and then consider the commonly observed non-uniformity, such as within wafer non-uniformity (WIWNU) in polish across the wafer.

3.4.1 Definitions of MRR and WIWNU

In the case of wafer-scale models as tabularized in Table 3.1, MRR and WIWNU are two primary CMP output performance variables of interest. The modeling effects presented in this thesis pertain to the wafer scale. The mathematical definitions of MRR and WIWNU are listed in Equations (3.16a and 3.16b). The schematic of the variations of wafer thickness before and after CMP is shown as Figure 3.11.

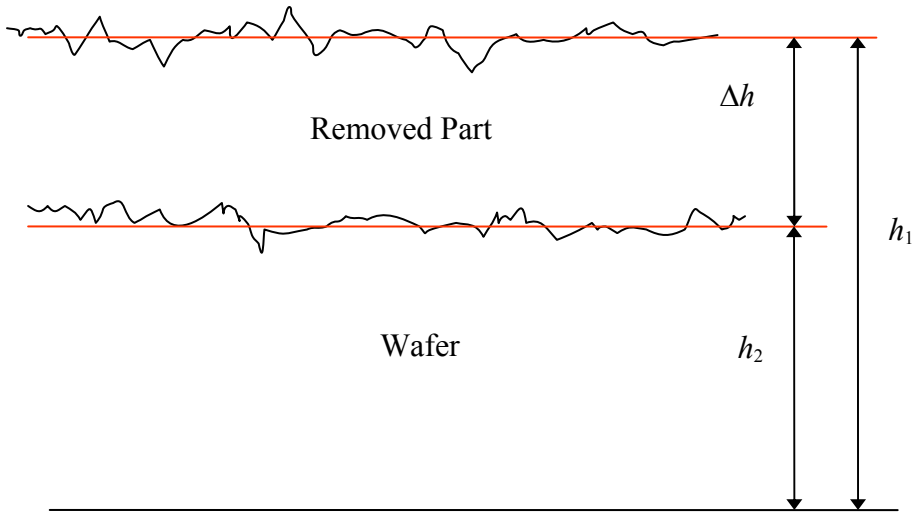
$$\text{MRR} = \frac{\boldsymbol{h}_1 - \boldsymbol{h}_2}{\Delta \boldsymbol{T}} \left(\text{Å}/\text{min}\right) \tag{3.16a}$$

$$\text{WIWNU} = \frac{\sigma_{h_2}}{\boldsymbol{h}_{2\_ave}} \times 100\% \tag{3.16b}$$

$$\Delta T = \text{Polishing time}\left(\text{min}\right), \quad \sigma_{h_2} = \text{Standard deviation of } h_2$$

where, $\boldsymbol{h}_1 - \boldsymbol{h}_2 = \Delta \boldsymbol{h} =$ thickness difference of wafer substrate before and after CMP.

At the wafer-scale, the major factors influencing the MRR and WIWNU mainly include the pressure distribution, velocity distribution, abrasive solid content in the slurry, pad surface hardness, and abrasive particle size [69]. Among these, the distributions of pressure, velocity, and abrasive solid content are considered to have significant influence on MRR and WIWNU. A concise description on the influence of each of these parameters on CMP performance is detailed in the following three subsections.



h_1 = average thickness before CMP
h_2 = average thickness after CMP

Figure 3.11 Schematic of variations of wafer thickness before and after CMP

3.4.2 Effect of Pressure Distribution

Preston's model in Equation (3.1) indicates a pressure dependency. Runnels *et al*. [70] report a model incorporating pressure dependencies to account for wafer scale non-uniformity. Wafer surface pressure distribution during CMP process depends on the applied down pressure, back pressure, and interfacial friction forces between the wafer and the polishing pad. Among them, the distribution of down pressure and back pressure across the wafer surface is highly dependent on the wafer carrier design (Figure 3.12). Significant innovation in carrier design to achieve either uniform or controllable pressure

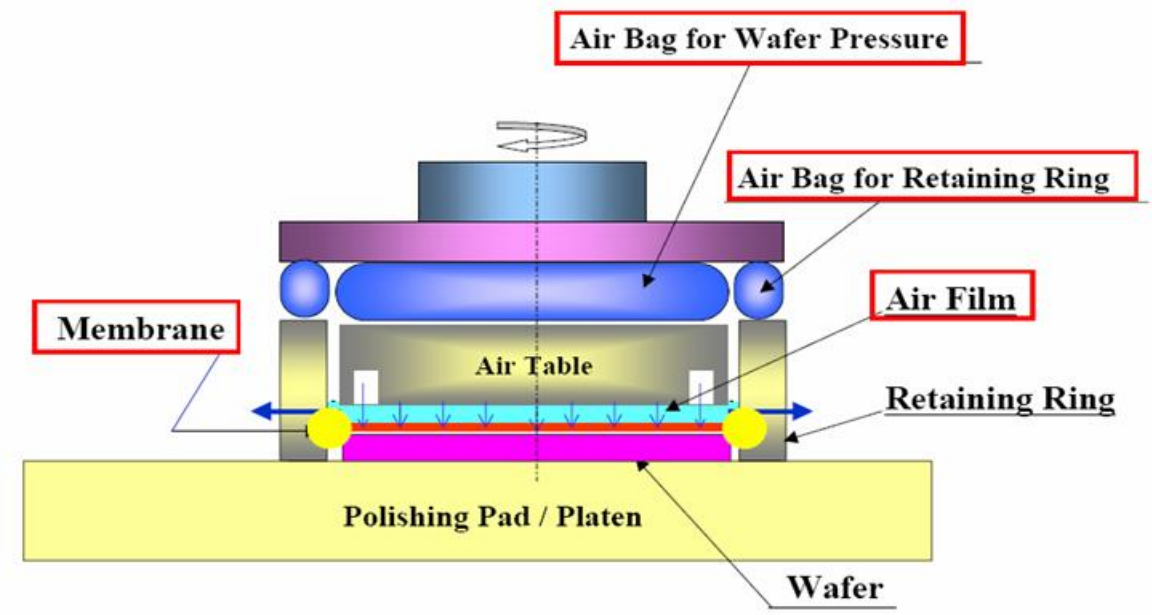


Figure 3.12 Example of Novel Design of Wafer Carrier on CMP Polisher [13]

distributions is an important area of development.

Except for the wafer carrier, the other related issues, such as wafer thickness, wafer warp and bow, thickness of slurry film across the wafer surface, uniformity of stress in such thin film across the wafer, have to do with the initial wafer-level uniformity. Zhang *et al*. [71] present the initial warpage (or warp and bow) can have significant impact on the polish performance. Certainly, they will influence the pressure distributions and dynamic balance of all active applied forces, which subsequently impact on the wafer-scale polish performance.

The interfacial contact conditions are characterized as direct-contact, semi-contact, and hydroplaning modes. The friction coefficient values tend to have different order ranges [14] depending on the contact mode as illustrated in Figure 3.13. The edge effects in direct-contact mode [72] and negative fluid pressure in semi-contact mode [73] cause

variations in friction forces across the wafer surface. Also, Wang *et al*. [72] have considered inherent variation due to the Von Mises stress concentrations at the edge of the wafer (conceptually, a downward pressure on the wafer causes lateral stress buildup near the edge of the wafer). Consequently, the average normal pressure profile is changed.

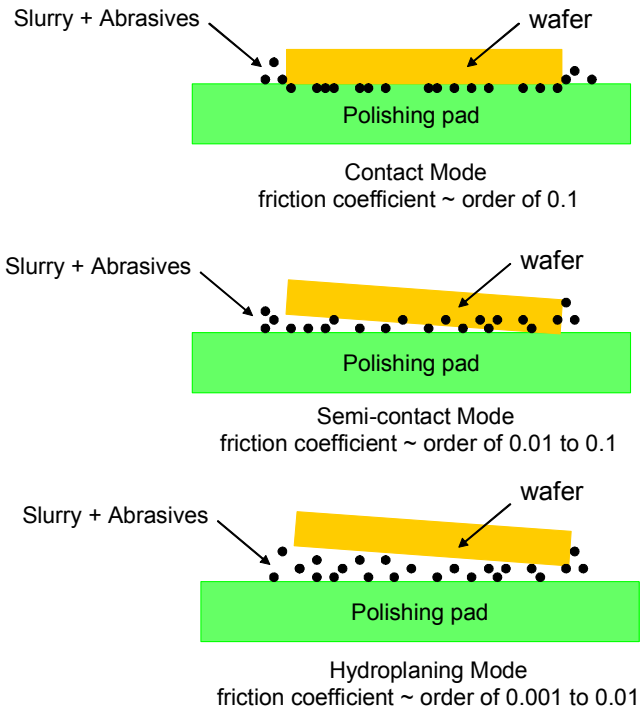


Figure 3.13 Wafer-Pad Contact Modes [14] (drawn by Lih)

In addition, the consideration and modeling of local pressure differences near the edge of the wafer has also been reported. These approaches examine discontinuities in the pad compression due to the edge of the wafer (either statically or dynamically including leading and trailing edge issues) to understand such effects as slow or fast polish in the several millimeters long "edge exclusion" at the wafer edge [74]. These explorations have contributed to the design of "active retaining ring" heads, where the wafer carrier

has a separately pressurized or controlled retaining ring to pre-compress the polish pad and enable uniform pressure distribution further out on the edge of the wafer. Moreover, the use of suitable back pressure can also help alleviate the pressure distribution across wafer surface, but do not completely eliminate the differences.

3.4.3 Effect of Velocity Distribution

As stated in Section 3.3, the Preston's relationship provides a "pointwise" dependency of MRR based on the relative velocity and applied pressure within some region of the wafer. A straightforward application of Preston's model as shown in Equation (2.2) is to study the wafer-level non-uniformity due to dependencies in the relative velocity (V_r) arising from machine configuration. Indeed, Preston's original work [42] contains an analysis of the relative velocity arising from rotary or platen based polishing apparatus. Figure 3.14 shows a typical rotary polishing machine configuration

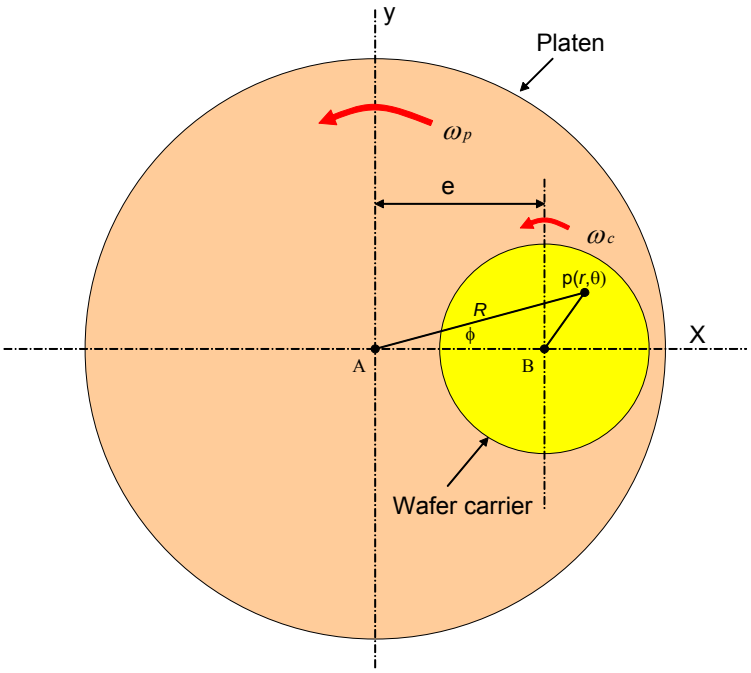


Figure 3.14 Schematic of platen and wafer carrier rotational motion [9] (drawn by Lih)

commonly used for CMP.

In above figure, **A** is the center of the polish platen and **B** is the axis of rotation of the wafer carrier. The circle of radius ***r*** centered at **B** is a ring of points on the wafer. The wafer carrier usually oscillates along **AB** to ensure even polishing pad degradation and improved uniformity. The relative velocity $\vec{V}_p$ at point ***p*** on the wafer can be determined as Equation (3.17).

$$
\begin{aligned}
&\vec{V}_p = \vec{V}_{pA} = \vec{V}_{wafer} + \vec{V}_{oscillation} - \vec{V}_{platen} \\
&\because \vec{V}_{wafer} = \vec{\omega}_c \times \vec{r} \\
&\qquad = \omega_c \cdot \hat{k} \times \left[r\cos\theta \cdot \hat{i} + r\sin\theta \cdot \hat{j} \right] = \hat{i} \cdot \left(-r\omega_c \sin\theta \right) + \hat{j} \cdot \left(r\omega_c \cos\theta \right) \\
&\because \vec{V}_{oscillation} = V_{oscillation} \cdot \hat{i} \\
&\because \vec{V}_{platen} = \vec{\omega}_p \times \vec{R} = \omega_p \cdot \hat{k} \times \left[\left(e + r\cos\theta \right) \cdot \hat{i} + r\sin\theta \cdot \hat{j} \right] \\
&\qquad = \hat{i} \cdot \left(-r\omega_p \sin\theta \right) + \hat{j} \cdot \left(e\omega_p + r\omega_p \cos\theta \right) \\
&\Rightarrow \vec{V}_p = \hat{i} \cdot \left(-r\omega_c \sin\theta + V_{oscillation} + r\omega_p \sin\theta \right) + \hat{j} \cdot \left(r\omega_c \cos\theta - e\omega_p - r\omega_p \cos\theta \right) \\
&\Rightarrow V_p = \left[\left(r\sin\theta \cdot \left(\omega_p - \omega_c \right) + V_{oscillation} \right)^2 + \left(r\cos\theta \cdot \left(\omega_c - \omega_p \right) - e\omega_p \right)^2 \right]^{1/2}
\end{aligned}
\tag{3.17}
$$

If the oscillation velocity ($V_{oscillation}$) is simply set to be equal to zero, the relative velocity at point p can be described as Equation (3.18)

$$
\begin{aligned}
V_p &= \left[\left(r\sin\theta \cdot \left(\omega_p - \omega_c \right) \right)^2 + \left(r\cos\theta \cdot \left(\omega_c - \omega_p \right) - e\omega_p \right)^2 \right]^{1/2} \\
&= e\omega_p \cdot \left[1 + \left(\frac{r}{e} \right)^2 \cdot \left(\frac{\omega_p - \omega_c}{\omega_p} \right)^2 + 2 \cdot \left(\frac{r}{e} \right) \cdot \left(\frac{\omega_p - \omega_c}{\omega_p} \right) \cdot \cos\theta \right]^{1/2} \\
&= v_s \cdot \left(1 + x^2 + 2x\cos\theta \right)^{1/2}
\end{aligned}
\tag{3.18}
$$

here, $v_s = e\omega_p, \quad x = \left(\frac{r}{e} \right) \cdot \left(\frac{\omega_p - \omega_c}{\omega_p} \right)$

Given V_p at any point on the ring with radius r, the effective (time-averaged) relative velocity at any point at a distance r from the center B can be determined as Equation (3.19).

$$V_{effective}(r) = \frac{v_s}{\pi} \cdot \int_0^{\pi} \left(1 + x^2 + 2x\cos\theta\right)^{1/2} d\theta \tag{3.19}$$

In conclusion, if the rotational speeds of the wafer carrier and platen are synchronized, the relative speed is constant across the entire wafer surface. On the other hand, when the speeds of the platen and wafer carrier are mismatched, the center of the always experiences a reduced velocity compared to the outer edges of the wafer, as shown in Figure 3.15.

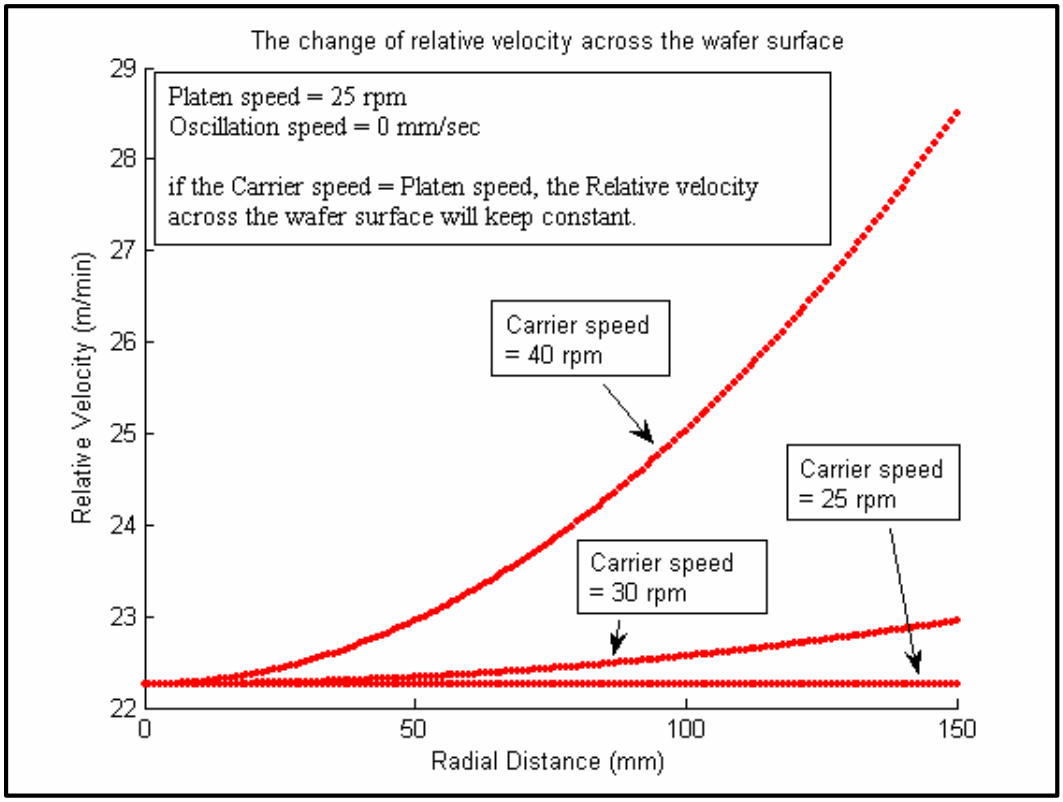


Figure 3.15 Relative Velocity Dependence on Wafer Carrier and Platen Speeds

The platen speed ω_p is close to the wafer carrier speed ω_c, the velocity distribution is more uniform [75], resulting in evenly distributed abrasive particles over the contact interface between the wafer and polishing pad, and more uniform MRR across the wafer surface, i.e., more planar wafer surface or lower WIWNU. In fact, there is, however, always a small offset maintained between the rotational speeds of the platen and the wafer carrier to avoid synchronized polishing. In addition, to accommodate incoming profile variations across the wafer surface and to increase the MRR, the speeds of the platen and the wafer carrier are varied significantly. In a typical CMP process, the carrier is also swept along the surface of the polishing pad to enable effective slurry transport underneath the wafer, removal of debris generated during polishing, and to expose the wafer to different areas along the pad radius [76].

3.4.4 Effect of Abrasive Particle Concentration

The MRR in solid contact mode usually increases linearly with the abrasive solid content [77]. However, this linear increase only holds for a limited range of abrasive weight concentrations as shown in Figure 3.16. When there are no or a few abrasives in the slurry, the MRR is close to zero and dominate by isotropic wet etching and dissolution. Therefore, there is no planarization that can be realized in this range. A small increase in abrasive content in this region usually leads to a rapid increase in MRR. Conversely, when the abrasive solid content is larger than a certain value, the MRR will reach a constant value [78]. This phenomenon of material removal saturation is shown schematically in Figure 3.16 where the total contact area between the wafer and the pad

surface is fully occupied by the active abrasives. A further increase in abrasive solid content cannot increase the number of abrasives in the contact area [26, 64]. Figure 3.16 shows three regions of MRR with increase in abrasive solid content.

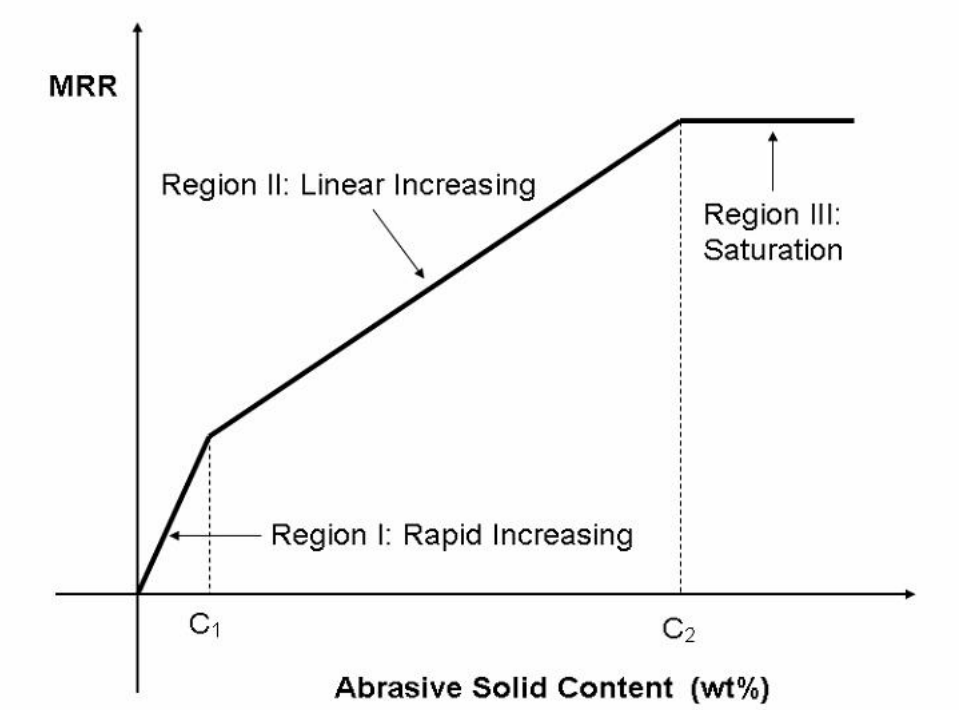


Figure 3.16 Three regimes of MRR variation relative to abrasive solid content [27, 28] (drawn by Lih)

In addition to these, other parameters, such as the polishing time, wafer, and abrasive properties also have a significant influence on MRR and WIWNU. Not only are the relationships nonlinear, but also the interactive effects of a combination of these parameters are significant. Apparently, these problems increase a great amount of hindrance to construct accurate process models for CMP.

3.5 Non-Analytical Modeling for CMP Process

Wang *et al*. [79-81] proposed a *Neural-Taguchi* method combining neural networks Taguchi array for experimental designs to seek the best parameter set for maximum MRR and minimum WIWNU. Five input parameters (i.e., solid content or abrasive concentration, down pressure, back pressure, platen speed, and polishing time) and two out performances (i.e., MRR and WIWNU) were chosen to explore the unknown dynamic relationships between inputs and outputs. In addition, the ANOVA was applied to estimate contributing degree of each input parameter to two outputs. Signal/Noise (S/N) ratio analysis was used to indicate the quality of output. CMP experiments on the basis of L_{25} Taguchi array was conducted, and 25 input-output pairs of experiment data were generated.

Furthermore, neural network models were constructed using the above 25 data sets for modeling MRR and WIWNU. Next, a hybrid performance index as shown in Equation (3.20) was applied to find out the optimal process input setting.

$$\boldsymbol{J = a \times MRR\,(S/N) + b \times WIWNU(S/N)} \tag{3.20}$$

Non-analytical neural network models were expected to obtain accurate simulation results of MRR and WIWNU. Additionally, two process outputs were combined together in Equation (3.20) to form a single-objective optimization problem (SOOP).

3.6 Predicaments of Analytical Modeling for CMP Process

With unceasingly increase of the customers' demands form the global markets and the relentless competitions from the business competitors, the higher product quality and

the higher production rate become the most crucial survival requirements to the manufacturing industry. In particular, it is more severe in global semiconductor manufacturing. Form Chapter II, the CMP process combining with the ULSI has become a core technology to the fabrication of current semiconductor wafers. Meanwhile, new materials for next-generation deep sub-micron processes, such as copper, low-k dielectrics, are rapidly replacing with old materials. Comparing with so quick evolutions in semiconductor industry, the development of analytical models for CMP process is still trapped in a number of dilemmas.

Modeling and simulation are critical to transfer CMP from an engineering 'art' to an engineering 'science'. Numerous research efforts in CMP modeling have been made in the last decade (also as shown in Figure 3.17) but the available analytical MRR and surface non-uniformity models as reviewed in the previous Sections cannot precisely provide accurate prediction or estimation during the polishing process. Novel modeling ideas and methods are urgently needed. Further, the new modeling tools should be able to fully combine with the current advanced monitoring and control systems. Through continuously online learning, new CMP models should be rapidly self-adjusted for making the best decisions to increase production rate and improve product quality.

Obviously, current analytical CMP models cannot achieve above goals. Hence, the use of Neural Networks (NN) [79-81], ANFIS [82] and other Evolutionary Algorithms (EA) become deserved ways for precisely capturing the highly dynamic complexity of CMP processes.

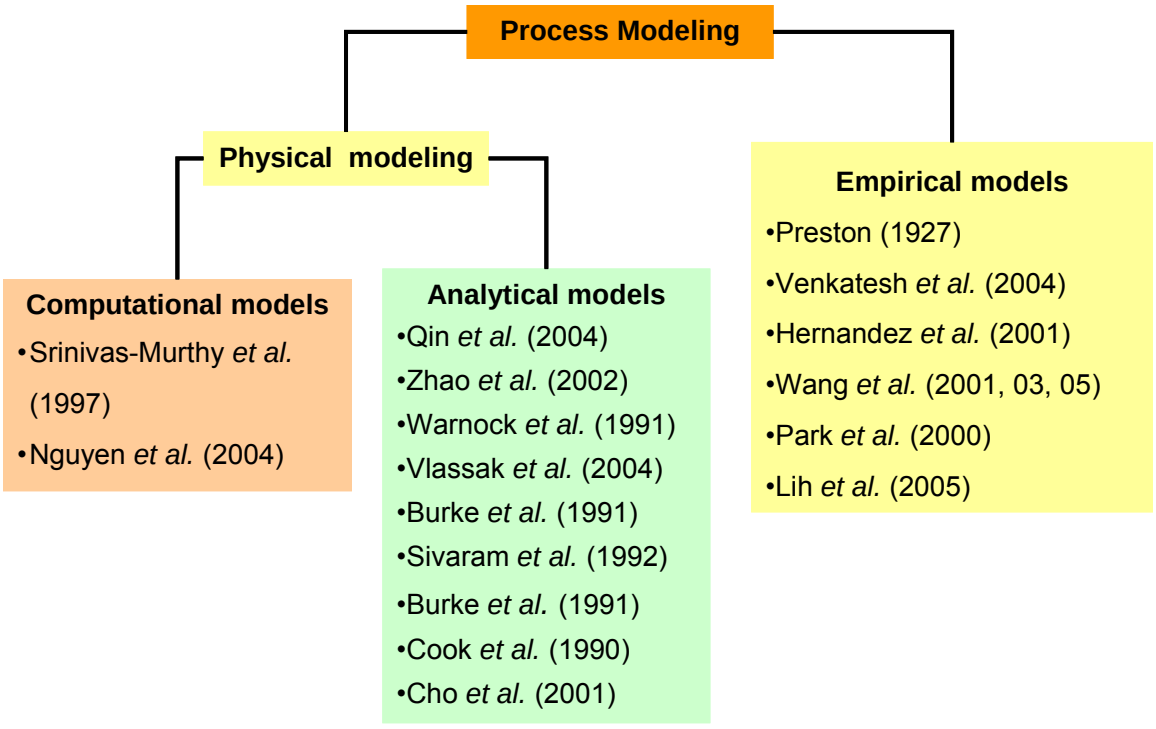


Figure 3.17 Other Relevant Literature in CMP

3.7 Our Approaches to CMP Process Modeling

In this investigation, more accurate process models and optimization of CMP process were emphasized. Well-trained neural networks can accurately capture the highly nonlinear dynamic relationships between input and output spaces of unknown systems. Several MRR and WIWNU models constructed using feedforward neural networks were proposed. ANFIS (Adaptive-based-Network Fuzzy Inference Systems) provides another option for modeling CMP process. Also, given suitable membership function and fuzzy rules, ANFIS can vividly emulate complex behaviors between inputs and outputs. Moreover, real-value Genetic Algorithms (GA) from the Evolutionary Algorithms (EA) was applied to entirely search unknown relationships between inputs and outputs. Lastly,

the concepts from Multi-objective Evolutionary Algorithms (MOEA) combined most accurate models of MRR and WIWNU constructed by the above-mentioned methods were used to optimizing the CMP process for obtaining optimal input settings to arrive at higher MRR and lower WIWNU.

CHAPTER IV

NEURAL NETWORKS, ANFIS, AND EVOLUTIONARY ALGORITHMS

4.1 Introduction

In 1959, the Rockefeller Foundation sponsored a conference at Dartmouth College that had as its scope – The potential use of computers and simulation in every aspect of learning and any other feature of intelligence. After that conference, the term "artificial intelligence" came into common use.

The term "artificial intelligence" (AI), in its broadest sense, encompasses a number of technologies that includes, but is not limited to, expert systems (ES), neural networks (NN), evolutionary algorithms (EA), fuzzy inference systems (FIS), cellular automata, chaotic systems, and anticipatory systems. Most of these technologies have their origins in biological or behavioral phenomena related to humans or animals, and many of these technologies are simple analogs of human and animal systems. Hybrid intelligent systems generally involve two, three or more of these individual AI technologies that are either used in series or integrated in a way to produce advantageous results through synergetic interactions. In this thesis the emphasis is placed on using neural networks and fuzzy inference systems to build process models.

In data or information processing, the objective is generally to gain an understanding of the phenomena involved and to evaluate relevant parameters

quantitatively. This is usually accomplished through "modeling" of the systems, either experimentally or analytically (using mathematical and physical principles). As stated in Chapter II, all CMP models are developed to date analytical. Practically, most hybrid systems relate experimental data to systems or models. Once the model of a system is constructed, there are various procedures (e.g., statistical regression, sensitivity analysis) that can be carried out to gain a better understanding of the system. Such experimentally derived models give insight into the nature of the system behavior that can be used to enhance mathematical and physical models.

There are, however, many situations in which the phenomena involved are very complex, such as the CMP process, and often not well understood and for which first principle models are not possible. Even more often, physical measurements (e.g., endpoint detection in the CMP process) of the pertinent quantities are very difficult and expensive. These difficulties lead us to explore the use of neural network and fuzzy inference systems as a way of obtaining models based on experimental measurements.

Multi-objective evolutionary algorithms are used to search optimal process parameter sets, and provide useful guidance to redesign the relative experiments for re-training the models constructed by using neural networks and fuzzy inference systems.

Neural networks and fuzzy inference systems represent two distinct methodologies that deal with uncertainty. Uncertainties that are important include both those in the model or description of the systems involved as well as those in the variables. These uncertainties usually arise from system complexity (often including nonlinearities). Neural networks approach the modeling representation by using precise inputs and

outputs which are used to "train" a generic model which has sufficient degrees of freedom to formulate a good approximation of the complex relationship between the inputs and the outputs.

In fuzzy systems, the reverse situation prevails. The input and output variables are encoded in "fuzzy" representations, while their interrelationships take the form of well-defined IF-THEN rules. Zadeh [83] observed that the uncritical pursuit of precision may not only be unnecessary but actually a source of error that led him to the notion of a fuzzy set.

4.2 Fundamentals of Neural Networks

4.2.1 Introduction

Research in neural networks has experienced three consecutive cycles of enthusiasm and skepticism. The first peak, dating back to the 1940's is due to McCullough and Pitt's pioneering work [84]. The second period of intense activity occurred in the 1960's which featured Rosenblatt's *perceptron convergence theorem* [85]. Because Minsky and Papert [86] pointed out that perceptrons were limited to learning in linearly separable classes, the interests in neural networks have been diminished for almost 20 years. Since the early 1980's, neural networks have received considerable renewed interest. The major development behind this resurgence include Hopfield's *energy approach* [87] in 1982, and the *back-propagation learning algorithm* for multilayer perceptrons (multilayer feed-forward networks) which was first proposed by Werbos [88], reinvented several times, and popularized by Rumelhart *et al*. [89]. He

discovered a method of enabling a network to learn to discriminate between classes of patterns that are not linearly separable. They called the method "backward propagation of error" (currently, called back-propagation learning algorithm), which is based on the *gradient descent method*. Neural networks have the learning capability of adapting its parameters based on the desired input-output pairs.

4.2.2 Biological and Artificial Neurons

A *biological neuron* (Figure 4.1) is the fundamental cellular unit of the brain's nervous system. It is a simple processing element that receives and combines signals from other neurons through input paths called *dendrites*. If the combined input signal is

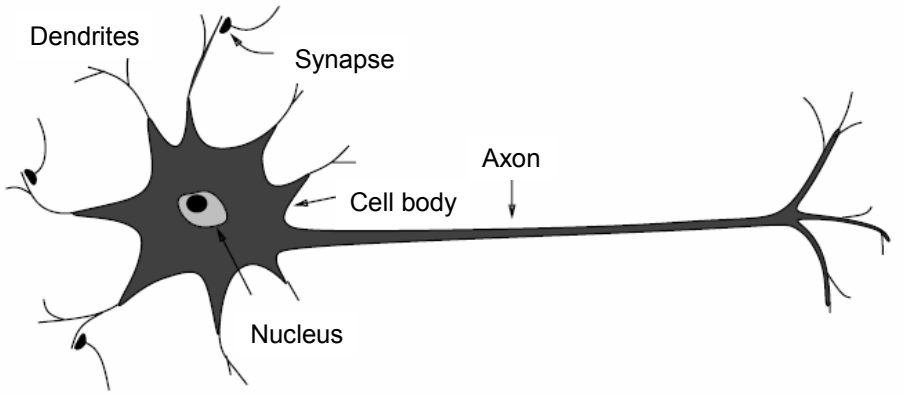


Figure 4.1 Sketch of a Biological Neuron [4]

strong enough, the neuron "fires", producing an output signal along the axon that connects to the dendrites of many other neurons. Biologically, each signal coming into a neuron along a dendrite passes through a *synapse* or *synaptic junction*. This junction is an infinitesimal gap in the dendrite that is filled with neurotransmitter fluid that either accelerates or retards the flow of electrical charges. The fundamental actions of the

neuron are chemical in nature, and this neurotransmitter fluid produces electrical signals that go to the nucleus or *soma* of the neuron. The adjustment of the impedance or conductance of the synaptic gap is a critically important process. Indeed, these adjustments lead to memory and learning. As the synaptic strengths of the neurons are adjusted, the brain "learns" and stores information.

An *artificial neuron* is a model whose components have direct analogs to the components of an actual neuron. Figure 4.2 shows the schematic representation of an artificial neuron. The input signals are represented by $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$. These signals are continuous variables, not the discrete electrical pulses that occur in the brain. Each of

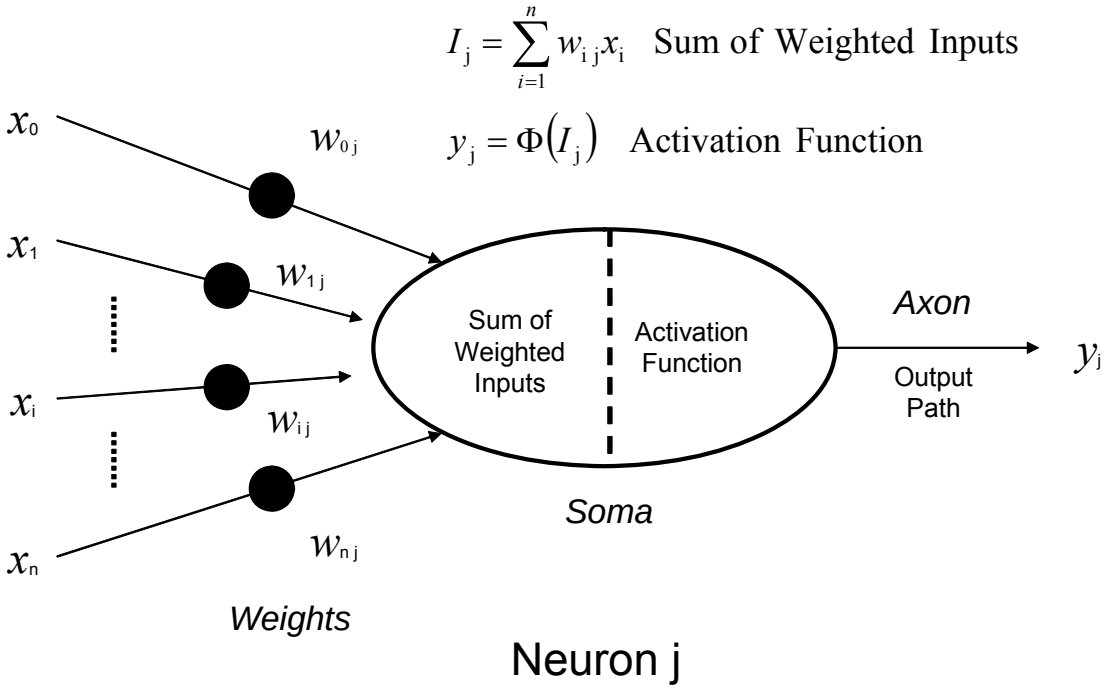


Figure 4.2 Schematic Representation of an Artificial Neuron [10]

these inputs is modified by a *weight* (also called the *synaptic weight*) whose function is analogous to that of the synaptic junction in a biological neuron. These weights can be

either positive or negative. Each processing element consists of two parts. The first part simply aggregates (sums) the weighted inputs resulting in a quantity I; the second part is effectively a nonlinear filter, usually called the *activation function* (or *transfer function*), through which the combined signal flows.

An obvious generalization in an artificial neuron is to use activation function other than the threshold function, e.g., a piecewise linear, sigmoid, or Gaussian, shown in Figure 4.3. Taking the sigmoid function for example, it is by far the most frequently used function in neural networks. Also, this function is strictly an increasing function that exhibits smoothness and has the desired asymptotic properties. The standard sigmoid function is the logistic function, defined by

$$\boldsymbol{g}(\boldsymbol{x}) = \left(1 + \exp\left(-\beta \cdot \boldsymbol{x}\right)\right)^{-1} \tag{4.1}$$

where β is the slope parameter.

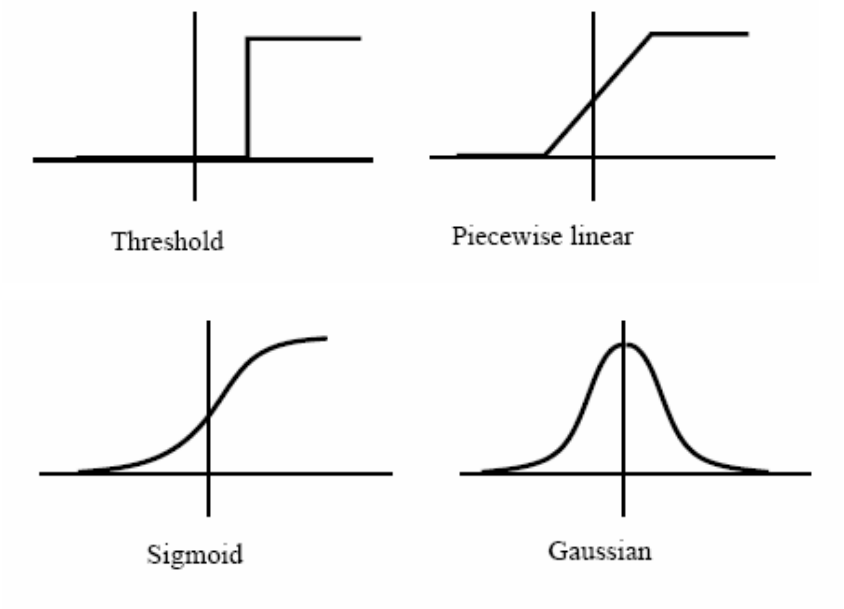


Figure 4.3 Examples of different types of Activation Functions [10]

4.2.3 Architectures of Artificial Neural Networks

An assembly of artificial neurons is called an artificial neural network (in short, called neural network in this research) which is thought by many to loosely model the biological nervous system. Like our brains, neural networks have the following attributes [90]:

- Made up of highly interconnected nodes (also neurons or units) that are capable of high speed parallel processing.
- Robust and fault tolerant.
- Flexible and can adjust to new environments through learning.
- Can deal with information that is fuzzy, probabilistic, noisy or inconsistent.
- Store knowledge or information in its connections.

For these reasons the field of neural networks has also been given such names as *Connectionism*, *Parallel Distributed Processing* (PDP), and *Computational Neurobiology*. Although inspired by the neuroscience, neural network are in reality an alternative computational paradigm that operates according to some relatively simple mathematical constructs.

Neural networks can be viewed as weighted directed graphs in which nodes are artificial neurons and directed edges (with weights) are connections from the outputs of some neurons to the inputs of the other neurons. On the basis of connection pattern (or architecture), neural networks can be grouped into two major categories [10] as shown in Figure 4.4: (1) *feedforward networks* in which no loop exists in the network architecture, and (2) *feedback* (or *recurrent*) *networks* in which loops exist because of feedback

connections. The most common family of feedforward networks is a layered network in which neurons are organized into layers with connections strictly in one direction from one layer to another. Figure 4.4 also shows typical networks of each category.

Different connectivities yield different network behaviors. Generally speaking, feedforward networks are static networks, i.e., given an input, they produce only one set of output values, not a sequence of values. Feedforward networks are memoryless in the sense that the response of a feedforward network to an input is independent of the previous state of the network. Feedback or recurrent networks are dynamic systems. Upon presenting a new input pattern, the outputs of the neurons are computed. Because of the feedback paths, the inputs to each neuron are then modified, which leads the network to enter a new state.

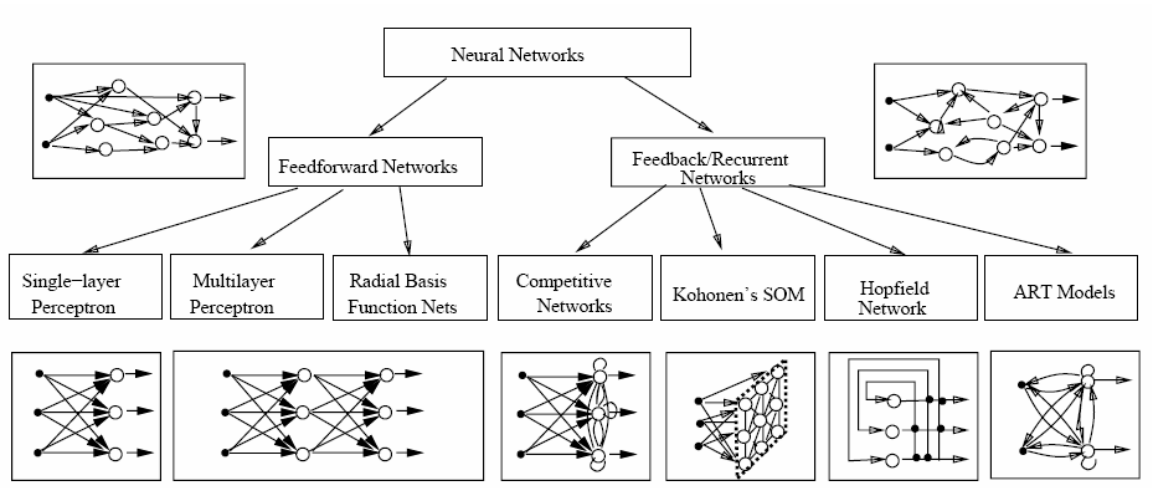


Figure 4.4 Taxonomy of network architectures [17]

A neural network can be defined as – a data processing system consisting of a large number of simple, highly interconnected processing elements (i.e., artificial neurons) in

an architecture inspired by the structure of the cerebral cortex of the brain. These processing elements are usually organized into a sequence of layers or slabs with full or random connections between the layers as shown in Figure 4.4.

The neural network models for modeling highly complicated CMP processes undertaken by this research effort are restricted to neural networks that are used for prediction purpose only. The neural network paradigm chosen for this research is the *feedforward multilayer backpropagation trained artificial neural network*. The essential ingredients to understanding this paradigm are summarized in Subsection 4.2.5 and Section 5.2.

4.2.4 Applications of Neural Networks [4, 10]

Neural networks, inspired by biological nervous systems, are composed of many simple elements, with adjustable weights, connecting and operating in parallel. By adjusting the values of weights and connections between elements, neural networks can represent or model complex nonlinear relationships or specific functions, and they are very powerful at classification of phenomena into pre-selected categories used in the training process. In brief, there exist many important applications of neural network to the following classes of challenging problems of interest to industry and academia.

- **Pattern classification:** This task is to assign an input pattern (e.g. speech waveform or handwritten symbol) represented by a feature vector to one of pre-specified classes (Figure 4.5). Well-known applications of pattern classification are character

recognition, speech recognition, ECG, EEG waveform classification, blood cell classification, and printed circuit board inspection.

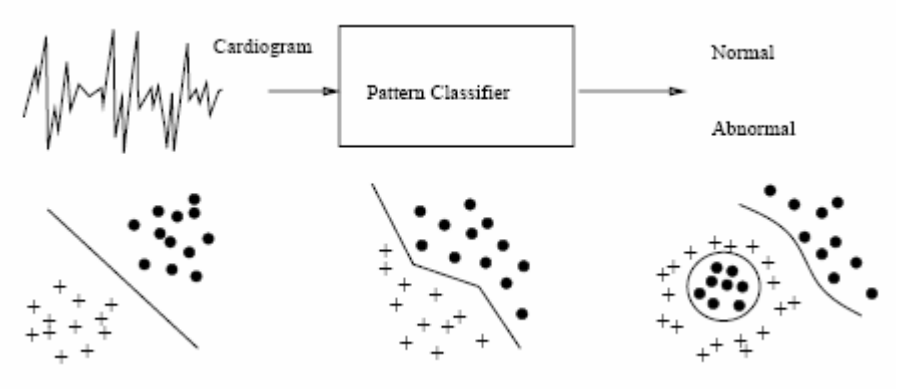


Figure 4.5 Pattern Classification [17]

- **Clustering / categorization:** Also known as unsupervised pattern classification. There are no training data with known class labels. A clustering algorithm explores the similarity between the patterns and places similar patterns in a cluster (Figure 4.6). Well-known clustering applications include data mining, data compression, and exploratory data analysis.

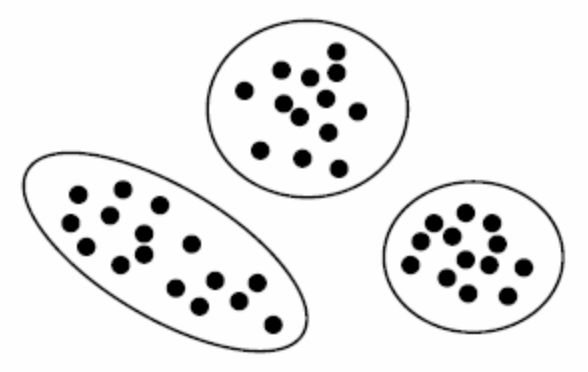


Figure 4.6 Clustering and Categorization [17]

- **Function approximation:** Given a set of training patterns (input-output pairs), this task is to find an estimate outputs after training this neural network (Figure 4.7). Various engineering and scientific modeling problems require function approximation.

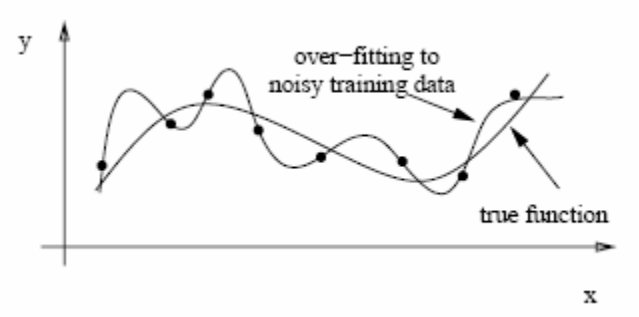


Figure 4.7 Function Approximation [17]

- **Prediction / forecasting:** Given a set of samples in a time sequence, this task is to predict the sample at the future time interval (Figure 4.8). Prediction/forecasting has a significant impact on decision-making in business, science, and engineering. Stock market prediction and weather forecasting are typical application of prediction /forecasting techniques.

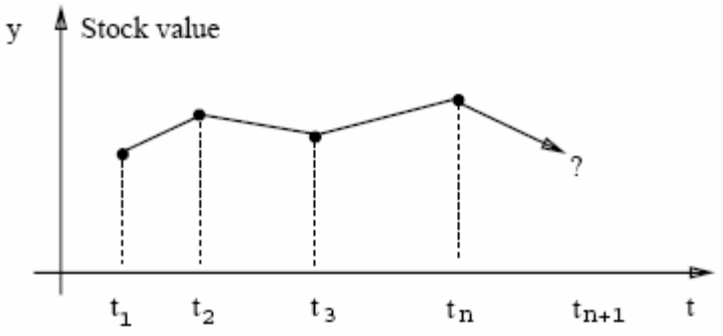


Figure 4.8 Prediction and forecasting [17]

- **Optimization:** The goal of an optimization algorithm is to find a solution satisfying a set of constraints such that an objective function is maximized or minimized. A classical optimization problem is the Traveling Salesman Problem (TSP) as shown in Figure 4.9.

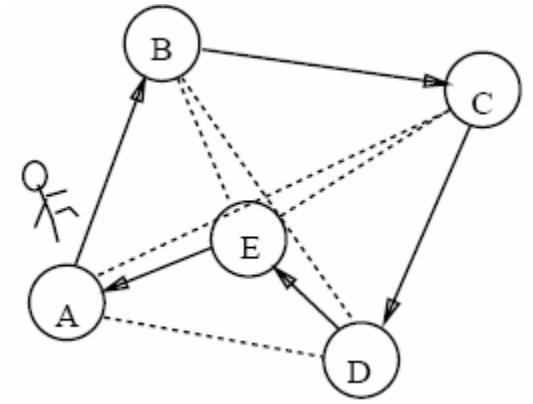


Figure 4.9 Optimization [17]

- **Content-addressable memory:** The content in the memory can be recalled by a partial input or distorted content (Figure 4.10)

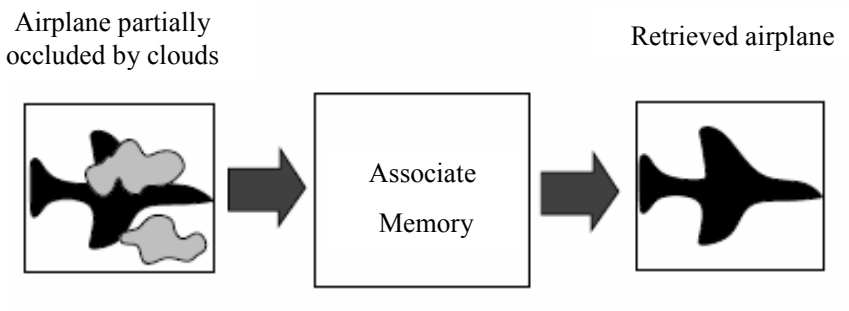


Figure 4.10 Content-addressable Memory [17]

- **Control:** in model-reference adaptive control, the goal is to generate a control inputs such that the system follows a desired trajectory determined by the reference model. An example of model reference adaptive control is the engine idling speed control (Figure 4.11).

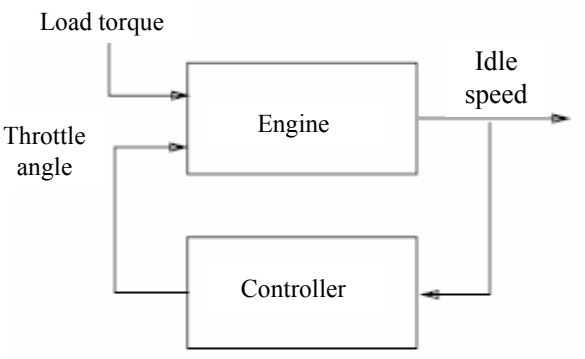


Figure 4.11 Control System [17]

Even a large number of approaches have been proposed for solving the problems described above, the field of neural networks has provided alternative approaches for solving these problems.

As stated above, neural networks have been successfully applied to a variety of function approximation or classification problems (quality inspection, weather forecasting, speech/image understanding, automatic control and robot systems, character recognition, sonar classification, etc.). Problems that are characteristically sparse and noisy with highly complex nonlinear relationship are good candidates for neural network applications.

Multilayer feedforward networks have been shown to belong to the class of nonparametric estimators of unknown mappings; that is, multilayer feedforward networks

provide a particular form of function f; where f describes the relationship between x and y, $f: x \to y$, so that y may be determined for any given value of x. Because they are nonparametric, they do not require distributional or model assumptions.

4.2.5 Multilayer Feedforward Neural Networks

Since Rosebblatt [85] first introduced single layer perceptrons, many neural network models have been proposed and investigated. These models can be classified according to various criteria, such as *supervised* or *unsupervised* learning algorithms, binary or continuous node values, feedforward or recurrent network architectures, adjustable or fixed parameters, and uniform or hybrid node functions.

A single neuron (or node), or a single layer of neurons can only solve problems that have linear solutions. But, if the problems are limited and more complex, they will require nonlinear partitioning. As stated in Section 4.2.2, a typical neural network consists of three or more layers of neurons. The first or input layer receives external signals or information from the environment. The output layer transmits to environment, the network's response to the input. Between the input and output layers are one or more hidden layers.

In this research, the focus is on modeling problems associated with desired input-output data sets; so the relevant neural networks should have *adjustable parameters* that are updated by a supervised learning rule. Since the primary goal of these networks is to achieve a desired input-output mapping, they are often called *mapping networks*.

Weights (or adjustable parameters) connect neurons to neurons. For this class of networks, nodes are connected between layers and not within layers. Learning or training occurs through iterative adjustments of the weights, where the network evolves from an initially random state to equilibrium. Supervised learning requires a teacher, i.e., a known input-output pair. During training, the network's output is compared to the target or desired output. The weights are adjusted to reduce the error or difference between the network's output and target. The goal of all learning procedures is to ultimately minimize the error. Once trained, the input to the network is fed-forward through its neurons and weights to produce an output. Cybenko [91], Funahashi [92] and Hornik *et al*. [93] have shown that the *multilayer feedforward sigmoidal logistic architecture* can approximate any continuous function, given a sufficient number of hidden neurons.

Figure 4.12 shows a typical four-layer feedforward networks whose layers are successively connected in a feedforward fashion without connections between neurons in the same layer and without feedback connections between layers. For simplicity, this network will be referred to as a [5-6-7-2] structure according to the number of neurons in each layer, i.e., five neurons in the first input layer, six neurons in the first hidden layer, seven neurons in the second hidden layer, and two neurons in the last output layer.

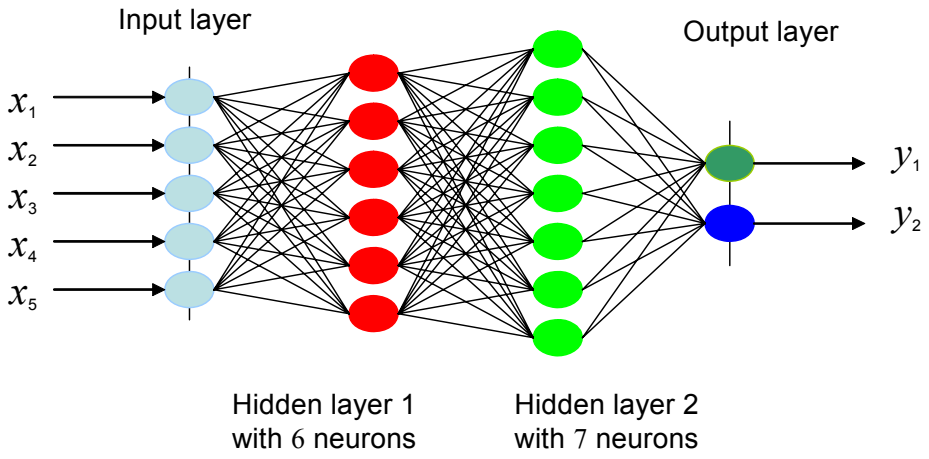


Figure 4.12 Multilayer feedforward network of [5-6-7-2] structure [10]

The most popular class of multilayer feedforward networks is *multilayer perceptrons* in which each computational unit employs either the threshold function or the sigmoid function. Multilayer perceptrons are capable of forming arbitrarily complex decision boundaries and can represent any Boolean function [86].

4.2.6 Learning Algorithms

Fundamental characters and important applications of neural networks are mentioned in prevision sections. They will be, however, useless before neural networks are effectively and sufficiently trained. A learning process in the context of neural networks can be viewed as the problem of updating network architecture and connection weights so that a network can efficiently perform a specific task. Most of the time, the network must learn the connection weights from the available training patterns (or data sets). Improvement in performance is achieved over time through iteratively updating the weights in the network. Instead of having to specify a set of rules such as fuzzy systems,

neural networks appear to learn from the given collection of representative examples. This is one of the major advantages of neural networks over traditional expert systems.

In order to understand or design a learning process, one must first have a model of the environment in which a neural network operates, i.e., what information is available to the neural network. Second, one must understand how weights in the network are updated, i.e., what are the learning rules which govern the updating process. There are three main learning paradigms, namely, (1) supervised, (2) unsupervised, and (3) hybrid leaning.

In *supervised learning* as depicted in Figure 4.13, the network is provided with a correct answer to every input pattern. Weights are determined so that the network can produce answers as close as possible to the known correct answers.

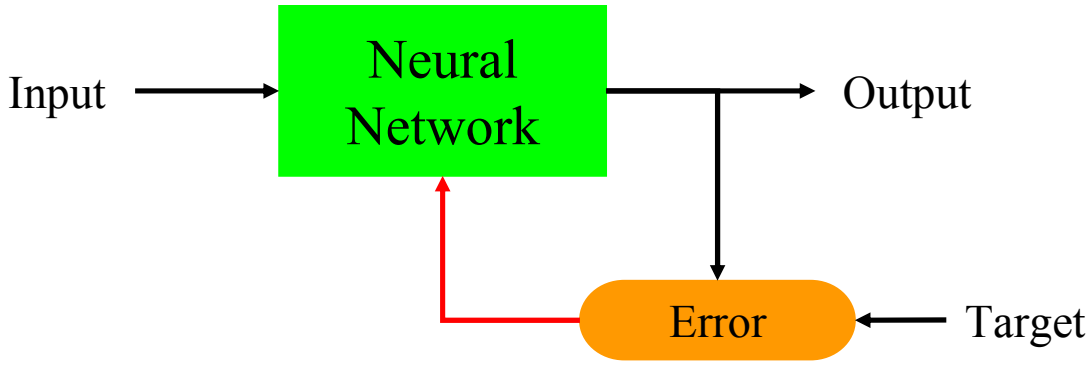


Figure 4.13 Supervised Learning Network [4]

In contrast, *unsupervised learning*, as illustrated in Figure 4.14, does not require any correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlations between patterns in the data, and organizes patterns into categories from these correlations. *Hybrid learning* combines supervised learning and unsupervised learning. Typically, a portion of weights in the

network are determined using supervised learning, while the others are obtained from unsupervised learning.

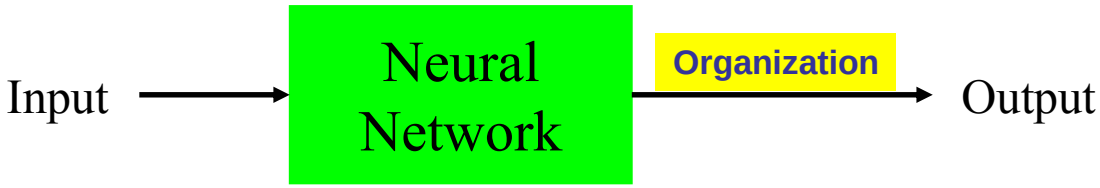


Figure 4.14 Unsupervised learning network [4]

Furthermore, learning theory must address three fundamental and practical issues associated with learning from samples [10]: (1) *capacity*, (2) *sample complexity*, and (3) *time complexity*. *Capacity* concerns how many patterns can be stored, and what functions and decision boundaries can be formed by a network. *Sample complexity* determined the number of training patterns needed to train the network in order to guarantee a valid generalization. Too few patterns may cause "over-fitting", wherein the network performs well on the training data set, but poorly on independent test patterns drawn from the same distribution as the training patterns as shown in Figure 4.7. *Computation complexity* refers to the time requirement for learning algorithm to estimate a solution from the training patterns. Currently, many existing learning algorithms have high computational complexity. Designing efficient algorithms for neural network learning is still a very active research topic. Basically, there are four fundamental types of learning rules: (1) *error-correction*, (2) *Boltzmann*, (3) *Hebbian*, and (4) *competitive learning*. The well-known backpropagation learning algorithm used in the backpropagation neural networks (or BPNN described in Section 4.2.8), which is the main neural network in this research,

is based on the error-correction principle. The learning rule of error-correction is mentioned in detail in next section.

4.2.7 Error-correction Rules

In the supervised learning paradigm, the network is given a desired output for each input pattern. During the learning process, the actual output, y, generated by the network may not equal to the desired output, d. The basic principle of error-correction learning rules is to use the error signal $(d - y)$ to modify the correction weights such that this error will be gradually reduced. The well-known perceptron learning rule is based on this error-correction principle. A perceptron consists of a single neuron with adjustable weights, $w_j, j = 1,2,\ldots,n$, and threshold value μ in threshold activation function, as shown in Figure 4.15. Given an input column vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, the net input to the neuron (before applying the threshold function) is given by

$$v = \sum_{j=1}^{n} w_j \cdot x_j - \mu \tag{4.2}$$

The output y of the perceptron is $+1$ if $v > 0$, and 0 otherwise. In a two-class classification problem, the perceptron assigns an input pattern to one class if $y = 1$, and to the other class if $y = 0$. The linear equation given by

$$\sum_{j=1}^{n} w_j \cdot x_j - \mu = 0 \tag{4.3}$$

defines the decision boundary (a hyperplane in the *n*-dimensional input space) which divides the space into two halves.

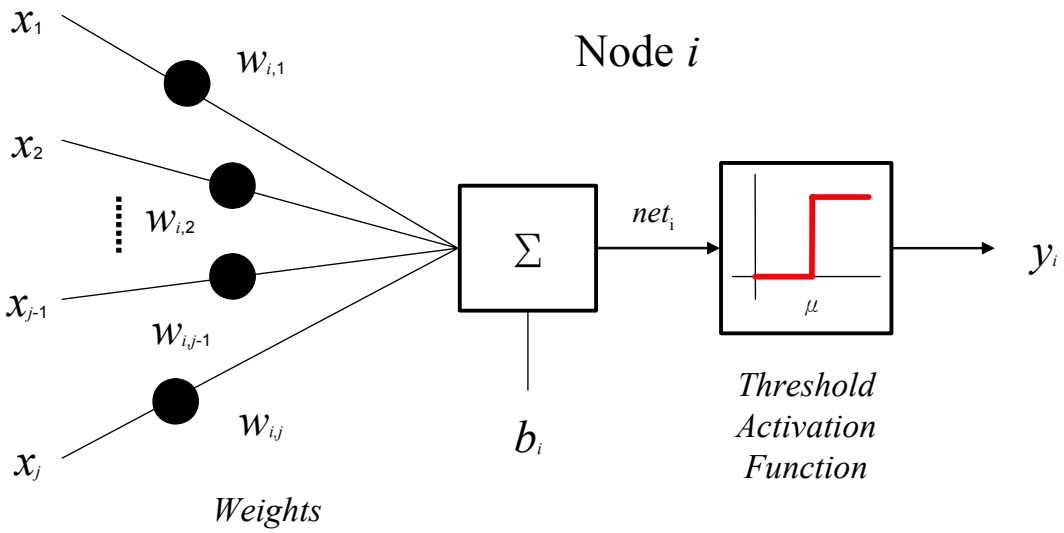


Figure 4.15 Perceptron Neuron of Threshold Activation Function [10]

Rosenblatt [85] developed a learning procedure to determine the weights and threshold in a perceptron, given a set of training patterns. The perceptron learning procedure can be described as follows.

1. Initialize the weights and threshold to small random numbers.
2. Present a pattern vector $\mathrm{x} = [x_1, x_2, \ldots, x_j]$, and evaluate the output of the neuron.
3. Update the weights according to $w_{ij}(n+1) = w_{ij}(n) + \eta \cdot (d - y_i) \cdot x_j$, where d is the desired output, n is iteration number, and $\eta\,(0.0 < \eta < 1.0)$ is the gain (step size).

Rosenblatt proved that if the training patterns are drawn from two linearly-separable classes, then the perceptron learning procedure will coverage after a finite number of iterations (also called *perceptron convergence theorem*). Other activation functions can also be used, which lead to different learning characteristics. However, a single layer

perceptron can only separate linearly separable patterns, as long as a monotonic activation function is used.

4.2.8 Backpropagation Learning Algorithm

The development of the *backpropagation learning algorithm* (or backpropagation algorithm), based on the above-mentioned *error-correction rules*, for determining weights in a multilayer perceptron has made these networks more successful in the applications of process modeling, classification, etc.

In this research, this multilayer feedforward network, by using the backpropagation learning algorithm, is simply called "*backpropagation neural network* (BPNN)" which is composed of a number of interconnected computing units (i.e. neurons, or simply nodes), each of which performs a summing and nonlinear mapping function (*activation* or *transfer* function) in a parallel manner.

The most commonly used activation functions are of sigmoidal and hyperbolic tangent type that approximates the threshold function (also known as *signum*, *step function* or *hard-limiter*) and yet provides differentiability with respect to input signals. These functions are describes as:

$$\text{Threshold function}: \quad \text{sgn}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{4.4}$$

$$\text{Sigmoid function}: \quad \text{sig}(x) = \frac{1}{1+\exp(-x)} \tag{4.5}$$

$$\text{Hyperbolic tangent function}: \quad \tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} \tag{4.6}$$

The backpropagation algorithm is a procedure for adjusting the weights of any feedforawrd network such that the network learns the functional mapping of a set of inputs to the desired outputs or targets. In other words, BPNN is a supervised neural network. This learning algorithm was first derived by Werbos [88] as "dynamic feedback" in his Harvard dissertation in statistics. It was not popularized until it was published by Rumelhart *et al*. [89].

This backrpopagation approach to learning is a myopic optimization procedure based on the gradient (or steepest) descent method. The objective is to find a set of weights that will minimize the squared error over all training patterns (i.e., samples of input-output pairs). There are two phases included in this algorithm: a forward pass followed by a reverse pass depicted in detail in Sections 4.2.8.1 and 4.2.8.2.

In the forward pass, the input signals propagate from the network input layer to the output layer. In the reverse pass, the calculated error signals propagate backward through the network, where they are used to adjust the weights. The calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the next layer. In the reverse pass, the weights of the output layer are adjusted first since the target value of each output node is available to guide the adjustment of the associated weights. Next, the weights of middle layers (or hidden layers) are sequentially adjusted.

Accordingly, the backpropagation learning algorithm [89], by using a gradient descent method is to minimize the square-error cost function in Equations (4.9) and (4.10), and is given by the following steps and flow chart as illustrated in Figure 4.16 [4]:

Step 1: Initialize the weights to small random values.

Step 2 : Input a training pair from the training sets.

Step 3 : Calculate the network output.

Step 4 : Calculate the error between the network output and the target value.

Step 5 : Calculate the change of each weight and each bias.

Step 6 : Adjust the weights and biases of the network layer in a way that minimizes this error, calculated in Step 5.

Step 7 : Input new training pair from the training sets

Step 8 : Repeat steps 2 ~ 7 for each pair of input-output vectors in the training set until the error for the entire system is acceptably low.

In step 1, initializing the weights to small random values is to ensure that the network is not saturated by large values of weights. If all weights start at equal values, and the desired performance requires unequal weights, the network would not train at all.

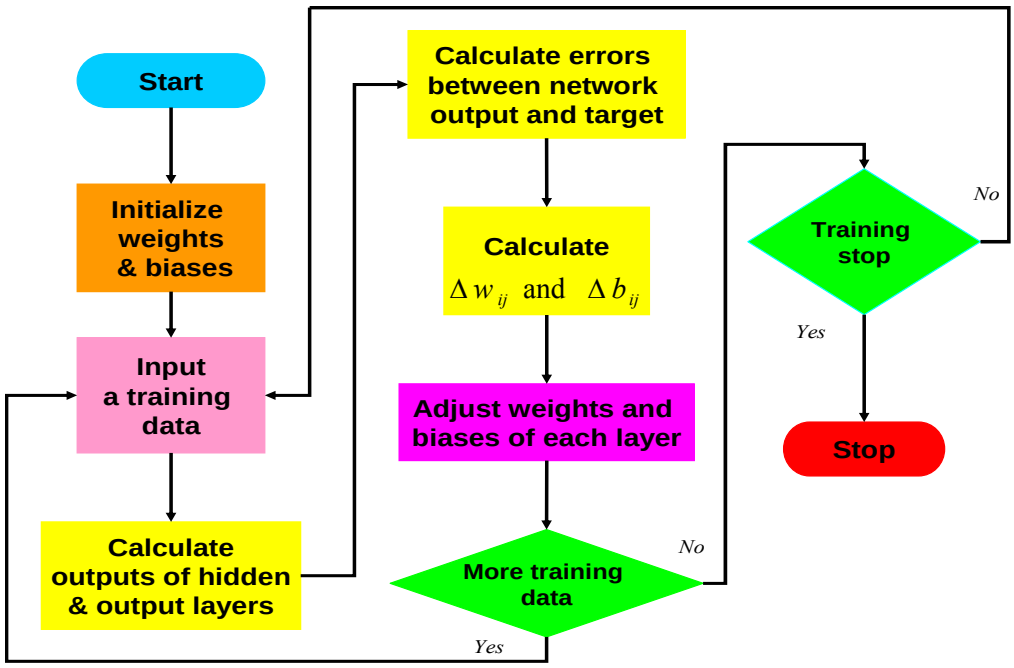


Figure 4.16 Flow chart of Backpropagation Learning Algorithm [4]

4.2.8.1 Calculation of Weights for the Output-layer Nodes [10]

Before discussing further, recalling the net input of a node which is defined as the weighted sum of the incoming signals plus a threshold or bias. For instance, the output of node *k* in Figures 4.17 and Figure 4.18 is given by

$$y_k = F_k(net_k) = F_k\left(\sum_{j=1}^{Nq} w_{kj} \cdot v_j + \theta_{rk}\right) \tag{4.7}$$

where $F_k(\bullet)$ is the activation function (see Equations (4.4) to (4.6))

net_k = the weighted sum of the incoming signals

v_j = the *j*-th input to the *k*-th node in the layer *r*

w_{kj} = the weight of the *j*-th input to the *k*-th node in the layer *r*

θ_{rk} = the bias (or threshold value) of the *k*-th node in the layer *r*

y_k = the output to the *k*-th node in the layer *r*

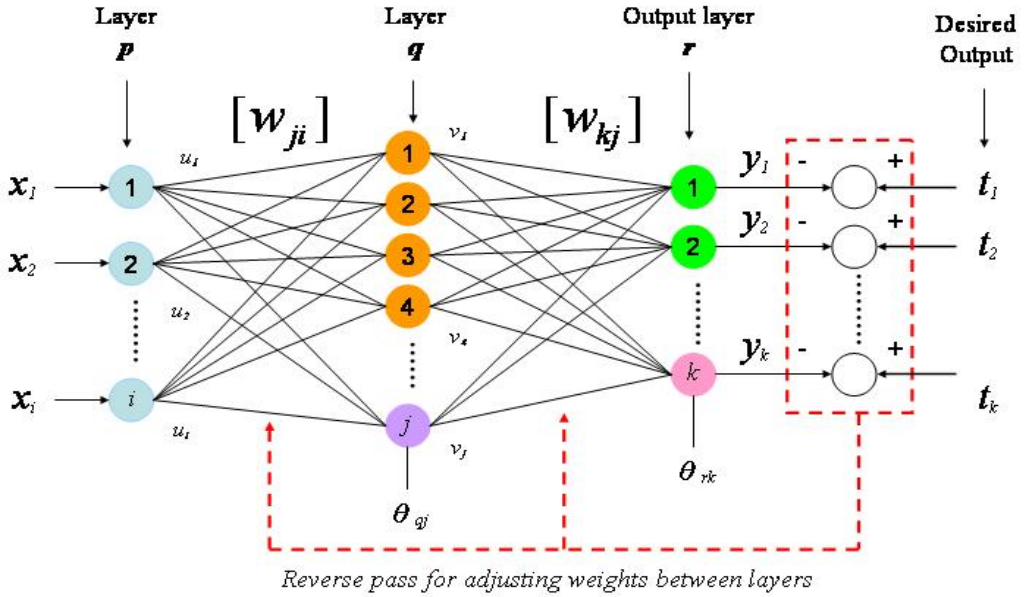


Figure 4.17 Forward and Backward Pass of Multi-layered Neural Network [10]

The numbers of nodes in layers *p*, *q*, and *r* are *Np*, *Nq*, and *Nr*, respectively. The [$\boldsymbol{u}_i$] and [$\boldsymbol{v}_j$] vectors are the output vectors of the hidden layer *p* and *q*. The [$\boldsymbol{y}_k$] is the output vector of the output layer *r*. The [$\boldsymbol{\theta}_{qj}$] and [$\boldsymbol{\theta}_{rk}$] are the bias vectors of the layers *q* and *k*.

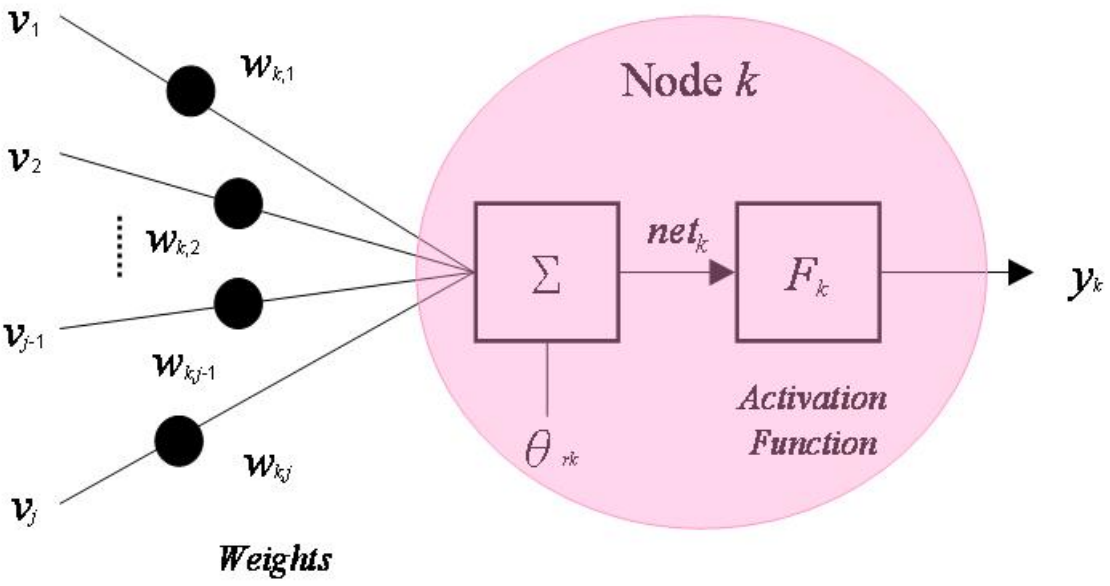


Figure 4.18 Multiple-input of Node *k* in Figure 4.16 [10]

Taking input-output ($\boldsymbol{R}^{Np} \rightarrow \boldsymbol{R}^{Nr}$) mapping for example and denoting $\boldsymbol{w}_{ji}$ as the weight on the connection between the *j*-th node in the layer $\boldsymbol{q}$ to *i*-th node in layer $\boldsymbol{p}$, let $\left\{\left(\boldsymbol{x}^{(1)},\boldsymbol{t}^{(1)}\right),\left(\boldsymbol{x}^{(2)},\boldsymbol{t}^{(2)}\right),\ldots,\left(\boldsymbol{x}^{(M)},\boldsymbol{t}^{(M)}\right)\right\}$ be a set of *M* training patterns (input-output pairs), where $\boldsymbol{x}^{(m)} \in \boldsymbol{R}^{Np}$, $\boldsymbol{m} = 1 \cdots \boldsymbol{M}$ is the input vector in the *Np*-dimensional pattern space, and $\boldsymbol{t}^{(m)} \in \boldsymbol{R}^{Nr}, \boldsymbol{m} = 1 \cdots \boldsymbol{M}$ is the output vector in the *Nr*-dimensional objective space. After *n* learning circulations, the *error function* and *instantaneous square error value* of the *k*-th node in the output layer $\boldsymbol{r}$ can be defined as below:

$$\boldsymbol{e}_k(\boldsymbol{n}) = \boldsymbol{t}_k(\boldsymbol{n}) - \boldsymbol{y}_k(\boldsymbol{n}), \text{ and } \frac{1}{2}\boldsymbol{e}_k^2(\boldsymbol{n}) \tag{4.8}$$

The *summation of instantaneous square error values* of all nodes in the output layer $\boldsymbol{r}$ can be described as:

$$\boldsymbol{E}(\boldsymbol{n}) = \frac{1}{2}\sum_{\boldsymbol{k}=1}^{\boldsymbol{Nr}} \boldsymbol{e}_{\boldsymbol{k}}^{2}(\boldsymbol{n}) = \frac{1}{2}\sum_{\boldsymbol{k}=1}^{\boldsymbol{Nr}} \{\boldsymbol{t}_{\boldsymbol{k}}(\boldsymbol{n}) - \boldsymbol{y}_{\boldsymbol{k}}(\boldsymbol{n})\}^{2} \tag{4.9}$$

where *Nr* is the number of nodes in the output layer $\boldsymbol{r}$.

Further, the *average error function* of a set of *M* training patterns is defined as follow:

$$\boldsymbol{E}_{\boldsymbol{ave}}(\boldsymbol{n}) = \frac{1}{\boldsymbol{M}}\sum_{\boldsymbol{n}=1}^{\boldsymbol{M}} \boldsymbol{E}(\boldsymbol{n}) \tag{4.10}$$

From Equations (4.7), (4.9) and (4.10), both $E(n)$ and $\boldsymbol{E}_{\boldsymbol{ave}}(\boldsymbol{n})$ are functions of weights, w_{ij} and can be represented as the *square-error cost functions* of network learning from the training data sets. The only difference between these two error functions is that first one is called as *pattern learning*, and the second one is called as *batch learning*. Again, the major objective of training networks is to minimize the $E(n)$ or $\boldsymbol{E}_{\boldsymbol{ave}}(\boldsymbol{n})$.

For simplicity, $E(n)$ is used as the cost function for the following derivation. The input of node *k* in the layer *r* is

$$\boldsymbol{net}_{\boldsymbol{k}} = \sum_{\boldsymbol{j}=1}^{\boldsymbol{Nq}} \boldsymbol{w}_{\boldsymbol{kj}} \cdot \boldsymbol{v}_{\boldsymbol{j}} + \theta_{\boldsymbol{rk}} \tag{4.11}$$

The output of this node is

$$\boldsymbol{y}_{\boldsymbol{k}} = \boldsymbol{F}_{\boldsymbol{k}}(\boldsymbol{net}_{\boldsymbol{k}}) = \boldsymbol{F}_{\boldsymbol{k}}\left(\sum_{\boldsymbol{j}=1}^{\boldsymbol{Nq}} \boldsymbol{w}_{\boldsymbol{kj}} \cdot \boldsymbol{v}_{\boldsymbol{j}} + \theta_{\boldsymbol{rk}}\right) \tag{4.7}$$

If $\boldsymbol{F}_{\boldsymbol{k}}(\bullet)$ is the *sigmoid* function as depicted in Equation (4.5), the $\boldsymbol{y}_{\boldsymbol{k}}$ can be expressed as:

$$y_k = F_k(net_k) = \frac{1}{1+\exp(-net_k)} = \frac{1}{1+\exp\left(-\left(\sum_{j=1}^{Nq} w_{kj}\cdot v_j + \theta_{rk}\right)\right)} \tag{4.12}$$

From Equation (4.9), the *square error cost function* (or *error function*) after *n* learning circulations can be described as the function of weights and biases:

$$E(n) = \frac{1}{2}\sum_{k=1}^{Nr} e_k^2(n) = \frac{1}{2}\sum_{k=1}^{Nr}\left\{t_k(n) - \frac{1}{1+\exp\left(-\left(\sum_{j=1}^{Nq} w_{kj}(n)\cdot v_j(n) + \theta_{rk}(n)\right)\right)}\right\}^2 \tag{4.13}$$

Usually, the biases, $[\theta_{rk}(n)]$, are constants.

According to the *least-mean-square error algorithm* and gradient descent method, the change in weight, $\Delta w_{kj}(n)$, is proportional to the negative of the rate of change of the square error with respect to that weight, $-\frac{\partial E(n)}{\partial w_{kj}(n)}$. Using the change rule, $\frac{\partial E(n)}{\partial w_{kj}(n)}$ can be derived as Equation (4.14)

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = \frac{\partial E(n)}{\partial net_k(n)}\frac{\partial net_k(n)}{\partial w_{kj}(n)} \tag{4.14}$$

Applying Equation (4.1), we get

$$\frac{\partial net_k(n)}{\partial w_{kj}(n)} = \frac{\partial}{\partial w_{kj}(n)}\left(\sum_{j=1}^{Nq} w_{kj}(n)\cdot v_j(n) + \theta_{rk}(n)\right) = v_j(n), \text{ here, } \theta_{rk}(n) = -1 \tag{4. 15}$$

Now, we can define the gradient function, $\delta_k(n)$, as

$$\delta_k(n) = -\frac{\partial E(n)}{\partial net_k(n)} = \frac{\partial E(n)}{\partial y_k(n)}\frac{\partial y_k(n)}{\partial net_k(n)} = -\{t_k(n) - y_k(n)\}\cdot F_k'(net_k(n)) \tag{4.16}$$

Then, the change of weights after *n* learning circulations can be written as

$$\Delta \boldsymbol{w}_{kj}(\boldsymbol{n}) = \eta_q \cdot \delta_k(\boldsymbol{n}) \cdot \boldsymbol{v}_j(\boldsymbol{n}) = \eta_q \cdot \{\boldsymbol{t}_k(\boldsymbol{n}) - \boldsymbol{y}_k(\boldsymbol{n})\} \cdot \boldsymbol{F}_k'(\boldsymbol{net}_k(\boldsymbol{n})) \cdot \boldsymbol{v}_j(\boldsymbol{n}) \qquad (4.17)$$

where, η_q is the learning rate which can determine the modification amplitude for the gradient descent method. Usually, η_q is set as a small positive value between 0 and 1. The adjusted weights can be obtained by the Equation (4.18).

$$\boldsymbol{w}_{kj}(\boldsymbol{n}+1) = \boldsymbol{w}_{kj}(\boldsymbol{n}) + \Delta \boldsymbol{w}_{kj}(\boldsymbol{n}) = \boldsymbol{w}_{kj}(\boldsymbol{n}) + \eta_q \cdot \delta_k(\boldsymbol{n}) \cdot \boldsymbol{v}_j(\boldsymbol{n}) \qquad (4.\ 18)$$

According to Equation (4.17), it is required to calculate the $\delta_k(\boldsymbol{n})$ first for obtaining $\Delta \boldsymbol{w}_{kj}(\boldsymbol{n})$.

Obviously, in the reverse pass, the weights of the output layer *r* are adjusted first since the target value of each output node is available to guide the adjustment of the associated weights. Next, the weights of the hidden layers *p* and *q* will be adjusted, respectively. The problem is that the hidden-layer nodes have no target values. Hence, the training is more complicated because the error must be back propagated through the network, including the nonlinear functions, layer by layer.

4.2.8.2 Calculation of Weights for the Hidden-layer Nodes [4]

Similar to the previous statement, according to the *least-mean-square error algorithm* and gradient descent method, the change in a weight in the hidden layer, $\Delta \boldsymbol{w}_{ji}(\boldsymbol{n})$, is proportional to the negative of the rate of change of the square error with

respect to that weight, $-\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{w}_{ji}(\boldsymbol{n})}$. Using the change rule, $\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{w}_{ji}(\boldsymbol{n})}$ can be derived as

Equation (4.14)

$$\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{w}_{ji}(\boldsymbol{n})} = \frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{net}_{j}(\boldsymbol{n})} \frac{\partial \boldsymbol{net}_{j}(\boldsymbol{n})}{\partial \boldsymbol{w}_{ji}(\boldsymbol{n})} \tag{4.19}$$

Similarly from Equation (4.11), we get

$$\frac{\partial \boldsymbol{net}_{j}(\boldsymbol{n})}{\partial \boldsymbol{w}_{ji}(\boldsymbol{n})} = \frac{\partial}{\partial \boldsymbol{w}_{ji}(\boldsymbol{n})}\left(\sum_{i=1}^{Np} \boldsymbol{w}_{ji}(\boldsymbol{n}) \cdot \boldsymbol{u}_{i}(\boldsymbol{n}) + \theta_{qj}(\boldsymbol{n})\right) = \boldsymbol{u}_{i}(\boldsymbol{n}), \text{ here, } \theta_{qj}(\boldsymbol{n}) = -1 \tag{4.20}$$

Now, we can define the gradient function, $\delta_{j}(\boldsymbol{n})$, as

$$\delta_{j}(\boldsymbol{n}) = -\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{net}_{j}(\boldsymbol{n})} = \left[\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}\right] \frac{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}{\partial \boldsymbol{net}_{j}(\boldsymbol{n})} = \left[\sum_{k=1}^{Nr}\left(-\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{net}_{k}(\boldsymbol{n})}\right) \frac{\partial \boldsymbol{net}_{k}(\boldsymbol{n})}{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}\right] \frac{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}{\partial \boldsymbol{net}_{j}(\boldsymbol{n})} \tag{4.21}$$

From Equation (4.7), we can derive the following expression.

$$\boldsymbol{v}_{j} = \boldsymbol{F}_{j}(\boldsymbol{net}_{j}) = \boldsymbol{F}_{j}\left(\sum_{i=1}^{Np} \boldsymbol{w}_{ji} \cdot \boldsymbol{u}_{i} + \theta_{qj}\right) \Rightarrow \frac{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}{\partial \boldsymbol{net}_{j}(\boldsymbol{n})} = \boldsymbol{F}_{j}^{'}(\boldsymbol{net}_{j}(\boldsymbol{n})) \tag{4.22}$$

From the equation (4.11), we can derive Equation (4.23).

$$\frac{\partial \boldsymbol{net}_{k}(\boldsymbol{n})}{\partial \boldsymbol{v}_{j}(\boldsymbol{n})} = \frac{\partial}{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}\left[\sum_{k=1}^{Nr}\left(\boldsymbol{w}_{kj}(\boldsymbol{n}) \cdot \boldsymbol{v}_{j}(\boldsymbol{n})\right) + \theta_{rk}(\boldsymbol{n})\right] = \boldsymbol{w}_{kj}(\boldsymbol{n}) \tag{4.2.23}$$

Combining Equations (4.16), (4.22), and (4.23), we can arrive at Equation (4.24).

$$\begin{aligned} \delta_{j}(\boldsymbol{n}) &= -\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{net}_{j}(\boldsymbol{n})} = \left[\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}\right] \frac{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}{\partial \boldsymbol{net}_{j}(\boldsymbol{n})} = \left[\sum_{k=1}^{Nr}\left(-\frac{\partial \boldsymbol{E}(\boldsymbol{n})}{\partial \boldsymbol{net}_{k}(\boldsymbol{n})}\right) \frac{\partial \boldsymbol{net}_{k}(\boldsymbol{n})}{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}\right] \frac{\partial \boldsymbol{v}_{j}(\boldsymbol{n})}{\partial \boldsymbol{net}_{j}(\boldsymbol{n})} \\ &= \left[\sum_{k=1}^{Nr} \delta_{k}(\boldsymbol{n}) \cdot \boldsymbol{w}_{kj}(\boldsymbol{n})\right] \cdot \boldsymbol{F}_{j}^{'}(\boldsymbol{net}_{k}(\boldsymbol{n})) \end{aligned} \tag{4.24}$$

Therefore, the change of weights in the hidden layer can be written as

$$\Delta \boldsymbol{w}_{ji}(\boldsymbol{n}) = \eta_p \cdot \delta_j(\boldsymbol{n}) \cdot \boldsymbol{u}_i(\boldsymbol{n}) = \eta_p \cdot \left[\sum_{k=1}^{Nr} \left(\delta_k(\boldsymbol{n}) \cdot \boldsymbol{w}_{kj}(\boldsymbol{n}) \right) \right] \boldsymbol{F}_j'(\boldsymbol{net}_k(\boldsymbol{n})) \cdot \boldsymbol{u}_i(\boldsymbol{n}) \quad (4.25)$$

If there are more than one hidden layers of nodes, this process moves through the network, layer by layer to the input, adjusting the weights as it goes. When finished, a new training input is applied and the process starts all over again till an acceptable error is reached. At that point, the network is trained.

4.2.9 Limitations of Neural Networks

Unlike fuzzy inference systems (see Section 4.3), it is difficult to give physical meaning to each connection weight in the neural networks. Also, the precision of the outputs is sometimes limited because the variables are effectively treated as analog variables (even when implemented on a digital computer), and "minimization of least squares errors" does not mean "zero error".

Furthermore, the performance of a neural network is controlled by a number of design parameters, such as the *network topology*, *type of transfer function*, *learning coefficients*, *moment parameter,* and so on. In practice, a fixed combination of these parameters which can be applied to every problem does not exist. Identifying the best combination of parameters for a particular application is usually done empirically and is an ongoing area of neural network research activities.

Choosing the network size is a critical step in the network model building. A network with too few nodes may not be capable of learning the training data, while a network with too many nodes will *memorize* the training examples and generalize poorly

(i.e., overfitting) [90]. Selecting the optimal network size and architecture (i.e., the number of hidden layers and nodes, transfer functions, etc.), which will capture the uncertainties and structures of the data, is an initial obstacle needed to be overcome.

Besides, "*the Achilles heel*" of neural networks is the need for substantial data that are representative and cover the entire range over which different variables are expected to change. If the number of free parameters, or weights, in a network is equal to or exceed the number of training examples, or exemplars, that are available to be used in the training phase to determine the appropriate weight values. This is analogous to the problem of an underdetermined system, where the number of unknown variables exceed the numbers of equations and cannot be determined or one or more of these variables will have multiple, ambiguous values that may not satisfy an additional equation that is subsequently introduced. In a neural network, this situation results in sparse training of the network weights. Such a sparsely trained network may perform adequately in training, but may fail when subsequently presented with new inputs that result in mappings to high-dimensional areas of the weight space that were determined by only a few training exemplars.

The resulting output of the network from these sparse mappings is unlikely to yield meaningful results. In order to avoid this problem, a common *rule-of-thumb* among neural network designers is to assume the ratio of training exemplars to network weights a minimum of 10:1.

4.3 Fundamentals of Fuzzy Inference Systems

4.3.1 Introduction

In the literature, there are two kinds of justification for fuzzy inference systems (or fuzzy system for short) theory [7]:

- The real world is too complicated to obtain the precise descriptions; therefore approximation (or fuzziness) must be introduced in order to obtain a reasonable, yet trackable model.
- Living in the current information era, human knowledge becomes increasingly important. There needs a theory to formulate human knowledge in a systematic manner and put it into engineering system, together with other information such as mathematical models and sensory measurements.

The above two justifications characterizes the unique features of fuzzy system theory and justifies the existence of fuzzy systems theory as an independent branch in engineering fields.

For many practical systems, the important information comes from two major sources: the first source is human experts who describe their knowledge about the system in natural languages. The second source is sensory measurements and mathematical models that are derived according to the physical laws. An important task, therefore, is to combine these two types of information into the system designs. To achieve this combination, a key problem is how to formulate human knowledge into a similar framework used to formulate sensory measurements and mathematical models. Essentially, fuzzy systems can effectively perform this transformation.

Fuzzy systems based on *fuzzy numbers*, *fuzzy sets* and *fuzzy IF-THEN rules*, have emerged as one of the most active and fruitful areas for research, because of the capability of capturing the approximate qualitative aspects of human knowledge and reasoning. Moreover, fuzzy systems are also called knowledge-based or rule-based systems. A fuzzy IF-THEN rule is an IF-THEN statement in which some words are characterized by continuous *membership functions* as the example given below – human drivers usually use the following type of rule to drive a car in normal situations:

IF the speed of a car is high, THEN apply less force to the accelerator (4.26)

where the words "high" and "less" are characterized by the two membership functions shown in Figure 4.26. Of course, more rules will be needed for more complicated and real situations. A fuzzy system is constructed from a collection of all fuzzy IF-THEN rules.

In Figure 4.19, m_i and n_i (i =1, 2, 3) are membership values; s_i and f_i (i =1, 2, 3) are car speeds and forces to the accelerator. Different rules may have different membership functions, which precisely translates the fuzzy linguistic descriptions to non-fuzzy membership values.

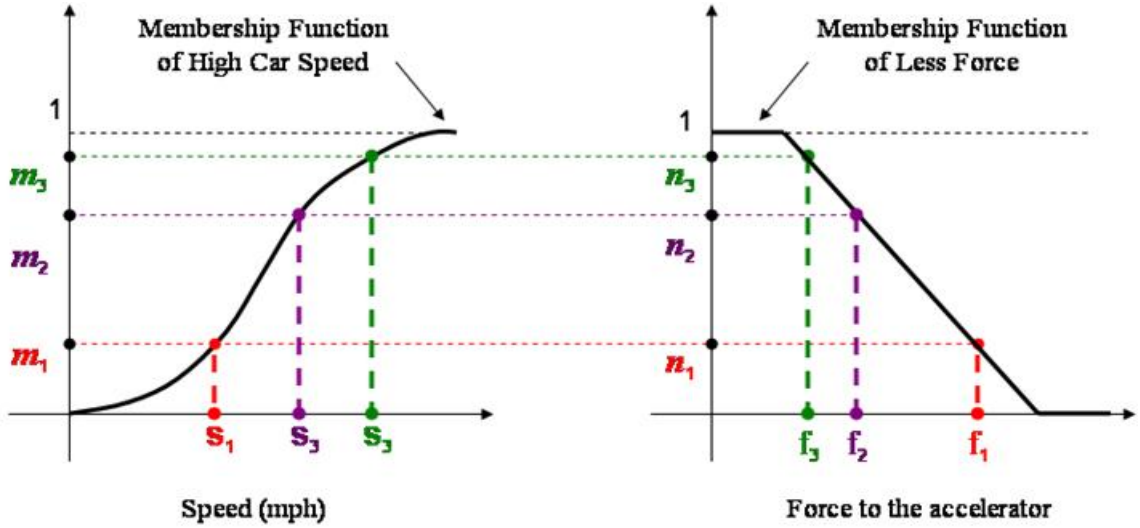


Figure 4.19 Membership Functions of Fuzzy IF-THEN Rules

Besides, fuzzy systems can address the imprecision of the input and output variables directly by defining them with the aforementioned fuzzy numbers and fuzzy sets that can be expressed in linguistic terms (e.g. *cold*, *warm*, and *hot*). They also allow far greater flexibility in formulating system descriptions at the appropriate level of detail. Hence, fuzziness has a lot to do with the parsimony and the accuracy and efficiency of a description. This means that complex process behavior can be described in general terms without precisely defining the complex (usually nonlinear) phenomena involved. Parapharsing the law of *Occam's Razor*, the philosophical principle holding that more parsimonious description are more representative of nature. We may say that fuzzy descriptions are more parsimonious and hence easier to formulate and modify, more tractable, and perhaps more tolerant of change and even failure.

Figure 4.20 shows an example of trade-off between precision and significance which are something that humans have been managing for a very long time. Obviously, precision descriptions cannot truly express the emergent situation as shown in Figure

4.20. On the contrary, parsimonious linguistic descriptions can fully and accurately deliver the emergency and significance.

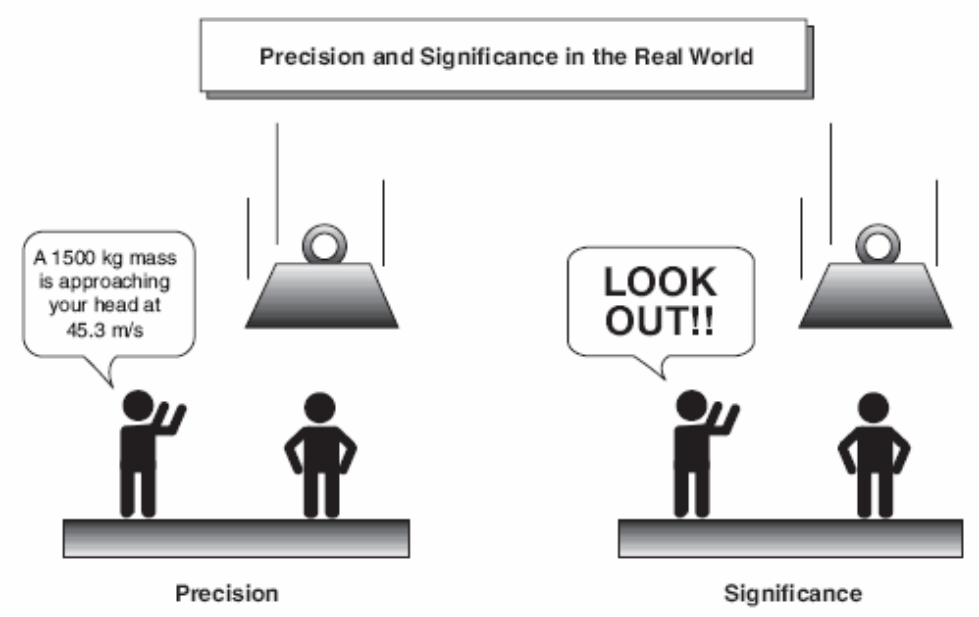


Figure 4.20 Precision and Significance in the Real World [2]

"*So far the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality*" – Albert Einstein.

4.3.2 Fuzzy Sets

A classical set is a set with a crisp boundary. For example, a classical set N of real numbers greater than 100 can be expressed as

$$N = \{x \mid x > 100\}, x \in R \tag{4.27}$$

where there is a clear, unambiguous boundary 100 such that if x is greater than this number, then x belongs to the set N; otherwise x does not belong to this set. Although

classical sets (or crisp sets) are suitable for various applications and have proven to be an important tool for mathematics and computer science, as mentioned in Subsection 4.3.1, they can not fully or accurately reflect the nature of human concepts and thoughts, which tend to be abstract and imprecise.

In contrast to a classical set, a fuzzy set, as the name implies, is a set without a crisp boundary. That is, the transition from "belong to a set" to "not belong to a set" is gradual, and this smooth transition is characterized by membership functions that give fuzzy sets flexibility in modeling the commonly used linguistic expressions, such as "the water is hot" or "the temperature is high." Zadeh [83] pointed out such imprecisely defined sets or classes play an important role in human thinking, particularly in the domains of *pattern recognition*, *communication of information*, and *abstraction*.

If X is a collection of objects denoted generically by x, then a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \tag{4.28}$$

where $\mu_A(x)$ is called the membership function for the fuzzy set A. This membership function maps each element of X to a membership grade (or membership value) between 0 and 1. If the value of the membership function $\mu_A(x)$ is restricted to either 0 or 1, then A is reduced to a classical set and $\mu_A(x)$ is the characteristic function of A. Obviously, the definition of a fuzzy set is a simple extension of the definition of a classical set in which the characteristic function is permitted to have any value between 0 and 1. Usually, X is referred to as the *universe of discourse* (or simply the *universe*), and it may consist of discrete (ordered or non-ordered) objects or continuous space.

4.3.3 Fuzzy Inference Systems

There are three types of fuzzy inference systems that are commonly used in practice. They are (1) pure fuzzy systems, (2) Takagi-Sugeno-Kang (TSK) fuzzy systems [94, 95], and (3) fuzzy systems with fuzzifier and defuzzifizer.

The basic configuration of a pure fuzzy system is shown in Figure 4.21. The fuzzy rule-base represents the collection of fuzzy IF-THEN rules. The fuzzy inference engine (or decision-making unit) combines these fuzzy IF-THEN rules into a mapping from fuzzy sets in the input space U to fuzzy sets in the output space V. The main problem of this fuzzy system is that its inputs and outputs are fuzzy sets (i.e., words in natural languages), whereas in engineering systems the inputs and outputs are real-value variables.

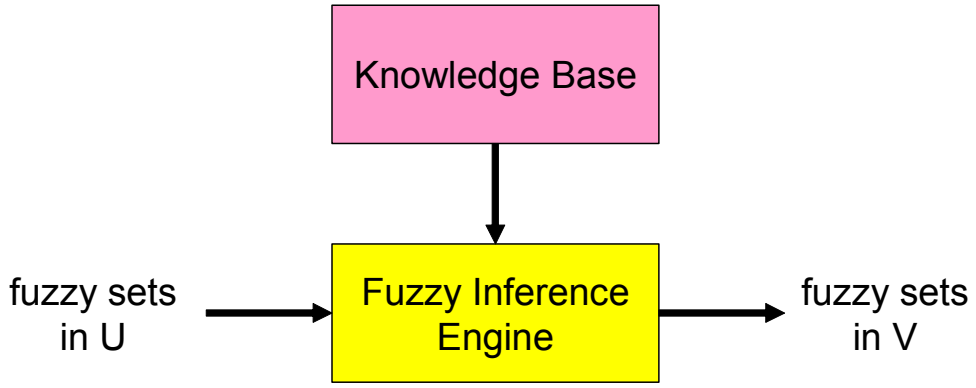


Figure 4.21 Basic Configuration of pure fuzzy systems [7]

Takagi and Sugeno [95] and Sugeno and Kang [94] proposed another fuzzy system, namely, Takagi-Sugeno-Kang fuzzy system, whose inputs and outputs are real-valued variables. The problems of previous pure fuzzy system can be solved by considering the

TSK systems use rules in the following form, instead of considering the fuzzy IF-THEN rules in the form of Equation (4.26).

IF the speed x *of a car is high, THEN the force to the accelerator is* $y = c \cdot x$ (4.29)

where the word "high" has the same meaning as in Equation (4.26), and c is a constant. Comparing Equation (4.26) with Equation (4.29), the THEN part of the rule changes from a description using words in natural languages to a simple mathematical formula. This change makes it easier to combine the rules. In fact, the TSK fuzzy system, as shown in Figure 4.22, is a weighted average of the values in the THEN parts of the rules. However, there exist two main problems in TSK fuzzy system: (1) because of the mathematical expression in the THEN part, there may not provide a natural framework to represent human knowledge, and (2) there is not much freedom left to apply different principles in fuzzy logic, so that the versatility of fuzzy systems is not well-represented in this framework.

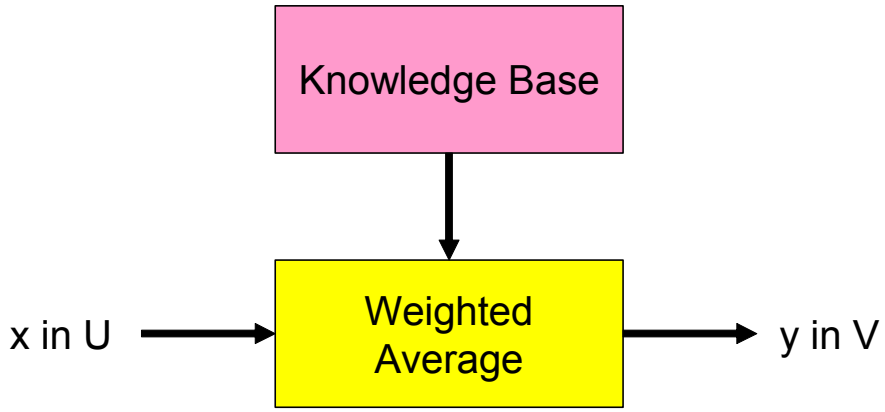


Figure 4.22 Basic configuration of Takagi-Sugeno-Kang fuzzy system [7]

In order to solve the problems of TSK fuzzy system, adding the fuzzifier and defuzzifier into this system will be required. The fuzzifier can transform the real-

valued variables into the fuzzy sets to the inputs; on the contrary, the defuzzifizer can transform the fuzzy sets back to the real-values variables to the outputs, as shown in Figure 4.23. Because of these transformations, the knowledge-based fuzzy systems can be widely used in engineering applications (e.g. control, signal processing, process modeling, communication systems) in the same manner as the performances of mathematical models and sensory measurements. Consequently, the analysis and design of the resulting combined systems can be performed in a mathematically rigorous fashion.

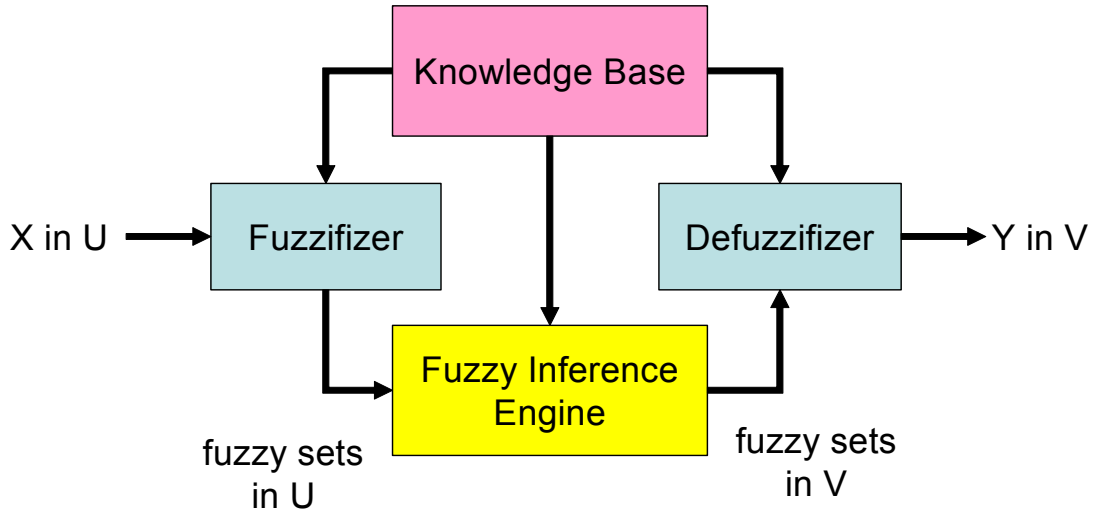


Figure 4.23 Basic Configuration of fuzzy systems with fuzzifier and defuzzifier [7]

Conclusively, combining with the above three systems, fuzzy inference systems (FIS) are also known as fuzzy-rule-based systems or fuzzy controllers when used as controllers. Basically a fuzzy inference system is composed of five functional blocks as shown in Figure 4.24

- **Rule base** – containing a number of fuzzy if-then rules.

- **Database** – defining the membership functions of the fuzzy sets used in the fuzzy rules.
- **Decision-making unit** – performing the inference operations on the rules.
- **Fuzzification interface** – transforming the crisp inputs into degrees of match with linguistic values.
- **Defuzzification interface** – transforming the fuzzy results of the inference into a crisp output.

Usually, the rule base and the database are jointly referred to as the **knowledge base**

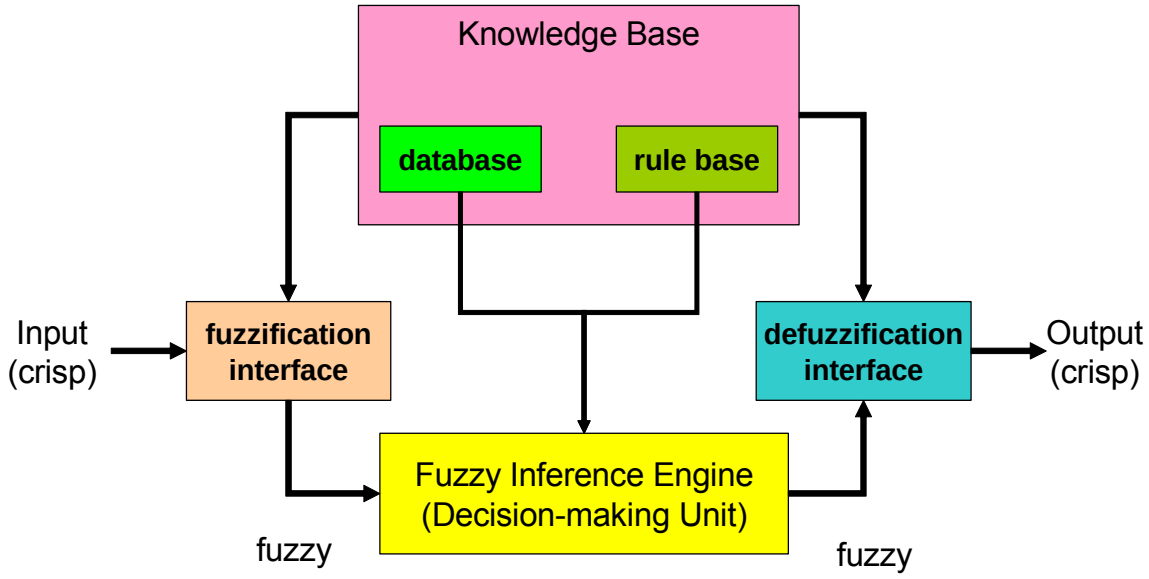


Figure 4.24 Fuzzy Inference System [7]

4.3.4 Fuzzy Reasoning Mechanisms [8]

Fuzzy reasoning, also known as approximate reasoning, is an inference procedure that derives conclusions from a set of fuzzy IF-THEN rules and known facts. The steps of fuzzy reasoning performed by fuzzy inference systems are:

1. Fuzzification – comparing the input variables with the membership functions on the premise part to obtain the membership values (or compatibility measures) of each linguistic label.
2. Product – combining the membership values on the premise part to get the firing strength (weight) of each rule through a specific *T-norm* operator, usually *multiplication* or *minimum*.
3. Generating the qualified consequent (either fuzzy or crisp) of each rule depending on the firing strength.
4. Defuzzification – aggregating the qualified consequents to produce a crisp output.

In past years, several types of fuzzy rule reasoning have been proposed and most fuzzy inference systems can be classified into three types, according to the types of fuzzy reasoning and fuzzy IF-THEN rules employed:

Type 1 : The final output is the weighted average of each rule's crisp output, induced by the rule's firing strength and output membership functions. However, the output membership functions used in this type must be monotonically non-decreasing (Figure 4.25)

Type 2 : The final output is obtained by applying the maximum operation to the qualified fuzzy outputs, each of which is derived by the minimum of firing strength and the output membership function of each rule. Various schemes have been proposed to choose the final crisp output based on the overall fuzzy output (for example, center of area, bisector of area, or mean of maximum) (Figure 4.26).

Type 3 : The output of each rule is a linear combination of inputs plus a constant term, and the final output is the weighted average of each rule's output (Figure 4.27).

Figures 4.25, to 4.27 show different types of fuzzy rules and fuzzy reasoning mechanisms by utilizing a two-input fuzzy inference system.

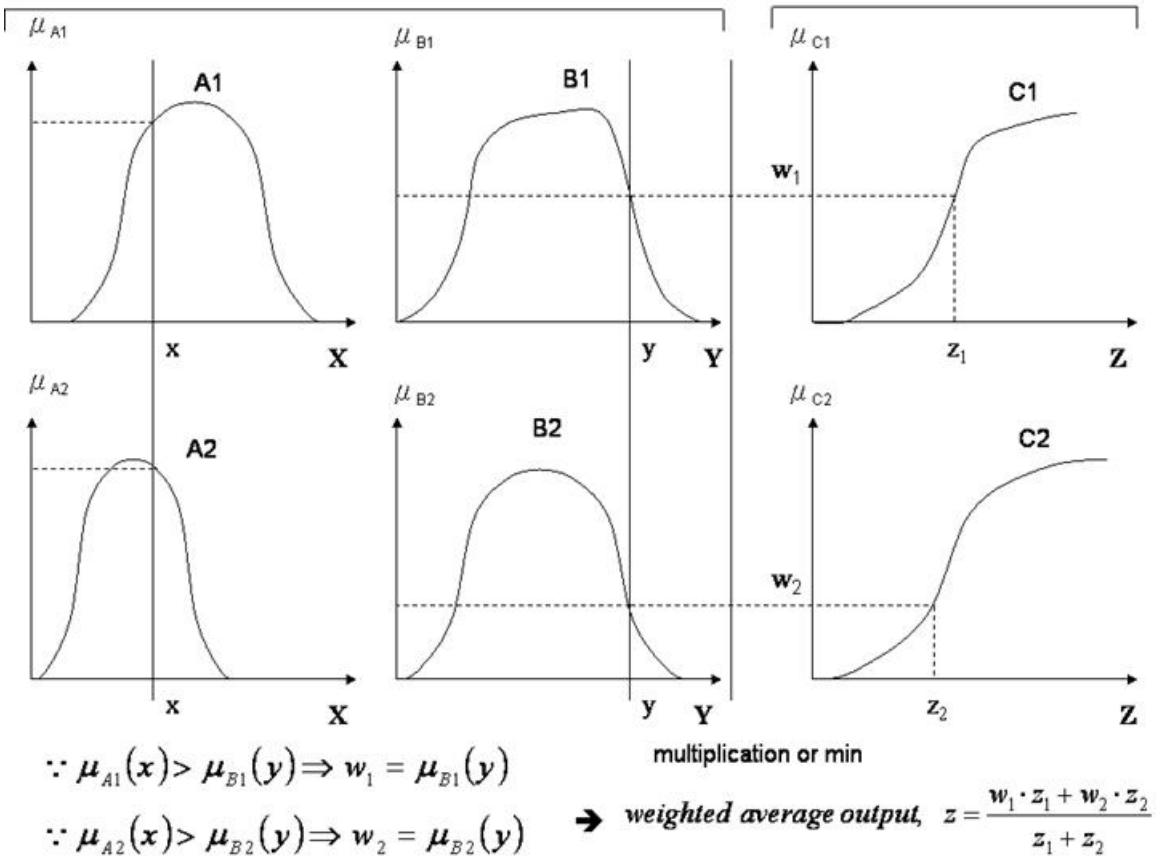
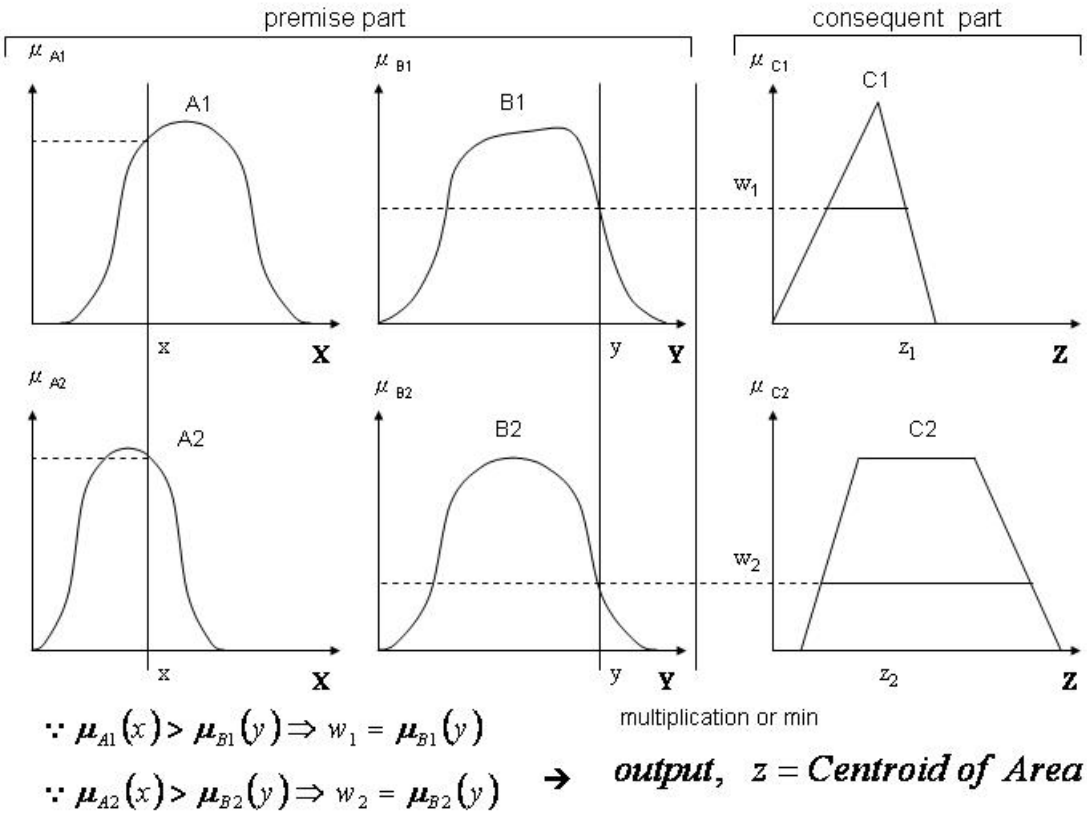


Figure 4.25 Type 1 – Fuzzy Reasoning Mechanisms [24]



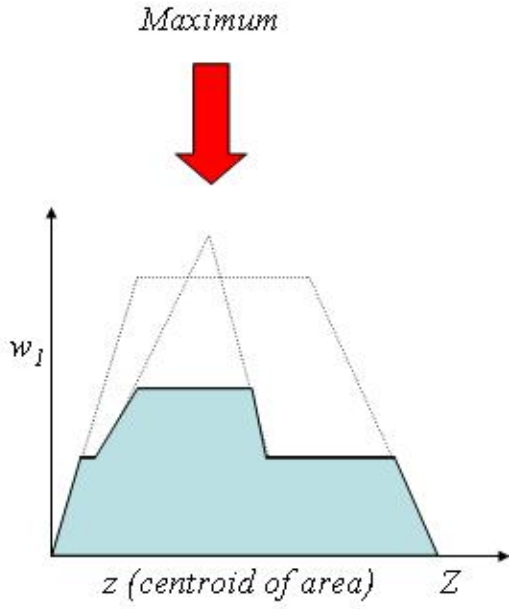


Figure 4.26 Type 2 – Fuzzy Reasoning Mechanisms [24]

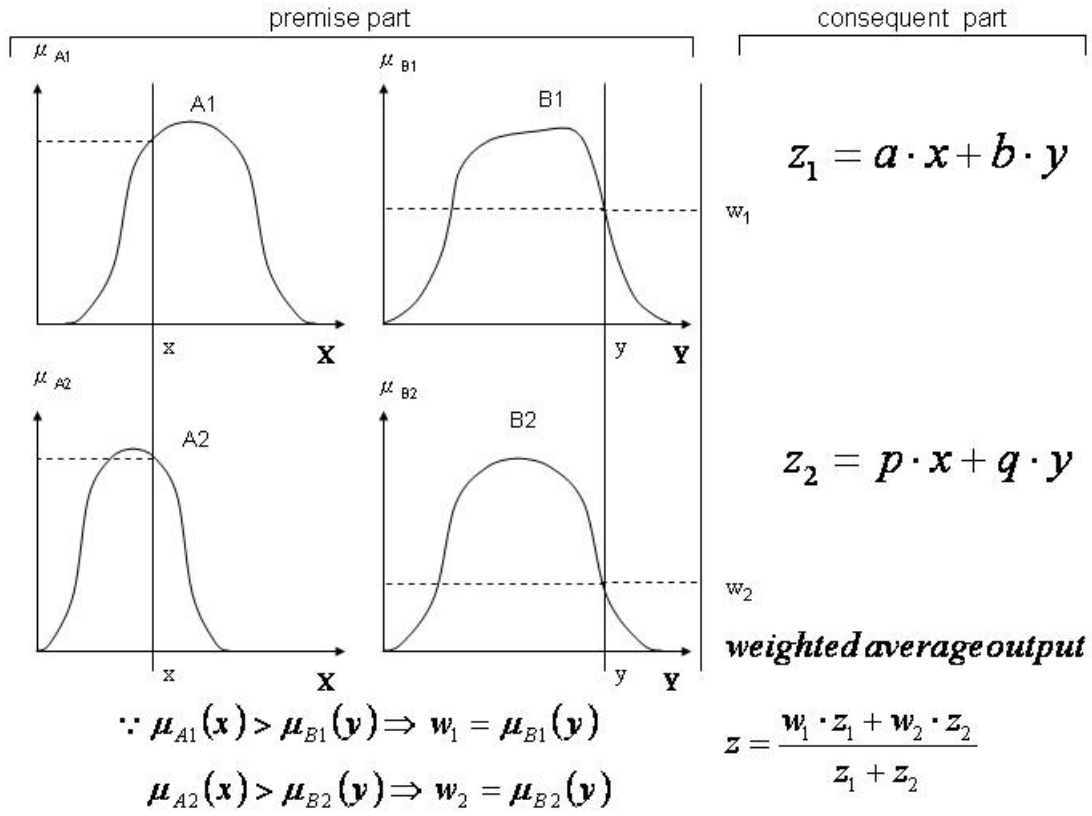


Figure 4.27 Type 3 - Fuzzy Reasoning Mechanisms [24]

4.3.5 Limitations in Fuzzy Inference Systems

There are some drawbacks in the fuzzy inference systems. For example, the rules of combining membership functions, such as the *minimum-maximum* (*min-max*) rule for conjunctive (AND) and disjunctive (OR) reasoning is not robust at all. Simply speaking, this *min-max* rule is definitely not the way to imitate human reasoning. Another disadvantage includes that fuzzy systems give the same importance to all factors that are to be combined. For example, it is possible that the role of soil depth or rock permeability is not of the same importance to soil erosion as the role of slope. Consequently, they

create some useless fuzzy rules. As such, effectively building the complete and robust fuzzy rules is a predestinate challenge.

In brief, neural networks and fuzzy inference systems are quite different, and each has unique capabilities that are useful in information processing. Yet, they often can be used to accomplish the same results in different ways. For instant, they can speed the unraveling and specifying the mathematical relationships among the numerous variables in a complex dynamic process. Both can be used to control nonlinear systems to a degree not possible with conventional linear control systems. Prevailingly, the unique advantages and capabilities of these two technologies can also be combined in a synergistic way, such as *Adaptive-based-Network Fuzzy Inference System* or *Adaptive Neuro-Fuzzy Inference System* (ANFIS) [24] is a very successful example.

In the following sections, the basic concepts, operations and structures of fuzzy inference systems are presented, followed by an introduction of ANFIS and its off-line training and on-line learning algorithms.

4.4 Adaptive-based -Networks Fuzzy Inference System (ANFIS)

4.4.1 Introduction

ANFIS is a class of *adaptive feedforward network* which is functionally equivalent to a fuzzy inference system and focuses on the learning capability of neural networks. That is, ANFIS uses the strategy of fuzzy systems to tune the neural network model in an adaptive manner. Thus all the design methodologies for neural network models become applicable to fuzzy inference models. The learning rule combines backpropagation and

the *Kalman Filter algorithm* which can speed up the learning process. Combining the advantages of neural networks and fuzzy inference systems, ANFIS can construct nonlinear process models based on consideration of the human-knowledge (represented as fuzzy IF-THEN rules, membership functions) with input-output data pairs for neural network training.

4.4.2 Types of ANFIS

ANFIS can be classified into two main types based on the input space partitioning method: *grid* and *scatter* partition [24]. Grid partition (GP) generates rules by enumerating all possible combinations of membership functions of all input parameters. This leads to an exponential explosion even when the number of inputs is moderately large. For instance, for ANFIS with 7 inputs, each of which has 3 membership functions (or divided by 3 levels), the GP creates 3^7 (=2187) rules. Conversely, scatter partition (also known as subtractive clustering (SC)) generates minimal number of rules equal to the same number of membership functions (or levels) for each input. As previously mentioned, these two partitioning methods are used in this thesis.

4.4.3 Adaptive Feedforward Networks

As stated in the beginning of Section 4.4.1, ANFIS is an adaptive feedforward network. An adaptive feedforward network is a network structure consisting of a number of nodes (or neurons) connected through directional links. Each node represents a process unit, and the links between nodes specify the causal relationship between the connected

nodes. All or parts of the nodes are “adaptive”, which means the outputs of these nodes depend on *modifiable parameters* pertaining to these nodes. Different from the *adjustable parameters*, or *weights*, used in previous neural networks, they are located between the nodes. Usually, a *node function* (or *neuron function*) is a parameterized function with modifiable parameters. By changing these parameters, we can change the node function as well as the overall behavior of the adaptive network.

These parameters of an adaptive network are distributed into its nodes, so each node has a local parameter set. In Figure 4.28, if a node’s parameter set is not empty, then its node function depends on the parameter values; a square shape is used to represent this kind of adaptive node. On the other hand, if a node has an empty parameter set, then its function is fixed; a circle is used to denote this type of fixed node.

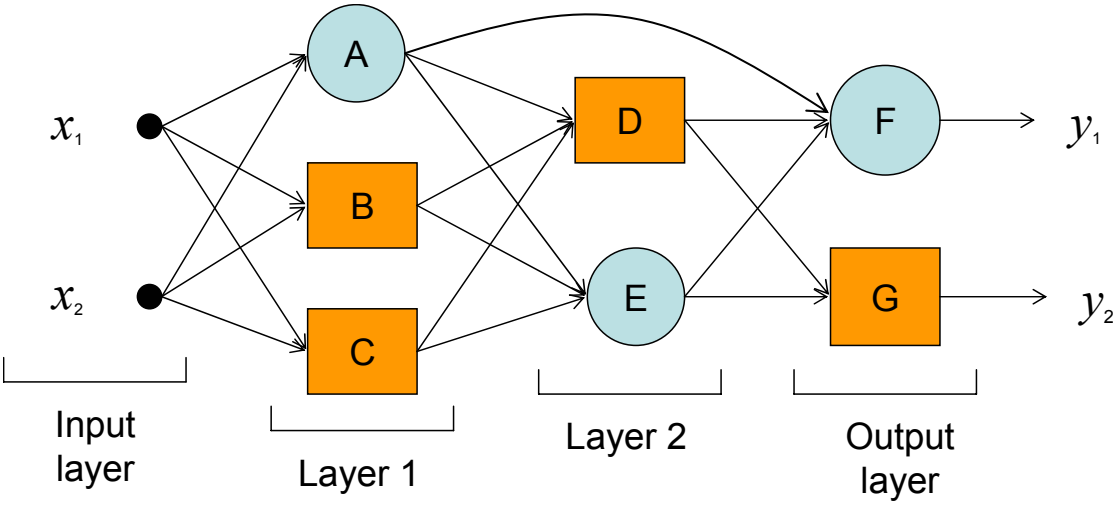


Figure 4.28 Feedforward Adaptive Network [4]

Conceptually, an adaptive feedforward network is actually a static mapping between its input and output spaces. This mapping may be either a simple linear relationship or a highly nonlinear one, depending on the network structure (arrangement

of nodes, connections, and so on) and the functionality for each node. The goal is to construct a network for achieving a desired nonlinear mapping that is regulated by a data set consisting of desired input-output pairs of a target system (also called training data set) to be modeled.

The procedures which are followed in adjusting the parameters to improve the network's performance are often referred to as the *learning rules*, *learning algorithms*, or *adaptation algorithms*. Usually, a network's performance is measured as the discrepancy between the desired output and the networks' output under the same input conditions. This discrepancy is called the *error measure* and it can be assumed in different forms for different applications. Generally, a learning rule is derived by applying a specific optimization technique to a given error measure.

4.4.4 Fuzzy Rules and ANFIS Architectures

Figures 4.29 and 4.30 show examples of ANFIS-GP and ANFIS-SC architectures [8]. Both have the same two inputs, each of which has three membership functions, and one output. ANFIS-GP model has nine fuzzy rules and ANFIS-SC model has three fuzzy rules. Tables 4.1 and 4.2 list the fuzzy, if-then, rules for these two examples. These rules, originally proposed by Takagi and Sugeno [96], have fuzzy sets involved only in the premise parts. The consequent parts are described by non-fuzzy equations of the input variables (e.g., electric current) as follows:

$$\textbf{If current is high, then power} = (\textbf{current})^2 \times (\textbf{resistance}) \tag{4.30}$$

where "high" in the premise part is a linguistic label characterized by appropriate

membership functions. The premise part of a rule delineates a fuzzy space, while the consequent part specifies the output within this fuzzy space. As summarized in Tables 4.1 and 4.2, the format of a generic fuzzy rule is given as

If x is A_i and y is B_i, then $f_i = p_i \cdot x + q_i \cdot y + r_i$

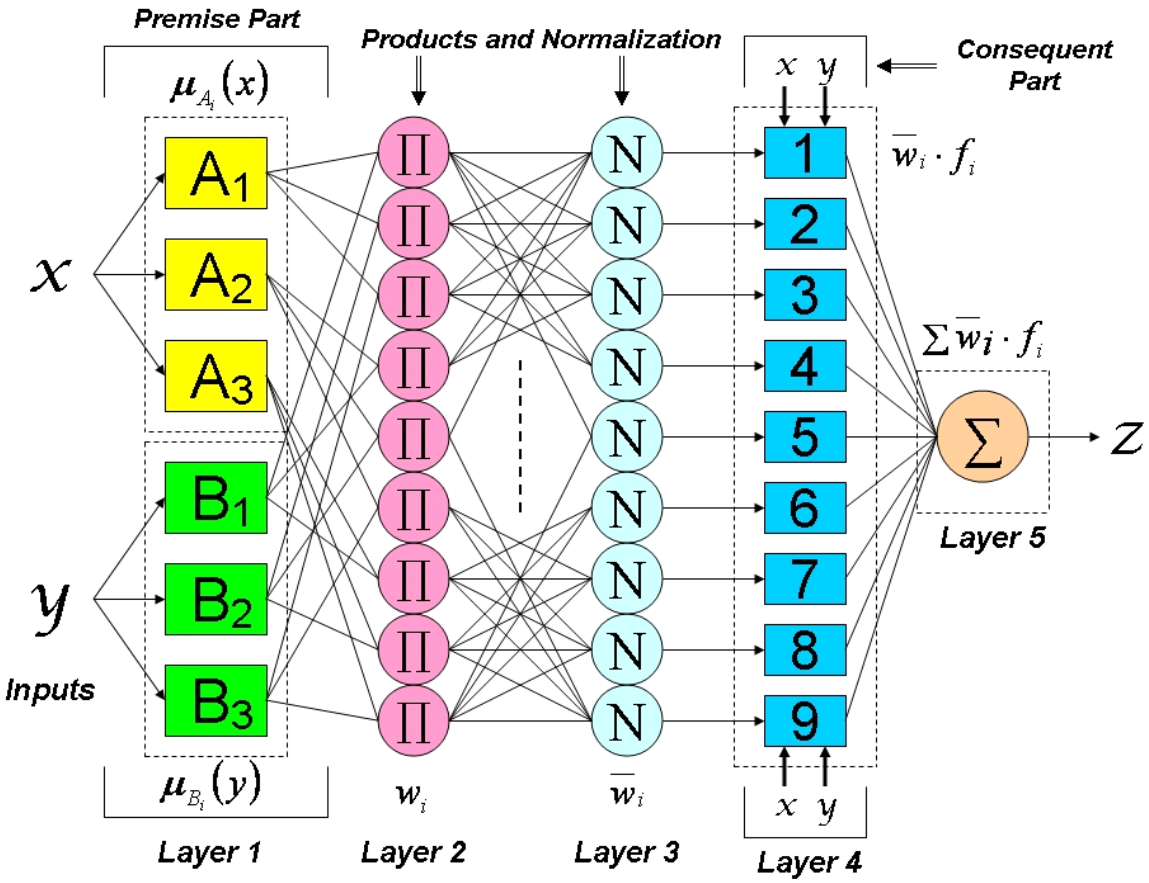


Figure 4.29 Architecture of ANFIS-GP with nine Fuzzy Rules [24]
(designed and redrawn by Lih)

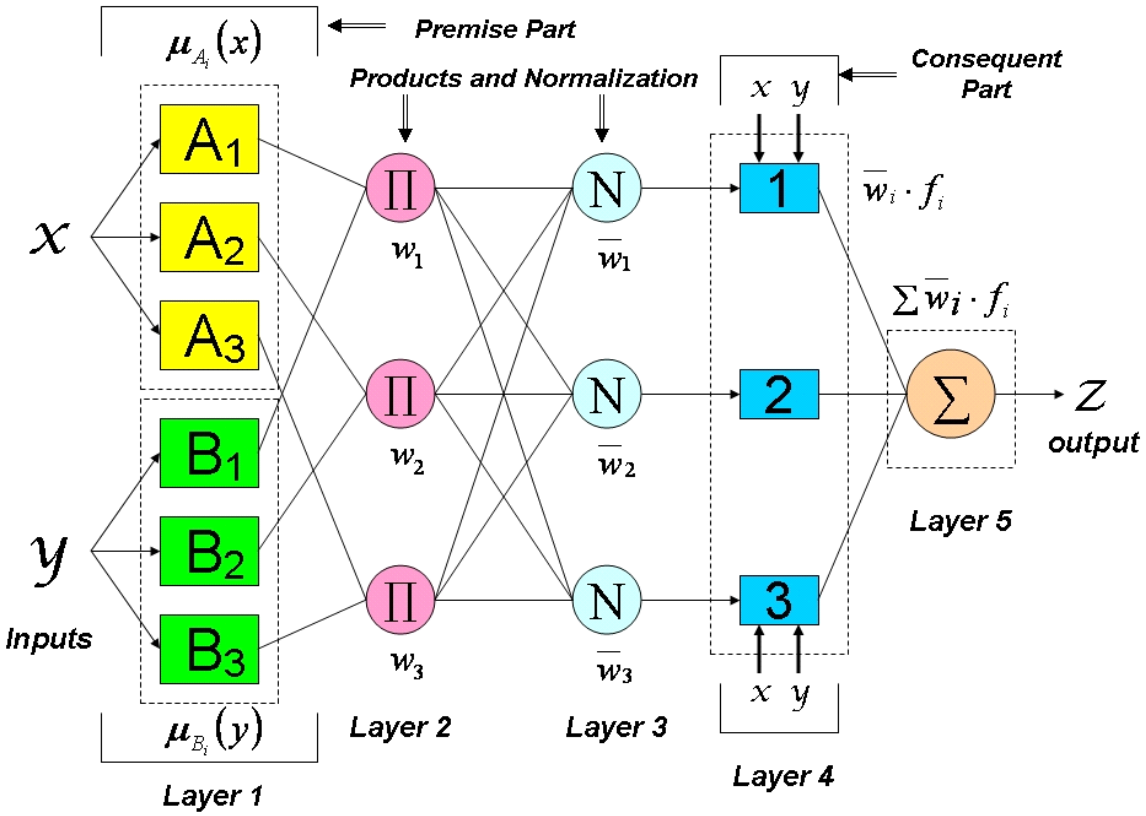


Figure 4.30 Architecture of ANFIS-SC with three Fuzzy Rules [24]
(designed and redrawn by Lih)

Table 4.1 Fuzzy, IF-THEN, rules of ANFIS

Grid Partition (GP) Model

Rules	If		Then
Rule #	x	y	f
1	A_1	B_1	$f_1=p_1x+q_1y+r_1$
2	A_1	B_2	$f_2=p_2x+q_2y+r_2$
3	A_1	B_3	$f_3=p_3x+q_3y+r_3$
4	A_2	B_1	$f_4=p_4x+q_4y+r_4$
5	A_2	B_2	$f_5=p_5x+q_5y+r_5$
6	A_2	B_3	$f_6=p_6x+q_6y+r_6$
7	A_3	B_1	$f_7=p_7x+q_7y+r_7$
8	A_3	B_2	$f_8=p_8x+q_8y+r_8$
9	A_3	B_3	$f_9=p_9x+q_9y+r_9$
* A_i and B_i are linguistic descriptions in premise part, characterized by relative membership functions			
** p_i, q_i and r_i are consequent constants decided during training process			

Table 4.2 Fuzzy, IF-THEN, Rules of ANFIS

Scatter Partition (SC) Model

Rules	If		Then
Rule #	x	y	f
1	A_1	B_1	$f_1=p_1x+q_1y+r_1$
2	A_2	B_2	$f_2=p_2x+q_2y+r_2$
3	A_3	B_3	$f_3=p_3x+q_3y+r_3$

4.4.5 ANFIS Transfer Functions

Five network layers are used in ANFIS to perform the following fuzzy inference steps: (1) fuzzification, (2) product, (3) normalization, (4) defuzzification, and (5) summation output. For simplicity, we take ANFIS-SC model (as illustrated in Figure 4.30) an example to explain the functions of each layer as described below:

Layer 1: Transferring input data to fuzzifized values through membership functions.

$$\begin{gathered} \mu_{A_i}, \mu_{B_i} \text{ are membership functions} \\ \text{input data } [x] \text{and } [y], i = 1,2,3 \\ [x] \Rightarrow [\mu_{A_i}(x)] \text{ and } [y] \Rightarrow [\mu_{B_i}(y)] \end{gathered} \tag{4.31}$$

Consider, for example, a bell-shaped membership function.

$$\mu_{A_i}(x) = \frac{1}{1 + \left[\left(\frac{x - c_i}{a_i} \right)^2 \right]^{b_i}} \tag{4.32}$$

where $[a_i, b_i, c_i]$ is the premise parameter set. As the values of these parameters are changed, the bell-shaped functions will vary accordingly.

Layer 2: Multiplying the incoming signals and sending the products out.

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), i = 1,2,3 \tag{4.33}$$

Each node output represents the firing strength of a rule.

Layer 3: Normalizing the firing strengths.

$$\overline{w}_i = \frac{w_i}{\sum_i w_i}, i = 1,2,3 \tag{4.34}$$

Layer 4: Defuzzification

$$\overline{w}_i \cdot f_i = \overline{w}_i \cdot (p_i \cdot x + q_i \cdot y + r_i), i = 1,2,3 \tag{4.35}$$

where (p_i, q_i, r_i) is the consequent parameter set, which is determined during the training process. If p_i and q_i are zeros (i.e. f_i = constant), this FIS is called zero-order-Sugeno model; alternatively, first-order-Sugeno model.

Layer 5: Summation of all incoming signals.

$$overall \quad output \quad = \sum_i \overline{w}_i \cdot f_i = \frac{\sum_i w_i \cdot f_i}{\sum_i w_i}, i = 1,2,3 \tag{4.36}$$

The output of each rule is a linear combination of input variables plus a constant term and the final output is the weighted average of each rule's output.

Similar to NN, ANFIS modeling starts by partitioning the data sets (input-output data pairs) into training and testing (or validating) data. The training data set is used to find the initial premise parameters for fuzzy membership functions by equally spacing each membership function. When the values of the premise parameters are fixed, the overall predicted MRR or WIWNU can be expressed as a linear combination of consequent parameters [24]

4.4.6 Adaptive Networks and Basic Learning Rule [8]

An ANFIS architecture is a typical adaptive network as illustrated in Figure 4.28 which is a multilayer feedforward network. As previously mentioned in Section 4.4.2, each node in an adaptive network performs a particular function (node function) on incoming signals as well as a set of parameters pertaining to this node. The formulae for the node functions may vary from node to node, and the choice of each node function depends on the overall input-output function which the adaptive network is required to carry out. Note that the links in an adaptive network only indicate the flow direction of signals between nodes; no weights are associated with the links.

To reflect different adaptive capabilities, we use both circle and square nodes in an adaptive network. A square node (adaptive node) has parameters while circle node (fixed node) has none. The parameter set of an adaptive network is the union of the parameter sets of each adaptive node. In order to achieve a desired input-output mapping, these parameters are updated according to a given training data and a gradient-based learning procedure is described in the following.

The primary training rule for an ANFIS is the error backpropagation algorithm based upon the gradient descent (steepest descent) theory which uses the same method that is used in neural networks. The essence of the error backpropagation algorithm is the evaluation of the contribution of each parameter in each layer to the output error measure (or energy function) function. However, the error back backpropagation algorithm suffers from slow convergence speed due to random initial conditions and the possibility of

trapping in local minima. Jang [97] proposed a hybrid learning method which combines error backpropagation and the least square estimate (LSE) to overcome the problems

Suppose that a given adaptive feedforward network in the layered representation has L layers and the l-th layer ($l = 0, 1, 2, \ldots, L; l = 0$ represents the input layer) has $N(l)$ nodes. Then the output and function of node i [i=1, …, $N(l)$] in layer l can be represented as $\boldsymbol{X}_i^l$ and f_i^l respectively, as shown in Figure 4.31. Since a node output depends on its incoming signals and its parameter set, we have

$$\boldsymbol{X}_i^l = \boldsymbol{f}_i^l\left(\boldsymbol{X}_1^{l-1}, \ldots \boldsymbol{X}_{N(l-1)}^{l-1}, \alpha, \beta, \gamma, \ldots\right) \tag{4.37}$$

where α, β, γ, etc. are the parameters pertaining to this node.

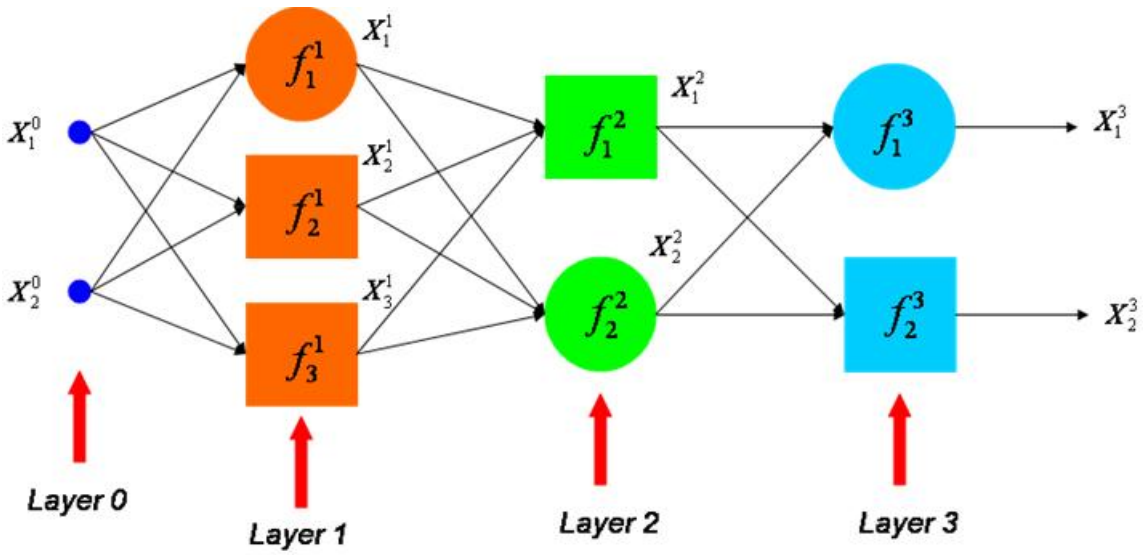


Figure 4.31 Layered Representation of Adaptive Network [8] (redrawn by Lih)

Assume that the given data set has P entries; then the error measure for the p-th ($1 \le p \le P$) entry of a training data set can be defined as the sum of squared errors:

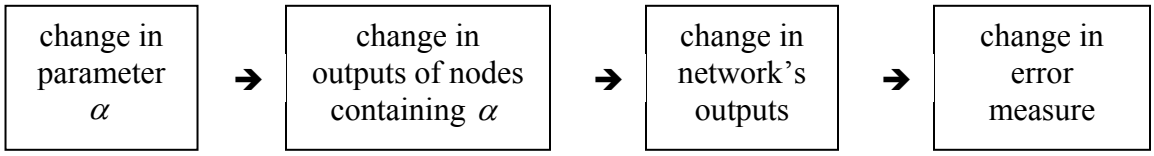
$$E_p = \sum_{m=1}^{N(L)} \left(T_{m,p} - O_{m,p}^{L} \right)^2 \tag{4.38}$$

where $T_{m,p}$ is the m-th component of p-th target output vector, and $O_{m,p}^{L}$ is the m-th component of actual output vector produced by the p-th input vector. Hence, the overall error measure is

$$E = \sum_{p=1}^{P} E_p \tag{4.39}$$

Actually, the definition of E_p in Equation (4.38) is not universal; other definitions of E_p are possible for specific situations or applications. In this thesis, Equation (4.38) is used.

To use the steepest descent method for minimizing the error measure, first we have to obtain the gradient vector. Before calculating the gradient vector, we should observe the following casual relationships:



where the arrows (➔) indicate the casual relationships. In other words, a small change in a parameter α will affect the output of the node containing α; this in turn will affect the output of the final layer and thus the error measure. Therefore, the basic concept in calculating the gradient vector is to pass a form of derivative information starting from the output layer and going to the backward layer by layer until the input layer is reached.

In order to develop a learning procedure that implements gradient descent in E over the parameter space, first the error signal (or error rate), $\varepsilon_{i,p}^{L}$, for a p-th training data with respect to the output of node i and layer l is calculated as:

$$\varepsilon_{i,p}^{l} = \frac{\partial^{+} E_p}{\partial O_{i,p}^{l}} \tag{4.40}$$

Werbos called this expression, the *ordered derivative* [88]. The difference between the ordered derivative and the ordinary partial derivative lies in the way the function is to be differentiated viewed. For an internal node output $O_{i,p}^{l}$ (where $l \neq L$), the partial derivative $\frac{\partial E_p}{\partial O_{i,p}^{l}}$ is equal to zero, since E_p does not depend on $O_{i,p}^{l}$ directly. However, it is obvious that E_p does depend on $O_{i,p}^{l}$ indirectly, since a change in $O_{i,p}^{l}$ will propagate through indirect paths to the output layer and thus produce a corresponding change in the value of E_p. Therefore, $\varepsilon_{i,p}^{l}$ can be viewed as the ratio of these two changes when they are made infinitesimal.

Consider here, a simple adaptive network as shown in Figure 4.32, where z is a function of x and y, and y is in turn a function of x:

$$\begin{cases} z = g(x, y) \\ y = f(x) \end{cases} \tag{4.41}$$

For the ordinary partial derivative $\frac{\partial z}{\partial x}$, assume that all the other input variables (in this case, y) are constant:

$$\frac{\partial z}{\partial x} = \frac{\partial g(x,y)}{\partial x} \tag{4.42}$$

In other words, we assume that the inputs x and y to the function g are independent, without paying attention to the fact that y is actually a function of x. For the ordered derivative, we take this indirect casual relationship into consideration:

$$\frac{\partial^+ z}{\partial x} = \frac{\partial g(x, f(x))}{\partial x} = \frac{\partial g(x,y)}{\partial x}\Big|_{y=f(x)} + \frac{\partial g(x,y)}{\partial y}\Big|_{y=f(x)} \frac{\partial f(x)}{\partial x} \tag{4.43}$$

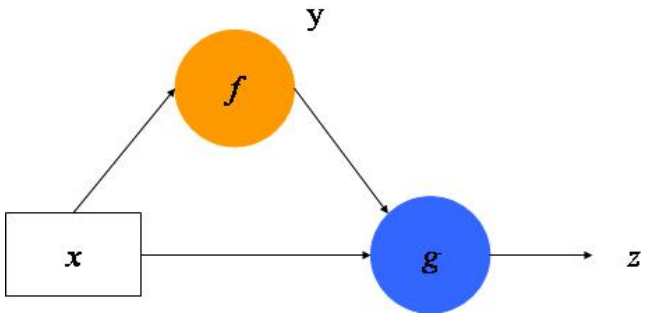


Figure 4.32 Simple Adaptive Network for Ordered Derivative and Ordinary Partial Derivative [8] (redrawn by Lih)

Therefore, the ordered derivative takes into consideration both the direct and indirect paths that lead to the casual relationship.

As the above derivation, the error signal for the i-th output node (at layer L) can be calculated directly:

$$\varepsilon_{i,p}^{L} = \frac{\partial^+ E_p}{\partial O_{i,p}^{L}} = \frac{\partial E_p}{\partial O_{i,p}^{L}} \tag{4.44}$$

Applying the definition in the Equation (4.20), we have:

$$\varepsilon_{i,p}^{L} = -2\left(T_{i,p} - O_{i,p}^{L}\right) \tag{4.45}$$

For the internal node at (l, i), the error signal can be derived by the chain rule:

$$\varepsilon_{i,p}^{l} = \underbrace{\frac{\partial E_p}{\partial O_{i,p}^{k}}}_{\substack{\text{error signal} \\ \text{at layer } l}} = \sum_{m=1}^{N(l+1)} \underbrace{\frac{\partial E_p}{\partial O_{m,p}^{l+1}}}_{\substack{\text{error signal} \\ \text{at layer } l+1}} \frac{\partial O_{m,p}^{l+1}}{\partial O_{i,p}^{l}} \tag{4.46}$$

where $0 \le l \le L-1$. That is, the error rate of an internal node can be expressed as a linear combination of the error rate of the nodes in the next layer. Therefore, for all l and i ($0 \le l \le L$ and $1 \le i \le N(l)$), we can find $\varepsilon_{i,p}^{l} = \partial^{+} E_p / \partial O_{i,p}^{l}$ by first applying Equation (4.44) once to get the error rates at the output layer, and then applying Equation (4.46) iteratively until we reach the desired layer l. The underlying procedure is called back-propagation, since the error rates are obtained sequentially from the output layer back to the input layer.

The gradient vector is defined as the derivative of the error measure with respect to each parameter. So we have to apply the chain rule again to find the gradient vector. Now, if α is an adjustable parameter of the i-th node at layer k, we have:

$$\frac{\partial^{+} E_p}{\partial \alpha} = \frac{\partial^{+} E_p}{\partial O_{i,p}^{l}} \frac{\partial O_{i,p}^{l}}{\partial \alpha} = \varepsilon_{i,p}^{l} \frac{\partial f_{i,p}^{l}}{\partial \alpha} \tag{4.47}$$

Note that if the parameter α is allowed to be shared between different nodes, then Equation (4.47) should be changed to a more general form:

$$\frac{\partial^{+} E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial^{+} E_p}{\partial O^*} \frac{\partial f^*}{\partial \alpha} \tag{4.48}$$

where S is the set of nodes containing α as a parameter; x^* and f^* are the output and function, respectively, of a generic node in S.

The derivative of the overall error measure E with respect to α is

$$\frac{\partial^+ E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial^+ E_p}{\partial \alpha} \tag{4.49}$$

Accordingly, for simple steepest descent without line minimization, the update formula for the generic parameter α is given by

$$\Delta\alpha = -\eta \frac{\partial^+ E}{\partial \alpha} \tag{4.50}$$

in which η is the learning rate. The effectiveness and convergence of the gradient descent method depend significantly on the value of the learning rate. In general, there is no signal learning rate suitable for different training cases. However, some common rules, such as those given in the following still apply: (1) when broad minima yield small gradient values, then a larger value of η will result in a more rapid convergence; (2) for problems with steep and narrow minima, a small value of η must be chosen to avoid overshooting the solution. This leads the learning rate that can be expressed as Equation (4.51) to satisfy the common rules.

$$\eta = \frac{\kappa}{\sqrt{\sum_{\alpha} \left(\frac{\partial E}{\partial \alpha} \right)^2}} \tag{4.51}$$

where κ is the step size, the length of each transition along the gradient direction in the parameter space. Usually, we can change the step size to vary the speed of convergence.

4.4.7 The Hybrid Learning Algorithm: Off-Line Learning [24]

As mentioned earlier, though the gradient method can be applied to identify the parameters in an adaptive network, the method is generally slow and likely to become

trapped in local minima. The following explains how the hybrid learning rule, proposed by Jang [97], overcomes the problems of implementation.

For simplicity, assume that the network has only one output

$$output = F\left(\vec{I}, \mathbf{S}\right) \tag{4.51}$$

where $\vec{I}$ is the input set and $\mathbf{S}$ is the parameter set. The parameter set S can be decomposed into two sets,

$$\mathbf{S} = \mathbf{S}_1 \oplus \mathbf{S}_2 \tag{4.52}$$

where $\oplus$ represents the direct sum, $\mathbf{S}_1$ is a nonlinear parameter set, and $\mathbf{S}_2$ is a linear parameter set. For given values of element of $\mathbf{S}_1$, we can incorporate the P training data into Equation (4.51) and obtain a matrix representation:

$$\mathbf{AX} = \mathbf{B} \tag{4.53}$$

where $\mathbf{X}$ is an unknown vector whose elements are parameters in $\mathbf{S}_2$. Let $|\mathbf{S}_2| = M$; then the dimensions of $\mathbf{A}$, $\mathbf{X}$, and $\mathbf{B}$ are $P \times M, M \times 1$, and $P \times 1$, respectively. Since P (number of training data set) is usually greater than M (number of linear parameters), we have an over-determined problem and generally there is no exact solution. Instead, a least squares estimate (LSE) of $\mathbf{X}$, $\mathbf{X}^*$, is sought to minimize the square error $\|\mathbf{AX} - \mathbf{B}\|^2$. The most well-known form of $\mathbf{X}^*$ uses the pseudo-inverse of $\mathbf{A}$:

$$\mathbf{X}^* = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{B} \tag{4.54}$$

where $\mathbf{A}^T$ is the transpose of $\mathbf{A}$, and $\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T$ is the pseudo-inverse of $\mathbf{A}$, if $\left(\mathbf{A}^T\mathbf{A}\right)$ is nonsingular. Equation (4.54) is computationally intensive because of the matrix inverse

operation; moreover, it is ill-defined if $\left(\mathbf{A}^T\mathbf{A}\right)$ is singular. As a result, recursive formulas are employed to compute the LSE of **X**. A recursive method is more efficient (especially when *M* is small) and can be easily modified for an on-line version for systems with changing characteristics. Specifically, let the *i*-th row vector of matrix **A** be $\mathbf{a}_i^T$ and the *i*-th element of **B** be $\mathbf{b}_i^T$; then **X** can be calculated iteratively using the recursive formulas:

$$\left.\begin{aligned}\mathbf{X}_{i-1} &= \mathbf{X}_i + \mathbf{S}_{i-1}\mathbf{a}_{i-1}\left(\mathbf{b}_{i-1}^T - \mathbf{a}_{i-1}^T\mathbf{X}_i\right)\\ \mathbf{S}_{i-1} &= \mathbf{S}_i - \frac{\mathbf{S}_i\mathbf{a}_{i-1}\mathbf{a}_{i-1}^T\mathbf{S}_i}{1+\mathbf{a}_{i-1}^T\mathbf{S}_i\mathbf{a}_{i-1}}, i = 0,1,\ldots,P-1\end{aligned}\right\} \tag{4.55}$$

where $\mathbf{S}_i$ is called the *covariance matrix* and the least squares estimate $\mathbf{X}^*$ is equal to $\mathbf{X}_P$. The initial condition of Equation (4.55) is $\mathbf{X}_0 = 0$ and $\mathbf{S}_0 = \gamma\,\mathbf{I}$, where γ is a positive large number and ***I*** is the identity matrix of dimension $M \times M$. When dealing with multi-output adaptive networks (the output in Equation (4.53) is a column vector). Equation (4.55) can still be applied except that $\mathbf{b}_i^T$ is the *i*-th row of matrix **B**.

The recursive least squares estimate of **X** can be interpreted as a *Kalman Filter* for the process

$$\begin{aligned}\mathbf{X}(k+1) &= \mathbf{X}(k)\\ \mathbf{Y}(k+1) &= \mathbf{A}(k)\mathbf{X}(k) + noise\end{aligned} \tag{4.56}$$

where $\mathbf{X}(k) = \mathbf{X}_k$, $\mathbf{Y}(k) = \mathbf{b}_k$, and $\mathbf{A}(k) = \mathbf{a}_k$. For this reason, Equation (4.55) is sometimes loosely referred to as the *Kalman Filter algorithm*.

Now the gradient descent method and the least squares estimate can be combined to update the parameters in an adaptive network. Each epoch of this hybrid learning

procedure is composed of a forward pass and a backward pass. In the forward pass, input data is supplied and functional signals go forward to calculate each node output until the matrices **A** and **B** in Equation (4.53) are obtained. The parameters in $\mathbf{S}_2$ are identified by the recursive least squares formula in Equation (4.55). After identifying the parameters in $\mathbf{S}_2$, the functional signals are generated in a forward direction until the complete error measure is calculated. In the backward pass, the error signals (or error rate, the derivative of the error measure with respect to each node output) propagate from the output end toward the input end and the parameters in $\mathbf{S}_1$ are updated by the gradient descent of Equation (4.48).

Not only can this hybrid learning algorithm decrease the dimension of the search space in the gradient method, but also, in general, cut down substantially the convergence time.

4.4.8 The Hybrid Learning Algorithm: On-Line Learning

The learning paradigm is vital for the on-line parameter identification for systems with changing or unknown characteristics. To modify the batch learning rule to its on-line version, the gradient descent method should be based on E_p, instead of E. Strictly speaking, this is not a truly gradient search procedure that minimizes E; yet the approach will approximate this minimization if the learning rate is small.

For the recursive formulas, the effects of old data pairs need to be reduced as new data pairs become available. One simple method for doing this is to add a *forgetting factor* λ to the original recursive formulas:

$$\left.\begin{aligned}\mathbf{X}_{i-1} &= \mathbf{X}_i + \mathbf{S}_{i-1}\mathbf{a}_{i-1}\left(\mathbf{b}_{i-1}^T - \mathbf{a}_{i-1}^T\mathbf{X}_i\right) \\ \mathbf{S}_{i-1} &= \frac{1}{\lambda}\left(\mathbf{S}_i - \frac{\mathbf{S}_i\mathbf{a}_{i-1}\mathbf{a}_{i-1}^T\mathbf{S}_i}{1+\mathbf{a}_{i-1}^T\mathbf{S}_i\mathbf{a}_{i-1}}\right), i = 0,1,\ldots,P-1\end{aligned}\right\} \quad (4.57)$$

where the value of λ is between 0 and 1. The smaller the value of λ is, the faster the effect of old data decay. But a value of λ that is too small may sometimes cause numerical instability and thus should be avoided [24].

4.4.9 Summary

A fuzzy inference system is composed of five functional blocks – rule base, database, decision-making unit, fuzzification interface, and defuzzification interface. Fuzzy inference systems are generally classified into three types according to the types of fuzzy reasoning and fuzzy IF-THEN rules employed. Figures 4.25 to 4.27 show the commonly used fuzzy IF-THEN rules and fuzzy reasoning mechanisms.

The ANFIS architecture consists of five layers as shown in Figures 4.29 and 4.30. The layer functions are described by Equations (4.30) to (4.31). In order to use the ANFIS for modeling, a set of data is first collected for off-line training. The off-line training procedure is composed of a forward pass and a backward pass. In the forward pass, the linear parameters are identified by the recursive least squares estimate while the nonlinear parameters are fixed. In the backward pass, the linear parameters are fixed and the nonlinear parameters are obtained by the gradient descent method. Then a new set of data is collected on-line for testing the ANFIS model using on-line learning. For the on-

line learning algorithm, a forgetting factor is added to the original recursive formulas and the gradient descent method is used based on the error measure at each epoch.

4.5 Fundamentals of Evolutionary Algorithms (EA)

4.5.1 Introduction

Different schools of evolutionary algorithms have evolved during the last 30 years: *genetic algorithms* (GA), mainly developed in the USA by Holland [98], *evolutionary strategies* (ES), developed in Germany by Rechenberg [99] and Schwefel [100], and *evolutionary programming* (EP) by Fogel *et al*.[101]. Each of these constitutes a different approach; however, they are inspired by the same principles of natural evolution.

More clearly, the EA are a stochastic global search method that mimics the metaphor of natural biological evolution. EA operate on a *population* of potential solutions applying the principle of *survival of the fittest* to produce (hopefully) better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

EA model natural processes, such as *selection*, *recombination*, *mutation*, *reinsertion*, *migration*, *locality*, and *neighborhood*. Figure 4.33 shows the structure of a simple

evolutionary algorithm. EA work on populations of individuals instead of single solutions. In this way the search is performed in a parallel manner.

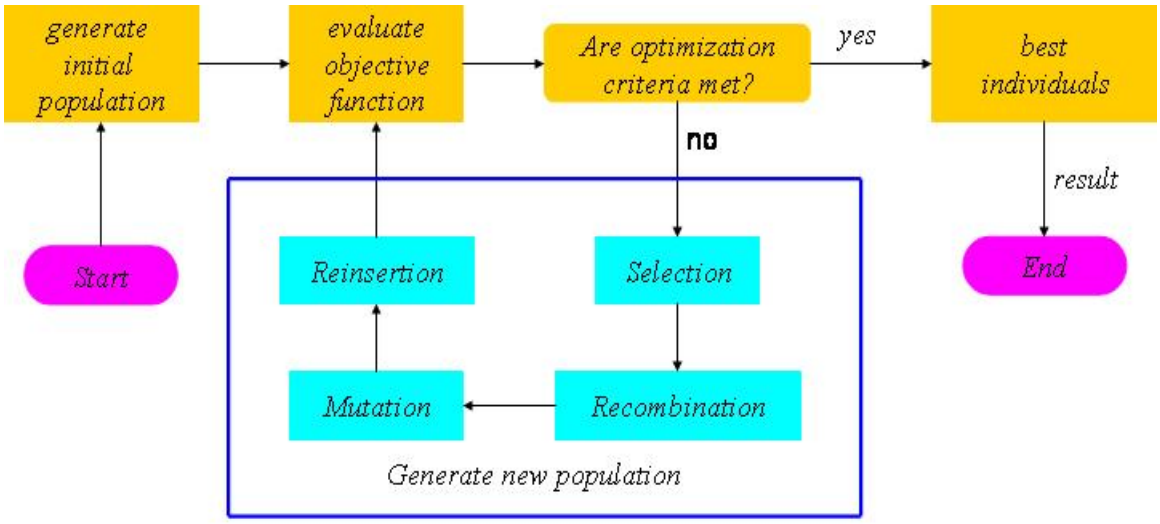


Figure 4.33 Structure of a Single Population Evolutionary Algorithm [1]
(redrawn by Lih)

At the beginning of the computation a number of individuals (the population) are randomly initialized. The objective function is then evaluated for these individuals. The first generation is produced. If the optimization criteria are not met by the creation of a new generation, a new cycle will keep starting as follows. Individuals are selected according to their *fitness* for the production of offspring. Parents are recombined to produce offspring. All offsprings will be mutated with a certain probability. The fitness of the offspring is then computed. The offspring are inserted into the population replacing the parents, producing a new generation. This cycle is performed until optimization criteria are reached.

A single population evolutionary algorithm is powerful and performs well on a

wide variety of problems. However, better results can be obtained by introducing multiple subpopulations. Every subpopulation evolves over a few generations isolated (like the single population evolutionary algorithms) before one or more individuals are exchanged between the subpopulation. The multi-population evolutionary algorithm models the evolution of a species in a way more similar to nature than the single population evolutionary algorithm. Figure 4.34 shows the structure of such an extended multi-population evolutionary algorithm.

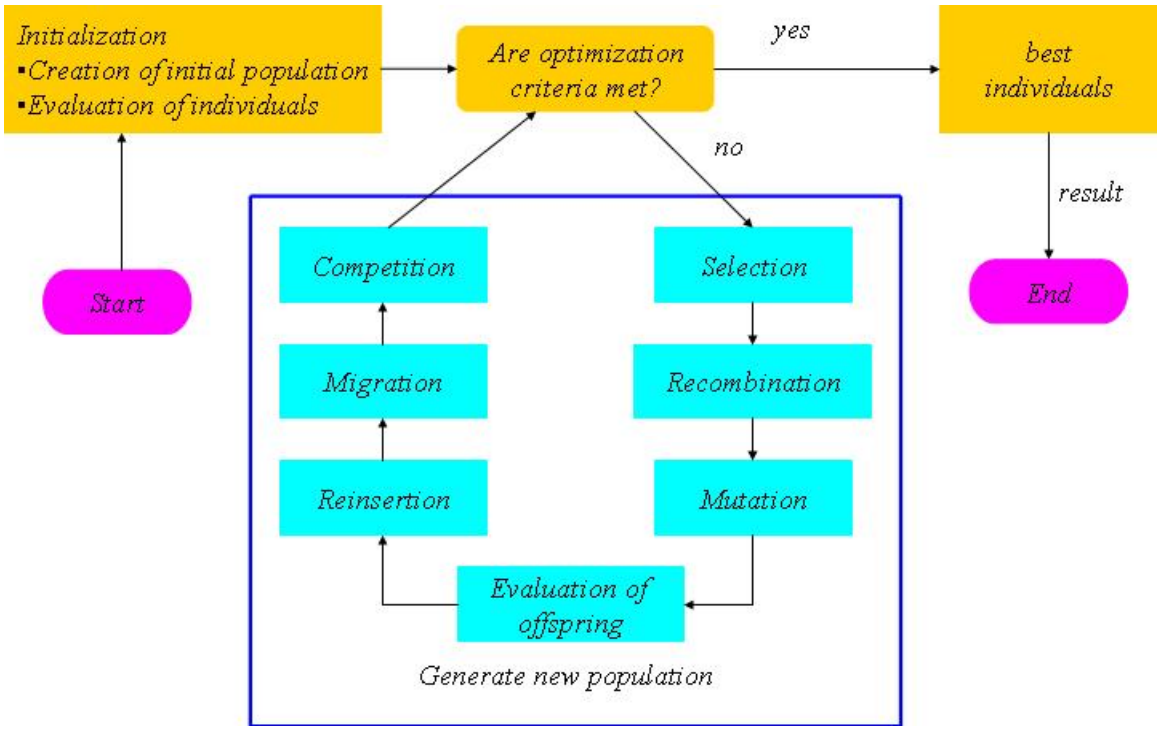


Figure 4.34 Structure of an Extend Multi-population Evolutionary Algorithm [1] (redrawn by Lih)

From the above discussion, it can be seen that evolutionary algorithms differ substantially from more traditional search and optimization methods. The most significant differences are:

- Evolutionary algorithms search a population of points in parallel, not just a single

point.

- Evolutionary algorithms do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.
- Evolutionary algorithms use probabilistic transition rules, not deterministic rules.
- Evolutionary algorithms are generally more straightforward to apply because no restrictions for the definition of the objective function exist.
- Evolutionary algorithms can provide a number of potential solutions to a given problem. The final choice is left to the user.

Thus, in cases where the particular problem does not have one individual solution, for example, a family of *Pareto-optimal* solutions, as in the case of *multi-objective optimization* and *scheduling problems*, then the evolutionary algorithms are potentially useful for identifying these alternative solutions simultaneously, as stated in the following Section 4.6. Figure 4.35 shows the problem solution using evolutionary algorithms.

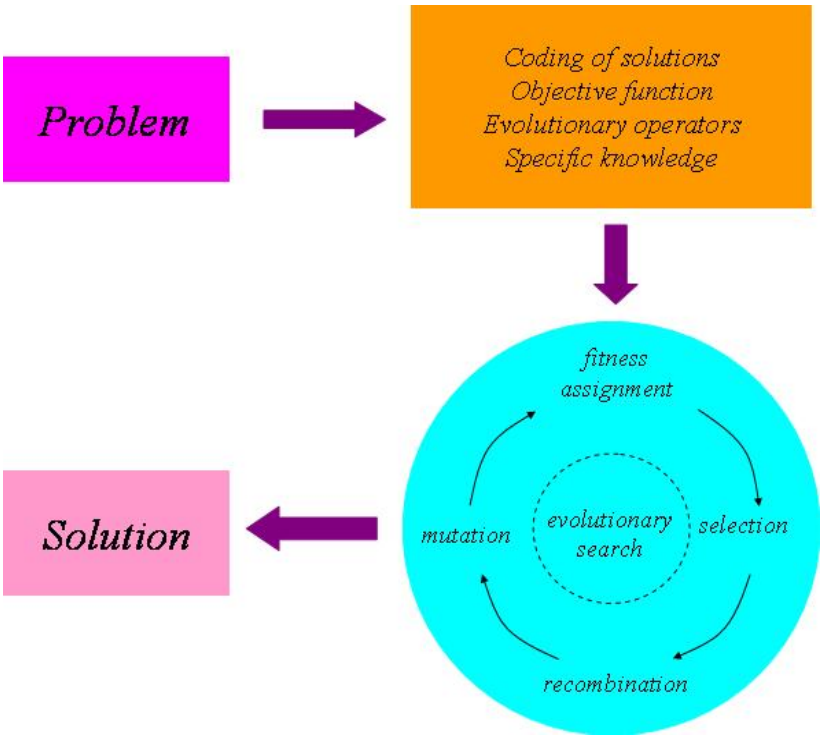


Figure 4.35 Problem Solution using Evolutionary Algorithms [1] (redrawn by Lih)

In this research, genetic algorithms are used to model the CMP process as the neural networks and ANFIS, and also to search the best input settings for obtaining the optimal output performances i.e., higher MRR and lower WIWNU.

4.5.2 Genetic Algorithms (GA)

As the name suggests, genetic algorithms (GA) borrow their working principle from natural genetics. According to the classification in the previous subsection, GA belong to EA. Clearly, GA search and optimize the procedures that are motivated by the principles of *natural genetics* and *natural selection*. GA use operators that are analogous to the *evolutionary processes of mating* (or "*crossover*" at the gene level), *mutation* and *natural selection* to explore multi-dimensional parameter spaces. In principal, GA can be applied

to any problem where the variables to be optimized ("genes") can be encoded to form a string ("chromosome") - as shown in Figure 4.36.

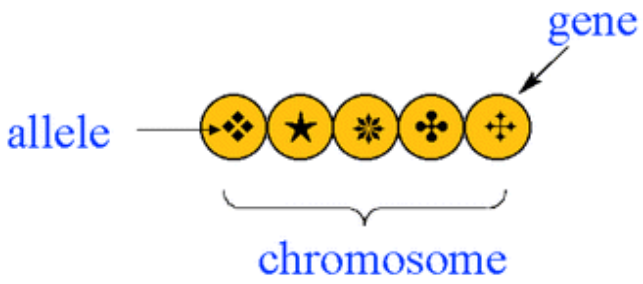


Figure 4.36 Gene, Allele and Chromosome in GA [3]

Each string represents a trial solution of the problem. By analogy with biology, the values of the individual variables are known as "alleles". The working principles of GA are very different from that of most classical optimization techniques. Over the last decade, GA have been extensively used as search and optimization tools in various problem domains, including sciences, commerce, and engineering. The primary reasons for their success are their broad applicability, ease of use, and global perspective [5].

As stated above, GA are search algorithms for optimization on the basis of the mechanics of natural selection and genetics. The power of these algorithms is derived from a very simple heuristic assumption that the best solution will be found in the regions of solution space containing high proposition of good solutions, and that these regions can be identified by judicious and robust sampling of the solution space.

4.5.2.1 Genotypes (Chromosome Values) and Phenotypes (Decision Variables)

Individuals, or current approximations are encoded as *strings*, *chromosomes*,

composed over some alphabet(s), so that the *genotypes* (chromosome values) are uniquely mapped onto the decision variable (*phenotypic*) domain as depicted in Figure 4.37.

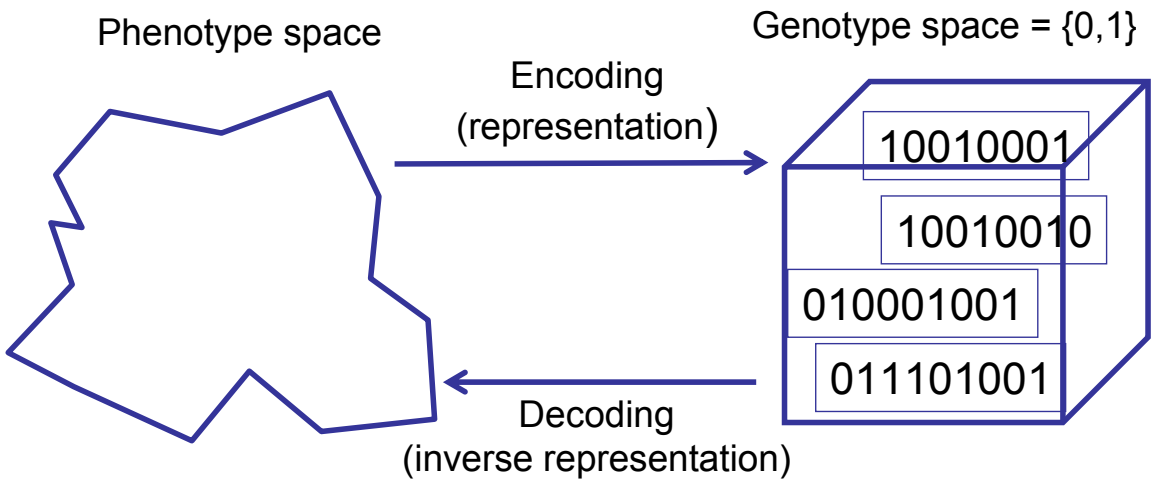
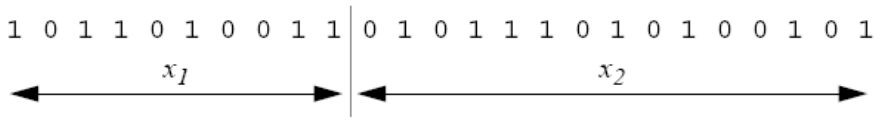


Figure 4.37 Phenotype space and genotype space [3] (redrawn by Lih)

The most commonly used representation in GA is the binary alphabet [0, 1], although other representations can be used, e.g. *ternary*, *integer*, *real-valued* etc. For example, a problem with two variables, x_1 and x_2, may be mapped onto the chromosome structure in the following way:



where x_1 is encoded with 10 bits and x_2 with 15 bits, possibly reflecting the level of accuracy or range of the individual decision variables. Examining the chromosome string in isolation yields no information about the problem we are trying to solve. It is only with the decoding of the chromosome into its phenotypic values that any meaning can be applied to the representation. However, as described below, the search process will

operate on this encoding of the decision variables, rather than the decision variables themselves, except, of course, where *real-valued genes* are used.

Basically, the mechanics of GA are simple, involving copying of binary strings and the swapping of the binary strings (or chromosomes). The simplicity of operation and computational efficiency are the two main attractions of the GA approach. The computations are carried out in three stages to get a result in one generation or iteration. The three stages are (1) *Reproduction* (or selection operator), (2) *Recombination* (or Crossover), (3) *Mutation*. These will be briefly discussed in the following.

4.5.2.2 Reproduction and Fitness

Reproduction (or selection operator) is the first procedure of the genetic operators. It is a process in which some of the chromosomes are copied into a separate place called the *mating pool*, in proportional to their *fitness values*. Having decoded the chromosome representation into the decision variable domain, it is possible to assess the performance, or *fitness*, of individual members of a population. This is done through an *objective function* that characterizes an individual's performance in the problem domain. In the natural world, this would be an individual's ability to survive in its present environment. Thus, the objective function establishes the basis for selection of pairs of individuals that will be mated together during *reproduction*.

During the reproduction phase, each individual is assigned a *fitness value* derived from its raw performance measure given by the objective function. This value is used in the selection to bias towards more fit individuals. Highly fit individuals, relative to the

whole population, have a high probability of being selected for *mating*, whereas less fit individuals have a correspondingly low probability of being selected. This implies that strings with higher fitness values will have a higher probability of contributing more strings as the search progresses.

4.5.2.3 Recombination (or Crossover)

This operator, second among the genetic operators, is mostly responsible for the progress of the search. Once the individuals have been assigned a fitness value, they can be chosen from the population, with a probability according to their relative fitness, and recombined to produce the next generation. Genetic operators manipulate the characters (genes) of the chromosomes directly, using the assumption that certain individual's gene codes, on average, produce fitter individuals. The *recombination operator* is used to exchange genetic information between pairs, or larger groups, of individuals. In other words, it may swap the parent chromosomes partially, causing offspring to be generated. In this, a crossover site along the length of the chromosome is selected randomly, and the portions of the chromosomes beyond the crossover site are swapped. The simplest recombination operator is a *single-point crossover*.

Consider the two parent binary strings:

$$P_1 = 1\ 0\ 0\ 1\ 0\ \mathbf{1}\ \mathbf{1}\ \mathbf{0}$$

$$P_2 = 1\ 0\ 1\ 1\ 1\ \mathbf{0}\ \mathbf{0}\ \mathbf{0}$$

If an integer position, i, is selected uniformly at random between 1 and the string length, l, minus one [1, l-1], and the genetic information exchanged between the individuals about

this point, then two new offspring strings are produced. The two offsprings below are produced when the crossover point $i = 5$ is selected,

$$O_1 = 1\ 0\ 0\ 1\ 0\ \mathbf{0}\ \mathbf{0}\ \mathbf{0}$$

$$O_2 = 1\ 0\ 1\ 1\ 1\ \mathbf{1}\ \mathbf{1}\ \mathbf{0}$$

This crossover operation is not necessarily performed on all strings in the population. Instead, it is applied with a probability, *Pc*, when the pairs are chosen for breeding.

4.5.2.4 Mutation

A further genetic operator, called *mutation*, is then applied to the new chromosomes, again with a set probability, *Pm*. It is usually the last operator in GA, and this is the occasional random alteration (with small probability) of the value of a chromosome position. Mutation causes the individual genetic representation to be changed according to some probabilistic rule. In the binary string representation, mutation will cause a single bit to change its state from 1 to 0, or vice versa. So, for example, mutating the fourth bit of O_1 leads to the new string,

$$O_{1m} = 1\ 0\ 0\ \mathbf{0}\ 0\ 0\ 0\ 0$$

Mutation is generally considered to be a background operator that ensures that the probability of searching a particular subspace of the problem space is never zero. This has the effect of tending to inhibit the possibility of converging to a *local optimum*, rather than the *global optimum*.

After recombination and mutation, the individual strings are then, if necessary, decoded, the objective function evaluated, a fitness value assigned to each individual and individuals selected for mating according to their fitness, and so the process continues

through subsequent generations. In this way, the average performance of individuals in a population is expected to increase, as good individuals are preserved and bred with one another and the less fit individuals die out. The operations of GA are terminated when some criteria are satisfied, e.g., a certain number of generations, a mean deviation in the population, or when a particular point in the search space is encountered.

4.5.3 Reproduction

The primary objective of the reproduction operator is to make duplicates of good solutions and eliminate bad solutions in a population, while keeping the population size constant. This is achieved by performing the following tasks:

- Identify good (usually above-average) solutions in a population.
- Make multiple copies of good populations.
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population.

There exists a number of ways to achieve the above tasks. Some common methods include *tournament selection, proportionate selection*, *ranking selection* and *Elitism* [21].

4.5.3.1 Tournament Selection

In the tournament selection, tournaments are played between two solutions and the better one is chosen and placed in the mating pool. Two other solutions are picked again and another slot in the mating pool is filled with a better solution. If carried out systematically, each solution can be made to participate in exactly two tournaments. The

best solution in a population will win both times, thereby making two copies of it in the new population. Using a similar argument, the worst solution will lose in both tournaments and will be eliminated from the population. In this way, any solution in a population will have zero, one, or two copies in the new population.

4.5.3.2 Proportionate Selection

In the proportionate selection method, solutions are assigned copies, the number of which is proportional to their fitness values. The average fitness of all population members is f_{avg}, a solution with a fitness f_i gets an expected f_i / f_{avg} number of copies. The implementation of this selection operator can be thought of as *roulette-wheel mechanism*, where the wheel is divided into *N* (population size) divisions, where the size of each is marked in proportion to the fitness of each population member. Thereafter the wheel is spun *N* times, each time choosing the solution indicated by the pointer, as shown in Figure 4.38. This figure shows a roulette wheel for six individuals having different fitness values. Since the fifth individual has a higher fitness value (or larger division area) than any other, it is expected that the *roulette wheel selection* (RWS) will choose the fifth solution more often than any other solution.

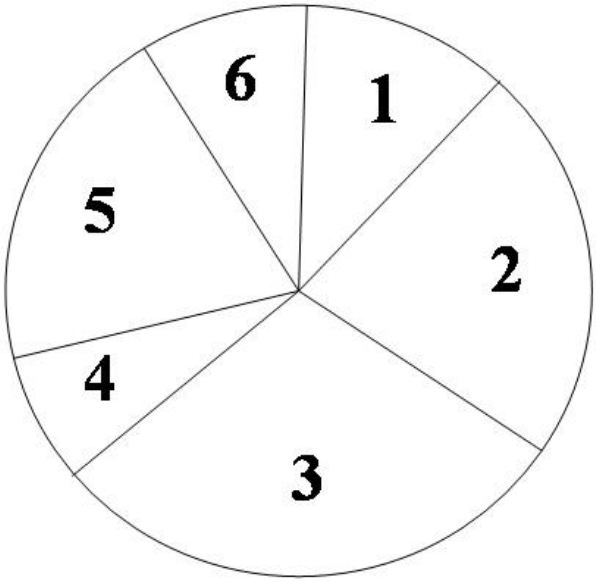
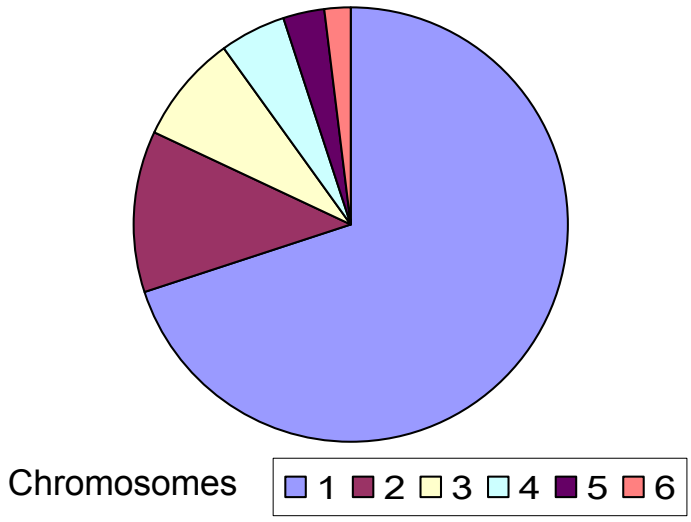


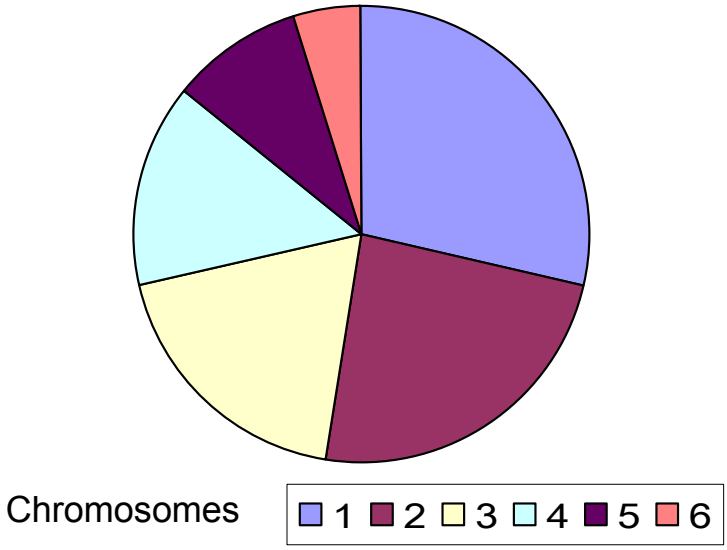
Figure 4.38 Roulette-wheel Selection for GA [5]

4.5.3.3 Ranking Selection

The previous type of selection will have problems when there are large differences between the fitness values. For example, if the best chromosome fitness is 90% of the sum of all fitnesses then the other chromosomes will have very few chances to be selected. Rank selection ranks the population first and then every chromosome receives fitness value determined by this ranking. The worst will have the fitness ***1***, the second worst ***2*** etc. and the best will have fitness ***N*** (number of chromosomes in population). Each member in the sorted list is assigned a fitness value equals to the rank of the solution in the list. Thereafter, the proportionate selection operator as previously stated is applied with the ranked fitness values, and *N* solutions are chosen for the mating pool. Figure 4.39 shows how the situation changes after changing the fitness to the numbers determined by the ranking.



(a) Situation before Ranking (Graph of Fitness)



(b) Situation after ranking (Graph of Order Number)

Figure 4.39 Ranking Selection for GA

Now all the chromosomes have a chance to be selected. However, this method can lead to slower convergence because the best chromosomes do not differ so much from the other ones.

4.5.3.4 Elitism

As the name suggests, the elitism is to use an elite-preserving operator which favors the elites of a population to give them an opportunity to be directly carried over to the next generation. When creating a new population by crossover and mutation, there exists a big chance, that the best chromosome might be lost. Elitism is the name of the method that first copies the best chromosome (or few best chromosomes) to the new population. The rest of the population is constructed in ways described above. Elitism can rapidly increase the performance of GA, because it prevents a loss of the best found solution.

4.5.4 Binary-coded and Real-parameter Recombination (or Crossover)

In 1975, Holland [98] proposed the binary-coded representation (such as stings 01111010 and 10000111) in the very first application in GA and still remains popular to the present. Until 1989, Antonisse [102] demonstrated the real-parameter representation for GA. Real-parameter (or real-valued encoding) GA open a new attractive way to real world problems because it eliminates the computational penalties incurred by translating test solutions into and out of the binary format. In this research, real-valued encoding was applied to the GA for modeling and optimizing the CMP process.

When binary-coded GA needs to be used to handle problems having a continuous search space, a number of difficulties arise. For example, one of the difficulties is the inability to achieve any arbitrary precision in the optimal solution. In binary-coded GA, the string length must be chosen a priori to enable it to achieve a certain precision in the solution. More the required precision, the longer is the string length. For longer strings, the population size requirement should be also large, thereby increasing the computational complexity of the algorithm. Since a fixed coding scheme is used to code the decision variables, variable bounds must be such that they bracket the optimum variable values. Since in many problems this information is not usually known a priori, this may cause some difficulty in using binary-coded GA in such problems.

Since real parameters are used directly (without any string coding), solving real-parameter optimization problems is step easier when compared to the binary-coded GA. Unlike in the binary-coded GA, decision variables can be directly used to compute the *fitness values*. Since the *selection operator* works with the fitness values, any selection operator used with the binary-coded GA can also be used in real-parameter GA. However, the difficulty arises with the *search operators* (i.e. crossover and mutation procedures) for real-parameter GA.

A recombination (or crossover) operator is applied next to the strings of the mating pool. The reproduction operator cannot create any new solution in the population. It only makes more copies of good solutions at the expense of not-so-good solutions. The creation of new solutions is performed by crossover and mutation operators. Like the reproduction operator, there exists a number of crossover operators in the GA literature

[103], but in almost all crossover operators, two strings are picked from the mating pool at random and some portion of the strings are exchanged between the strings to create two new strings.

4.5.4.1 Binary-coded Crossover

As an example in Section 4.5.1, a single-point crossover operator is shown in Figure 4.40. This is performed by randomly choosing a crossover site along the string and by exchanging all bits on the right side of the crossing site

Single-point Binary Crossover

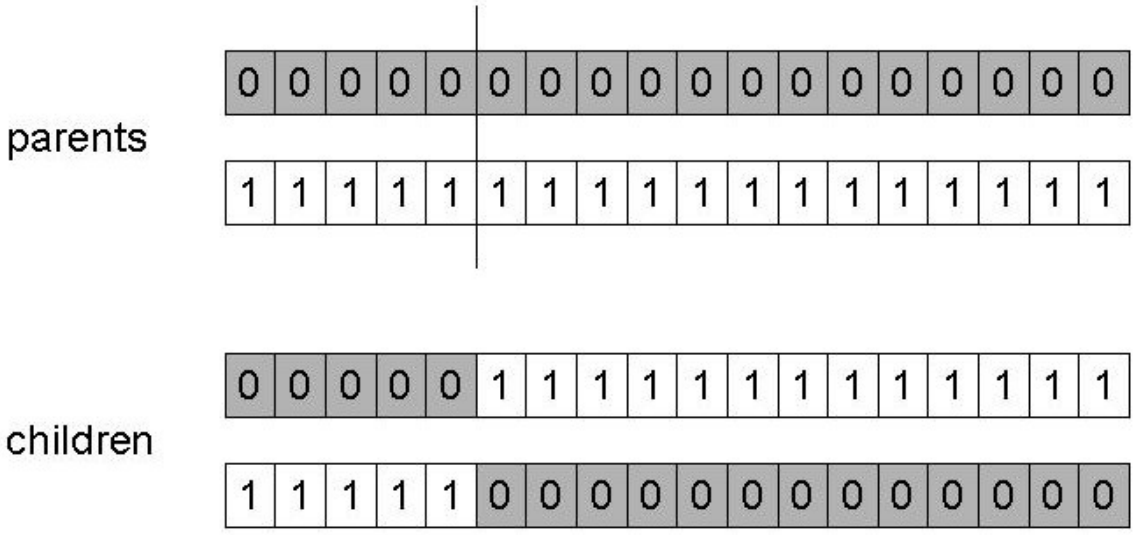


Figure 4.40 Single-point Binary Crossover Operator

Multi-point Binary Crossover

Figure 4.41 shows an example of *n*-point crossover mechanisms in which *n* random crossover points are chosen and the parents' chromosomes are split along these points.

Thereafter, the split parts are alternated between parents and then glued together to form two new children chromosomes.

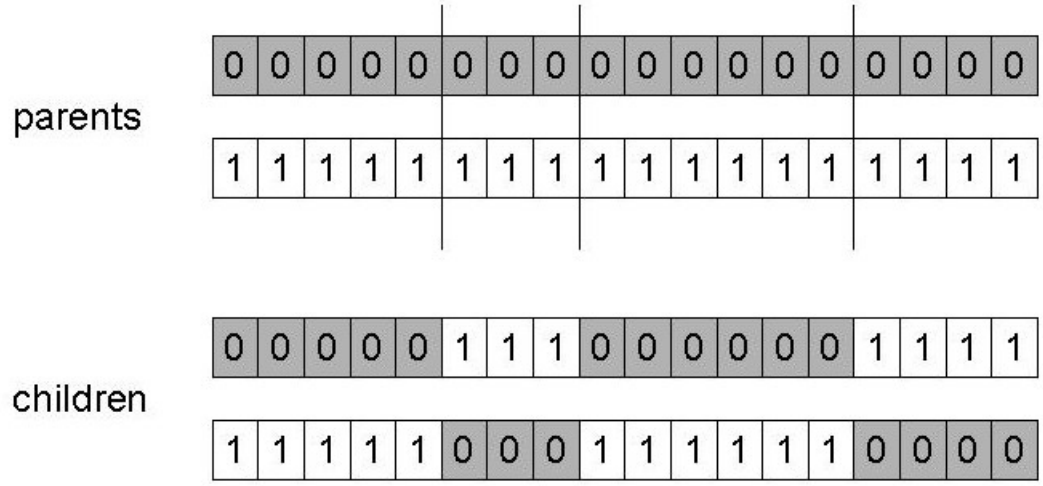


Figure 4.41 *n*-point Binary Crossover Mechanism

Uniform Binary Crossover

Similarly, Figure 4.42 exhibits the uniform crossover mechanism.

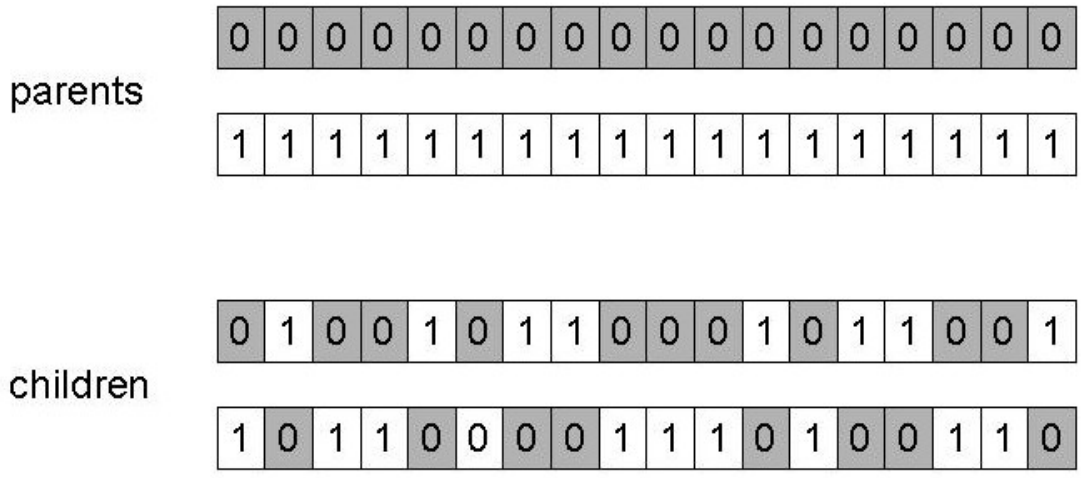


Figure 4.42 Uniform Binary Crossover Mechanism [21]

4.5.4.2 Real-parameter Recombination

In the above binary-coded crossover for GA, decision variables are coded in finite-length strings and exchanging portions of two parent string is easier to implement and visualize. Simply flipping a bit to perform mutation is also convenient and resembles a natural mutation event. In real-parameter GA, the main challenge is how to use a pair of real-parameter decision variable vectors to create a new pair of offspring vectors or how to perturb a decision variable vector to a mutated vector in a meaningful manner. As in such cases the term "crossover" is not that meaningful, they can be best described as *blending* operators. Herrara *et al*. [104] provided a good overview of many real-parameter crossover and mutation operators. Generally, there exist two basic types of recombination rules used in real-parameter GA: *line* and *intermediate* recombination rules.

Line Recombination

Figure 4.43 shows that line recombination can generate any point on the line defined by the parents within the limits of the perturbation, α, for a recombination in two variables. Three line recombination rules (*Linear*, *BLX-*$\alpha$and *Fine-adjusting recombination*) as stated in the following paragraphs are taken for explorations.

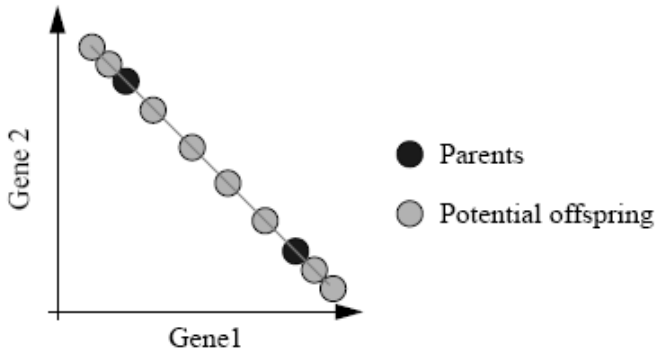


Figure 4.43 Geometric Effect of Line Recombination [21]

Linear Recombination

One of the earliest implementations was reported by Wright [105], where a ***linear recombination*** operator created the following three solutions:

$$0.5\left(\boldsymbol{x}_i^{(1,t)} + \boldsymbol{x}_i^{(2,t)}\right) \tag{4.58a}$$

$$\left(1.5 \cdot \boldsymbol{x}_i^{(1,t)} - 0.5 \cdot \boldsymbol{x}_i^{(2,t)}\right) \tag{4.58b}$$

$$\left(-0.5 \cdot \boldsymbol{x}_i^{(1,t)} + 1.5 \cdot \boldsymbol{x}_i^{(2,t)}\right) \tag{4.58c}$$

from two parent solutions $\boldsymbol{x}_i^{(1,t)}$ and $\boldsymbol{x}_i^{(2,t)}$ at generation t with the best two solutions being chosen as offspring as shown in Figure 4.44.

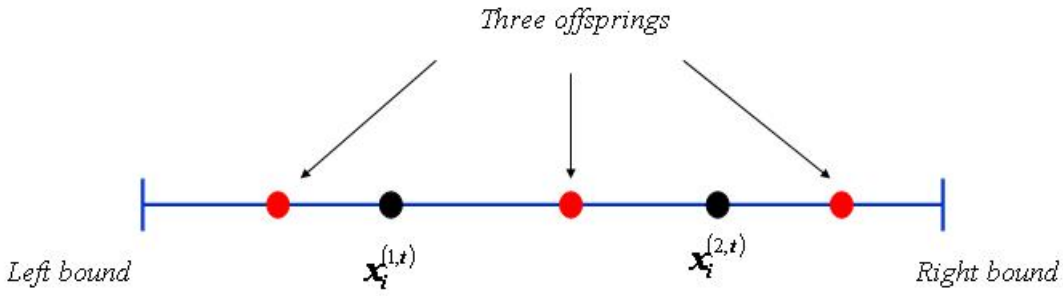


Figure 4.44 Linear Recombination for real-parameter GA [19] (redrawn by Lih)

Blend Crossover (or BLX-α Recombination)

Goldberg [21] and Eshelman and Schaffer [106] suggested a ***blend crossover*** (***BLX-α***) operator for real-parameter GA. For two parent solutions $\boldsymbol{x}_i^{(1,t)}$ and $\boldsymbol{x}_i^{(2,t)}$ at generation t (assuming $\boldsymbol{x}_i^{(1,t)} < \boldsymbol{x}_i^{(2,t)}$), the BLX-$\alpha$ randomly picks a solution in the range $\left[\boldsymbol{x}_i^{(1,t)} - \alpha \cdot \left(\boldsymbol{x}_i^{(2,t)} - \boldsymbol{x}_i^{(1,t)}\right), \boldsymbol{x}_i^{(2,t)} + \alpha \cdot \left(\boldsymbol{x}_i^{(2,t)} - \boldsymbol{x}_i^{(1,t)}\right)\right]$. This crossover operator is illustrated in Figure 4.45. Thus, if $\boldsymbol{u}_i$ is a random number between 0 and 1, the following is an offspring:

$$\boldsymbol{x}_i^{(2,t+1)} = \left(1 - \gamma_i\right) \cdot \boldsymbol{x}_i^{(1,t)} + \gamma_i \cdot \boldsymbol{x}_i^{(2,t)} \tag{4.59}$$

where $\gamma_i = \left(1 + 2\alpha\right) \cdot \boldsymbol{u}_i - \alpha$. If α is zero, this crossover creates a random solution in the range $\left(\boldsymbol{x}_i^{(1,t)}, \boldsymbol{x}_i^{(2,t)}\right)$. Usually, the BLX-0.5 (with α = 0.5) can perform better than BLX operators with any other α value, and the fact γ_i is uniformly distributed for a fixed value of α.

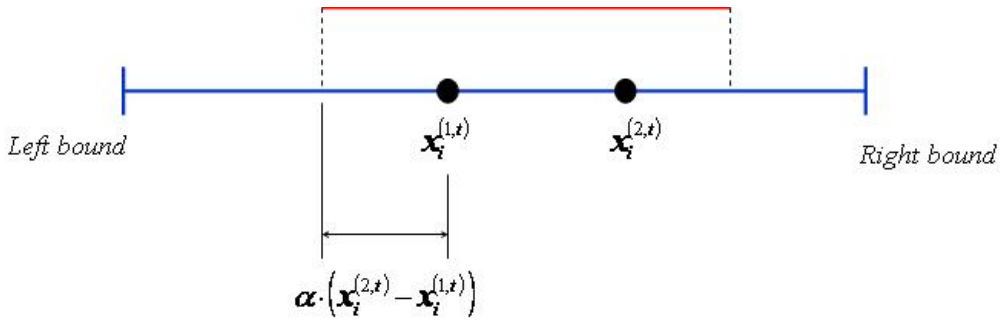


Figure 4.45 Blend Crossover (BLX-α) for real-parameter GA [19] (redrawn by Lih)

Fine-adjusting Recombination

As previously mentioned, the chromosomes of the offspring are usually very similar to those of their parents to keep the good genes in the next generation. In this subsection,

a new real-parameter crossover namely, a ***fine-adjusting recombination***, is developed through the fine-tuning parents' chromosomes to generate the new chromosomes of their offspring according to a following rules:

$$\text{Putting closer} \quad \begin{aligned} \boldsymbol{x}_1'(\boldsymbol{t}+1) &= \boldsymbol{x}_1(\boldsymbol{t}) + \sigma(\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t})) \\ \boldsymbol{x}_2'(\boldsymbol{t}+1) &= \boldsymbol{x}_2(\boldsymbol{t}) - \sigma(\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t})) \end{aligned} \tag{4.60}$$

$$\text{Pulling away} \quad \begin{aligned} \boldsymbol{x}_1'(\boldsymbol{t}+1) &= \boldsymbol{x}_1(\boldsymbol{t}) + \sigma(\boldsymbol{x}_2(\boldsymbol{t}) - \boldsymbol{x}_1(\boldsymbol{t})) \\ \boldsymbol{x}_2'(\boldsymbol{t}+1) &= \boldsymbol{x}_2(\boldsymbol{t}) - \sigma(\boldsymbol{x}_2(\boldsymbol{t}) - \boldsymbol{x}_1(\boldsymbol{t})) \end{aligned} \tag{4.61}$$

where $\boldsymbol{x}_1(\boldsymbol{t})$ and $\boldsymbol{x}_2(\boldsymbol{t})$ are parents for crossover to generate the offspring, $\boldsymbol{x}_1'(\boldsymbol{t}+1)$ and $\boldsymbol{x}_2'(\boldsymbol{t}+1)$ are two new generated offsprings, and σ (scaling factor) is a small random number between 0 and 1.

In other words, according to the previous reproduction procedure, good solutions with the high fitness values were chosen to put into the mating pool. After the operations of Equations (4.60) and (4.61), two new slightly-changed offspring were generated from two randomly-picked parents in the mating pool. Further, Equations (4.60) and (4.61) can be combined as

$$\begin{aligned} \boldsymbol{x}_1'(\boldsymbol{t}+1) &= \boldsymbol{x}_1(\boldsymbol{t}) + \sigma \cdot \cos\theta \cdot (\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t})) \\ \boldsymbol{x}_2'(\boldsymbol{t}+1) &= \boldsymbol{x}_2(\boldsymbol{t}) - \sigma \cdot \cos\theta \cdot (\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t})) \end{aligned} \tag{4.62}$$

where θ is limited between $-\pi$ and π, and σ is randomly taken from [-0. 5, 0.5]. In reality, the term "$\cos\theta$" is used to randomly generate small positive or negative values. The rules of the Equation (4.62) are used in this research for CMP process modeling.

Consider the following two parents with 3 variables each:

Parent 1 12 25 5

Parent 2 123 4 34

The chosen $\sigma \cdot \cos\theta$ for this example are (the same for all variables of parents):

Sample 1 0.5

Sample 2 0.1

The new individuals are calculated as:

Offspring 1 67.5 14.5 19.5

Offspring 2 23.1 22.9 7.9

Intermediate Recombination

Intermediate recombination is a method only applicable to real variables (and not binary variables). Given a real-valued encoding of the chromosome structure, intermediate recombination is a method of producing new phenotypes around and between the values of the parents' phenotypes. Offspring are produced according to the following rules,

$$\begin{aligned} \boldsymbol{x}_1'(\boldsymbol{t}+1) &= \boldsymbol{x}_1(\boldsymbol{t}) + \beta \cdot (\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t})) \\ \boldsymbol{x}_2'(\boldsymbol{t}+1) &= \boldsymbol{x}_2(\boldsymbol{t}) - \beta \cdot (\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t})) \end{aligned} \tag{4.63}$$

where β is a scaling factor usually chosen uniformly at random over some interval, typically $[-\boldsymbol{d}, 1+\boldsymbol{d}]$ and $\boldsymbol{x}_1(\boldsymbol{t})$ and $\boldsymbol{x}_2(\boldsymbol{t})$ are parent chromosomes. The value of the parameter $\boldsymbol{d}$ ($\boldsymbol{d} \in [0, 1]$) defines the size of the area for possible offspring. Each variable in the offspring is the result of combining the variables in the parents according to the above expression with a new β chosen for each pair of parent genes.

Consider the following two parents with 3 variables each:

Parent 1 12 25 5

Parent 2 123 4 34

The chosen β for this example are (different for variables of parents):

Sample 1 0.5 1.1 -0.1

Sample 2 0.1 0.8 0.5

The new individuals are calculated as:

Offspring 1 67.5 1.9 2.1

Offspring 2 23.1 8.2 19.5

In geometric terms, intermediate recombination is capable of producing new variables within a slightly larger hypercube than that defined by parents but constrained by the range of β as shown in Figure 4.46

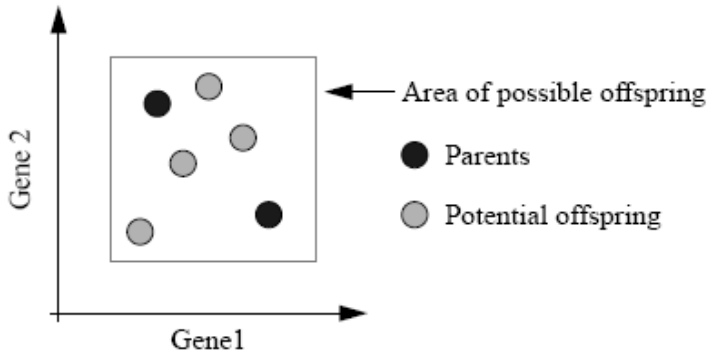


Figure 4.46 Geometric Effect of Intermediate Recombination [21]

4.5.5 Mutation

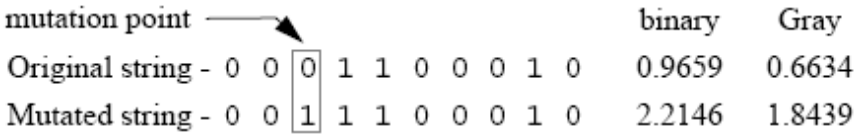
In natural evolution, mutation is a random process where one allele of a gene is replaced by another to produce a new genetic structure. In GA, mutation is randomly applied with low probability, $\boldsymbol{P}_{mutation}$, typically in the range 0.001 and 0.01, and modifies elements in the chromosomes. Usually considered as a background operator, the role of

mutation is often seen as providing a guarantee that the probability of searching any given string will never be zero and acting as a safety net to recover good genetic material that may be lost through the action of selection and crossover. Videlicet, the need of mutation is to keep diversity in the population and prevent from trapping in the local minima. The same as the previous recombination, the binary-coded and real-parameter mutations demonstrates the different rules to generate the offspring.

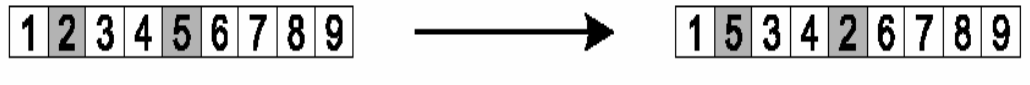
4.5.5.1 Binary Mutation

Single-point Mutation

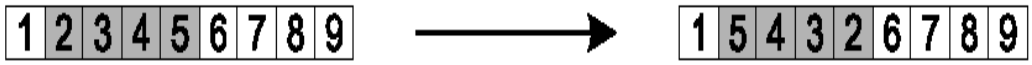
The effects of mutation on a binary string are illustrated in Figure 4.47. Figure 4.47 (a) shows a 10-bit chromosome representing a real value decoded over the interval [0, 10] using both standard and gray coding and a mutation point of 3 in the binary string. Here, binary mutation flips the value of the bit at the loci selected to be the mutation point. Given that mutation is generally applied uniformly to an entire population of strings, it is possible that a given binary string may be mutated at more than one point. Figure 4.47 (a) is a typical example of *single-point mutation*.



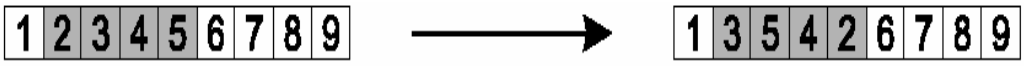
(a) Single-point mutation [19]



(b) Swap mutation



(c) Inversion mutation



(d) Scramble mutation

Figure 4.47 Binary Mutation

Swap Mutation

In Figure 4.47 (b), two alleles were picked at random and swapped their positions in the string. This method can preserve most of adjacency information.

Inversion Mutation

In Figure 4.47 (c), two alleles were picked at random and then inverted the substring between them. This method can preserve most adjacency information (only breaks two links) but disruptive of order information.

Scramble Mutation

In Figure 4.47 (d), a subset of genes was picked at random and randomly rearranged the alleles in those positions in this subset.

Although it may not happen in all the instances of a mutation operation to generate some better solutions than original, mutating a string with a small probability is not a random operation since the process has a bias for creating only a few solutions in the search space.

4.5.5.2 Real-parameter Mutation

As in the binary-coded GA a mutation is meant to have a local perturbation. In real-parameter GA a local perturbation in a predefined manner can also be useful in maintaining diversity in a population. With real-parameter (or non-binary) representation, mutation is achieved by either perturbing the gene values or random selection of new values within the allowed range. In the following, some of the most commonly used real-parameter mutation operators are presented.

Random Mutation

Equation (4.64) shows the rule of this simplest mutation which creates a solution randomly from the entire search space [107]:

$$\boldsymbol{y}_i^{(1,t+1)} = \boldsymbol{r}_i \cdot \left(\boldsymbol{x}_i^U - \boldsymbol{x}_i^L \right) \tag{4.64}$$

where $\boldsymbol{r}_i$ is a random number in [0 , 1]. Figure 4.48 shows this probability distribution with a continuous line. This operator is independent of the parent solution and is equivalent to a random initialization. Instead of creating a solution from the entire search space, a solution in the vicinity of the parent solution with a uniform probability distribution (shown with a dashed line in Figure 4.48) can also be chosen:

$$\boldsymbol{y}_i^{(1,t+1)} = \boldsymbol{x}_i^{(1,t)} + \left(\boldsymbol{r}_i - 0.5 \right) \cdot \Delta_i \tag{4.65}$$

where Δ_i is the user-defined maximum perturbation allowed in the *i*-th decision variable.

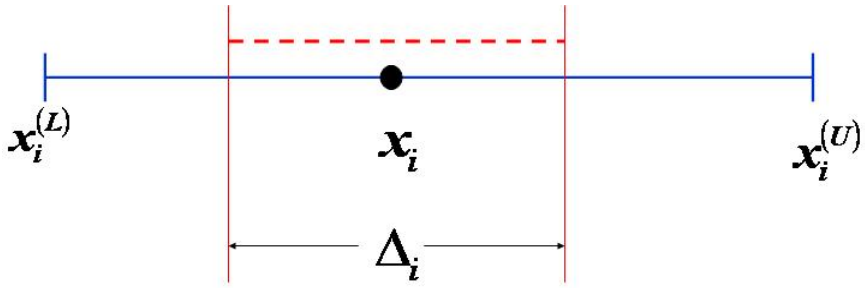


Figure 4.48 Random Mutation Operator for Real-parameter GA [19]

Non-uniform Mutation

The random mutation is the easiest way, but uncontrollable results reduce the applicability. Figure 4.48 demonstrates the probability of creating a solution closer to the parent is more than the probability of creating one away from it. However, as the generations (*t*) proceed, this probability of creating solutions closer to the parent gets higher and higher [107].

$$y_i^{(1,t+1)} = x_i^{(1,t+1)} + \tau \cdot \left(x_i^{(U)} - x_i^{(L)}\right) \cdot \left(1 - r_i^{\left(1 - t/t_{\max}\right)^b}\right) \tag{4.66}$$

Here, τ takes a Boolean value, -1 or 1, each with a probability of 0.5. The parameter $t_{\max}$ is the maximum number of allowed generations, while *b* is a user-defined parameter. In this way, from early on, the above mutation scheme acts like a uniform distribution, while in later generation it acts like *Dirac's function*, thus allowing a focused search.

Normally Distributed Mutation

A simple and popular method is to use a zero-mean Gaussian probability distribution:

$$\boldsymbol{y}_i^{(1,t+1)} = \boldsymbol{x}_i^{(1,t+1)} + \boldsymbol{N}\left(0\,,\sigma_i\right) \tag{4.67}$$

Here, the parameter σ_i is a fixed, user-defined parameter. This parameter is important and must be correctly set in a problem. Such a parameter can also be adaptively changed in every generation by some predefined rule.

Polynomial Mutation

The probability distribution can also be a polynomial function, instead of a normal distribution [19]

$$\boldsymbol{y}_i^{(1,t+1)} = \boldsymbol{x}_i^{(1,t+1)} + \left(\boldsymbol{x}_i^{(U)} - \boldsymbol{x}_i^{(L)}\right)\cdot \bar{\delta}_i \tag{4.68}$$

where the parameter $\bar{\delta}_i$ is calculated from the polynomial probability distribution $\boldsymbol{P}(\delta) = 0.5\cdot(\eta_m + 1)\left(1-|\delta|\right)^{\eta_m}$:

$$\bar{\delta}_i = \begin{cases} \left(2\boldsymbol{r}_i\right)^{1/(\eta_m+1)} - 1, & \text{if } \boldsymbol{r}_i < 0.5 \\ 1 - \left[2\left(1-\boldsymbol{r}_i\right)\right]^{1/(\eta_m+1)}, & \text{if } \boldsymbol{r}_i \geq 0.5 \end{cases} \tag{4.69}$$

This distribution is similar to the non-uniform mutation operator, although a fixed value of the parameter η_m is suggested here. The mutation operator is modified for two regions, i.e., $\left[\boldsymbol{x}_i^{(L)}\,,\boldsymbol{x}_i\right]$ and $\left[\boldsymbol{x}_i\,,\boldsymbol{x}_i^{(U)}\right]$, very similar to the previous non-uniform mutation operator. The difference is that in this mutation operator the shape of the probability distribution is directly controlled by an external parameter η_m and the distribution is not dynamically changed with generations.

Novel Random Mutation

The novel mutation rules used in this research are analogous to Equation (4.62) (for fine-adjusting recombination) as shown in Equation (4.70).

$$\begin{aligned} \boldsymbol{x}_1'(\boldsymbol{t}+1) &= \boldsymbol{x}_1(\boldsymbol{t}) + \varphi \cdot \cos\theta \cdot (\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t})) \\ \boldsymbol{x}_2'(\boldsymbol{t}+1) &= \boldsymbol{x}_2(\boldsymbol{t}) - \varphi \cdot \cos\theta \cdot (\boldsymbol{x}_1(\boldsymbol{t}) - \boldsymbol{x}_2(\boldsymbol{t})) \end{aligned} \tag{4.70}$$

where φ is also a scaling factor, but whose range, [-0.5 , 0.5], is usually larger than σ in Equation (4.62).

Additionally, the mutation rate, $\boldsymbol{P}_{mutation}$, which is typically from 0.001 to 0.01, is much smaller than recombination (or crossover) rate, $\boldsymbol{P}_{crossover}$ (0.1 ~0.5). However, Wright [105] and Janikow and Michalewicz [108] demonstrate how real-parameter GA may take advantage of higher mutation rates than binary-coded GA, increasing the level of possible exploration of the search space without adversely affecting the convergence characteristics. Indeed, Tate and Smith [109] argue that for codings more complex than binary, high mutation rates can be both desirable and necessary and show how, for a complex combinational optimization problem, high mutation and non-binary coding yielded significantly better solutions than the normal approach.

Summary

Currently, many other variations on the mutation operator have being proposed. For example, biasing the mutation towards individuals with lower fitness values to increase the exploration in the search without losing information from the fitter individuals [110] or parameterizing the mutation such that the mutation rate decreases with the population convergence [111].

Mühlenbein [112] introduced a mutation operator for the real-parameter GA that uses a non-linear term for the distribution of the range of mutation applied to gene values. It was claimed that by biasing mutation towards smaller changes in the gene values, mutation can be used in conjunction with recombination as a foreground search process.

Additionally, other mutation operations include that of *trade mutation* [113], whereby the contribution of individual genes in a chromosome is used to direct mutation towards weaker terms, and *reorder* (or *swap*) *mutation* [113], that swaps the positions of the bits or genes to increase diversity in the decision variable space.

4.5.6 Reinsertion

After producing offsprings they must be inserted into the population. This is especially important, if fewer offsprings are produced than the size of the original population. Another case is when not all offsprings are to be used at each generation or if more offsprings are generated than need be. By a reinsertion scheme which individuals should be inserted into the new population and which individuals of the population will be replaced by offspring are to be determined.

The selection algorithm used determines the reinsertion scheme including:

- Global reinsertion for all population based selection algorithm (such as roulette-wheel selection, stochastic universal sampling, truncation selection).
- Local reinsertion for local selection

4.5.7 Termination Criteria

Because the GA is a stochastic search method, it is difficult to formally specify convergence criteria. As the fitness of a population may remain static for a number of generations before a superior individual is found, the application of conventional termination criteria becomes problematic. A common practice is to terminate the GA after a pre-specified number of generations and then test the quality of the best members of the population against the problem definition. If no acceptable solutions are found, the GA may be restarted or a fresh search initiated.

4.6 Applications of Multi-objective Optimization to CMP

4.6.1 Multi-objective Optimization Problems (MOOP) in CMP

A multi-objective optimization problem (MOOP) implies simultaneous dealing with more than one objective function. In most practical decision-making problems, multiple objectives or multiple criteria are evident. In engineering practices, it is often a challenge to formulate a design when there are several criteria or design objectives to be met at the same time. In this research, higher MRR and lower WIWNU are two main objectives in the CMP process. Obviously, this is a typical MOOP. If the objectives are conflicting, then the problem becomes one of finding the best possible designs that satisfies the conflicting objectives under different trade-off scenarios.

Because of a lack of suitable solution methodologies, an MOOP has been mostly cast and solved as a single-objective optimization problem in the past. However, there exist a number of fundamental differences between the working principles of single and multi-objective optimization algorithms. In a single-objective optimization problem, the

task is to find one solution which optimizes the sole objective function. Extending the same idea to multi-objective optimization, it may be wrongly assumed that the task in a multi-objective optimization is to find an optimal solution corresponding to each objective function.

In reality, to a MOOP, it is usually possible to find a set of values for the decision variables that optimizes a set of objective functions. The set of decision variables that produces the optimal result is designated to be the optimal set. This optimal set is referred to as the Pareto optimal set (Figure 4.49) and it yields a set of possible answers from which we may choose the desired values of the design variables.

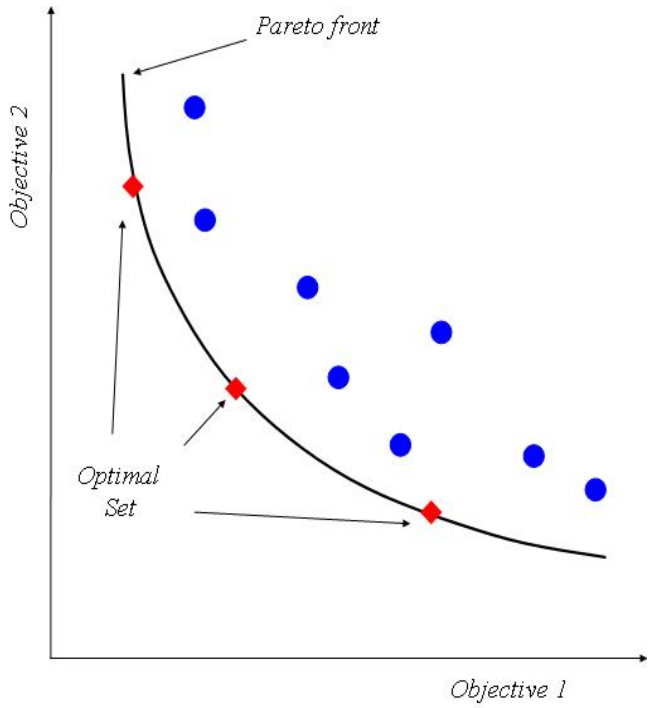


Figure 4.49 Definition of the Pareto Optimality for Two Conflicting Objectives [12]

As stated earlier, higher MRR and lower WIWNU are two main objectives in the CMP process. There are many process inputs (or decision variables) which affect these

two conflicting objectives. Except for constructing the accurate process models for MRR and WIWNU, searching the optimal set of process inputs to satisfy these two objectives is also necessary for advancements of semiconductor industry.

4.6.2 Concepts of Pareto Optimum

As shown in Figure 4.49, a set of points is said to be *Pareto optimal* (i.e., the red diamond points in the figure), if any movements of these red points not along the *Pareto front* will cause at least one of the other objectives to deteriorate from its current optimal value. Note that, on the basis of this concept, blue points in the Figure 4.48 are not Pareto optimal.

A more formal definition of Pareto optimality is given by [114]. Consider, without the loss of generality, the minimization of the *n* components $\boldsymbol{f}_{\kappa}, \boldsymbol{k} = 1,2,\ldots,\boldsymbol{n}$, of a vector function $\boldsymbol{F}$ of a vector variable $\boldsymbol{x}$ in a universe μ, where

$$\boldsymbol{F}(\boldsymbol{x}) = (\boldsymbol{f}_1(\boldsymbol{x}), \boldsymbol{f}_2(\boldsymbol{x}), \ldots, \boldsymbol{f}_n(\boldsymbol{x})) \tag{4.71}$$

Then a decision vector $\boldsymbol{x}_{\mu} \in \mu$ is said to be Pareto-optimal if and only if there is no $\boldsymbol{x}_{v} \in \mu$ for which $\boldsymbol{V} = \boldsymbol{F}(\boldsymbol{x}_{v}) = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ dominates $\boldsymbol{U} = \boldsymbol{F}(\boldsymbol{x}_{u}) = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$, i.e., there is no $\boldsymbol{x}_{v} \in \mu$ such that

$$\forall\, \boldsymbol{i} \in \{1, \ldots \boldsymbol{n}\}, \boldsymbol{v}_i \le \boldsymbol{u}_i \text{ and } \exists\, \boldsymbol{i} \in \{1, \ldots \boldsymbol{n}\}, \boldsymbol{v}_i < \boldsymbol{u}_i$$

The set of all Pareto-optimal decision vectors is called the Pareto-optimal set of the problem. The corresponding set of objective vectors is called the non-dominated set, or Pareto front. Apparently, the Pareto front dominates all other possible solutions and in

most cases, it is located on the boundary of the objective vector space as shown in Figure 4.49 for a two-objective optimization problem.

4.6.3 Problems in Conventional Methods for MOOP

Several methods have been recognized as popular decision-making methods for solving MOOP. Among all of these methods, *weighting objective method*, *goal programming method*, and *min-max optimum method* are the most representative ones. Although, these conventional algorithms have some differences in their design procedures, they all are based on a similar spirit that converts a MOOP into a single-objective optimization problem. These conversions are always directed by the preferences of the decision-maker. Meanwhile, gradient-based or simplex-based optimization techniques are usually applied as a searching tool for the optimal solution, which may result in a local optimum solution for complicated optimization problems.

In many real-world multi-objective optimization problems, however, a suitable solution for the overall problem can hardly be found via the previously mentioned methods since the objectives are different, sometimes even conflicting. Generally speaking, the simultaneous optimization of multiple, possibly competing, and conflicting objective functions are more attractive in that it seldom admits single, perfect solution. Instead, multi-objective optimization problems tend to be characterized by a family of alternatives that must be considered equivalent in the absence of information concerning the importance of each objective relative to others.

From the definition of the Pareto optimality [114], "an MOOP tends to be characterized by a family of trade-off solutions, which must be considered equivalent in the absence of the information of the relevance of each objective relative to the others" [114]. Therefore, with this spirit in mind, *Multi-objective Evolutionary Algorithms* (MOEA) have drawn more and more attention form researchers in this field.

4.6.4 Multi-objective Evolutionary Algorithms (MOEA)

Solving multi-objective scientific and engineering problems is generally a very difficult goal. In those particular optimization problems, such as MRR and WIWNU, the objectives often conflict across a high-dimensional problem space and may also require extensive computational resources. General multi-objective optimization problem (MOOP) solution methods range from linear objective function aggregation to Pareto-based techniques. In an attempt to stochastically solve problems of this generic class in an acceptable timeframe, specific multi-objective evolutionary algorithms (MOEA) were initially developed in the mid-eighties for application to the MOP domain. Since then, a forty-fold increase in the number of MOEA publications has been found for various solution techniques proposed, along with applications in numerous scientific and engineering disciplines [115].

In their early development, Evolutionary Algorithms (EA), a class of population-based optimization approaches, have been recognized to be well suited for multi-objective optimization. In EA, multiple individuals search for multiple solutions in parallel, advantageously producing a family of feasible solutions to the problem. The

ability to handle complex problems involving features, such as discontinuities, multi-modality and disjoint objective vector spaces reinforces the potential effectiveness of EA in multi-objective search and optimization, which is perhaps the problem area where EA most distinguish themselves from other algorithms [114].

Since the 1980's, several MOEA have been proposed as listed in Table 4.3 and applied [116]. These algorithms share the same purpose – approximate a uniformly distributed, near-optimal and near-complete Pareto front for a given MOOP. Generally, the approximation of the Pareto-optimal set involves two conflicting objectives: the distance to the true Pareto front is to be minimized while the diversity of the evolved solutions is to be maximized [117]. For the first objective, a Pareto-based fitness assignment (ranking scheme) is usually designed in some state-of-the-art MOEA [116] in order to guide the search towards the ideal Pareto optimal front. For the second objective, some successful MOEA provide a density estimation method to preserve the population diversity. In addition, several other techniques have also been adopted, such as *elitism scheme* [117, 118], *corwded comparison* [118], *archive truncation* [117].

Although all of these techniques are very important for MOEA, the fitness assignment scheme, population density preservation method and elitism archive are considered the most crucial approaches, which have been applied in all of the most successful MOEA.

A conceptual framework for MOEA is shown in Figure 4.50 [1]

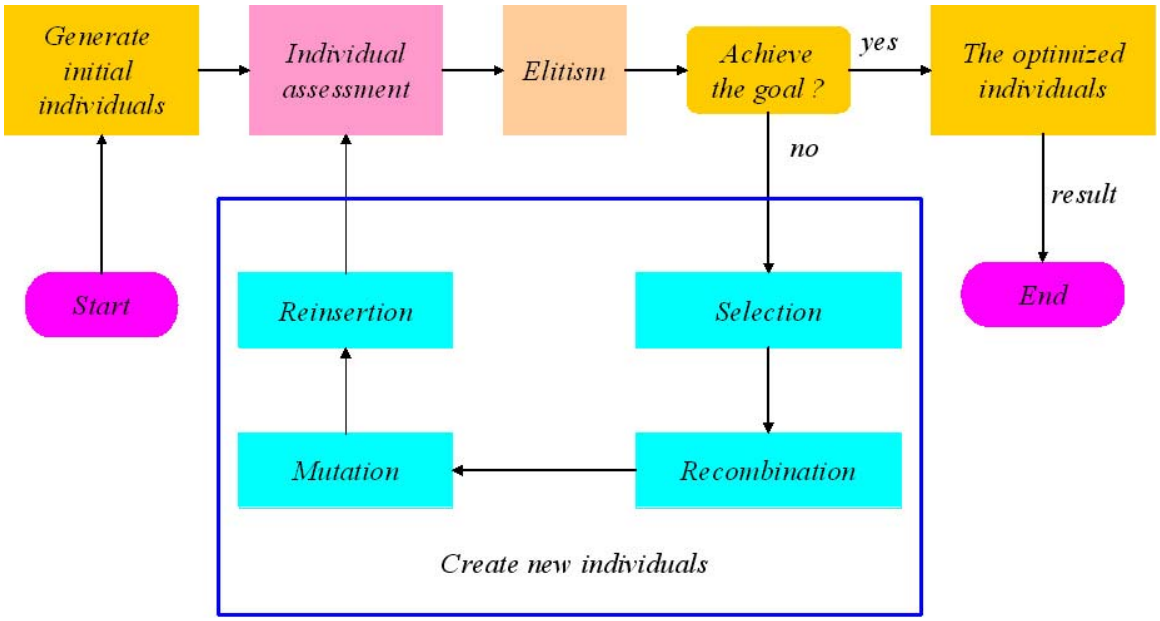


Figure 4.50 Conceptual Framework for MOEA [1] (redrawn by Lih)

In Table 4.3 a list commonly used MOEA in current MOOP is given. The biggest differences between MOEA listed in Table 4.3 are the blocks, "*Individual assessment*" and "*Elitism*". In order to evolve an evenly distributed population along Pareto front or to distribute the population at multiple optima in the search space, many methods have been proposed for the "*Individual assessment*" and "*Elitism*" procedures, including *fitness sharing* [119], *fitness scaling* [5], *sequential niching* [120], *dynamic niching* [121], *immune system* [122], *ecological GA* [123], *standard crowding* [124], *deterministic crowding* [125], *restricted tournament selection* [126], and *clearing* [127].

Table 4.3 Common MOEA used for MOOP [12]

Inventors	Year	MOEAs
Schaffer	1985	Vector Evaluated Genetic Algorithm (VEGA)
Fonseca & Fleming	1993	Multi-objective Genetic Algorithm (MOGA)
Horn *et al*.	1994	Niched Pareto Genetic Algorithm (NPGA)
Srinivas & Deb	1994	Nondominated Sorting Genetic Algorithm (NSGA)
Knowles & Corne	1999	Pareto Archived Evolution Strategy (PAES)
Zitzler & Thiele	1999	Strength Pareto Evolutionary Algorithm (SPEA)
Zitzler *et al*.	2001	Elitist Strength Pareto Evolutionary Algorithm (SPEA 2)
Deb *et al*.	2002	Elitist Nondominated Sorting Genetic Algorithm (NSGA-II)
Lu & Yen	2002	Rank-Density based Genetic Algorithm (RDGA)
Lu & Yen	2002	Dynamic Multi-objective Evolutionary Algorithm (DMOEA)

In this research, the Elitist Non-Dominated Sorting Genetic Algorithms (NSGA-II), described in the Section 4.5.6 is used for solving the higher MRR and lower WIWNU in CMP process to find out the optimal process setting range.

4.6.5 Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)

In NSGA-II, the offspring population $\boldsymbol{Q}_t$ is first created by using the parent population $\boldsymbol{P}_t$. However, instead of finding the non-dominated front of $\boldsymbol{Q}_t$ only, the first two populations are combined to form $\boldsymbol{R}_t$ of size $2\boldsymbol{N}$. Then, a non-dominated sorting is used to classify the entire population $\boldsymbol{R}_t$. Once the non-dominated sorting is over, the new population is filled by solutions of different non-dominated fronts, one at a time. The

filling starts with the best non-dominated front and continues with solutions of the second non-dominated front, followed by the third non-dominated front, and so on. Since the overall population size of $\boldsymbol{R_t}$ is $2\boldsymbol{N}$, not all fronts may be accommodated in $\boldsymbol{N}$ slots available in the new population. All fronts which could not be accommodated are simply deleted. When the last allowed front is being considered, there may exist more solutions in the last front than the remaining slots in the new population. This scenario is illustrated in Figure 4.51. Instead of arbitrarily discarding some members from the last front, it would be wise to use a niching strategy to choose the members of the last front, which reside in the least crowded region in that front.

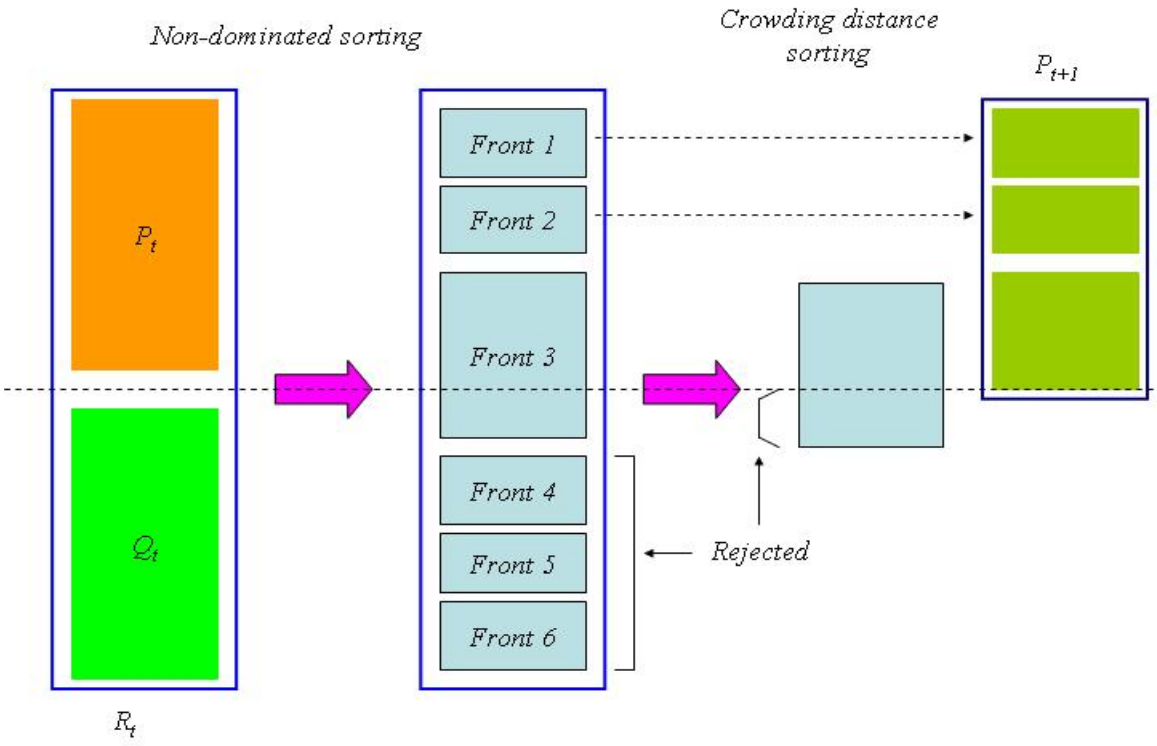


Figure 4.51 Schematic of the NSGA-II Procedure

In the following, the algorithm in a step by step format is outlined. Initially, a random population $\boldsymbol{P_t}$ is created. The population is sorted into different non-domination levels. Each solution is assigned a fitness which is equal to its non-domination level. Thereafter, the selection for reproduction, recombination, mutation, and reinsertion procedures are used to create an offspring population $\boldsymbol{Q_t}$ of size *N*. The NSGA-II procedure is outlined in the following:

Step 1 Combine parent and offspring populations and create $\boldsymbol{R_t} = \boldsymbol{P_t} \cup \boldsymbol{Q_t}$. Perform a non-dominated sorting to $\boldsymbol{R_t}$ and identify different fornts: $\boldsymbol{Front_i}, \boldsymbol{i} = 1,2,\ldots,$ etc.

Step 2 Set new population $\boldsymbol{P_{t+1}} = 0$. Set a counter $\boldsymbol{i} = 1$. Until $\left|\boldsymbol{P_{t+1}}\right| + \left|\boldsymbol{Front_i}\right| < \boldsymbol{N}$, perform $\boldsymbol{P_{t+1}} = \boldsymbol{P_{t+1}} \cup \boldsymbol{Front_i}$ and $\boldsymbol{i} = \boldsymbol{i} + 1$.

Step 3 Perform the *crowding-sort* procedure and include the most widely spread $\left(\boldsymbol{N} - \left|\boldsymbol{P_{t+1}}\right|\right)$ solutions by using the *crowding distance* values in the sorted $\boldsymbol{Front_i}$ to $\boldsymbol{P_{t+1}}$.

Step 4 Create offspring population $\boldsymbol{Q_{t+1}}$ form $\boldsymbol{P_{t+1}}$ by using the *crowded tournament selection*, *recombination*, and *mutation* operators.

In Step 3, the crowding-sort of the solutions of the front $\boldsymbol{i}$ is performed by using a *crowding distance metric*. The population is arranged in descending order of magnitude of the crowding distance values. In Step 4, a crowding tournament selection operator, which also uses the crowding distance, is used.

4.6.5.1 Crowding Distance

In order to evolve an evenly distributed population along Pareto front or to distribute the population at multiple optima in the search space, crowding distant is another important issue. More crowded parts should assign lower fitness values to each solution involved.

To get an estimate of the density of solution surrounding a particular solution i in the population, the average distance of two solutions is taken on either side of the solution i along each of the objectives. This quantity d_i serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices. In the Figure 4.52, the crowding distance of the i-th solution in its front is the average side-length of the cuboid (shown by a dashed box) [1].

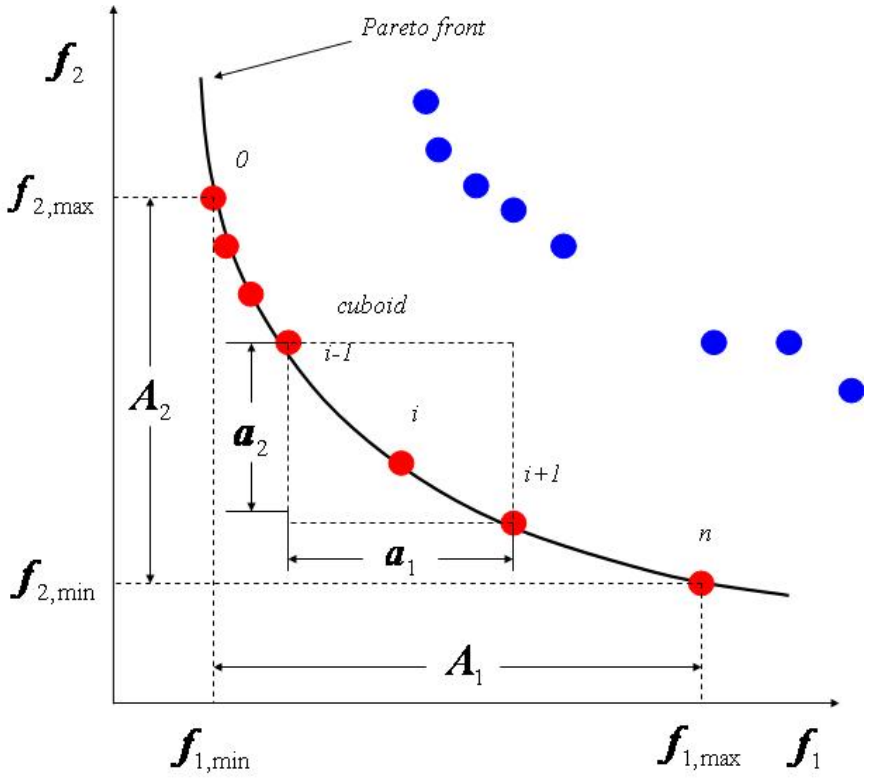


Figure 4.52 Calculation of Crowding Distance for NSGA-II [1] (drawn by Lih)

The following algorithm is used to calculate the crowding distance of each point in the Front.

Crowding Distance Assignment Procedures for the i-th solution in Figure 4.52:

Step 1 Initially, set the crowding distance of the solution i, $d_i = 0$.

Step 2 For the objective function, f_1, calculate the A_1 and a_1. Then calculate the d_i using Equation (4.71):

$$d_i = d_i + \frac{a_1}{A_1} \tag{4.71}$$

$$\Rightarrow d_i = 0 + \frac{a_1}{A_1} = \frac{a_1}{A_1}$$

Step 3 For the objection function, f_2, calculate the A_2 and a_2. Then calculate the d_i again using Equation (4.71):

$$\Rightarrow d_i = \frac{a_1}{A_1} + \frac{a_2}{A_2} = \frac{A_2 \cdot a_1 + A_1 \cdot a_2}{A_1 \cdot A_2}$$

The larger the crowding distance of the solution i is, the higher the fitness will be assigned. In other words, the solution i will have the higher probability to be copied to the next generation.

4.6.5.2 Crowded Tournament Selection Operator

The crowded comparison operator compares two solutions and returns the winner of the tournament. It assumes that every solution i has two attributes:

1. A non-domination rank r_i in the population.

2. A local crowding distance, d_i in the population.

As explained in the previous paragraphs, the crowding distance d_i of the solution i is a measure of the search space around i which is not occupied by any other solution in the population. Based on these two attributes, the crowded tournament selection operator can be defined as follows:

The solution i wins a tournament with another solution j, if any of the following conditions are true:

1. If solution i has a better rank, i.e., $r_i < r_j$.
2. If they have the same rank but the solution i has a larger crowding distance than solution j, that is, $r_i = r_j$ and $d_i > d_j$.

The first condition makes sure that the chosen solution lies on a better non-dominated front. The second condition resolves the tie of both solutions being on the same non-dominated front by deciding on their crowded distance. The one residing in a less crowded area (with a larger crowding distance d_i) wins.

CHAPTER V

MODELING AND OPTIMIZATION OF CMP PROCESS

5.1 Introduction

In this chapter, modeling of the CMP process (i.e., MRR and WIWNU) using Neural Networks (NN), ANFIS (GP and SC), and Genetic Algorithms (GA) is described in detail. Further, searching the setting ranges of input variables for the optimization of CMP (i.e., higher MRR and lower WIWNU) using multi-objective evolutionary algorithms (MOEA) is also introduced.

In this investigation, three cases of CMP experiments are used to model the CMP process. The experiment data sets for Case I and II were taken from those reported by Wang *et al*. [30, 79-81]. The experimental data sets for Case III were conducted by this author. Except for modeling CMP process, searching for the suitable input settings for process optimization is another main topic in this investigation.

In addition, three-stage CMP experiments in Case III is provided, each with 1875, 125, and 16 data sets respectively for constructing the CMP models, testing the models and verifying the input settings for process optimization.

5.2 CMP Experiments

5.2.1 Case I Experiments

The 27 source data sets for Case I [79, 81] are summarized in Table A-4 of Appendix III . Of this, 22 data sets were randomly chosen for training the NN and ANFIS (GP and SC) models, and the other 5 data sets for testing the trained models.

5.2.2 Case II Experiments

The 54 source data sets for Case II [79-81] are given in Table A-5 of Appendix III. Of this, the 45 data sets were randomly chosen, (similar to the Case I) for training the NN and ANFIS models, and the other 9 data sets for testing the trained models.

5.2.3 Case III Experiments

As discussed in Chapter III and referring to the experiments and conclusions from Wang *et al*.[30, 79-81] for Cases I and II, 5 main input variables (i.e., *abrasive concentration* or *solid content* (***Sc***), *down pressure* (***Pd***), *back pressure* (***Pb***), *platen speed*, (***Vp***), and *polishing time* (***T***)) and 2 important output performances (i.e., *material removal rate* (**MRR**) and *within wafer non-uniformity* (**WIWNU**)) for designing these CMP experiments were selected.

The average thickness of a wafer was measured via 49-point inspection as shown in Figure 5.1 before and after CMP. Then, Equations (3.16a) and (3.16b) in the Section 3.4.1 are used to calculate the MRR and WIWNU. Usually, the standard deviation of wafer thickness before CMP is larger than that after CMP.

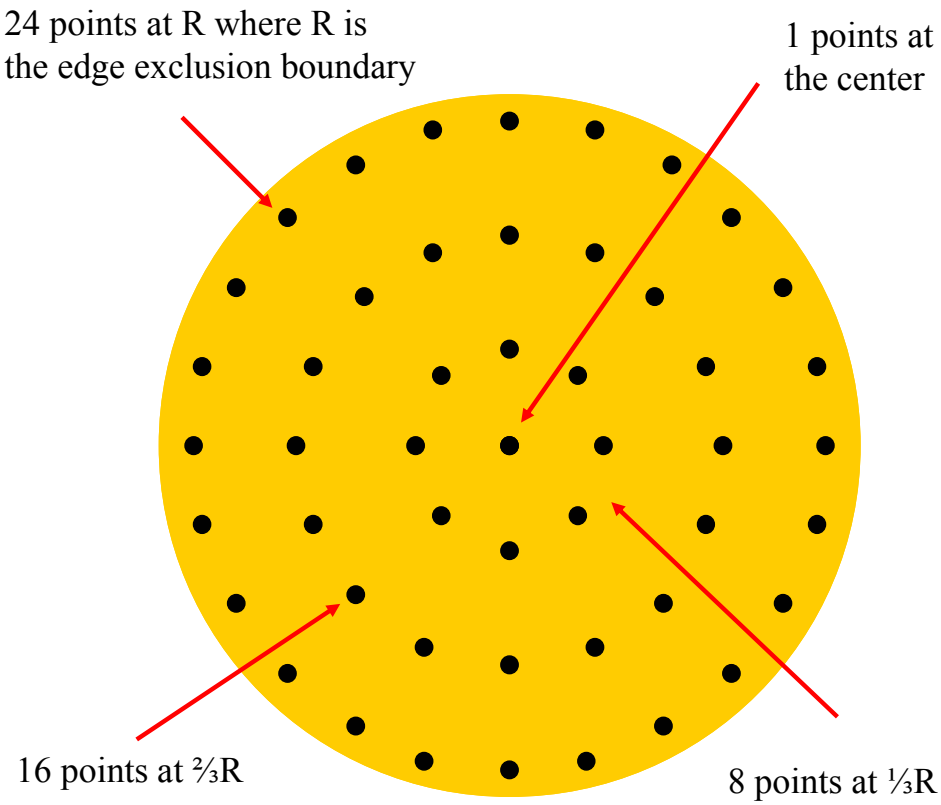


Figure 5.1 49-point Inspection to Measure Thickness Change of Wafer

Table 5.1 lists other experiment conditions used for Case III experiments.

Table 5.1 Experiment Conditions for Case III Experiments

Machine	IPEC/ Westech Avanti 372M Oxide CMP systems
Wafer	6" with Silicon Oxide coating
Slurry	Cabot SS25 (Fume silica 12 wt%, KOH formulated at pH 10.5)
Polishing Pad	Fujibo Polypas #FP85
Particle sizes	Primary 20 ~30 nm, Secondary 0.1 ~1 μm, average 0.25 μm
Carrier Speed = 25 rpm, Oscillation Speed = 2 mm/sec, Flow rate = 100 ml/min	

5.2.3.1 First-stage Experiments – Training data sets

For the requirements of highly accurate modeling using NN, the substantial experimental data sets for modeling construction is necessary. Therefore, the factorial design of experiment (DOE) is chosen to design the first-stage of the CMP experiments. The major reason for choosing the factorial DOE is to make an attempt to cover the entire input and output ranges for providing complete processing information, and to greatly increase the accuracy for process modeling. The experimental design includes 5 levels in ***Sc***, ***Pd***, ***Pb*** and ***Vp***, and 3 levels in ***T***. Totally, there consist of 1875 ($= 5 \times 5 \times 5 \times 5 \times 3$) trials are involved in the first-stage experiment.

Table 5.2 gives the defined range of each input variable used in this investigation.

Table 5.2 Defined Ranges of Input Variables

	Solid Content (***Sc***) weight %	Down Pressure (***Pd***) psi	Back Pressure (***Pb***) psi	Platen Speed (***Vp***) rpm	Polishing Time (***T***) sec
Level 1	5	4	0	20	40
Level 2	10	6	1	30	50
Level 3	15	8	2	40	60
Level 4	20	10	3	50	
Level 5	25	12	4	60	

5.2.3.2 Second-stage Experiment – Testing data sets

For validating the trained models using NN, ANFIS (GP and SC), and GA the second-stage experiments provide 125 testing data sets for testing the trained models. The settings of 5 input variables for each testing experiment are randomly chosen between their defined ranges. For example, the defined range of ***Sc*** ([5 , 25]) by weight percent is listed in Table 5.2.

For maintaining accuracy and less uncontrollable deviation in the CMP experiments, the experimental settings of the first- and second-stages are blended together during the polishing experiments.

5.2.3.3 Third-stage Experiments – Optimal Input Settings

Two main objectives of the experiments in this stage are to verify the predictions from the MOEA and to search for suitable settings of 5 input variables for process optimization via the guidance from the MOEA. In other words, the experiments of this stage are guided by the simulation results from the MOEA. The comparisons between the simulation and experimental results are listed in Chapter VI. Totally, 16 extra trial data sets are utilized to verify this idea.

5.3 Modeling Using Multilayer Feedforward Neural Networks

5.3.1 Difficulties Encountered prior to Neural Network Modeling

In training of the NN, the following four difficulties are usually encountered:

1. Deciding the portions of training data and testing data.
2. Deciding the transfer functions of neurons in the hidden and output layers.

3. Deciding the training methods.
4. Deciding the number of layers and neurons in the hidden level.

As a matter of fact, the first problem can be solved using the approach mentioned in the previous section. The training and testing data can be simultaneously obtained from the first- and second-stages of experiments. Due to high nonlinearity of the CMP process, the non-linear type transfer functions (such as "*tansig*") are expected to perform better in learning during the training period.

For obtaining good generalization, the Bayesian regulation method [128] which can minimize a combination of squared errors and weights and determine a correct combination was adopted to train the NN. Additionally, for faster learning, the Levenberg-Marquardt optimization will be also used. Under the Matlab programming platforms, the command "*trainbr*" [128] which covers these two methods will be employed to design the NN models.

5.3.2 Determining the Optimal Neural Network Architectures

Based on a common rule-of-thumb used among the experienced NN designers the ratio of *training exemplars* to *network weights* shows a minimum of 10:1 (as stated in Section 4.2.9). Taking Case III experiments as an example, the number of weights should be *no more than* 187 from the 1875 training data sets. Taking the [5-15-7-1] NN architecture for example, the number of weights can be calculated as

$$\text{Total Number of Weights of } [5\text{-}15\text{-}7\text{-}1] \text{ NN architerture}$$
$$= 5 \times 15 + 15 \times 7 + 7 \times 1 = 187$$

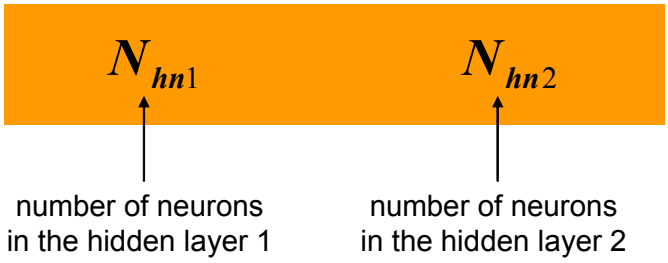
which means 5 neurons in the input layer, 1 neuron in the output layer, 15 and 7 neurons in the two hidden layers, respectively. Actually, because we use Bayesian regulation method for training NN, a slightly larger ratio (5:1) replaces the aforementioned rule-of-thumb for finding more potential NN architectures

Before determining the number of hidden layers and hidden neurons, "what is the best NN architecture" should be first defined. Undoubtedly, smaller training error (E_train) and smaller testing error (E_test) are two of the most important indices. Additionally, the capacity of the generalization is another important requirement. In brief, both the aforementioned concepts can be mathematically expressed as Equations (5.1) and (5.2). Typically, it can be viewed as a multi-objective optimization problem. The ideas learned from the multi-objective evolutionary algorithms (MOEA) can be very helpful to find out the optimal NN architecture.

$$E\_total = E\_train + E\_test \ , \ E\_total \text{ smaller is better} \tag{5.1}$$

$$G = \frac{E\_test}{E\_train} \ , G \text{ smaller is better} \tag{5.2}$$

According to the previous explanations in Section 5.3.1, only the problem 4 remains to be determined. Assume that there are at most two hidden layers to be used for our NN architecture. On the basis of the characteristics of this problem, we can use the real-parameter GA to form the following chromosomes shown in Figure 5.2:



Parameters \ Genes	N_{hn1}	N_{hn2}
Ranges	1 ~ 30	0~30
Step size	1	1

Figure 5.2 Real-parameter Chromosome of NN Architecture

The number of neurons in the input layer is fixed at 5 for 5 input variables and only one neuron is setup in the output layer for MRR or WIWNU. That is, the NN architectures of MRR and WIWNU models should possess the format of $[5 - N_{hn1} - N_{hn2} - 1]$. If N_{hn2} is zero, that means only one hidden layer exists in this NN architecture. Any chosen NN architecture should meet the requirements of the above-mentioned *designer's rule* first. The number of training cycles was set at 200, using the command "*trainbr*" under Matlab programming environment.

All combinations trained using 1875 training data sets and listed in Table 5.3 are qualified candidates, meeting the requirements of Equations (5.1) and (5.2). Fewer number of weights are preferred if they are undefeated by each other because fewer weights means shorter training time consuming.

In this example, the combination of # 18 as shown in Table 5.3 was preferred to be chosen to construct an NN model used.

Table 5.3 Optimal NN Architectures for Modeling MRR

	Best NN Architectures				Total weights	Mean Squre Training Error (E_train)	Mean Square Testing Error (E_test)	E_train + E_test	Generalization (E_test / E_train)
1	5	16	13	1	301	0.0028	0.0015	0.0043	0.5357
2	5	4	84	1	440	0.0043	0.002	0.0063	0.4651
7	5	5	85	1	535	0.0038	0.0018	0.0056	0.4737
8	5	16	13	1	301	0.0034	0.0016	0.0050	0.4706
9	5	15	14	1	299	0.0028	0.0015	0.0043	0.5357
11	5	13	3	1	107	0.0015	0.0011	0.0026	0.7333
12	5	17	4	1	157	0.0017	0.0012	0.0029	0.7059
13	5	13	8	1	177	0.0017	0.0012	0.0029	0.7059
14	5	13	8	1	177	0.002	0.0012	0.0032	0.6000
15	5	17	4	1	157	0.0017	0.0011	0.0028	0.6471
16	5	11	9	1	163	0.0019	0.0011	0.0030	0.5789
17	5	13	7	1	163	0.0017	0.0011	0.0028	0.6471
18	5	15	6	1	171	0.0019	0.0011	0.0030	0.5789

5.4 Modeling Using ANFIS

5.4.1 Grid Partition (GP) Method

Each pair of experiment data set (including inputs and outputs) in the training data can be viewed as a fuzzy rule. As described in Section 4.4.2, GP generates rules by enumerating all possible combinations of membership functions of all input parameters. For example, if the number of levels for each input variable used is the same as in Table 5.2, then some 1875 fuzzy rules are involved in the ANFIS-GP models for MRR and WIWNU, respectively.

5.4.2 Subtractive Clustering (SC) Method

In contrast to the GP method described in the previous subsection, the subtractive clustering (SC) (or scattering partition) generates minimal number of rules equal to the same number of membership functions (or levels) pre-determined in each input. How many rules are needed to describe an unknown system depends on the size and distribution of the given data in the entire input space. For Cases I and II, obviously, the size and distribution of the given data relative to the dimensions of the inputs and the outputs are sparse and non-uniform. These dilemmas increase the degree of difficulties to obtain accurate modeling results. Too few fuzzy rules assigned for the sparse-data case might result in the "fragmental model" as shown in Appendix II. Conversely, too many fuzzy rules used can lead to "overfitting-like" (i.e., small training error but much larger testing error) situation. For the sake of above-mentioned factors, the *trial-and-error* method is currently used to construct the ANFIS-SC models. Investigators in this research community are still attempting to find the efficient model-constructing methods. Besides, each constructed model should be check if "fragmental" situation occurred.

5.4.3 Newly-developed ANFIS-SC Modeling for the Case of Sparse-data

5.4.3.1 Origin

In many CMP polishing situations only sparse data is available. For example, if new slurry chemicals are introduced, the Fabs would like to know of its influence with minimal number of tests. Obviously, this will result in sparse data. Such sparse data sets can only provide fewer or fragmental process information for constructing models. Through correction rules during the training process, a model with very small training

error can be obtained, but this model usually cannot precisely portray the real features of the entire unknown system, namely, CMP problems. Clearly, the testing error appears as an important index from the present state to the real target. Our objective is to reduce the testing error step by step to the acceptable ranges as a sculpturing course. This is the core spirit behind the *fine-tuning technique* which will be introduced here.

CMP process is influenced by several process variables, some of which wield a significant influence on MRR and WIWNU, while the others might not influence at all. In other words, significant variables and insignificant variables provide different contributions to improve the accuracy of process modeling, and simultaneously to minimize simulation errors (e.g., testing error).

Figure 5.3 shows two curve-fitting results. After training, the errors at points 4 and 9 are large. But with a slight change of slopes of the fitted curve around points 3 and 8, the errors can be greatly reduced or changed almost without changes in the other points. In other words, if we can effect these fairy local, *slight changes*, we can effectively reduce or change the testing errors with almost no change in the training errors.

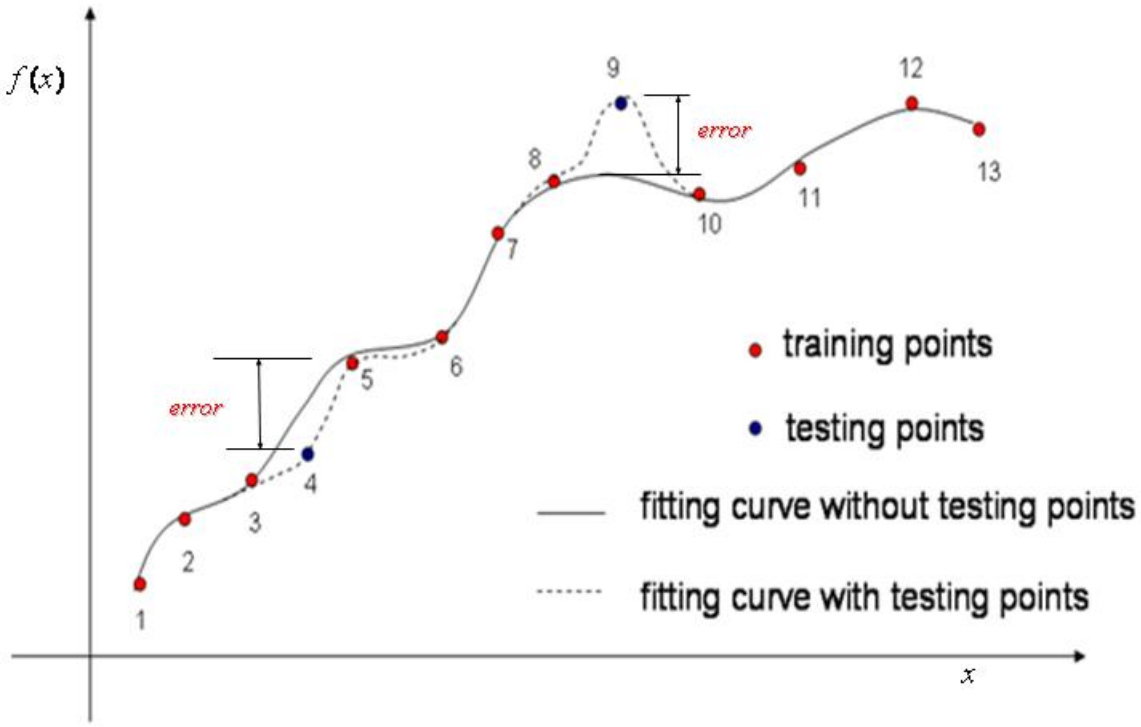


Figure 5.3 Influence of Slight Change on Testing Errors

5.4.3.2 Theoretical Concepts

As discussed in Section 4.4.2, the ANFIS-SC model generates a much small number of fuzzy rules compared to ANFIS-GP. Here, the number of rules is equal to the number of membership functions in each input. In other words, there is the minimal number of parameters in the consequent part (i.e., linear parameters), but maximal number of the parameters in the premise part (i.e., nonlinear parameters). Moreover, because more membership functions are applied to describe the input space, the influence of each membership function is much smaller than that in the ANFIS-GP model. Likewise, the varieties of training errors in the ANFIS-SC model will be substantially smaller than those in the ANFIS-GP model, for slight changes in the membership functions in the premise part. Now, if the input variables, whose membership functions

are to be slightly adjusted, are insignificant compared to the other important input variables, the training errors can be almost kept the same.

The parameters in the consequent part, however, vary following the trade-off of both forward-pass and backward-pass training processes as described in Sections 4.4.7 and 4.4.8. Consequently, the testing errors will still fluctuate, even though the change of the training errors is very small.

In conclusion, it is possible to continuously fine-tune the membership functions of insignificant input variables to keep infinitesimal variation in training errors, and, conversely, gradually reduce the testing errors. Also, because of the fewer number of fuzzy rules created in ANFIS-SC model, it is very advantageous to the sparse-data cases.

5.4.3.3 Training Procedures of ANFIS-SC Modeling

There are four major steps for training ANFIS-SC model for sparse-data case as follows:

Step 1 *Determining the significance of process inputs used in CMP experiments*: The conventional statistical analysis (e.g. ANOVA) can be used to determine the significance of each input parameter used in the CMP process.

Step 2 *Choosing a suitable number of membership functions and fuzzy rules*: A larger number of membership functions and fuzzy rules usually lead to smaller training errors, but that is not likely to happen for testing errors. Principally, the choice of the number of rules should be such that the resulting training errors and testing errors are both minimal and close to each other as possible as they can.

Step 3 *Adjusting the ANFIS-SC network parameters*: A two-step procedure is used to adjust the consequent parameters *p, q,* and *r* (as shown in Tables 4.1 and 4.2), as well as the premise parameters (i.e., weights *w*).

Step 4 *Fine-tuning rules and membership functions:* Adjusting the premise parameter sets of membership functions of the statistically insignificant process inputs or re-choosing the suitable membership functions to gradually reduce the testing errors. When premise parameter sets are adjusted, the fuzzified values (μ), firing strengths (w_i) and overall outputs will be sequentially and automatically varied. Step 4 needs to be repeated until the acceptable testing errors are reached.

During this tuning process, model errors (including training and testing errors) will either increase or decrease. The novel finding here is that when we adjust the membership functions of insignificant process inputs, it has little or no effects on the training errors. However, slight changes can constructively shrink the testing errors. In other words, this method can greatly improve the predictability of ANFIS simulation models. It is noteworthy that fewer membership functions and fuzzy rules selected are easy to implement fine-tuning adjustments but it may create a fragmentary simulation model (as shown in Appendix II). Conversely, fewer membership functions are involved in more fuzzy rules used in the ANFIS-GP model with greatly increase their individual influences on the overall output. In other words, slight adjustments even in each membership function of insignificant process inputs usually lead to large fluctuations of both training and testing errors.

5.5 Modeling Using Genetic Algorithms (GA)

5.5.1 Introduction

Process modeling is actually a search procedure to find the optimal (or most accurate) representation to describe or predict the behavior of an unknown system. Usually, some experiments and statistical analyses are needed to obtain an initial understanding of the relationships between the input variables and output performances. On the basis of discussions in the previous sections, it can be stated that neural networks and ANFIS models provide powerful methods, especially, if sufficient training data sets are available. In this section, an important development for another novel modeling method by GA is introduced here.

5.5.2 Modeling Methods Using GA

Two newly-developed methods for constructing CMP models are presented in this section. First, the pre-determined mathematical expression for process models is used in Method I. No pre-determined expression is proposed in Method II.

5.5.2.1 Procedures for Method I

1. Assuming a mathematical expression for the CMP model as Equation (5.1).

$$f_i(x_1, x_2, \cdots, x_n) = a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots + a_n \cdot x_n + \ldots + a_m \cdot x_1 x_2 \ldots x_n + \cdots$$

or

$$f_i(x_1, x_2, \cdots, x_n) = (x_1)^{a_1} \times (x_2)^{a_2} \times \cdots \times (x_n)^{a_n} + (x_1)^{b_1} + (x_2)^{b_2} + \cdots + (x_n)^{b_n} \quad (5.1)$$

where x_j = input variables, $f_i(\bullet)$ = the objective functions

unknown coefficients $= a_1, a_2, \cdots a_n, b_1, b_2, \cdots b_n \cdots$

2. Form all unknown coefficients and exponents as real-parameter chromosomes.

3. Sequentially put each pair of training data set (i.e., $x_1, x_2, \ldots x_n$ and f_i) into Equation (5.1) to calculate the error from each chromosome. The chromosome of smaller summation of errors is assigned to hold higher fitness value.
4. Reproduce the chromosome with high fitness values into the mating pool.
5. Recombination (or crossover) to generate new offspring.
6. Mutation to escape from the local minima.
7. Elite-preservation strategy to keep the optimal chromosomes in each generation.

5.5.2.2 Procedures for Method II

1. Involve symbol genes [+, -, *, /, ^, …] and terminal genes [*constants*, *pairs of input data*, …] together to randomly form reasonable mathematical expression, for example as

$$f_i = \frac{x \cdot z}{y} + x \cdot x - \frac{z}{xy} + 0.234 \tag{5.2}$$

2. Except for no mutation, rest of the steps are the same as those used in method 1

After generation by generation, the optimal process model with minimal summation errors should be generated.

5.5.2.3 Differences between ANFIS, NN and GA Models

No matter what method is used, an explicit and reasonable mathematical formula may be generated to describe an unknown system which we want to model. According to the generated mathematical expression, some important physical meanings or special

relationships between the input variables and output performances can be explained or found, if the generated GA model is accurate (i.e., training error and testing error are small enough). Undoubtedly, it is one of the most important advantages of GA which deserves to be emphasized. Even accurate ANFIS or NN models cannot provide this valuable information.

Clearly, to a GA modeling mentioned here, the need for a reliable pre-determined "formula" is a key requirement to Method I. In other words, if the other analyzing methods, such as statistical analysis, can provide a useful basic format first, what GA modeling does can be viewed as a fine-tuning process for all adjustable coefficients to search the best or optimal combination sets.

5.6 Process Optimization Using NSGA-II

After successfully modeling a complicated system, such as the CMP process, one immediate problem that follows is "*what are the available ranges of input settings to create optimal output performances* (i.e., higher MRR and lower WIWNU)?" Obviously, it is also a typical multi-objective optimization problem (MOOP). The models of NN and ANFIS are "*black boxes*" (as depicted in Figure 5.4), devoid of explicit mathematical expressions between inputs and outputs. Therefore, if those models are used as the objective functions, we cannot employ traditional methods (e.g. gradient-based methods, etc.) to solve multi-objective optimization problems (MOOP). Multi-Objective Evolutionary Algorithms (MOEA) are the best choices. Simulation results from these

techniques can guide us to search the available ranges of input settings for optimal output performance.

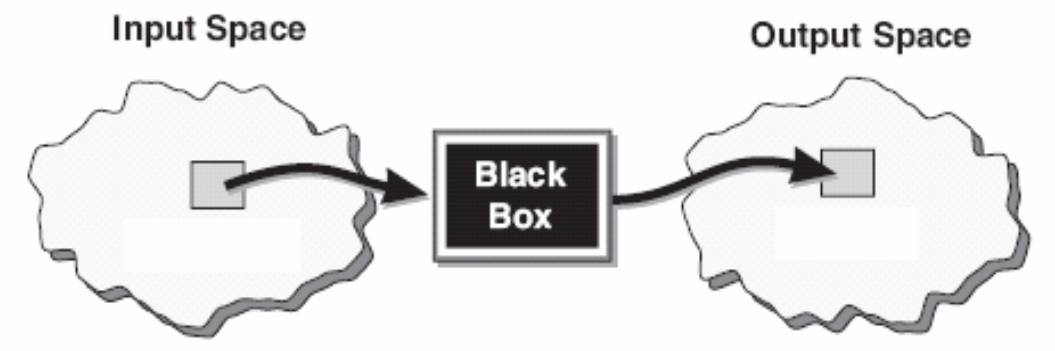


Figure 5.4 Black Box between Input Space and Output Space [2]

In this investigation, the *NSGA-II* (Elitist Non-dominated Sorting Genetic Algorithm) in MOEA was employed to search for the optimal input settings. Initially, a number of (e.g., 50 ~ 100) chromosomes, each with the format as shown in Figure 5.5, were randomly generated form an initial population. To some modeling tools (e.g., NN or GA), the genes in each chromosome should be scaled to a specific range, such as [-1 , 1]. To ANFIS, this step is not necessary. Thereafter, through the evolutionary operations detained in Section 4.6.5, a set of Pareto-optimal solutions (shown as the red diamond points along the Pareto front in Figure 4.51) are identified. According to these simulation solutions, confirmation experiments (i.e., the third-stage experiment in Case III) can be undertaken to verify the optimal input settings.

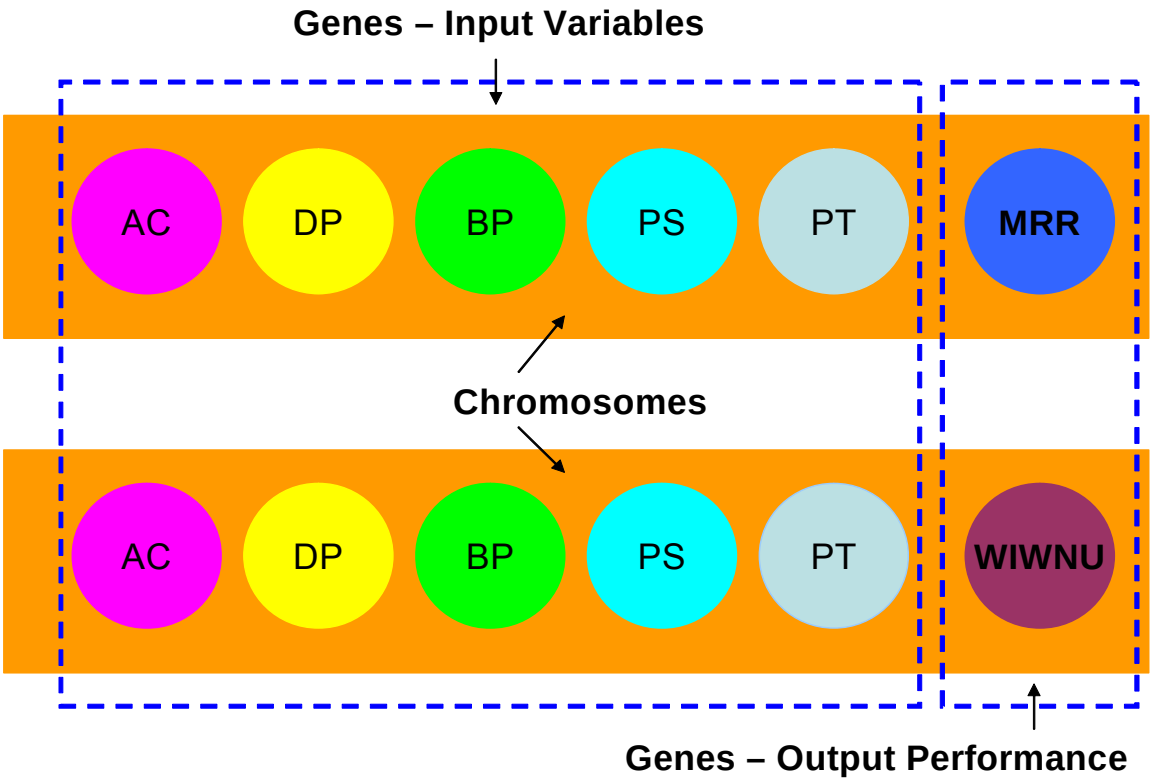


Figure 5.5 Chromosome Format for NSGA-II

CHAPTER VI

EXPERIMENT RESULTS AND ANALYSES

In this chapter, MRR and WIWNU in CMP process ware modeled and simulated using NN, ANFIS-GP, ANFIS-SC, and GA. Each of these models was constructed using the training data sets, and tested (with testing data sets) to investigate their accuracy and generalization capacities. The results from three different CMP experimental data cases (described in the Section 5.2) are presented in Sections 6.1 through 6.3 in this chapter. A fine-tuning technique for ANFIS-SC, models specifically suited for sparse data sets, is presented in Section 6.3.5. Section 6.4 includes discussion and summary of results from the above three sections. Comparison results, plots of MRR and WIWNU models are shown in the sections that subsequently follow.

6.1 MRR and WIWNU Models and Optimization of Process Parameters Using Case I Experimental Data

6.1.1 NN, ANFIS-GP, and ANFIS-SC Modeling

From the 27 source data sets [79, 81] (as summarized in Table A-4 of Appendix III), 22 data sets were randomly chosen as training data sets for obtaining NN and ANFIS models for capturing MRR and WIWNU characteristics. The remaining 5 data sets were used for testing the respective trained models. As described in the Section 5.4.3, fine-tuning procedures for membership functions were employed in the ANFIS-SC models

ANFIS-SC models for both MRR and WIWNU used 20 fuzzy rules (i.e., 20

membership functions in each input variable.) The prediction results are summarized in Tables 6.1 and 6.2 for MRR and WIWNU models, respectively. The results show that ANFIS-SC models for both MRR and WIWNU are more accurate compared to NN models as exemplified by a 50% reduction in testing errors compared to NN models and over three orders of magnitude reductions in the training errors. Figures 6.1 to 6.8 show the variations of model predictions relative to actual experimental prediction of MRR and WIWNU, respectively, with NN, ANFIS-GP and ANFIS-SC modeling.

Table 6.1 Statistical Mean and Standard Deviations of Training and Testing Errors from NN and ANFIS Models for **MRR** in CMP Process (Case I Experiment)

Results \ NN and ANFIS Model Types		NN 5-3-1 purelin	NN 5-3-1 tansig	ANFIS-GP 2-2-2-2-2	ANFIS-SC 20 Rules
22 Training Error %	Mean	14.55%	7. 70%	5.17%	~ 0 %
	Standard deviation	11.83%	7.04%	7.34%	~ 0 %
5 Testing Error %	Mean	14.25%	11.99%	6.71%	4.07%
	Standard deviation	9.88%	8.25%	8.33%	3.47%

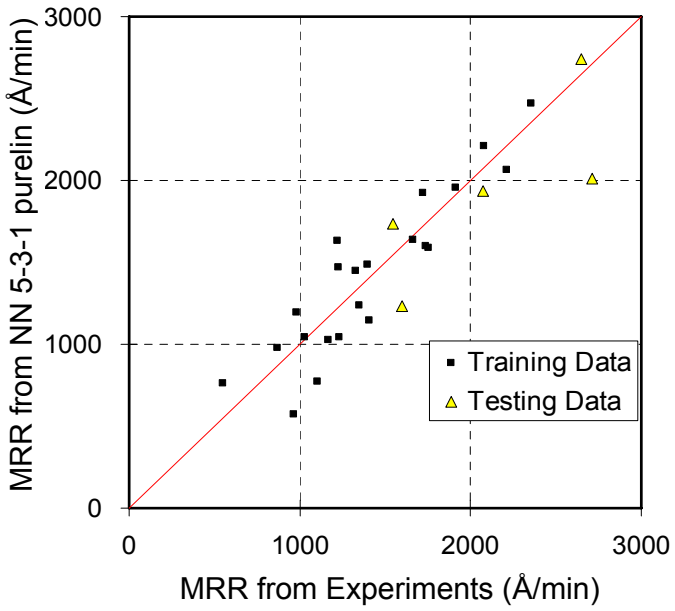


Figure 6.1 Results of NN 5-3-1 purelin Model of **MRR** (Case I Experiment)

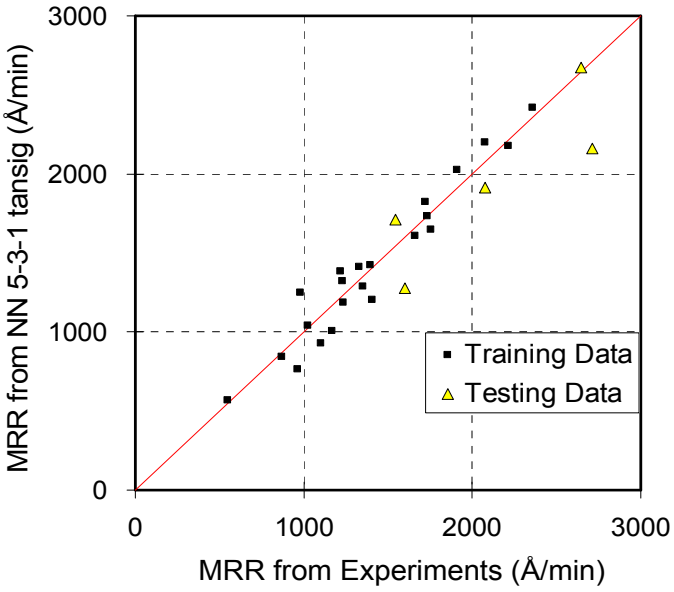


Figure 6.2 Results of NN 5-3-1 tansig Model of **MRR** (Case I Experiment)

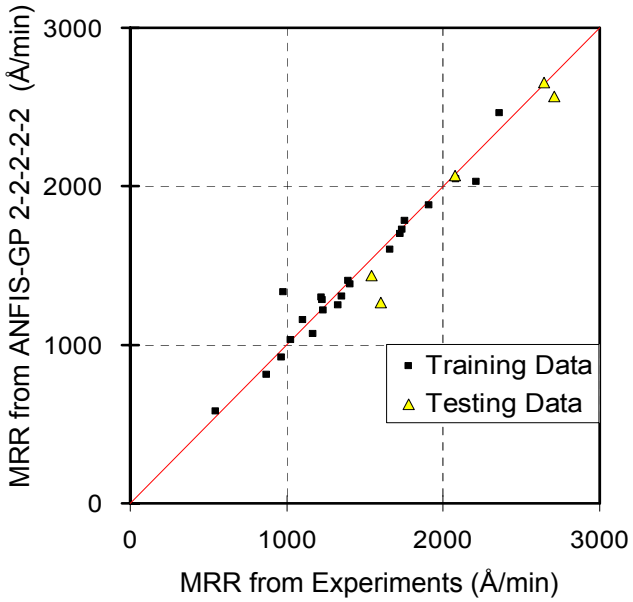


Figure 6.3 Results of ANFIS-GP 2-2-2-2-2 Model of **MRR** (Case I Experiment)

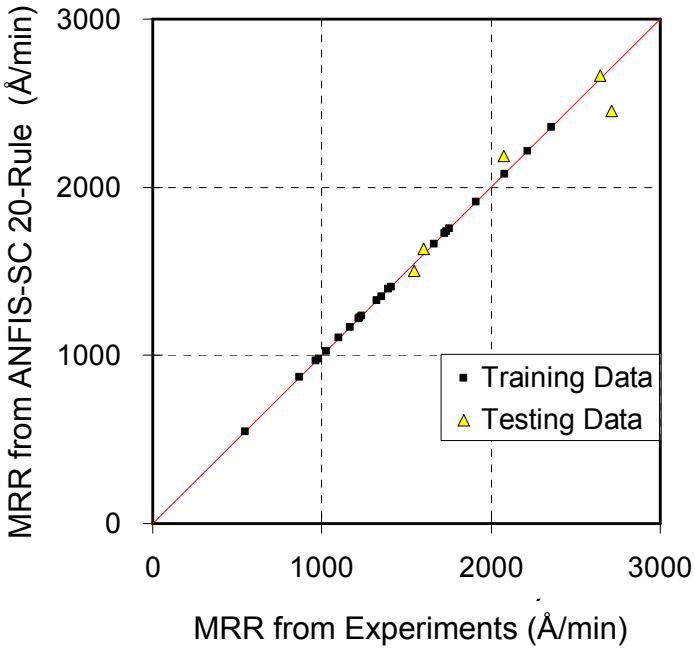


Figure 6.4 Results of ANFIS-SC 20 Rules Model of **MRR** (Case I Experiment)

Table 6.2 Statistical Mean and Standard Deviations of Training and Testing Errors from NN and ANFIS Models for **WIWNU** in CMP Process (Case I Experiments)

Results \ NN and ANFIS Model Types		NN 5-4-1 purelin	NN 5-4-1 tansig	ANFIS-GP 2-2-2-2-2	ANFIS-SC 20 Rules
22 Training Error %	Mean	26.27%	26.52%	10.29%	~ 0 %
	Standard deviation	24.50%	24.36%	14.85%	~ 0 %
5 Testing Error %	Mean	14.96%	16.96%	18.85%	4.55%
	Standard deviation	3.83%	2.95%	13.15%	1.58%

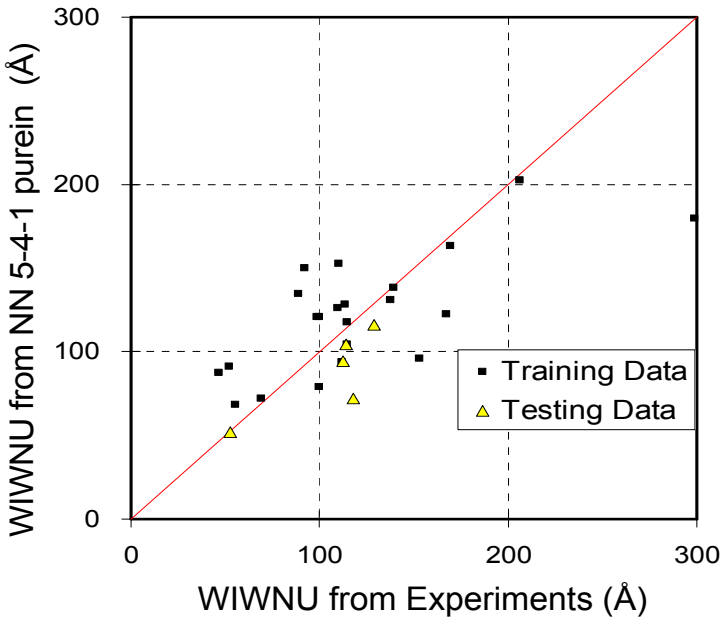


Figure 6.5 Results of NN 5-4-1 purelin Model of **WIWNU** (Case I Experiment)

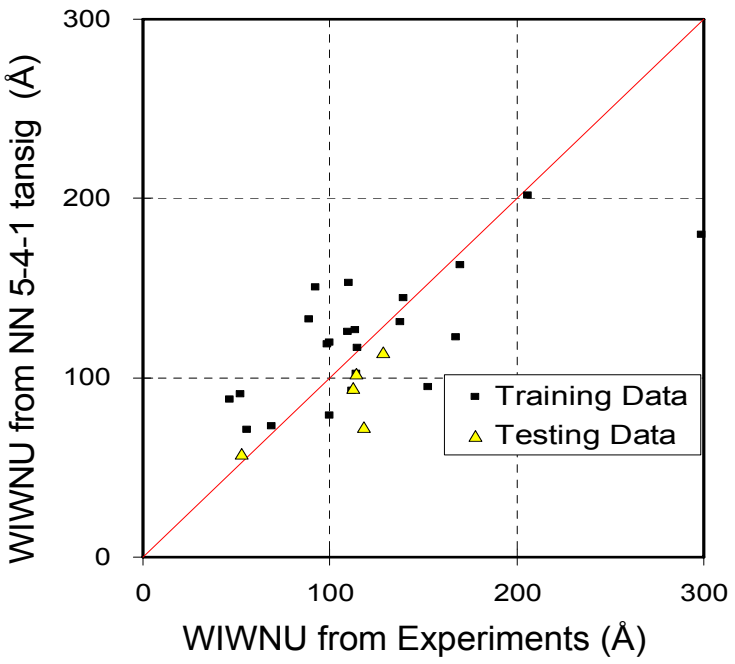


Figure 6.6 Results of NN 5-4-1 tansig Model of **WIWNU** (Case I Experiment)

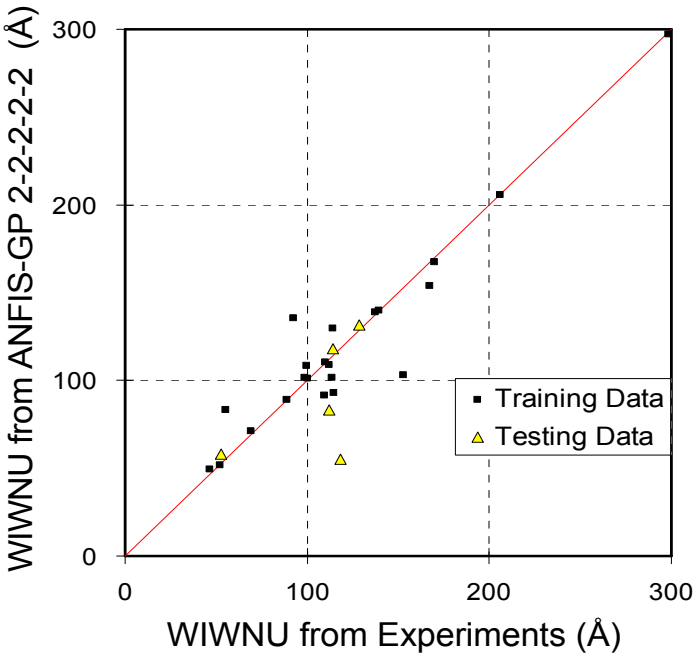


Figure 6.7 Results of ANFIS-GP 2-2-2-2-2 Model of **WIWNU** (Case I Experiment)

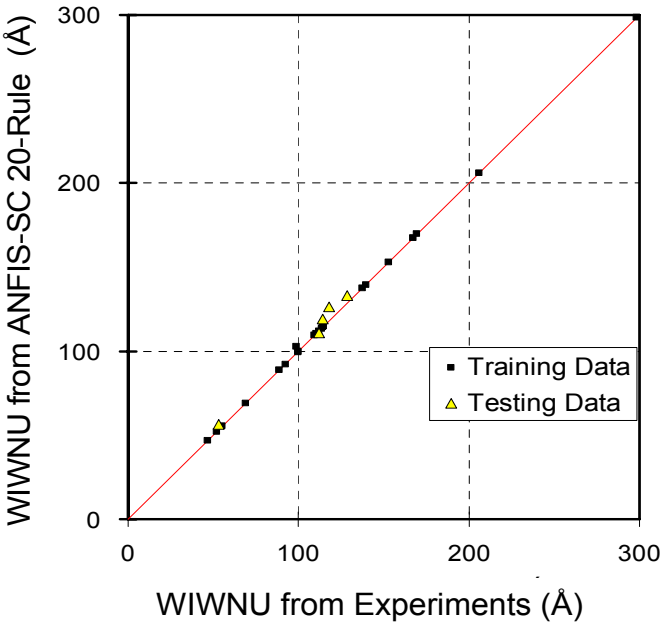


Figure 6.8 Results of ANFIS-SC 20 Rules Model of **WIWNU** (Case I Experiment)

6.1.2 GA Modeling from Case I Experimental Data

6.1.2.1 MRR Models

GA model for MRR was trained using 25 training data sets (as summarized in Table A-4 of Appendix III, row No. 1 ~ 25). The results are summarized in Figures 6.9 and 6.10. The minimum average error of 9.69% resulted after 1601 iterations and the GA Model for MRR is given by

$$\begin{aligned} \boldsymbol{MRR} &= \text{constant} \cdot \boldsymbol{Sc}^{a} \cdot \boldsymbol{Pd}^{b} \cdot \boldsymbol{Vp}^{c} \cdot \boldsymbol{Pb}^{d} \cdot \boldsymbol{T}^{e} \\ &= \frac{0.9547 \cdot \boldsymbol{Sc}^{0.2938} \cdot \boldsymbol{Pd}^{0.4957} \cdot \boldsymbol{Vp}^{0.3485}}{\boldsymbol{Pb}^{0.0230} \cdot \boldsymbol{T}^{0.0453}} \end{aligned} \tag{6.1}$$

It may be noted that inputs were linearly coded between 1 and 3. Their results match with the analytical explanations of CMP process given in Chapter III. Consequently, solid content (***Sc***), down pressure (***Pd***), and platen speed (***Vp***) are dominant factors in the

polishing process.

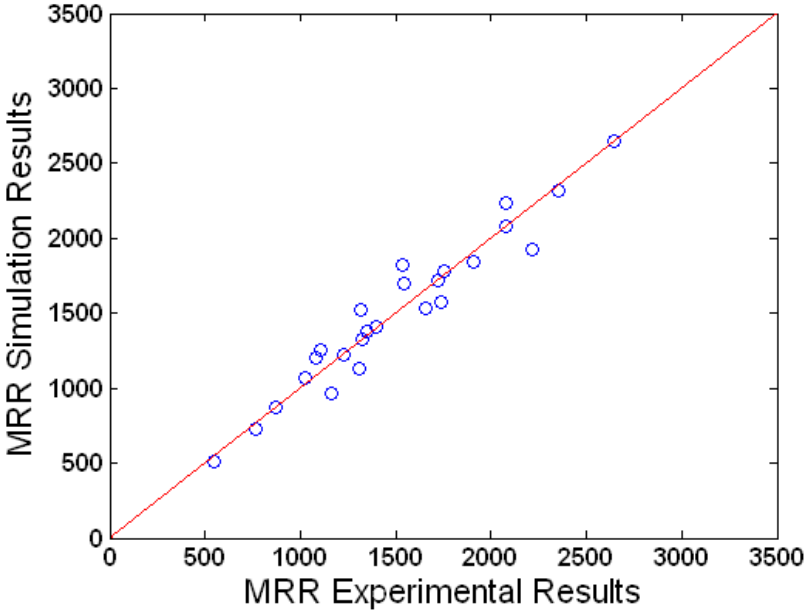


Figure 6.9 Comparisons of Simulation Results and Experimental Data for MRR

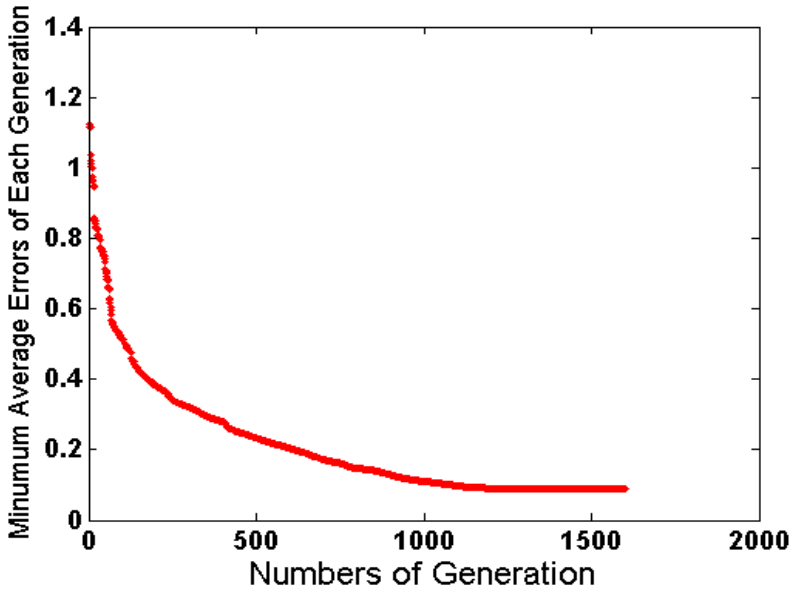


Figure 6.10 Minimal Average Training Errors for MRR – GA Model

6.1.2.2 WIWNU Models

Two WIWNU GA models were trained using the same 25 training data sets used in the previous MRR case. The results are summarized in Figures 6.11 to 6.14. The minimum average errors of 20.11% and 11.66% resulted, respectively. Both GA models I and II for WIWNU are given by Equations (6.2) and (6.3). Inputs were again linearly coded between 1 and 3. The results in Equation (6.2) show increasing the back pressure (***Pb***) can significantly decrease WIWNU.

$$\begin{aligned} \boldsymbol{WIWNU} &= \text{constant} \cdot \boldsymbol{Sc}^{a} \cdot \boldsymbol{Pd}^{b} \cdot \boldsymbol{Pb}^{c} \cdot \boldsymbol{Vp}^{d} \cdot \boldsymbol{T}^{e} \\ &= \frac{1.3057 \cdot \boldsymbol{Sc}^{0.5698} \cdot \boldsymbol{Pd}^{0.1100} \cdot \boldsymbol{Vp}^{0.1624} \cdot \boldsymbol{T}^{0.1264}}{\boldsymbol{Pb}^{0.4322}} \end{aligned} \tag{6.2}$$

$$\begin{aligned} \boldsymbol{WIWNU} &= \frac{\text{constant} \cdot \boldsymbol{Pb}^{a}}{\boldsymbol{Sc}^{b} \cdot \boldsymbol{Pd}^{c} \cdot \boldsymbol{Vp}^{d} \cdot \boldsymbol{T}^{e}} + \left(\boldsymbol{Sc}^{f} + \boldsymbol{Pd}^{g} + \boldsymbol{Pb}^{h} + \boldsymbol{Vp}^{i} + \boldsymbol{T}^{j} \right) \\ &= \frac{-3.6962 \cdot \boldsymbol{Pb}^{0.5214}}{\boldsymbol{Sc}^{1.2153} \cdot \boldsymbol{Pd}^{0.5099} \cdot \boldsymbol{Vp}^{0.3180} \cdot \boldsymbol{T}^{0.4905}} \\ &\quad + \left(\frac{1}{\boldsymbol{Sc}^{1.5989}} + \frac{1}{\boldsymbol{Pd}^{1.2761}} + \frac{1}{\boldsymbol{Pb}^{0.6463}} + \frac{1}{\boldsymbol{Vp}^{0.4038}} + \frac{1}{\boldsymbol{T}^{0.68524}} \right) \end{aligned}$$

(6.3)

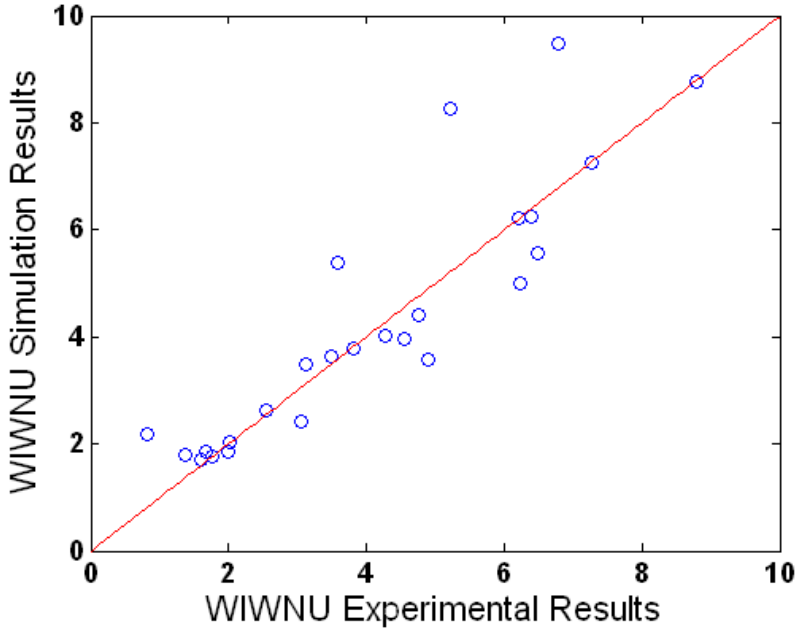


Figure 6.11 Comparison of Simulation Results and Experimental Data for WIWNU – Model I

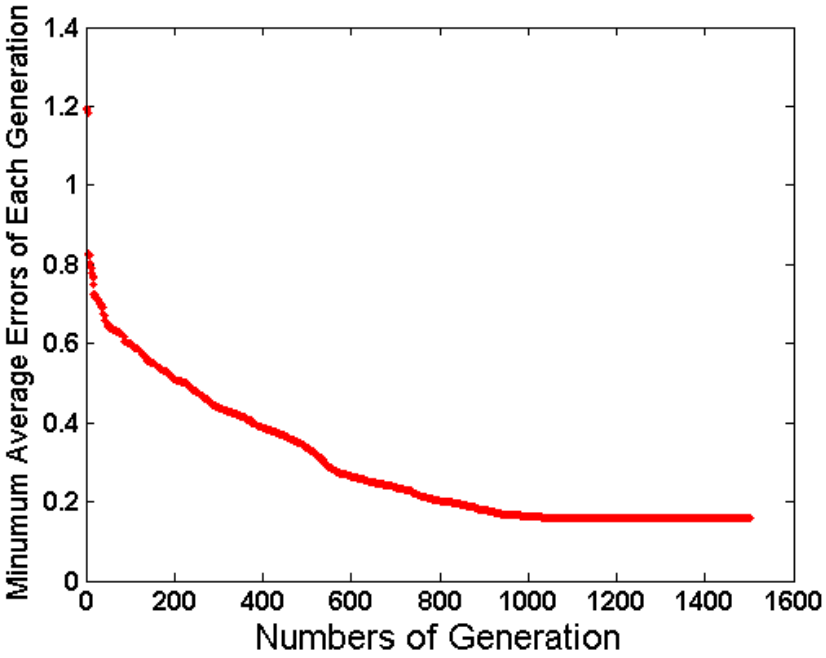


Figure 6.12 Minimal Average Training Errors for WIWNU – GA Model I

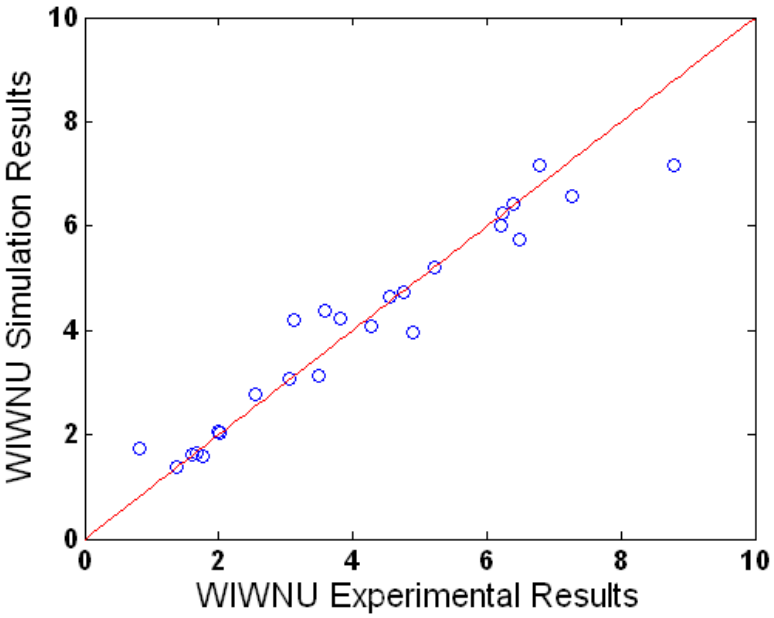


Figure 6.13 Comparisons of Simulation Results and Experimental Data for WIWNU – GA Model II

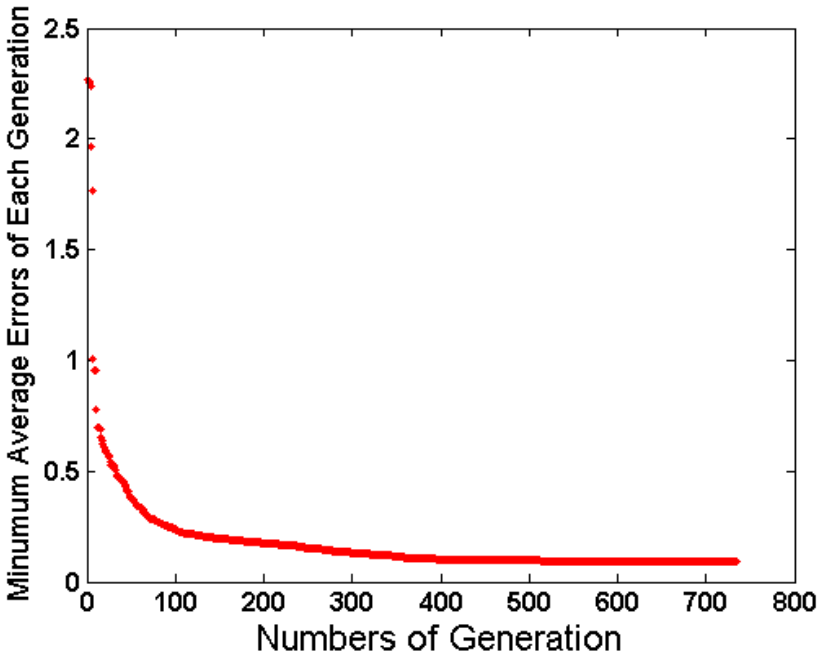


Figure 6.14 Minimal Average Training Errors for WIWNU – GA Model II

6.1.3 Optimal MRR and WIWNU in CMP Processes

The MRR and WIWNU discussed above are typical multi-objective optimization problems. Usually, higher MRR (or higher production rate) often conflicts with lower WIWNU (or higher quality assurance) in the CMP process. In fact, this is an unsolved challenge to the current semiconductor industry. Apart from how to precisely model them, how to find the balance points is the other critical issue. In this research, the process models, constructed by GA as stated in the previous section, will be used as the objective functions and simultaneously apply MOEA techniques to search for the non-dominated output results. On the basis of searched results and ranges of input settings, the advanced experimental plan can be re-designed to focus on the finer ranges for all input variables to obtain the optimal output performance of MRR and WIWNU. Also, new robust process models can be created.

In the case of MRR and WIWNU , the *NSGA-II* (Elitist Non-Dominated Sorting Genetic Algorithms) [1, 12, 129-131] was applied to search for the optimal solutions for these two objectives. Equations (6.3) and (6.4) are used as the objective functions for this multi-objective evolutionary optimization. Figure 6.15 shows the Pareto-front (non-dominated solutions) after 1000 iterations. The optimal input-and-output solutions (41 red points in Figure 6.15) along with the Pareto-front are given in Table 6.3 (including 20 Pareto-optimal input settings). These points can be viewed as the guidance to re-design the advanced experiments to search for the real optimal input settings and then verify the real output performance.

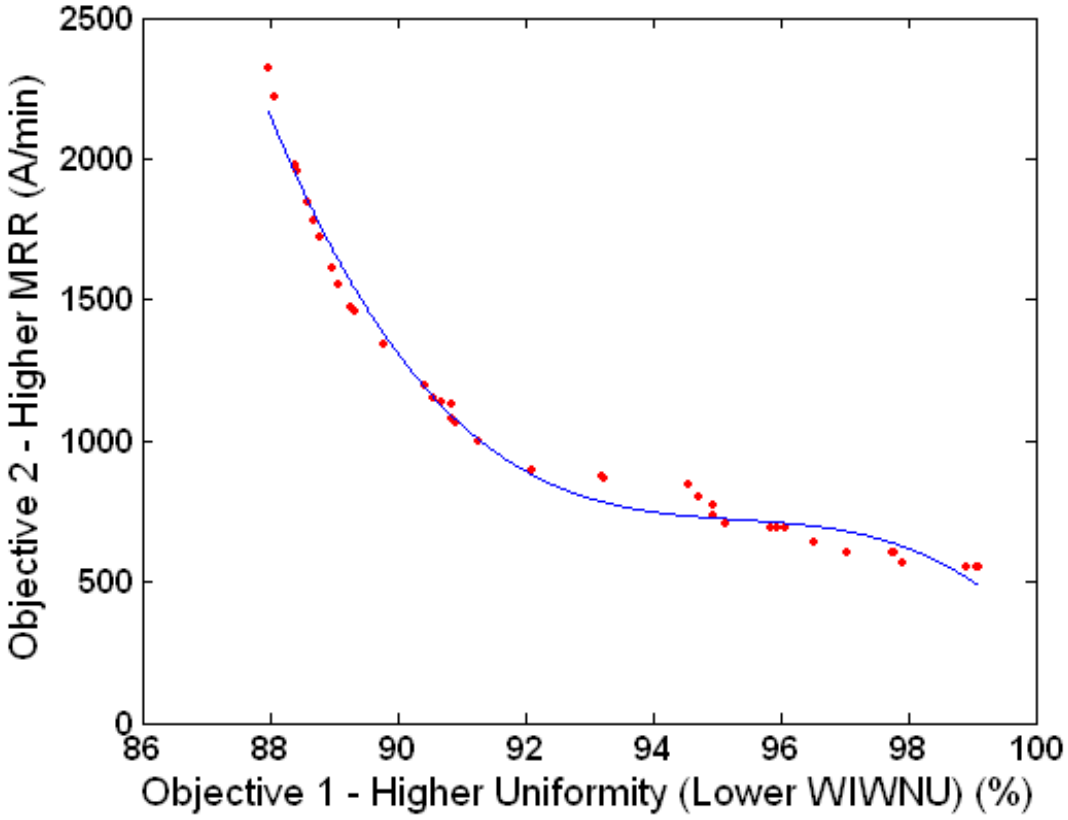


Figure 6.15 Pareto-front of Optimal MRR and Uniformity (or Lower WIWNU)

Table 6.3 Optimal Input-and Output Results for CMP Processes Using NSGA-II

Optimal Input settings	Solid Content (*Sc*) wt%	Down Prossure (*Pd*) psi	Back Pressure (*Pb*) psi	Platen Speed (*Vp*) rpm	Polishing Time (*T*) sec	MRR (Å/min)	WIWNU (%)
1	5.81	4.01	0.27	30.00	58.97	893.10	7.90
2	5.51	5.00	0.31	21.36	59.92	872.70	6.80
3	5.07	5.00	2.83	20.01	59.93	864.70	6.80
4	11.42	4.27	0.08	20.72	59.35	842.70	5.40
5	9.76	6.56	0.40	21.38	57.26	798.40	5.30
6	13.76	4.01	0.08	22.26	59.35	768.90	5.10
7	24.98	7.00	0.00	47.96	58.21	733.80	5.00
8	24.98	7.00	0.00	56.76	59.90	704.80	4.80
9	24.98	6.37	0.00	56.76	59.90	692.10	4.10
10	10.00	6.37	0.88	56.76	59.90	690.50	4.00
11	23.61	5.00	0.01	59.73	59.96	689.60	3.90
12	10.00	5.00	0.01	59.73	59.96	641.30	3.50
13	11.42	4.01	0.88	25.12	59.35	605.40	3.00
14	24.47	4.06	0.01	29.99	59.11	604.30	3.00
15	24.30	4.64	0.01	36.06	59.97	599.90	2.20
16	24.30	5.00	0.01	36.06	54.12	599.60	2.20
17	24.96	4.26	0.00	34.99	45.00	564.50	2.10
18	24.96	4.26	0.24	34.99	59.96	555.50	1.10
19	12.48	4.91	0.33	21.36	58.12	553.70	0.90
20	10.16	4.08	0.01	35.50	59.96	553.30	0.90

6.2 NN and ANFIS Models of MRR & WIWNU Using Case II Experimental Data

From 54 source data sets [79-81] (as tabulated in Table A-5 of Appendix III), 45 data sets were randomly chosen to be the training data for training NN and ANFIS

models and the remaining 9 data sets for testing the trained models. Two NN models (i.e., purelin and tansig) and two ANFIS models (i.e., grid partition (GP) and subtractive clustering (SC)) were used to develop models for MRR and WIWNU, respectively. Comparison of results and plots of MRR and WIWNU models are given in Tables 6.4 and Table 6.5, and Figure 6.16 to Figure 6.23.

Fine-tuning procedures for membership functions were employed for ANFIS-SC models for MRR and WIWNU. Here, 45 fuzzy rules in ANFIS-SC models for MRR and 48 rules for WIWNU models were used. Again, ANFIS-SC models for both MRR and WIWNU perform much more accurately than other ANFIS and NN models.

6.2.1 MRR Models

Table 6.4 Statistical Mean and Standard Deviations of Training and Testing Errors from NN and ANFIS Models for **MRR** in CMP Process (Case II Experiment)

Results \ NN and ANFIS Model Types		NN 7-4-1 purelin	NN 7-3-1 tansig	ANFIS-GP 2-2-1-3-1-1-2	ANFIS-SC 45 Rules
45 Training Error %	Mean	21.00%	11.03%	13.04%	~ 0 %
	Standard deviation	19.66%	8.42%	16.04%	~ 0 %
9 Testing Error %	Mean	17.50%	15.23%	15.16%	3.71%
	Standard deviation	11.42%	11.52%	12.32%	1.64%

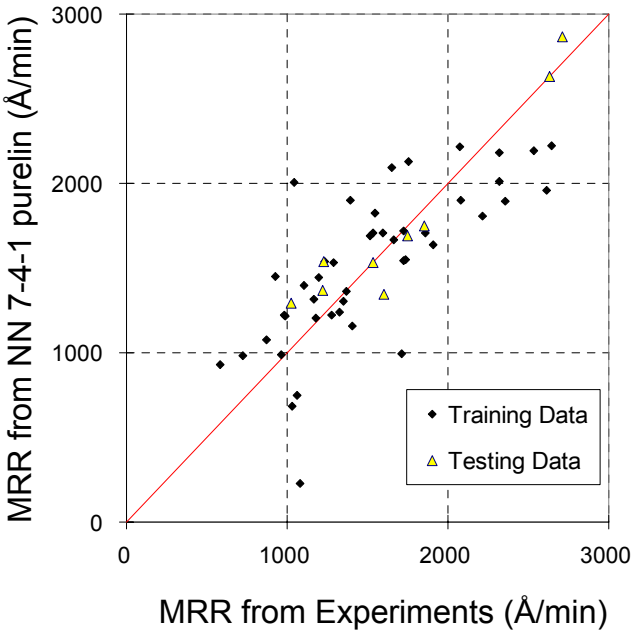


Figure 6.16 Results of NN 7-4-1 purelin Model of **MRR** (Case II Experiment)

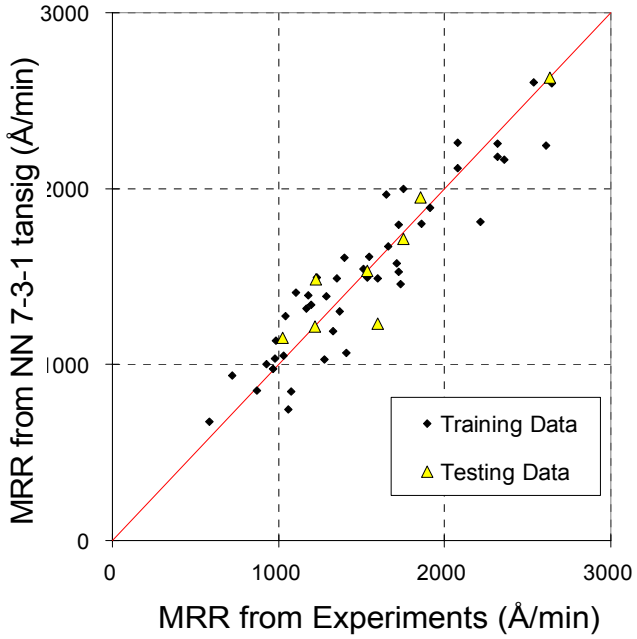


Figure 6.17 Results of NN 7-3-1 tansig Model of **MRR** (Case II Experiment)

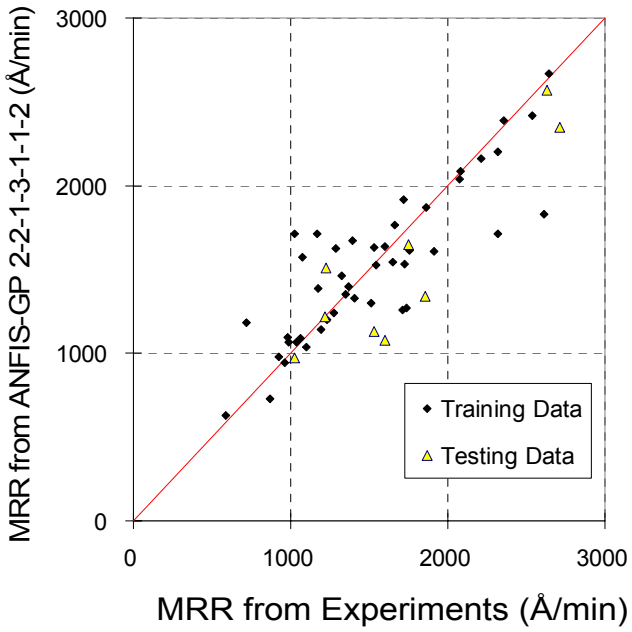


Figure 6.18 Results of ANFIS-GP 2-2-1-3-1-1-2 Model of **MRR** (Case II Experiment)

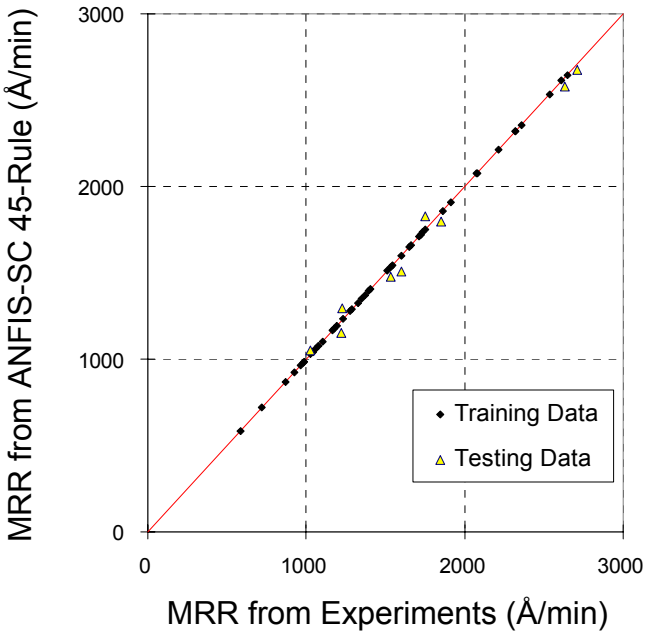


Figure 6.19 Results of ANFIS-SC 45 Rules Model of **MRR** (Case II Experiment)

6.2.2 WIWNU Models

Table 6.5 Statistical Mean and Standard Deviations of Training and Testing Errors from NN and ANFIS Models for **WIWNU** in CMP Process (Case II Experiment)

Results \ NN and ANFIS Model Types		NN 7-4-1 purelin	NN 7-4-1 tansig	ANFIS-GP 2-2-2-3-2-2-2	ANFIS-SC 48 Rules
45 Training Error %	Means	37.85%	15.98%	5.82%	~ 0 %
	Standard deviations	44.18%	20.39%	13.75%	~ 0%
9 Testing Error %	Means	81.67%	47.00%	54.24%	3.93%
	Standard deviations	166.99%	94.81%	78.08%	2.68%

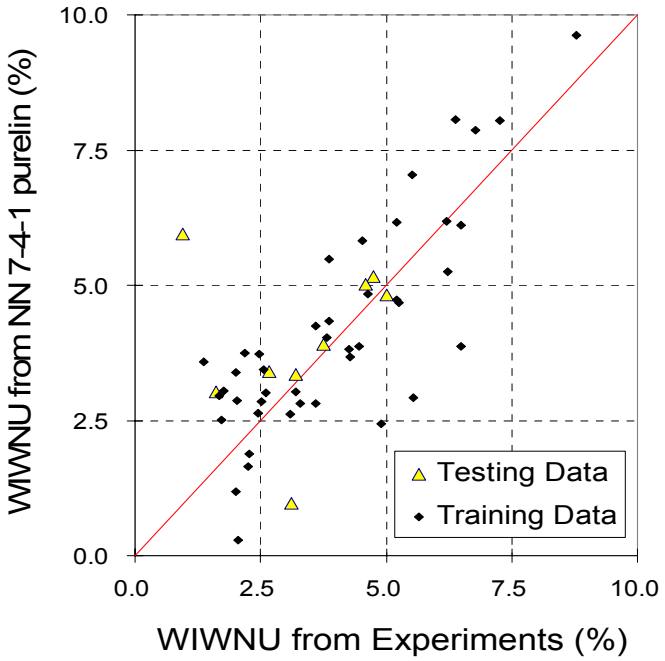


Figure 6.20 Results of NN 7-4-1 purelin Model of **WIWNU** (Case II Experiment)

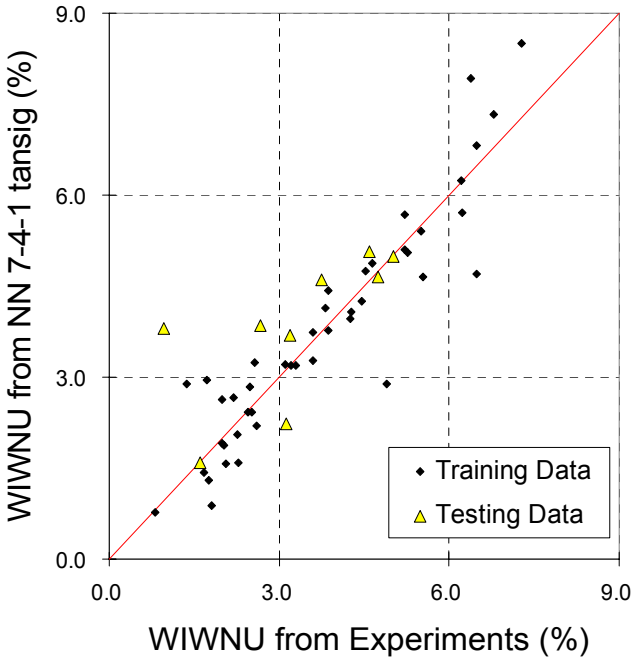


Figure 6.21 Results of NN 7-4-1 tansig Model of **WIWNU** (Case II Experiment)

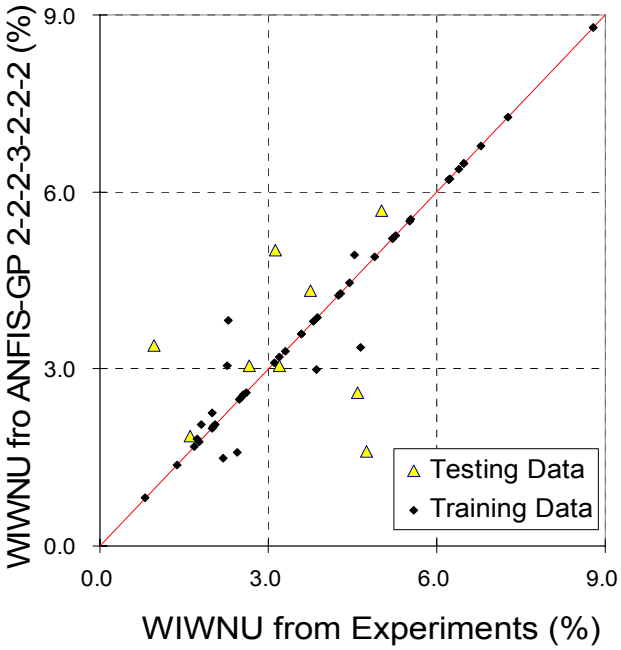


Figure 6.22 Results of ANFIS-GP 2-2-2-3-2-2-2 Model of **WIWNU** (Case II Experiment)

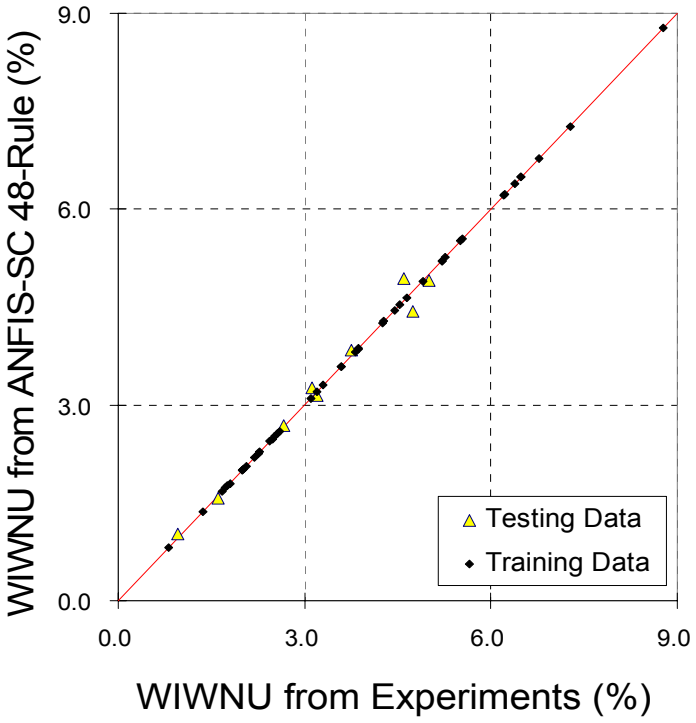


Figure 6.23 Results of ANFIS-SC 48 Rules Model of **WIWNU** (Case II Experiment)

6.3 MRR and WIWNU Models and Process Optimization Using Case III Experimental Data

As reported in Section 5.2.3, 1875 training data sets and 125 testing data sets generated via a factorial design of experiments (DOE), were used for MRR and WIWNU modeling to investigate the influence of the size of training data sets on the generalization of trained models, four different groups (I, II, III and IV) of training data sets were used to construct the MRR and WIWNU models.

Group I contains all the 1875 training data sets. Group II contains 512 data sets from the original 1875 training data sets as shown in the Table 6.6.

Table 6.6 Factorial DOE for Group II Case

Input Variables / Levels	Solid Content (***Sc***) weight %	Down Pressure (***Pd***) psi	Back Pressure (***Pb***) psi	Platen Speed (***Vp***) rpm	Polishing Time (***T***) sec
Level 1	5	4	0	20	40
Level 2	10	6	1	30	60
Level 3	20	10	3	50	
Level 4	25	12	4	60	

Group III consists of only 162 out of the original 1875 data sets for training as shown in Table 6.7. Finally, to these three groups, the same testing data sets were employed to test the established models. The 25 training data sets in Group IV were picked from the 1875 original training data sets based on $\boldsymbol{L}_{25}$ Taguchi array. The resulting data sets are identical to the ones in Case I experiment and listed in Table A-6 in the Appendix IV.

Table 6.7 Factorial DOE for Group III case

Input Variables / Levels	Solid Content (***Sc***) weight %	Down Pressure (***Pd***) psi	Back Pressure (***Pb***) psi	Platen Speed (***Vp***) rpm	Polishing Time (***T***) sec
Level 1	5	4	0	20	40
Level 3	15	8	2	40	60
Level 5	25	12	4	60	

Since the smallest training and testing errors resulted with the ANFIS-GP models of MRR and WIWNU, these models were chosen to generate the objective functions for NSGA-II. The third-stage confirmation experiments were implemented to verify the optimal results obtained using NSGA-II. Furthermore, an improved ANFIS-SC fine-tuning technique was applied to Group IV data sets to develop a new MRR model. Similarly, the original 125 testing data sets were used to test this new model.

6.3.1 Models Using ANFIS-GP

6.3.1.1 Results with Group I Data

Models of MRR Using ANFIS-GP 2-2-2-2-3 Linear for Group I Data

Architecture # 6 (as summarized in Table 6.8) was chosen to construct an ANFIS-GP model for Group I MRR case because the smallest *E_total* value and *Error ratio* values resulted for this case. The training and testing results are shown in Figures 6.24 and 6.25 and the percentage errors as follows.

Training error (%) = 2.0783 %

Testing error (%) = 2.2157 %

Standard deviaton of training error (%) = 1.4968 %

Standard deviaton of testing error (%) = 1.5322 %

Table 6.8 Comparisons of Different ANFIS-GP MRR Models (Group I)

MRR 1875	Structure (gaussian)	Type of Consequent part	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio (E_test/E_train)	Epochs
1	2-2-2-2-2	Linear	42.71	44.62	87.33	1.045	22
2	3-2-2-2-2	Linear	41.65	45.54	87.18	1.093	22
3	2-3-2-2-2	Linear	41.60	71.09	112.69	1.709	22
4	2-2-3-2-2	Linear	41.62	46.89	88.51	1.127	22
5	2-2-2-3-2	Linear	41.80	64.09	105.89	1.533	22
6	2-2-2-2-3	Linear	41.98	42.74	84.72	1.018	22
7	2-2-2-2-2	Constant	51.14	56.60	107.74	1.107	1000
8	3-2-2-2-2	Constant	50.92	56.92	107.85	1.118	1000
9	2-3-2-2-2	Constant	49.38	66.80	116.18	1.353	1000
10	2-2-3-2-2	Constant	50.80	55.99	106.78	1.102	1000
11	2-2-2-3-2	Constant	44.20	68.52	112.72	1.550	1000
12	2-2-2-2-3	Constant	50.89	51.20	102.09	1.006	1000

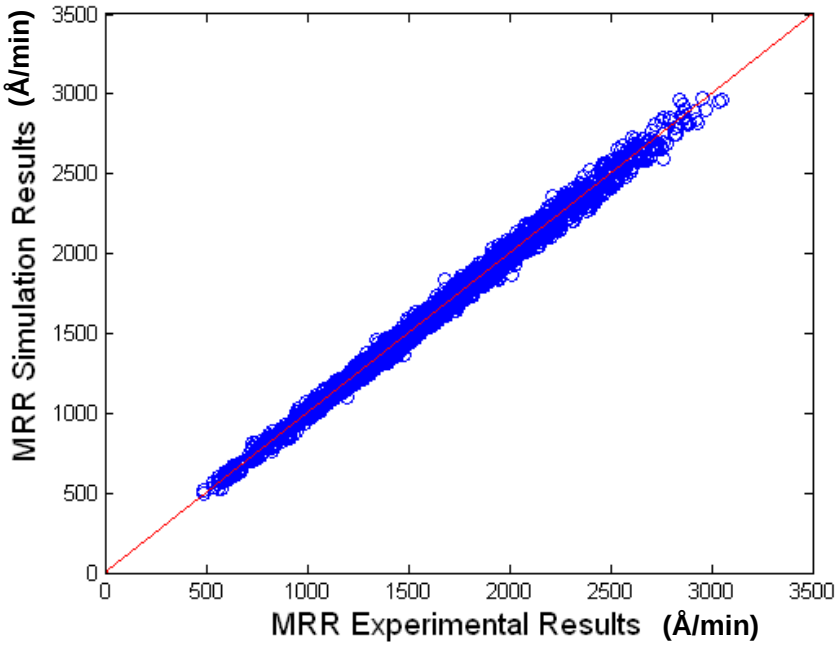


Figure 6.24 Training Results for MRR ANFIS-GP 2-2-2-2-3 Linear Model (Group I)

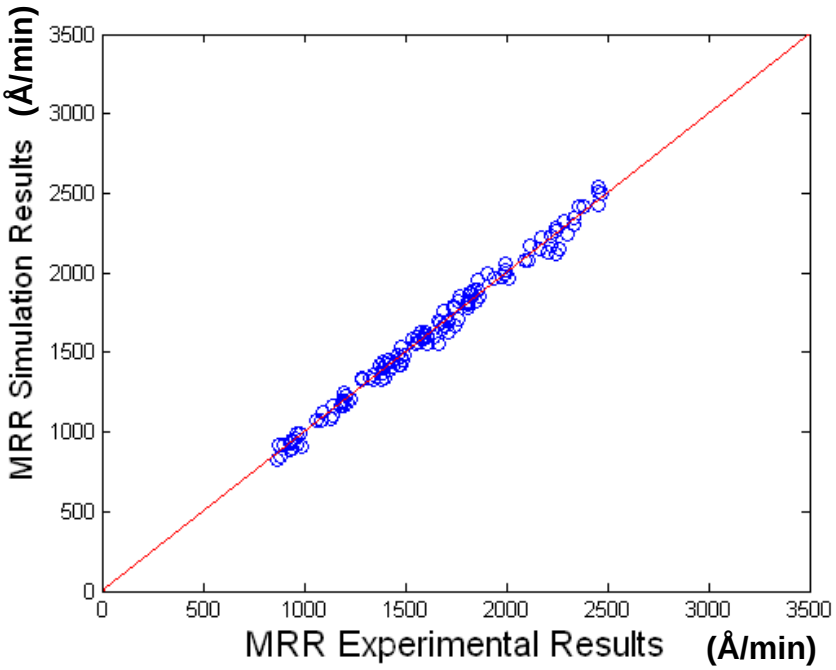


Figure 6.25 Testing Results for MRR ANFIS-GP 2-2-2-2-3 Linear Model (Group I)

Models of WIWNU Using ANFIS-GP 2-2-2-3-2 Linear for Group I Data

Architecture # 5 (as summarized in Table 6.9) was chosen to construct an ANFIS-GP model for Group I WIWNU case, because of the smaller *E_total* value and *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.26 and 6.27 and the percentage errors as follows.

Training error (%) = 9.1122 %

Testing error (%) = 9.6952 %

Standard deviation of training error (%) = 7.3268 %

Standard deviation of testing error (%) = 6.7805 %

Table 6.9 Comparisons of Different ANFIS-GP WIWNU Models (Group I)

WIWNU 1875	Structure (gaussian)	Type of Consequent part	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio (E_test/E_train)	Epochs
1	2-2-2-2-2	Linear	0.30	0.28	0.58	0.929	20
2	3-2-2-2-2	Linear	0.29	0.28	0.58	0.966	20
3	2-3-2-2-2	Linear	0.30	0.28	0.58	0.950	20
4	2-2-3-2-2	Linear	0.29	0.29	0.58	0.973	20
5	2-2-2-3-2	Linear	0.29	0.29	0.58	0.976	20
6	2-2-2-2-3	Linear	0.29	0.30	0.59	1.004	20
7	2-2-2-2-2	Constant	0.31	0.31	0.63	1.003	1000
8	3-2-2-2-2	Constant	0.31	0.31	0.63	1.007	1000
9	2-3-2-2-2	Constant	0.31	0.31	0.63	1.009	1000
10	2-2-3-2-2	Constant	0.31	0.33	0.64	1.058	1000
11	2-2-2-3-2	Constant	0.31	0.32	0.63	1.018	1000
12	2-2-2-2-3	Constant	0.31	0.26	0.57	0.843	1000

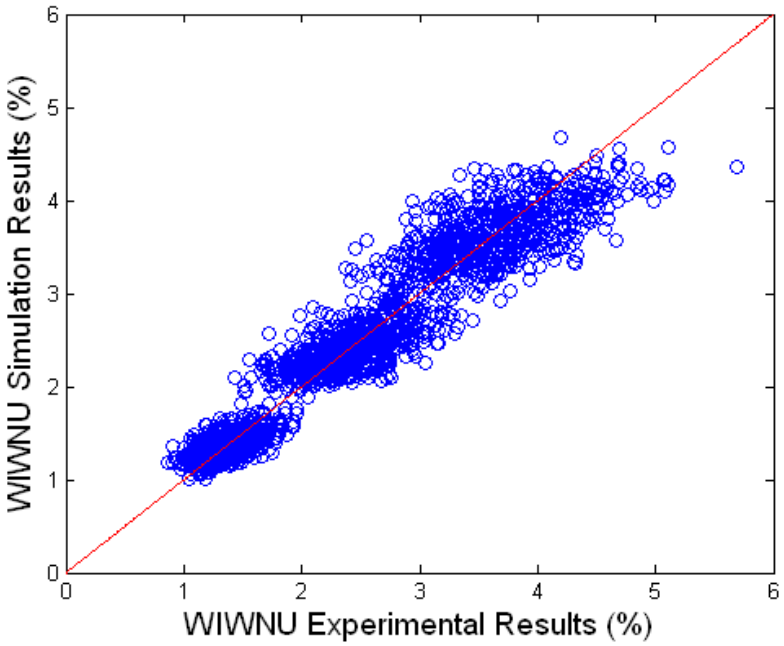


Figure 6.26 Training Results for WIWNU ANFIS-GP 2-2-2-2-3 Linear Model (Group I)

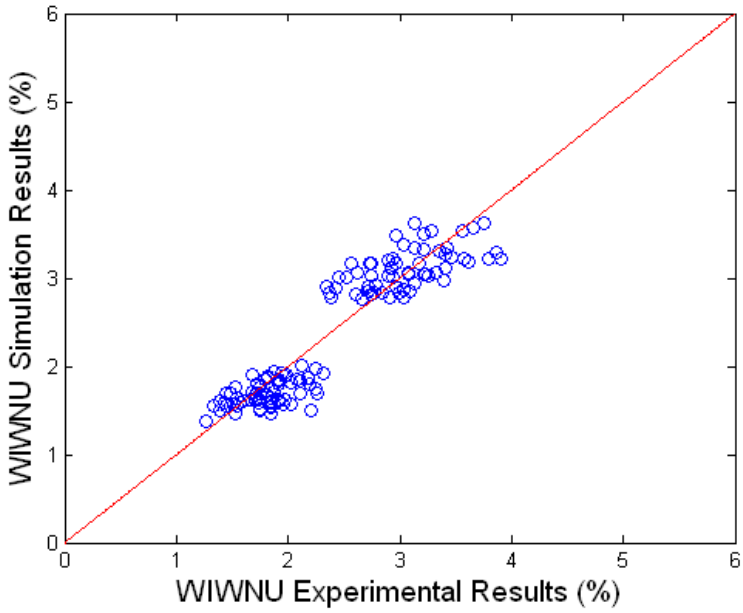


Figure 6.27 Testing Results for WIWNU ANFIS-GP 2-2-2-2-3 Linear Model (Group I)

6.3.1.2 Results with Group II Data

Models of MRR Using ANFIS-GP 2-2-2-2-2 Linear for Group II Data

Architecture # 1 (as summarized in Table 6.10) was chosen to construct an ANFIS-GP model for Group II MRR case, because of the smallest *E_total* value and *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.28 and 6.29 and the percentage errors as follows.

Training error (%) = 1.8051 %

Testing error (%) = 2.3884 %

Standard deviaton of training error (%) = 1.3543 %

Standard deviaton of testing error (%) = 1.7358 %

Table 6.10 Comparisons of Different ANFIS-GP MRR Models (Group II)

MRR 512	Structure (gaussian)	Type of Consequent part	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio (E_test/E_train)	Epochs
1	2-2-2-2-2	Linear	35.94	48.99	84.93	1.363	30
2	3-2-2-2-2	Linear	33.35	61.08	94.43	1.832	30
3	2-3-2-2-2	Linear	32.80	99.10	131.90	3.021	30
4	2-2-3-2-2	Linear	33.44	53.02	86.46	1.586	30
5	2-2-2-3-2	Linear	33.28	83.67	116.95	2.514	30
6	2-2-2-2-3	Linear	35.94	668.28	704.22	18.597	30
7	3-3-2-2-2	Constant	43.99	106.42	150.41	2.419	1000
8	3-2-2-2-2	Constant	48.23	57.56	105.79	1.194	1000
9	2-2-2-2-2	Constant	48.85	56.25	105.10	1.151	1000
10	4-3-2-2-2	Constant	43.26	107.14	150.40	2.477	1000
11	2-3-4-3-2	Constant	43.24	99.65	142.89	2.305	1000
12	2-4-4-2-3	Constant	42.51	67.97	110.48	1.599	1000

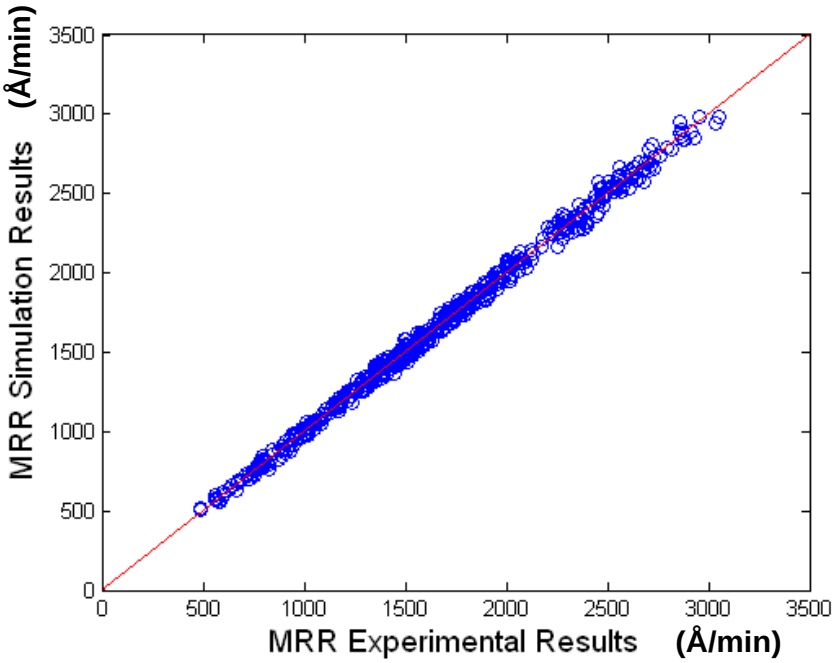


Figure 6.28 Training Results for MRR ANFIS-GP 2-2-2-2-2 Linear Model (Group II)

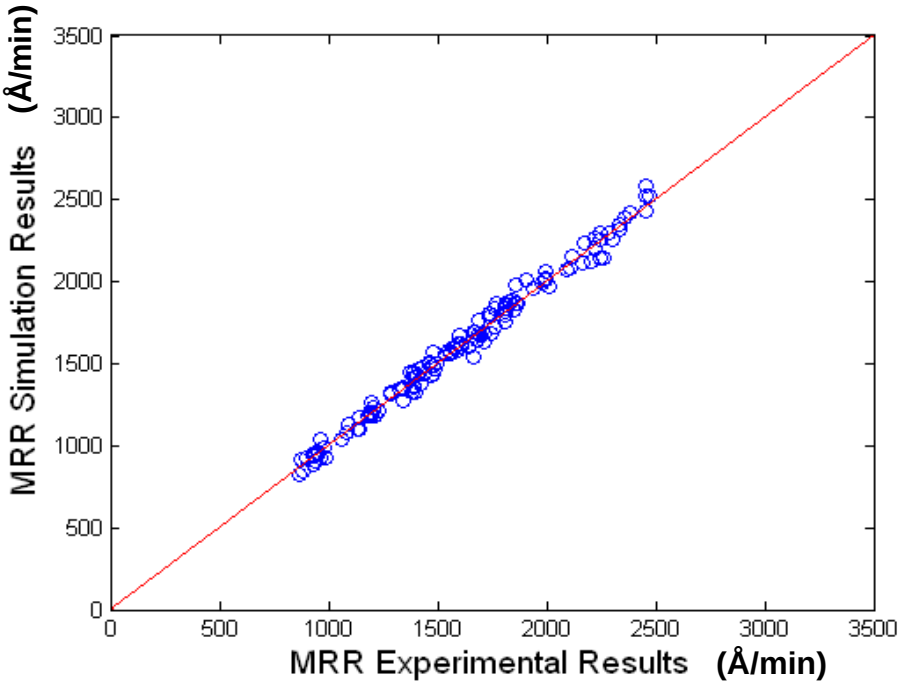


Figure 6.29 Testing Results for MRR ANFIS-GP 2-2-2-2-2 Linear Model (Group II)

Models of WIWNU Using ANFIS-GP 2-2-2-3-2 Linear for Group II Data

Architecture # 5 (as summarized in Table 6.11) was chosen to construct an ANFIS-GP model for Group II WIWNU case, because of the smallest *E_total* value and smaller *Error ratio* values resulted for this case. The training and testing results are summarized Figures 6.30 and 6.31 and the percentage errors as follows.

Training error (%) = 7.1631 %

Testing error (%) = 10.3907 %

Standard deviation of training error (%) = 5.5523 %

Standard deviation of testing error (%) = 7.8027 %

Table 6.11 Comparisons of Different ANFIS-GP WIWNU Models (Group II)

WIWNU 512	Structure (gaussian)	Type of Consequent part	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio (E_test/E_train)	Epochs
1	2-2-2-2-2	Linear	0.27	0.31	0.58	1.165	20
2	3-2-2-2-2	Linear	0.24	0.38	0.62	1.551	20
3	2-3-2-2-2	Linear	0.24	0.42	0.66	1.703	20
4	2-2-3-2-2	Linear	0.26	0.36	0.62	1.413	20
5	2-2-2-3-2	Linear	0.24	0.33	0.57	1.347	20
6	2-2-2-2-3	Linear	0.27	1.07	1.34	4.017	20
7	2-2-2-2-2	Constant	0.30	0.31	0.62	1.022	500
8	3-2-2-2-2	Constant	0.30	0.32	0.62	1.064	500
9	2-3-2-2-2	Constant	0.30	0.31	0.61	1.053	500
10	2-2-3-2-2	Constant	0.29	0.33	0.62	1.106	500
11	2-2-2-3-2	Constant	0.30	0.32	0.61	1.053	500
12	2-2-2-2-3	Constant	0.31	0.99	1.29	3.256	500

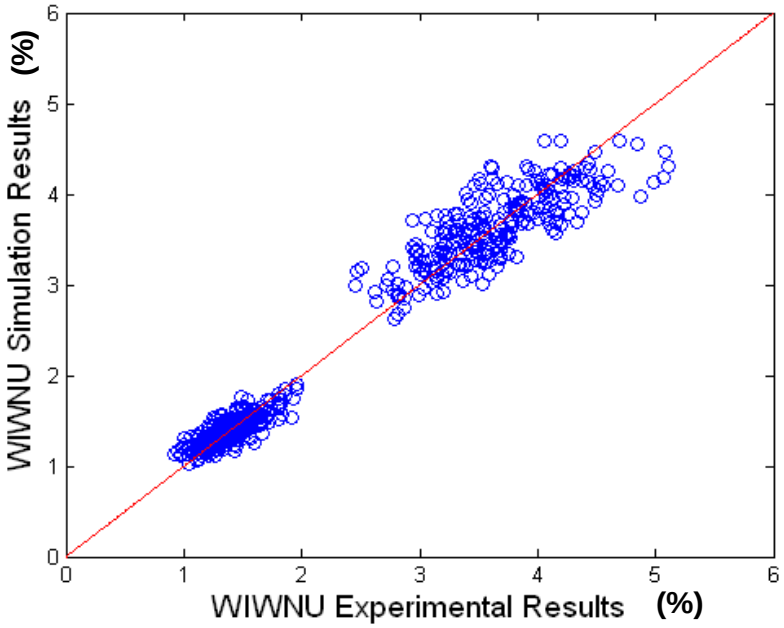


Figure 6.30 Training Results of WIWNU ANFIS-GP 2-2-2-3-2 Linear Model (Group II)

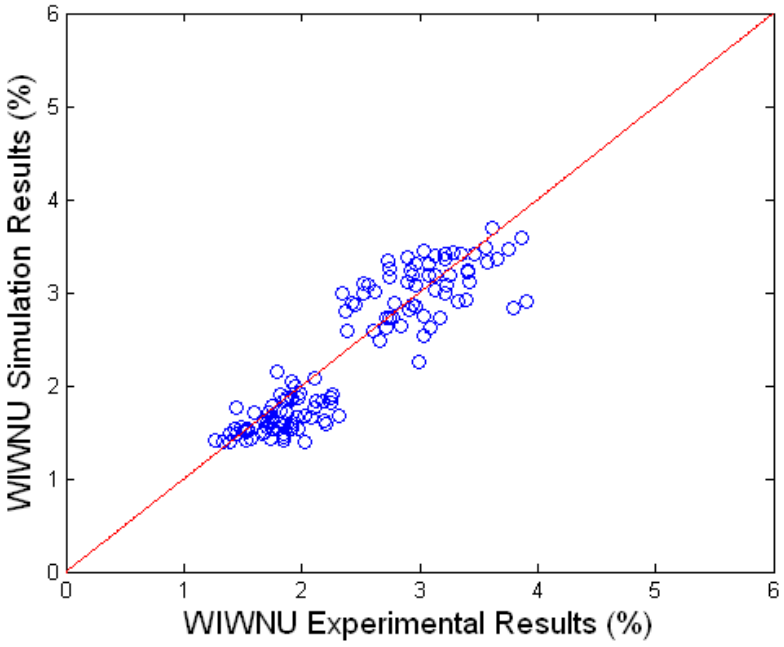


Figure 6.31 Testing Results of WIWNU ANFIS-GP 2-2-2-3-2 Linear Mode (Group II)

6.3.1.3 Results with Group III Data

Models of MRR Using ANFIS-GP 2-2-2-3-2 Linear for Group III Data

Architecture # 5 (as summarized in Table 6.12) was chosen to construct an ANFIS-GP model for Group III MRR case, because of the smallest *E_total* value and smaller *Error ratio* values resulted for this case. The training and testing results are illustrated in the below block and following Figure 6.32 and 6.3.33.

Training error (%) = 1.2973 %

Testing error (%) = 2.4711 %

Standard deviaton of training error (%) = 1.0517 %

Standard deviaton of testing error (%) = 2.0272 %

Table 6.12 Comparisons of Different ANFIS-GP MRR Models (Group III)

MRR 162	Structure (trimf)	Type of Consequent part	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio (E_test/E_train)	Epochs
1	2-2-2-2-2	Linear	31.0994	58.1761	89.28	1.871	20
2	3-2-2-2-2	Linear	23.8925	54.0825	77.98	2.264	20
3	2-3-2-2-2	Linear	26.3504	50.8865	77.24	1.931	20
4	2-2-3-2-2	Linear	26.2939	53.5203	79.81	2.035	20
5	2-2-2-3-2	Linear	25.3595	47.6812	73.04	1.880	20
6	2-2-2-3-2 (gaussmf)	Linear	25.2435	108.6660	133.91	4.305	20
7	2-2-2-2-3	Linear	31.0994	824.4910	855.59	26.511	20
8	2-2-2-2-2	Constant	46.9726	47.3185	94.29	1.007	200
9	3-2-2-2-2	Constant	45.0661	47.9782	93.04	1.065	200
10	2-3-2-2-2	Constant	43.9817	46.3683	90.35	1.054	200
11	2-2-3-2-2	Constant	45.0529	47.6616	92.71	1.058	200
12	2-2-2-3-2	Constant	38.8532	46.5339	85.39	1.198	200
13	2-2-2-3-2 (gaussmf)	Constant	38.8463	107.6120	146.46	2.770	200
14	2-2-2-2-3	Constant	46.9726	829.6780	876.65	17.663	200

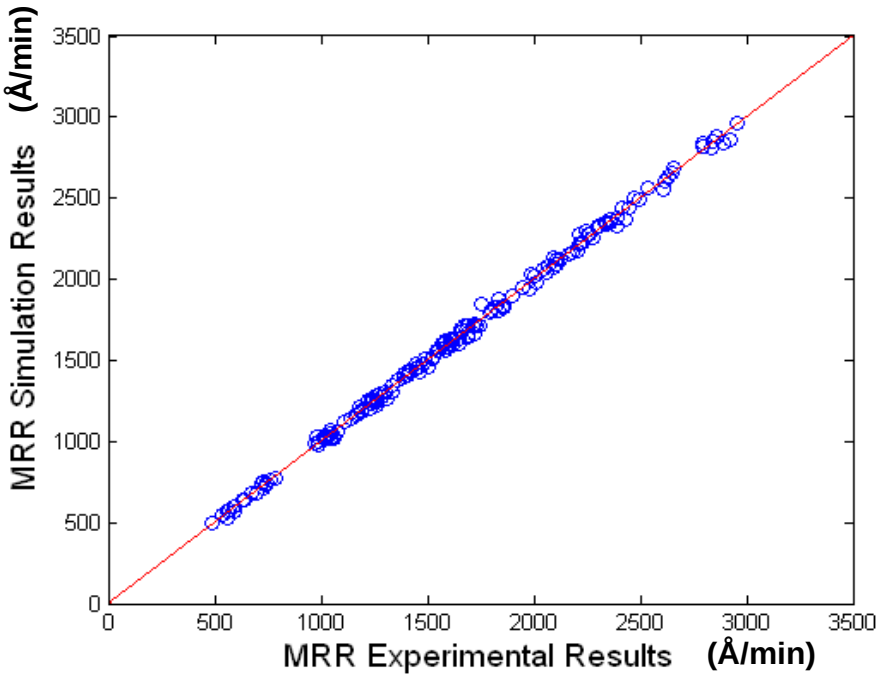


Figure 6.32 Training Results for MRR ANFIS-GP 2-2-2-3-2 Linear Model (Group III)

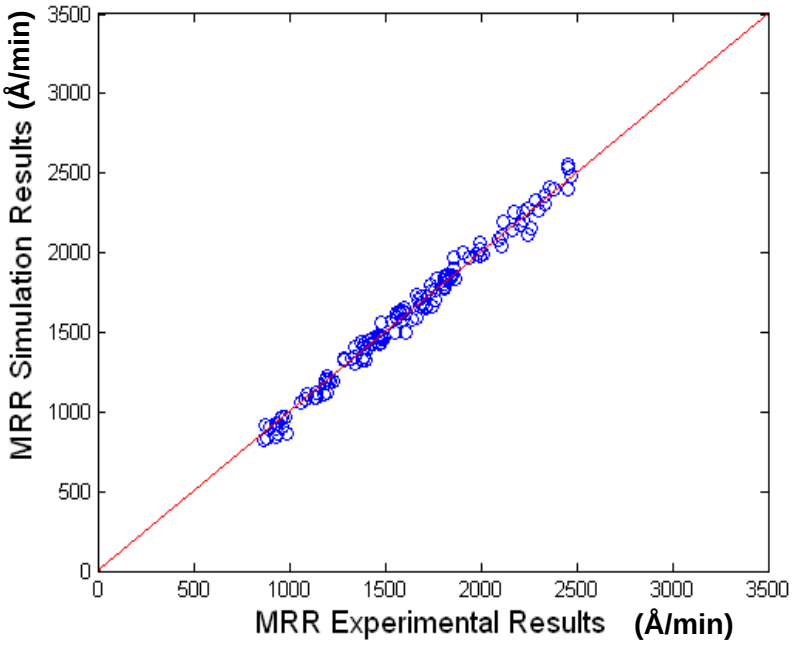


Figure 6.33 Testing Results for MRR ANFIS-GP 2-2-2-3-2 Linear Model (Group III)

Models of WIWNU Using ANFIS-GP 2-2-2-3-2 Linear for Group III Data

Architecture # 5 (as summarized in Table 6.13) was chosen to construct an ANFIS-GP model for Group III MRR case, because of the smallest *E_total* value and smaller *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.34 and 6.35 and the percentage errors as follows.

Training error (%) = 4.2443 %

Testing error (%) = 9.9340 %

Standard deviation of training error (%) = 3.6380 %

Standard deviation of testing error (%) = 7.2834 %

Table 6.13 Comparisons of Different ANFIS-GP WIWNU Models (Group III)

WIWNU 162	Structure (gaussian)	Type of Consequent part	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio (E_test/E_train)	Epochs
1	2-2-2-2-2	Linear	0.1894	0.3101	0.4995	1.638	20
2	3-2-2-2-2	Linear	0.1595	0.3004	0.4599	1.884	20
3	2-3-2-2-2	Linear	0.1409	0.3050	0.4459	2.164	20
4	2-2-3-2-2	Linear	0.1579	0.3014	0.4594	1.908	20
5	2-2-2-3-2	Linear	0.1396	0.3040	0.4435	2.177	20
6	2-2-2-3-2 (gaussmf)	Linear	0.1385	0.3820	0.5205	2.759	20
7	2-2-2-2-3	Linear	0.1894	1.2346	1.4240	6.520	20
8	2-2-2-2-2	Constant	0.3160	0.2850	0.6010	0.902	1000
9	3-2-2-2-2	Constant	0.2700	0.2966	0.5665	1.098	1000
10	2-3-2-2-2	Constant	0.2946	0.2901	0.5847	0.985	1000
11	2-2-3-2-2	Constant	0.2886	0.2904	0.5790	1.006	1000
12	2-2-2-3-2	Constant	0.3028	0.2870	0.5898	0.948	1000
13	2-2-2-3-2 (gaussmf)	Constant	0.2968	0.3402	0.6370	1.146	1000
14	2-2-2-2-3	Constant	0.3160	1.2062	1.5222	3.817	1000

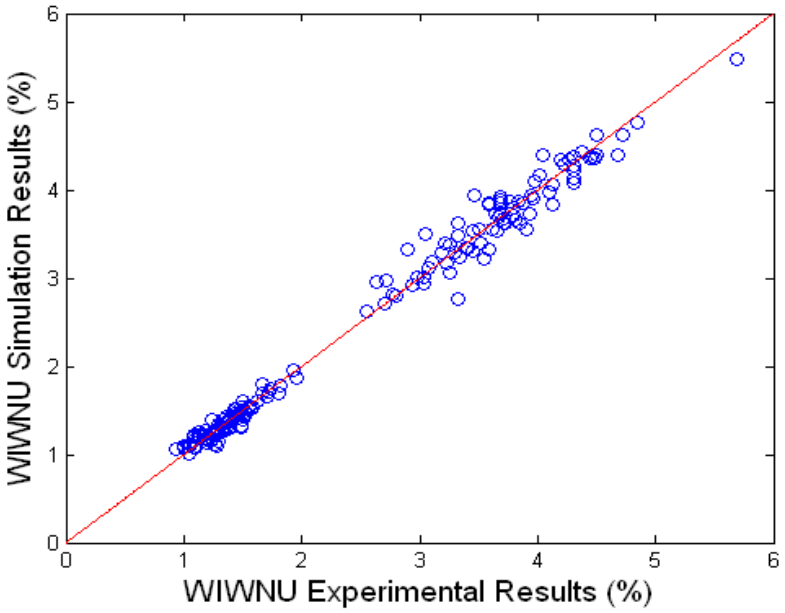


Figure 6.34 Training Results for WIWNU ANFIS-GP 2-2-2-3-2 Linear Model (Group III)

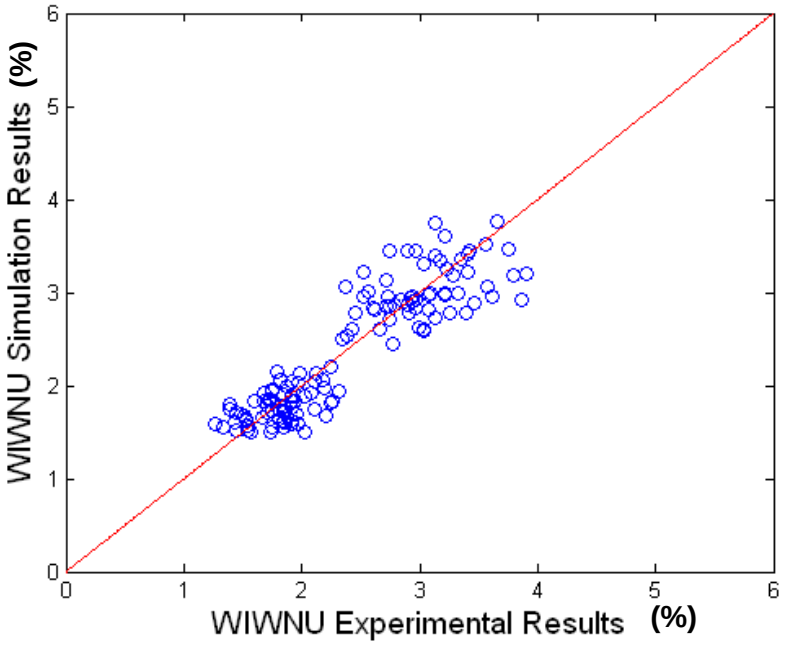


Figure 6.35 Testing Results for WIWNU ANFIS-GP 2-2-2-3-2 Linear Model (Group III)

6.3.2 Models Using ANFIS-SC

6.3.2.1 Results with Group I Data

Models of MRR Using ANFIS-SC 12-Rule for Group I Data

Architecture # 6 (as summarized in Table 6.140 was chosen to construct an ANFIS-SC model for Group I MRR case, because of the smallest *E_total* value and smaller *Error ratio* values resulted for this case. The training and testing results are summarized in Figure 6.36 and 6.37 and the percentage errors as follows.

Training error (%) = 2.2104 %

Testing error (%) = 2.2359 %

Standard deviation of training error (%) = 1.5837 %

Standard deviation of testing error (%) = 1.4431 %

Table 6.14 Comparisons of Different ANFIS-SC MRR Models (Group I)

MRR 1875	Range of Influence	Squash Factor	Accept Ratio	Reject Ratio	Rule numbers	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio E_test/E_train	Epochs
1	0.5	3.0	0.5	0.15	9	49.7283	43.8373	93.5656	0.88	30
2	1.1	1.1	0.5	0.15	12	45.1505	45.0219	90.1724	1.00	30
3	1.2	1.0	0.5	0.15	10	45.3162	43.0761	88.3923	0.95	30
4	1.0	1.0	0.5	0.15	19	45.898	45.5017	91.3997	0.99	30
5	1.2	0.9	0.5	0.15	14	43.6834	45.3846	89.068	1.04	30
6	1.3	0.8	0.5	0.15	12	44.1055	42.33	86.4355	0.96	30
7	1.6	0.5	0.5	0.15	12	43.8569	44.2327	88.0896	1.01	30
8	1.1	0.9	0.5	0.15	17	43.8528	44.475	88.3278	1.01	30

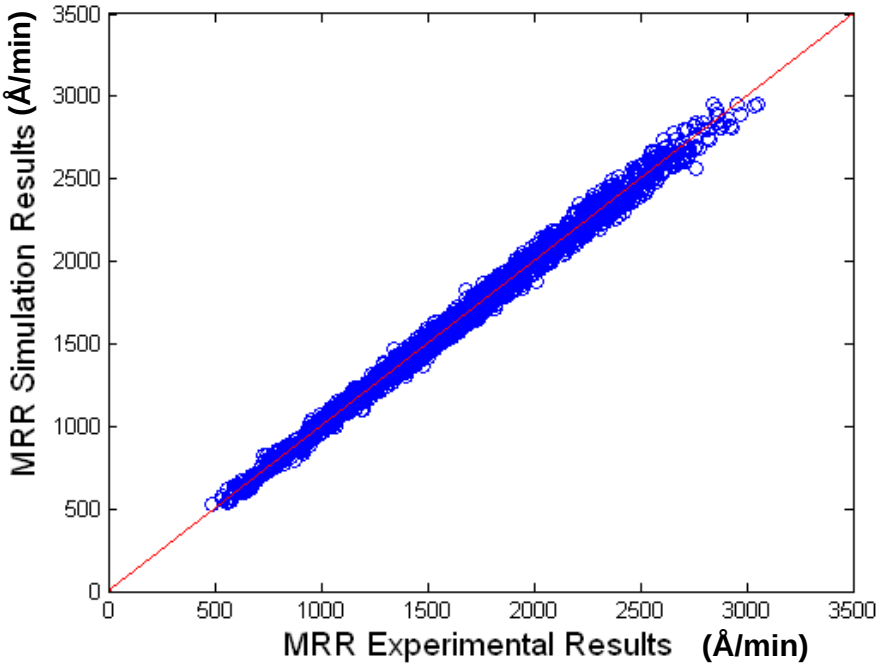


Figure 6.36 Training Results for MRR ANFIS-SC 12-Rule Model (Group I)

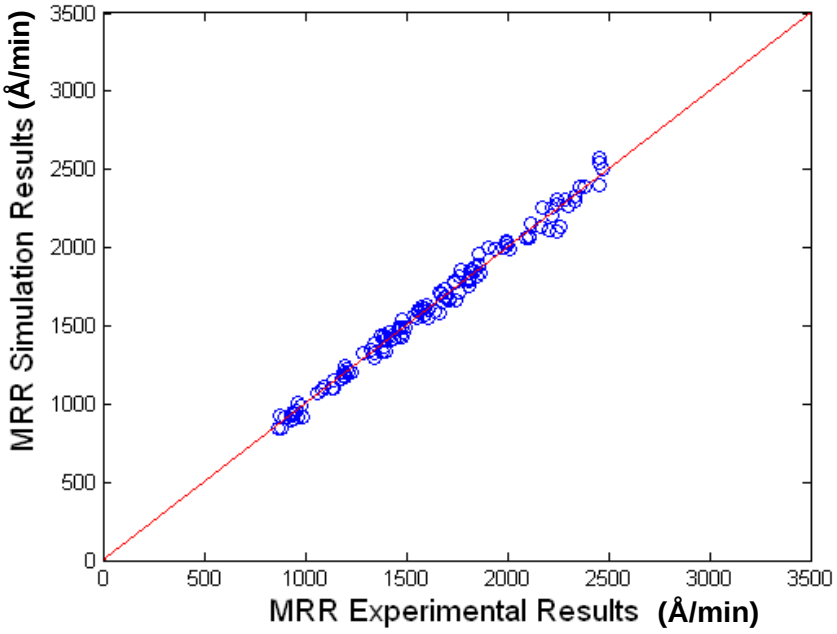


Figure 6.37 Testing Results for MRR ANFIS-SC 12-Rule Model (Group I)

Models of WIWNU Using ANFIS-SC 7-Rule for Group I Data

Architecture # 6 (as summarized in Table 6.15) was chosen to construct an ANFIS-SC model for Group I WIWNU case, because of the smaller *E_total* value and smallest *Error ratio* values resulted for this case. The training and testing are shown in Figures 6.38 and 6.39 and the percentage errors are tabulated in the following.

Training error (%) = 9.5765 %

Testing error (%) = 9.1558 %

Standard deviation of training error (%) = 7.7639 %

Standard deviation of testing error (%) = 6.1387 %

Table 6.15 Comparisons of Different ANFIS-SC WIWNU Models (Group I)

WIWNU 1875	Range of Influence	Squash Factor	Accept Ratio	Reject Ratio	Rule numbers	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio E_test/E_train	Epochs
1	2.0	0.4	0.5	0.15	20	0.30641	0.305446	0.61186	1.00	7
2	2.1	0.4	0.5	0.15	19	0.30566	0.26979	0.575453	0.88	30
3	2.2	0.4	0.5	0.15	18	0.30698	0.280139	0.587118	0.91	30
4	1.9	0.4	0.5	0.15	19	0.3052	0.280693	0.585897	0.92	6
5	2.2	0.5	0.5	0.15	5	0.31339	0.265065	0.578458	0.85	18
6	2.1	0.5	0.5	0.15	7	0.31243	0.265743	0.578169	0.85	21
7	2.0	0.5	0.5	0.15	8	0.31266	0.268186	0.580843	0.86	21
8	1.9	0.5	0.5	0.15	9	0.31262	0.274336	0.586952	0.88	30

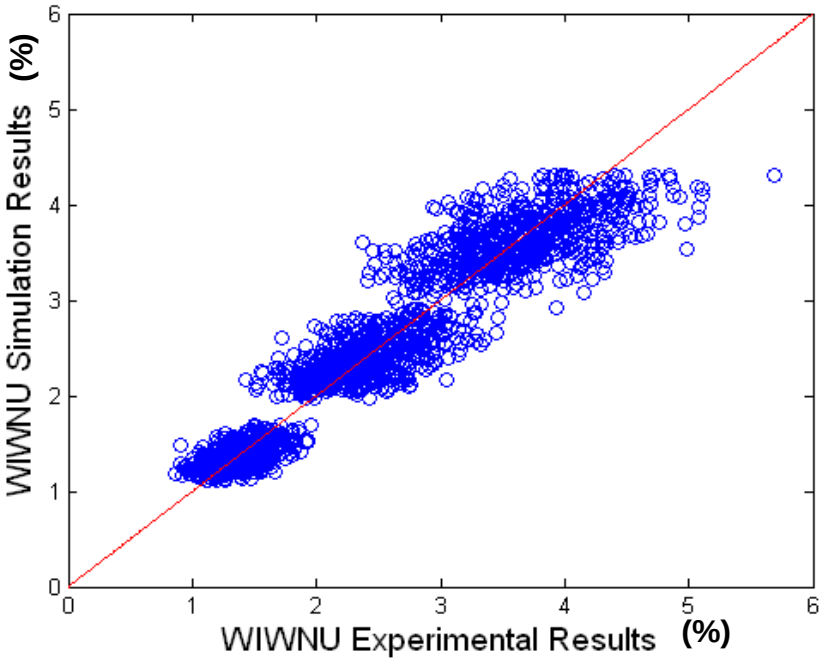


Figure 6.38 Training Results for WIWNU ANFIS-SC 7-Rule Model (Group I)

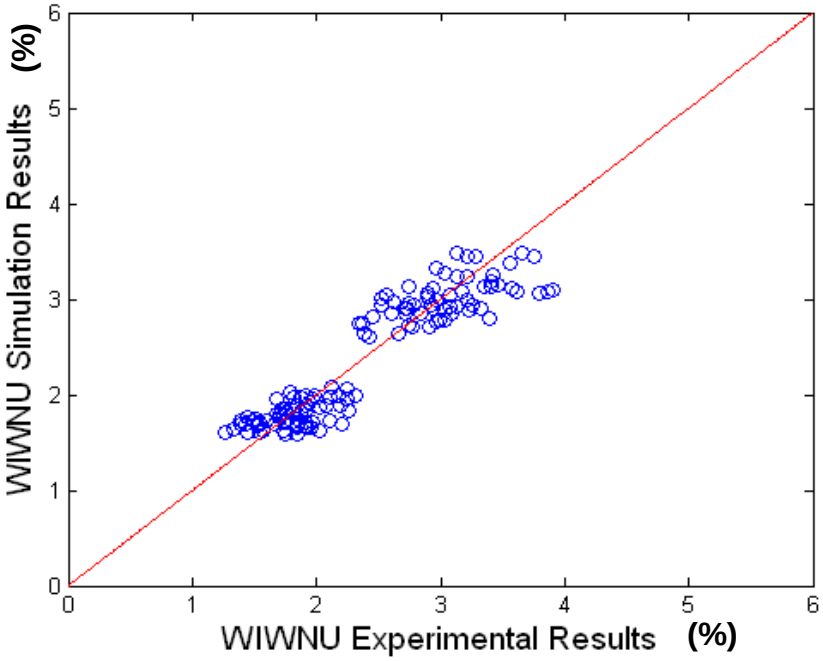


Figure 6.39 Testing Results for WIWNU ANFIS-SC 7-Rule Model (Group I)

6.3.2.2 Results with Group II Data

Models of MRR Using ANFIS-SC 12-Rule for Group II Data

Architecture # 5 (as summarized in Table 6.16) was chosen to construct an ANFIS-SC model for Group II MRR case, because of the smallest *E_total* value and second smallest *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.40 and 6.41 and the percentage errors are tabulated in the following.

Training error (%) = 2.1449 %

Testing error (%) = 2.3880 %

Standard deviation of training error (%) = 1.6009 %

Standard deviation of testing error (%) = 1.6098 %

Table 6.16 Comparisons of Different ANFIS-SC MRR Models (Group II)

MRR 512	Range of Influence	Squash Factor	Accept Ratio	Reject Ratio	Rule numbers	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio E_test/E_train	Epochs
1	0.6	2.5	0.5	0.15	29	36.035	64.594	100.629	1.79	30
2	0.5	2.4	0.5	0.15	33	35.593	76.105	111.699	2.14	30
3	3.0	0.4	0.5	0.15	19	36.341	168.949	205.290	4.65	30
4	8.0	0.2	0.5	0.15	12	41.396	63.265	104.661	1.53	30
5	8.1	0.2	0.5	0.15	12	41.363	44.852	86.215	1.08	30
6	10.0	0.1	0.5	0.15	35	41.796	44.377	86.172	1.06	30
7	9.0	0.2	0.5	0.15	25	41.425	45.565	86.990	1.10	30
8	7.0	0.3	0.5	0.15	9	41.288	47.873	89.161	1.16	30

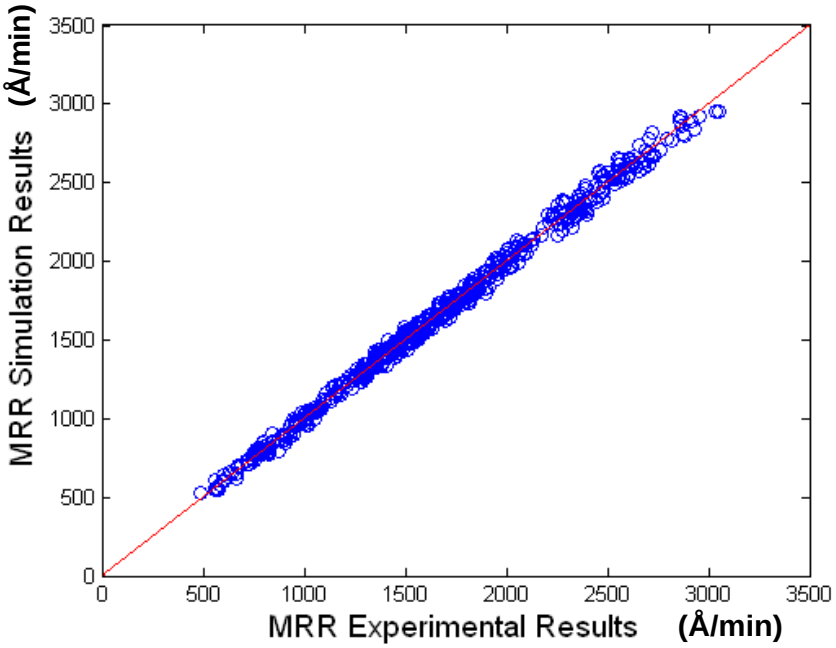


Figure 6.40 Training Results for MRR ANFIS-SC 12-Rule Model (Group II)

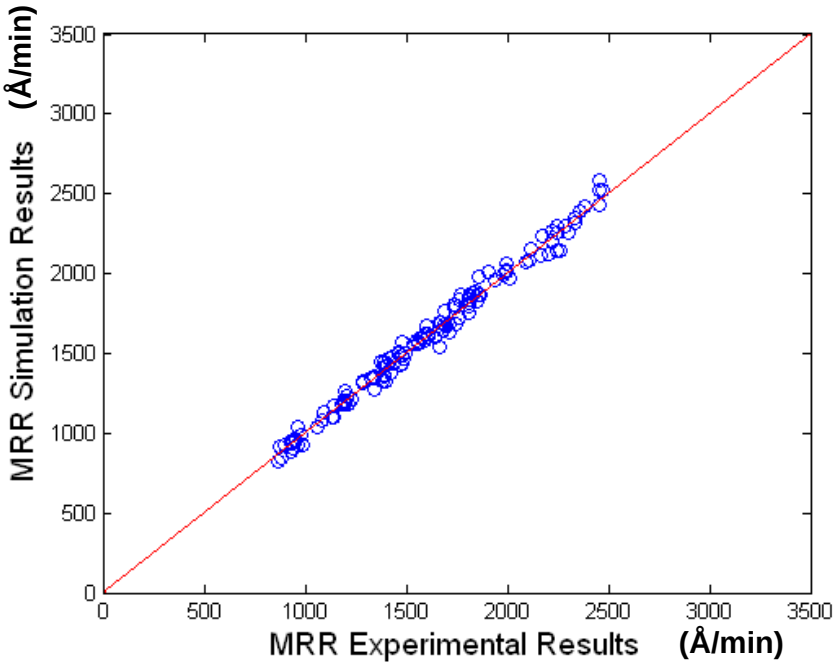


Figure 6.41 Testing Results for MRR ANFIS-SC 12-Rule Model (Group II)

Models of WIWNU Using ANFIS-SC 5-Rule for Group II Data

Architecture # 5 (as summarized in Table 6.17) was chosen to construct an ANFIS-SC model for Group II WIWNU case, because of the second smallest *E_total* value and smallest *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.42 and 6.43 and the percentage errors as follows.

Training error (%) = 9.1191 %

Testing error (%) = 9.4168 %

Standard deviation of training error (%) = 7.0589 %

Standard deviation of testing error (%) = 6.6441 %

Table 6.17 Comparisons of Different ANFIS-SC WIWNU Models (Group II)

WIWNU 512	Range of Influence	Squash Factor	Accept Ratio	Reject Ratio	Rule numbers	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio E_test/E_train	Epochs
1	9.0	0.20	0.5	0.15	8	0.30349	1.53292	1.836412	5.05	10
2	8.5	0.20	0.5	0.15	10	0.30352	0.71203	1.015547	2.35	10
3	8.5	0.25	0.5	0.15	6	0.30559	0.285083	0.590668	0.93	10
4	8.5	0.26	0.5	0.15	5	0.30598	0.274392	0.580368	0.90	10
5	8.6	0.24	0.5	0.15	7	0.30457	0.462671	0.767238	1.52	10
6	8.6	0.25	0.5	0.15	5	0.30598	0.274394	0.580374	0.90	10
7	8.7	0.25	0.5	0.15	6	0.30559	0.285083	0.590668	0.93	10
8	8.8	0.25	0.5	0.15	5	0.30599	0.274398	0.580385	0.90	10

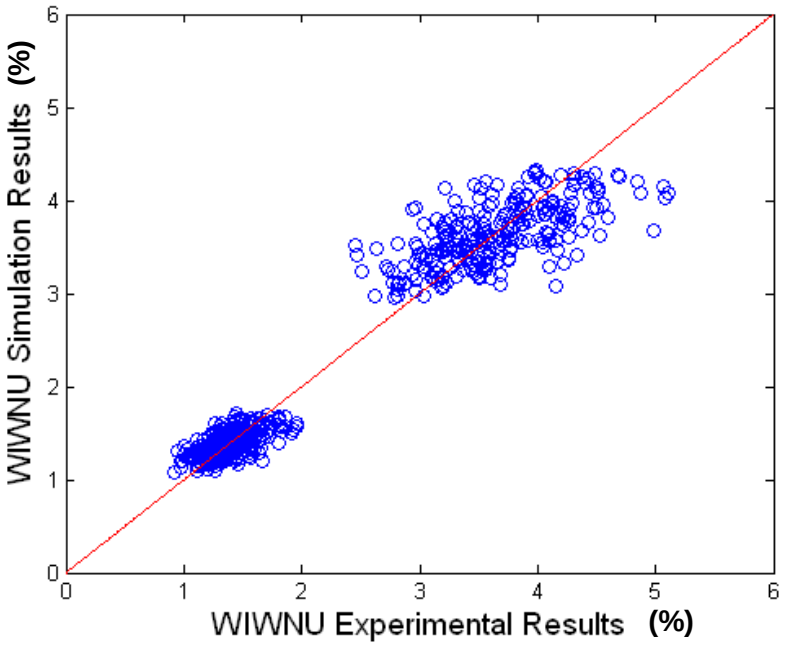


Figure 6.42 Training Results for WIWNU ANFIS-SC 5-Rule Model (Group II)

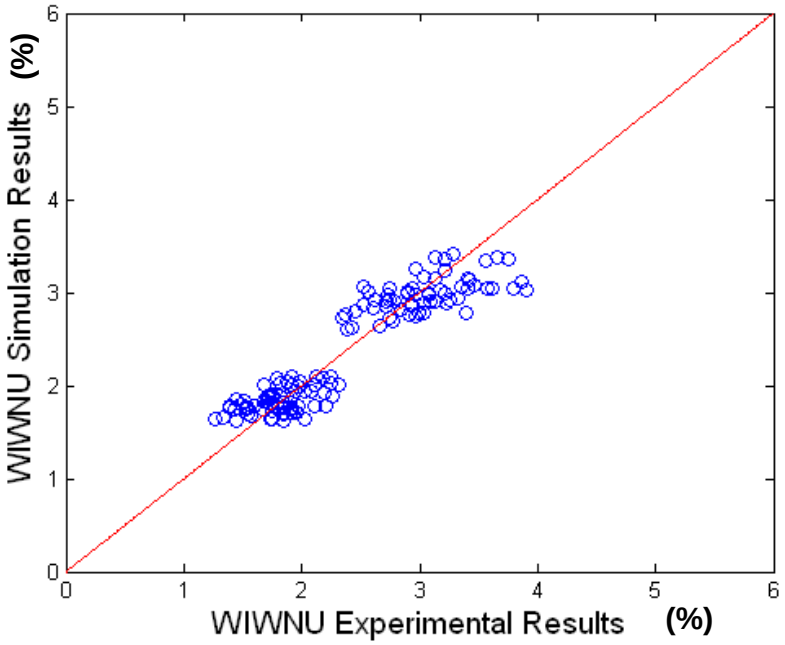


Figure 6.43 Testing Results for WIWNU ANFIS-SC 5-Rule Model (Group II)

6.3.2.3 Results with Group III Data

Models of MRR Using ANFIS-SC 24-Rule for Group III Data

Architecture # 7 (as summarized in Table 6.18) was chosen to construct an ANFIS-SC model for Group III MRR case, because of the second smallest *E_total* value and smallest *Error ratio* values resulted for this. The training and testing results are summarized in Figures 6.44 and 6.45 and the percentage errors as follows.

Training error (%) = 2.0228 %

Testing error (%) = 2.5703 %

Standard deviation of training error (%) = 1.5624 %

Standard deviation of testing error (%) = 1.9065 %

Table 6.18 Comparisons of Different ANFIS-SC MRR Models (Group III)

MRR 162	Range of Influence	Squash Factor	Accept Ratio	Reject Ratio	Rule numbers	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio E_test/E_train	Epochs
1	9.0	0.15	0.5	0.15	22	38.6822	51.1393	89.8215	1.32	20
2	9.0	0.16	0.5	0.15	17	38.8027	55.707	94.5097	1.44	20
3	9.0	0.17	0.5	0.15	16	38.9184	61.3742	100.2926	1.58	20
4	10.0	0.12	0.5	0.15	33	39.0754	50.8307	89.9061	1.30	20
5	9.5	0.12	0.5	0.15	32	38.9442	50.883	89.8272	1.31	20
6	9.0	0.14	0.5	0.15	31	38.7517	51.4539	90.2056	1.33	20
7	9.5	0.14	0.5	0.15	24	38.885	49.4205	88.3055	1.27	20
8	9.5	0.15	0.5	0.15	18	38.9309	54.6359	93.5668	1.40	20

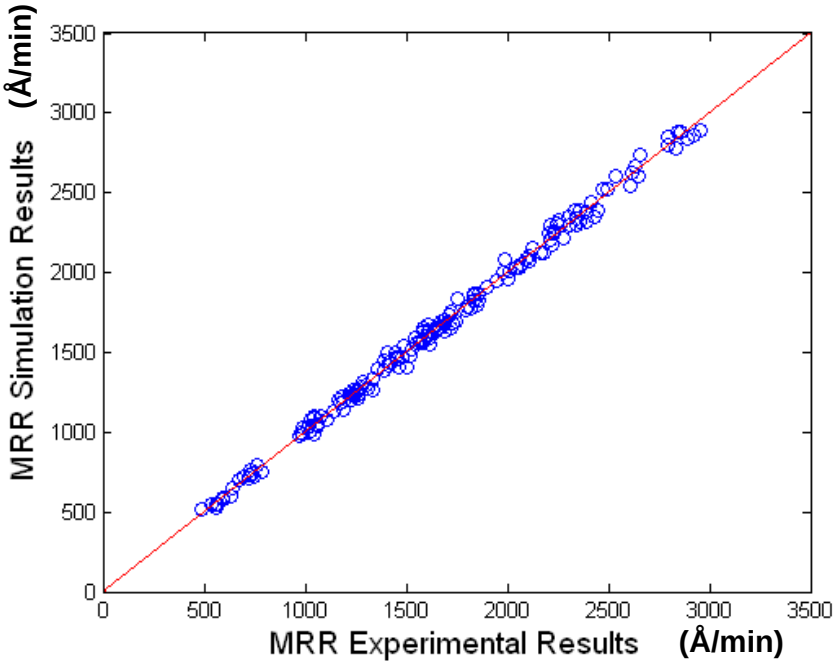


Figure 6.3.44 Training Results for MRR ANFIS-SC 24-Rule Model (Group III)

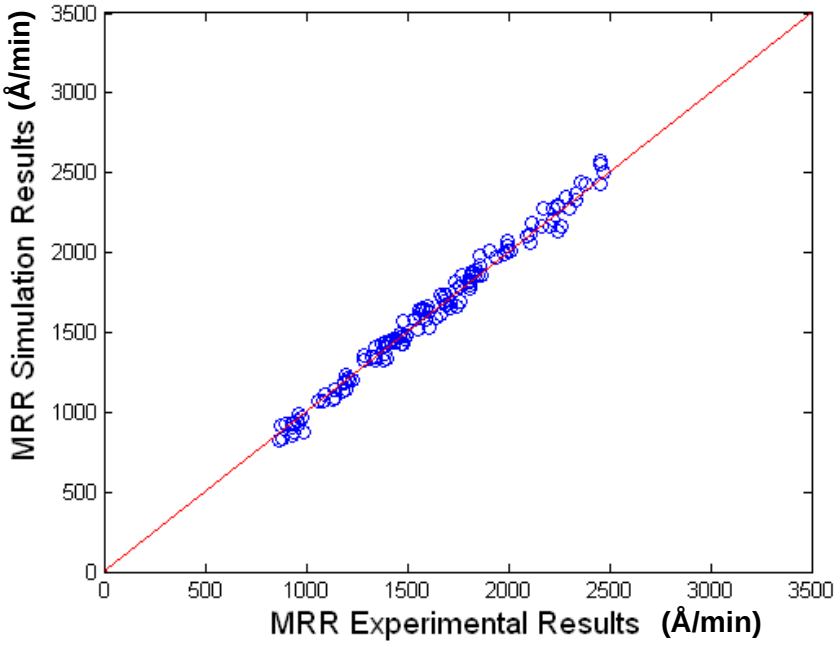


Figure 6.45 Testing Results for MRR ANFIS-SC 24-Rule Model (Group III)

Models of WIWNU Using ANFIS-SC 3-Rule for Group III Data

Architecture # 3 (as summarized in Table 6.19) was chosen to construct an ANFIS-SC model for Group III MRR case, because of the smallest *E_total* value and *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.46 and 6.47 and the percentage errors as follows.

Training error (%) = 9.3914 %

Testing error (%) = 9.8541 %

Standard deviation of training error (%) = 7.1057 %

Standard deviation of testing error (%) = 6.6393 %

Table 6.19 Comparisons of Different ANFIS-SC WIWNU Models (Group III)

WIWNU 162	Range of Influence	Squash Factor	Accept Ratio	Reject Ratio	Rule numbers	Training Error (E_train)	Testing Error (E_test)	E_total	Error Ratio E_test/E_train	Epochs
1	5.0	0.50	0.5	0.15	3	0.3243	0.3207	0.645032	0.99	20
2	5.1	0.50	0.5	0.15	3	0.3243	0.3198	0.644109	0.99	20
3	6.0	0.40	0.5	0.15	3	0.3245	0.3052	0.629667	0.94	20
4	5.0	0.40	0.5	0.15	4	0.3198	0.7979	1.117772	2.49	20
5	7.0	0.38	0.5	0.15	3	0.3248	0.3244	0.649274	1.00	20
6	8.0	0.28	0.5	0.15	4	0.3213	0.9547	1.276052	2.97	20
7	8.0	0.26	0.5	0.15	5	0.3191	1.2077	1.526795	3.79	20
8	8.2	0.24	0.5	0.15	6	0.3154	1.2347	1.550134	3.91	20

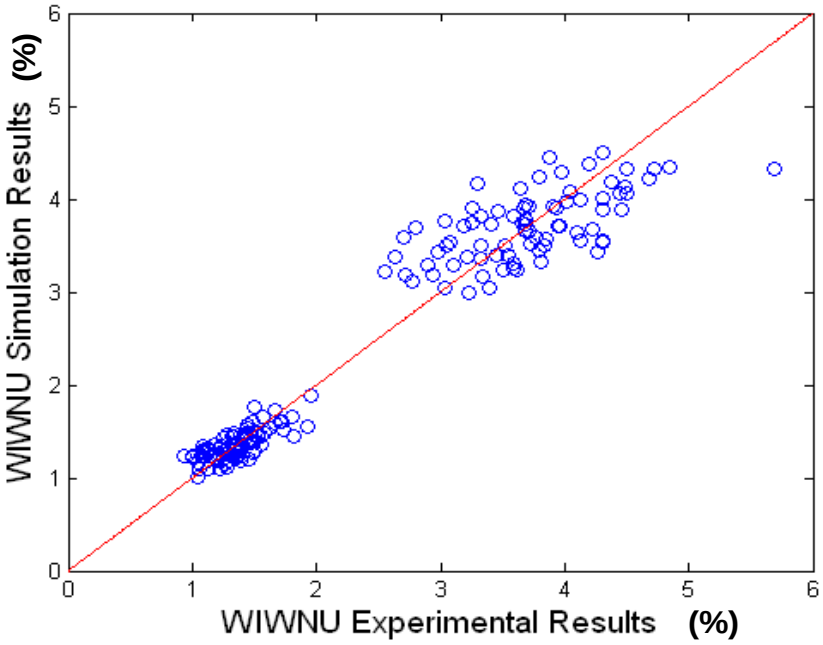


Figure 6.46 Training Results for WIWNU ANFIS-SC 5-Rule Model (Group III)

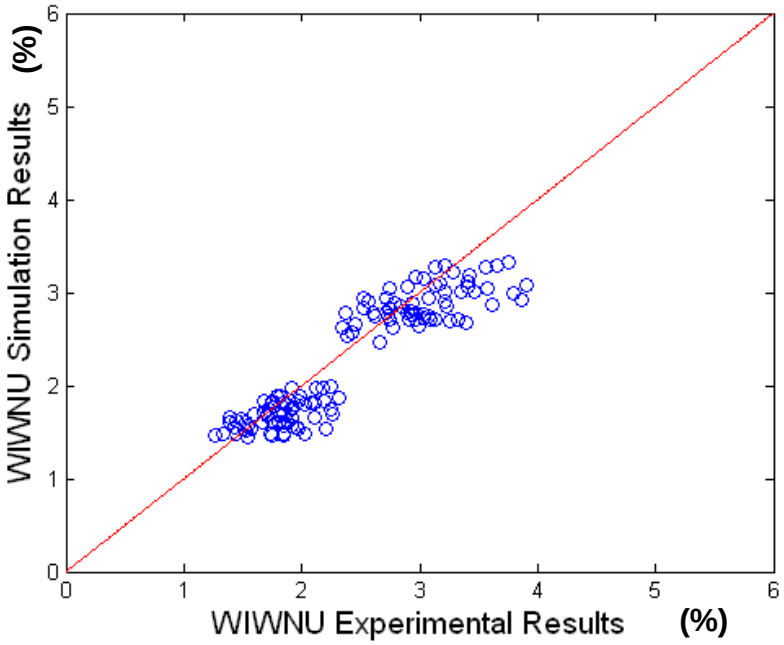


Figure 6.47 Testing Results for WIWNU ANFIS-SC 5-Rule Model (Group III)

6.3.3 Models Using Neural Networks

6.3.3.1 Results with Group I Data

Models of MRR Using NN 5-15-6-1 for Group I Data

Architecture # 18 (as summarized in Table 6.20) was chosen to construct an NN model for Group I MRR case, because of the smallest *E_total* value and *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.48 and 6.49 and the percentage errors as follows.

Training error (%) = 2.1219 %

Testing error (%) = 2.2389 %

Standard deviation of training error (%) = 1.5413 %

Standard deviation of testing error (%) = 1.5272 %

Table 6.20 Comparisons of Different NN MRR Models (Group I)

MRR 1875	Best NN Architectures				Total Weights	Mean Squre Training Error (E_train)	Mean Square Testing Error (E_test)	E_total	Error Ratio (E_test / E_train)
1	5	16	13	1	301	0.0028	0.0015	0.0043	0.5357
2	5	4	84	1	440	0.0043	0.002	0.0063	0.4651
7	5	5	85	1	535	0.0038	0.0018	0.0056	0.4737
8	5	16	13	1	301	0.0034	0.0016	0.0050	0.4706
9	5	15	14	1	299	0.0028	0.0015	0.0043	0.5357
11	5	13	3	1	107	0.0015	0.0011	0.0026	0.7333
12	5	17	4	1	157	0.0017	0.0012	0.0029	0.7059
13	5	13	8	1	177	0.0017	0.0012	0.0029	0.7059
14	5	13	8	1	177	0.002	0.0012	0.0032	0.6000
15	5	17	4	1	157	0.0017	0.0011	0.0028	0.6471
16	5	11	9	1	163	0.0019	0.0011	0.0030	0.5789
17	5	13	7	1	163	0.0017	0.0011	0.0028	0.6471
18	5	15	6	1	171	0.0019	0.0011	0.0030	0.5789

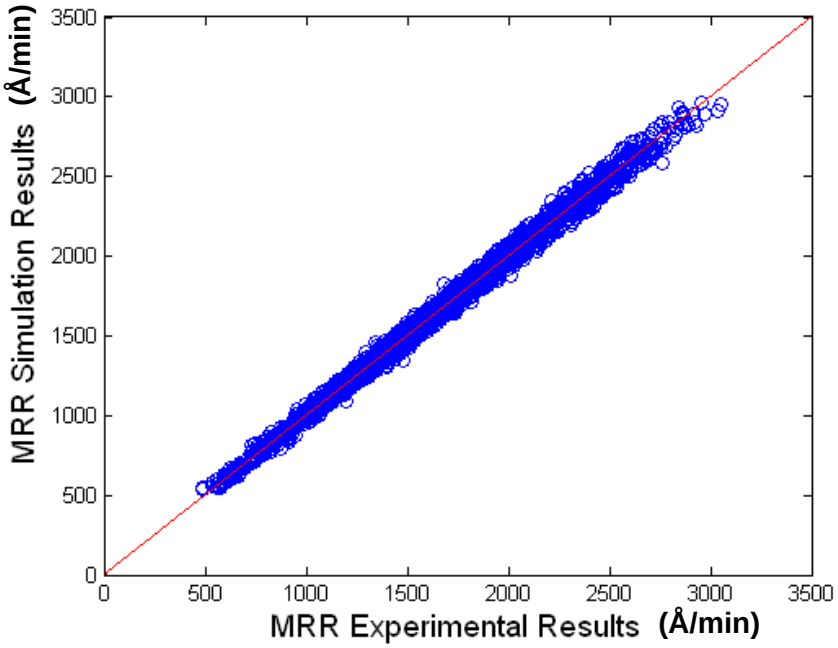


Figure 6.48 Training Results for MRR NN 5-15-6-1 Model (Group I)

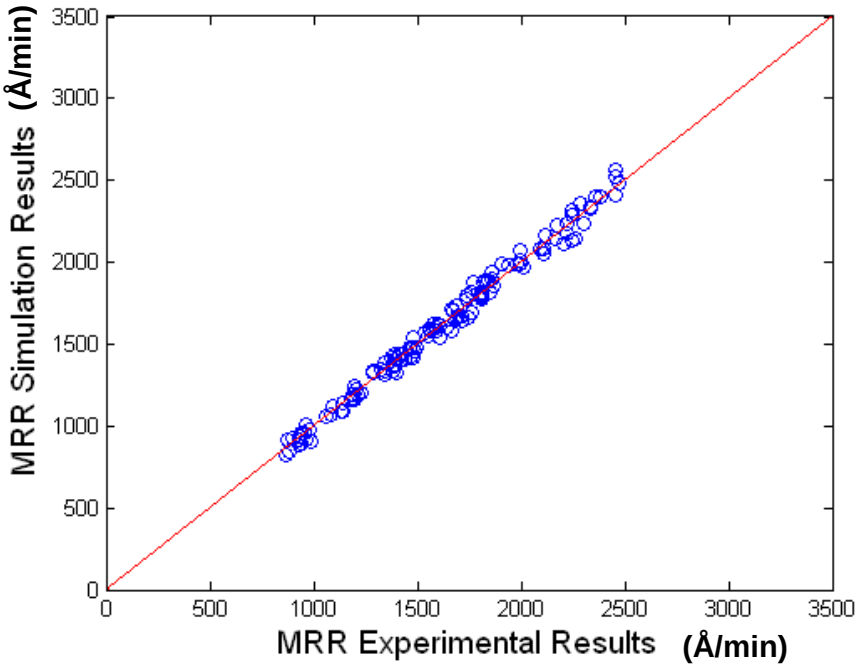


Figure 6.49 Testing Results for WIWNU NN 5-15-6-1 Model (Group I)

Models of WIWNU Using NN 5-15-13-1 for Group I Data

Architecture # 1 (as summarized in Table 6.21) was chosen to construct an NN model for Group I WIWNU case, because of the second smallest *E_total* value and smaller *Error ratio* values resulted for this case. The training and testing results were summarized in Figure 6.50 and 6.51 and the percentage errors as follows.

Training error (%) = 9.5851 %

Testing error (%) = 9.0600 %

Standard deviation of training error (%) = 7.7486 %

Standard deviation of testing error (%) = 6.0803 %

Table 6.21 Comparisons of Different NN WIWNU Models (Group I)

WIWNU 1875	Best NN Architectures				Total Weights	Mean Squre Training Error (E_train)	Mean Square Testing Error (E_test)	E_total	Error Ratio (E_test / E_train)
1	5	16	13	1	301	0.0028	0.0015	0.0043	0.5357
2	5	4	84	1	440	0.0043	0.002	0.0063	0.4651
3	5	5	85	1	535	0.0038	0.0018	0.0056	0.4737
4	5	15	14	1	299	0.0028	0.0015	0.0043	0.5357
5	5	15	2	1	107	0.0163	0.0122	0.0285	0.7485
6	5	25	0	1	125	0.0169	0.012	0.0289	0.7101
7	5	16	3	1	131	0.0168	0.0121	0.0289	0.7202
8	5	17	4	1	157	0.0168	0.0121	0.0289	0.7202

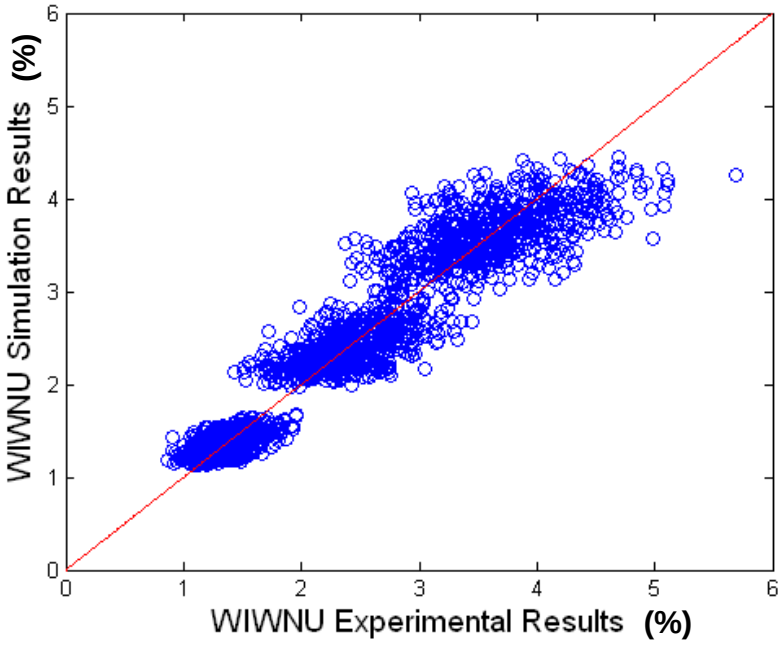


Figure 6.50 Training Results for WIWNU NN 5-16-13-1 Model (Group I)

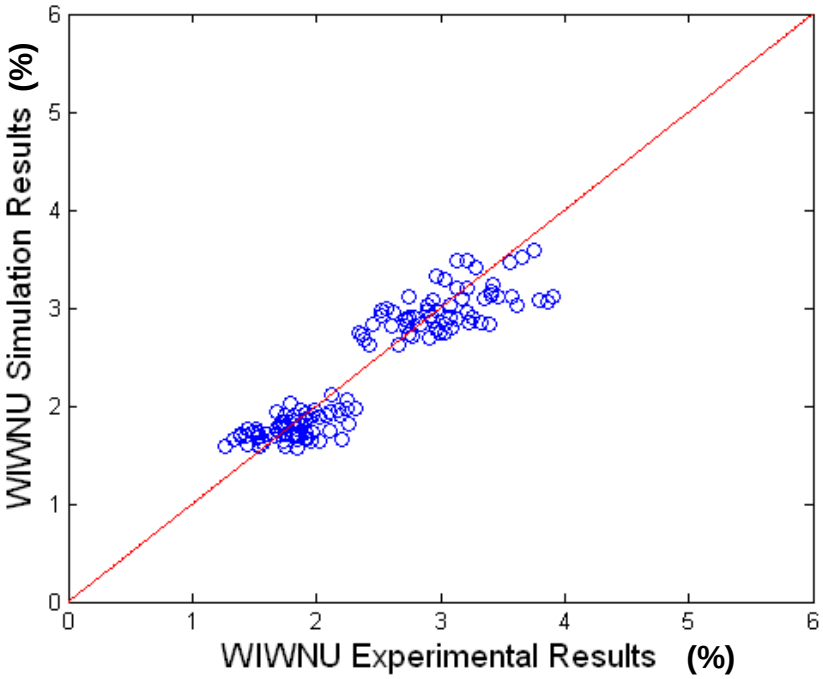


Figure 6.51 Testing Results for WIWNU NN 5-16-13-1 Model (Group I)

6.3.3.2 Results with Group II Data

Models of MRR Using NN 5-9-7-1 for Group II Data

Architecture # 1 (as summarized in Table 6.22) was chosen to construct an NN model for Group II MRR case, because of the smaller *E_total* value and *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.52 and 6.53 and the percentage errors as follows.

Training error (%) = 2.5199 %

Testing error (%) = 2.3992 %

Standard deviation of training error (%) = 2.1664 %

Standard deviation of testing error (%) = 1.5359 %

Table 6.22 Comparisons of Different NN MRR Models (Group II)

MRR 512	Best NN Architectures				Total weights	Mean Squre Training Error (E_train)	Mean Square Testing Error (E_test)	E_total	Error Ratio (E_test / E_train)
1	5	9	7	1	115	0.0048	0.0017	0.0065	0.3542
2	5	5	7	1	67	0.0053	0.002	0.0073	0.3774
7	5	3	1	1	19	0.0046	0.0019	0.0065	0.4130
8	5	3	5	1	35	0.0061	0.0022	0.0083	0.3607
9	5	3	9	1	51	0.0074	0.003	0.0104	0.4054
11	5	5	3	1	43	0.0048	0.0019	0.0067	0.3958
12	5	5	7	1	67	0.0053	0.002	0.0073	0.3774
13	5	7	1	1	43	0.0058	0.0017	0.0075	0.2931
14	5	7	5	1	75	0.0047	0.0019	0.0066	0.4043
15	5	7	9	1	107	0.0044	0.0021	0.0065	0.4773
16	5	8	12	1	148	0.0096	0.0054	0.0150	0.5625
17	5	9	3	1	75	0.0057	0.0021	0.0078	0.3684
18	5	10	10	1	160	0.0052	0.0022	0.0074	0.4231
19	5	11	5	1	115	0.0038	0.0017	0.0055	0.4474
20	5	11	1	1	67	0.0067	0.0025	0.0092	0.3731

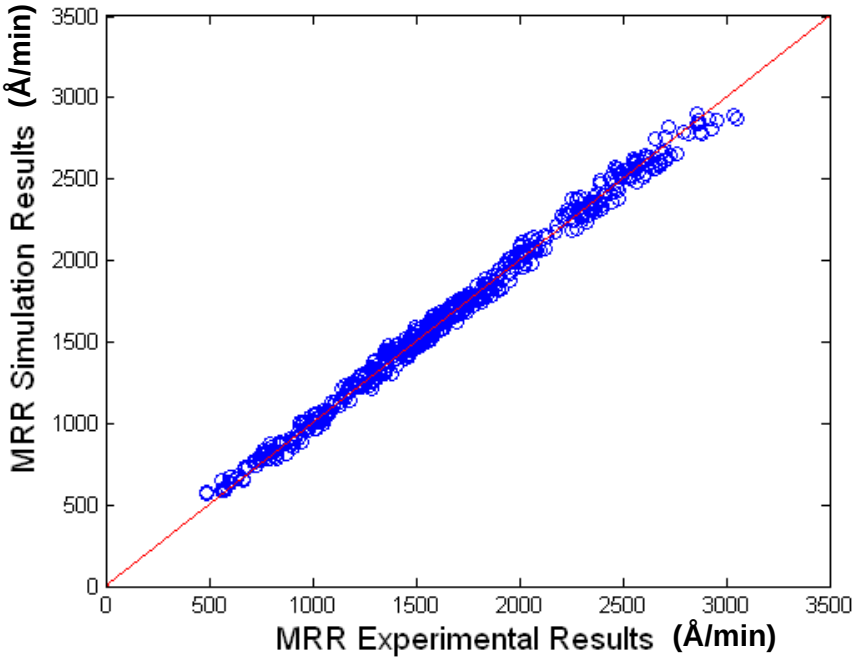


Figure 6.52 Training Results for MRR NN 5-9-7-1 Model (Group II)

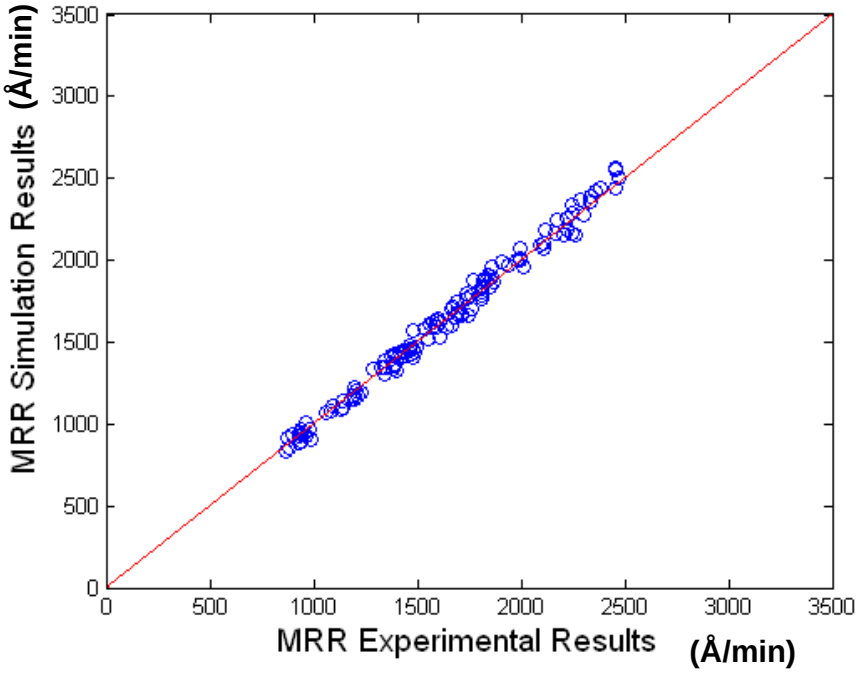


Figure 6.53 Testing Results for WIWNU NN 5-9-7-1 Model (Group II)

Models of WIWNU Using NN 5-2-5-1 for Group II Data

Architecture # 1 (as summarized in Table 6.23) was chosen to construct an NN model for Group II MRR case, because of the smallest *E_total* value and *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.54 and 6.55 and the percentage errors as follows.

Training error (%) = 9.1370 %

Testing error (%) = 10.9449 %

Standard deviation of training error (%) = 7.1590 %

Standard deviation of testing error (%) = 7.7574 %

Table 6.23 Comparisons of Different NN WIWNU Models (Group II)

WIWNU 512	Best NN Architectures				Total Weights	Mean Squre Training Error (E_train)	Mean Square Testing Error (E_test)	E_total	Error Ratio (E_test / E_train)
1	5	2	5	1	25	0.0215	0.0167	0.0382	0.7767
2	5	1	6	1	17	0.0219	0.0192	0.0411	0.8767
3	5	10	10	1	160	0.0219	0.0192	0.0411	0.8767
4	5	1	4	1	13	0.0219	0.0192	0.0411	0.8767
5	5	3	4	1	31	0.0213	0.0203	0.0416	0.9531
6	5	3	6	1	39	0.0213	0.0203	0.0416	0.9531
7	5	4	5	1	45	0.0213	0.021	0.0423	0.9859
8	5	4	7	1	55	0.0213	0.021	0.0423	0.9859
9	5	22	0	1	110	0.0213	0.021	0.0423	0.9859
10	5	4	7	1	55	0.0213	0.021	0.0423	0.9859
11	5	4	3	1	35	0.0213	0.0211	0.0424	0.9906
12	5	5	6	1	61	0.0212	0.0237	0.0449	1.1179
13	5	2	7	1	31	0.0214	0.0255	0.0469	1.1916
14	5	2	5	1	25	0.0214	0.0255	0.0469	1.1916
15	5	2	3	1	19	0.0214	0.0256	0.0470	1.1963
16	5	19	1	1	115	0.0206	0.0364	0.0570	1.7670
17	5	21	1	1	127	0.0206	0.0364	0.0570	1.7670
18	5	23	1	1	139	0.0206	0.0364	0.0570	1.7670

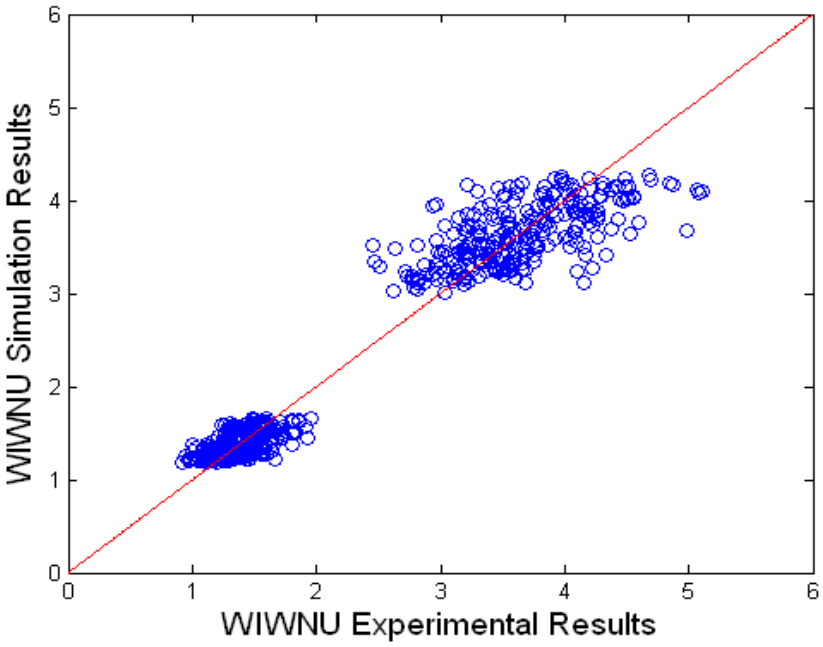


Figure 6.54 Training Results for WIWNU NN 5-2-5-1 Model (Group II)

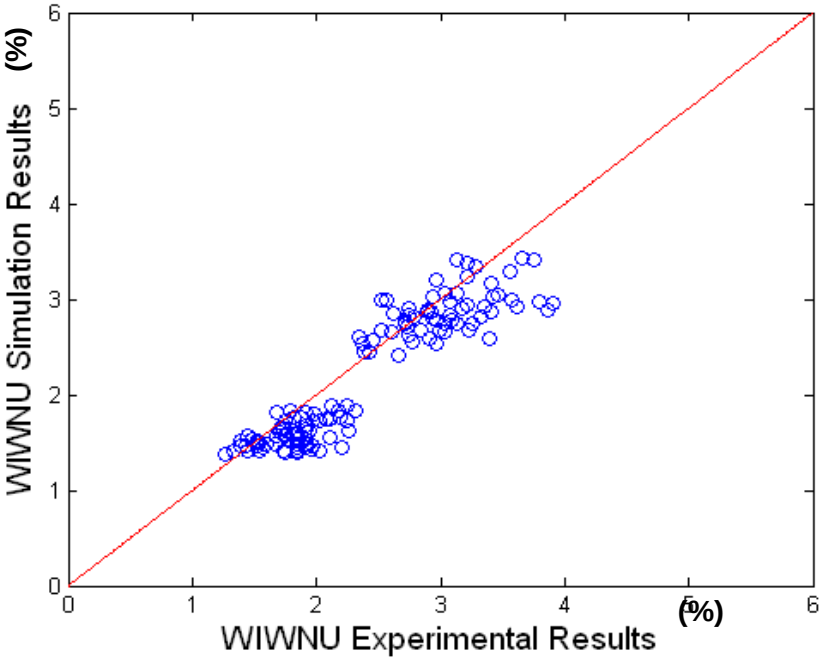


Figure 6.55 Testing Results for WIWNU NN 5-2-5-1 Model (Group II)

6.3.3.3 Results with Group III Data

Models of MRR Using NN 5-6-2-1 for Group III Data

Architecture # 1 (as summarized in Table 6.24) was chosen to construct an NN model for Group III MRR case, because of the second smallest *E_total* value and *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.56 and 6.57 and the percentage errors as follows.

Training error (%) = 5.4786 %

Testing error (%) = 3.4925 %

Standard deviation of training error (%) = 6.1808 %

Standard deviation of testing error (%) = 2.7797 %

Table 6.24 Comparisons of Different NN MRR Models (Group III)

MRR 162	Best NN Architectures				Total Weights	Mean Squre Training Error (E_train)	Mean Square Testing Error (E_test)	E_total	Error Ratio (E_test / E_train)
1	5	6	2	1	44	0.0069	0.0020	0.0089	0.2899
2	5	2	2	1	16	0.0097	0.0025	0.0122	0.2577
3	5	10	10	1	160	0.0097	0.0025	0.0122	0.2577
4	5	4	0	1	20	0.0097	0.0025	0.0122	0.2577
5	5	4	4	1	40	0.0089	0.0026	0.0115	0.2921
6	5	8	0	1	40	0.0064	0.0021	0.0085	0.3281
7	5	2	6	1	28	0.0288	0.0333	0.0621	1.1563

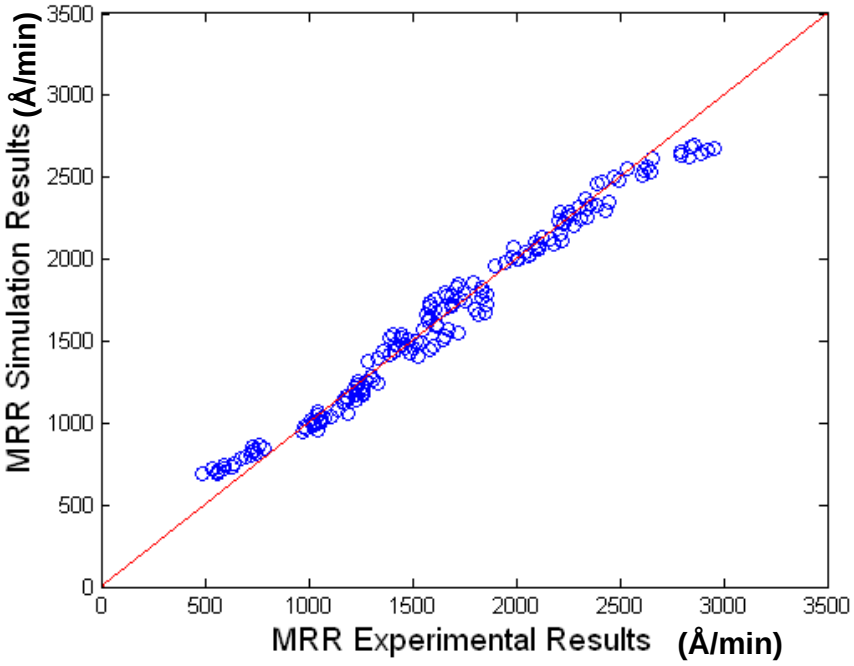


Figure 6.56 Training Results for MRR NN 5-6-2-1 Model (Group III)

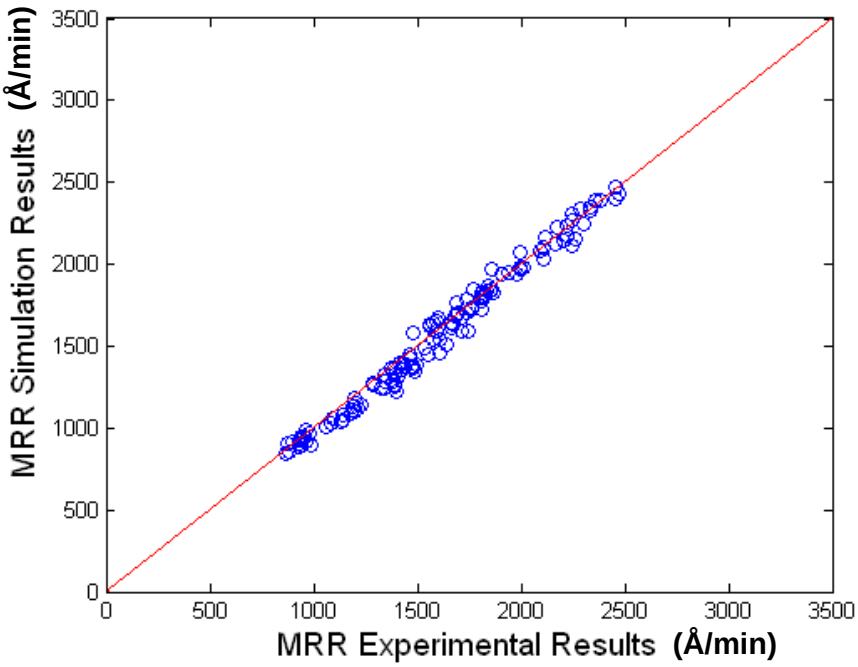


Figure 6.57 Testing Results for WIWNU NN 5-6-2-1 Model (Group III)

Models of WIWNU Using NN 5-2-3-1 for Group III Data

Architecture # 1 (as summarized in Table 6.25) was chosen to construct an NN model for Group III WIWNU case, because of the smallest *E_total* value and smaller *Error ratio* values resulted for this case. The training and testing results are summarized in Figures 6.58 and 6.59 and the percentage errors are tabulated in the following.

Training error (%) = 9.5333 %

Testing error (%) = 11.0140 %

Standard deviation of training error (%) = 7.0761 %

Standard deviation of testing error (%) = 7.4818 %

Table 6.25 Comparisons of Different NN WIWNU Models (Group III)

WIWNU 162	Best NN Architectures				Total Weights	Mean Squre Training Error (E_train)	Mean Square Testing Error (E_test)	E_total	Error Ratio (E_test / E_train)
1	5	2	3	1	19	0.016	0.0154	0.0314	0.9625
2	5	2	5	1	25	0.0161	0.0154	0.0315	0.9565
3	5	3	2	1	23	0.0155	0.0168	0.0323	1.0839
4	5	2	1	1	13	0.0163	0.0193	0.0356	1.1840
5	5	3	0	1	15	0.0163	0.0193	0.0356	1.1840
6	5	3	4	1	31	0.0153	0.0218	0.0371	1.4248

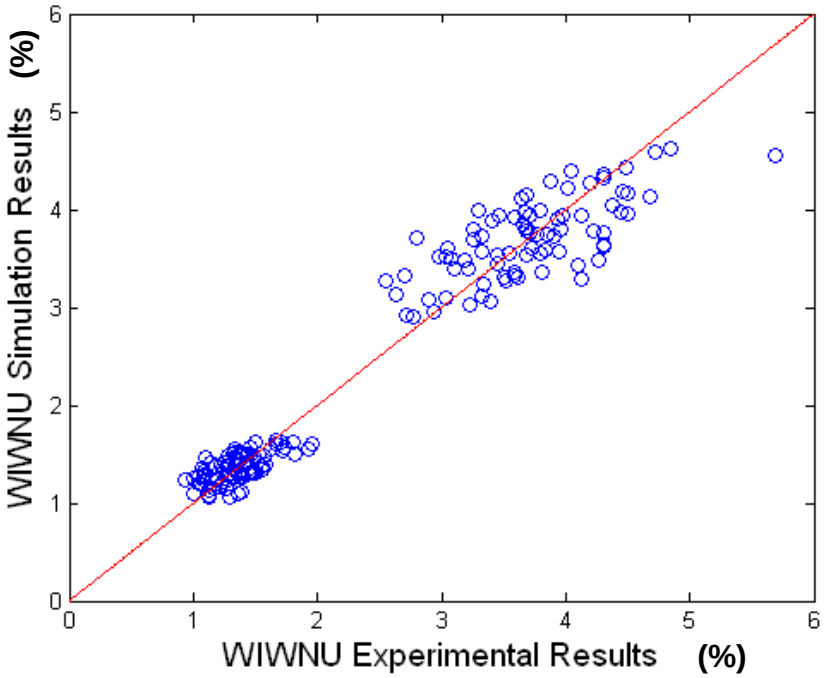


Figure 6.58 Training Results for WIWNU NN 5-2-3-1 Model (Group III)

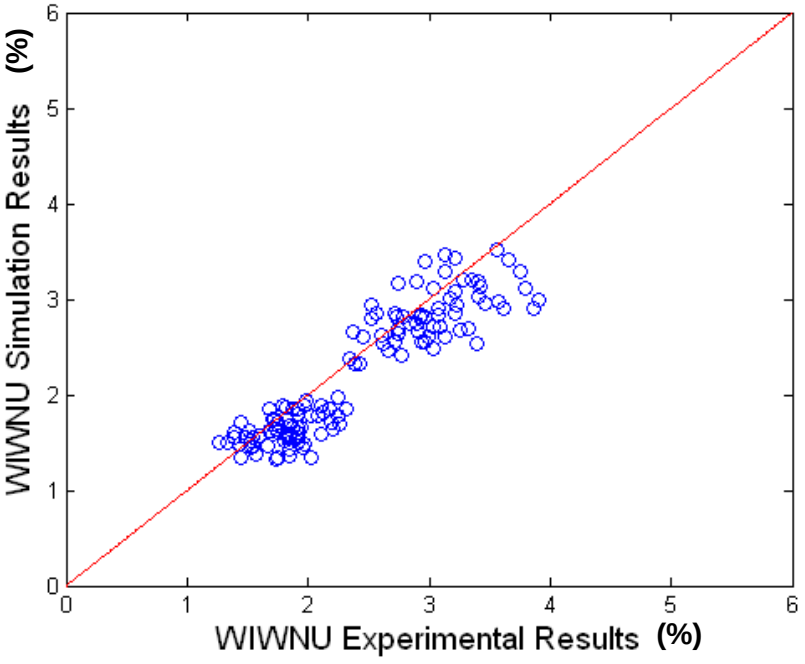


Figure 6.59 Testing Results for WIWNU NN 5-2-3-1 Model (Group III)

6.3.4 Models Using Genetic Algorithms

6.3.4.1 Results with Group I Data

GA Models of MRR with Using Group I Data

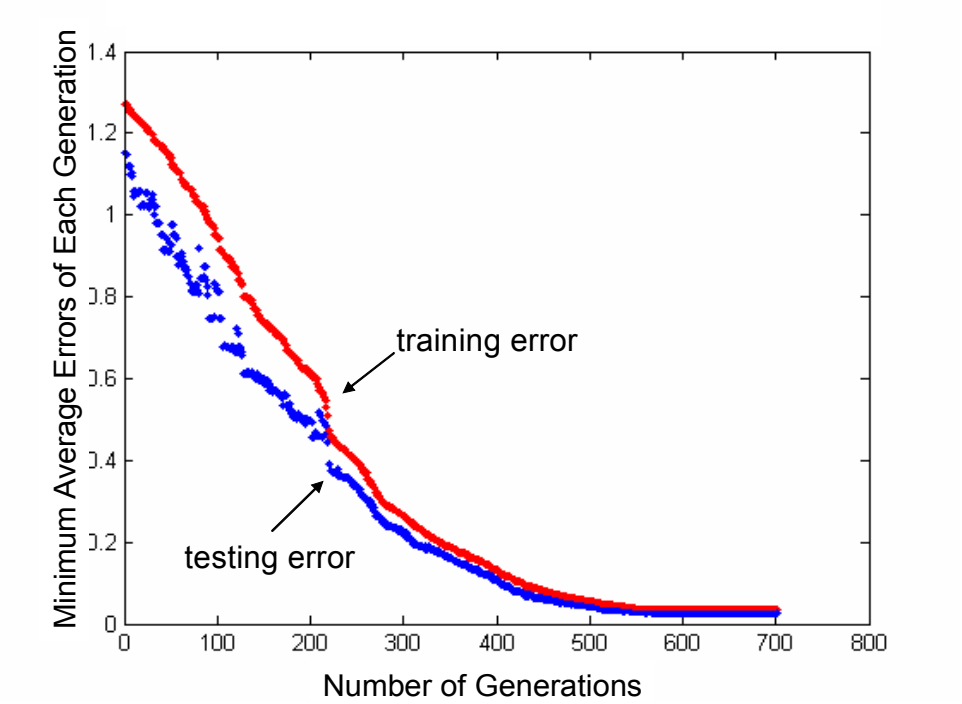


Figure 6.60 Training results for MRR using GA Model (Group I)

Initially, the MRR model structure was assumed to be of the form

$$MRR = C \cdot Sc^{u} \cdot Pd^{v} \cdot Pb^{w} \cdot Vp^{x} \cdot T^{y} \tag{6.1}$$

Here 50 chromosomes with different values for $[C\ u\ v\ w\ x\ y]$ were randomly generated to for an initial population. All the input parameters are coded between [1 , 3]. Through evolutionary operations, using a GA, a search for the best or optimal chromosomes (or solutions) was performed through a number of iterations ranging from 700 to 1200.

After 701 iterations, the training and testing errors have both converged to 0.034 as shown in the above Figure 6.60. The best solution, and the training and testing errors are tabulated in the following. The training and testing results are given in Figures 6.61 and 6.62.

# of generations = 701
Best solution = [1.2256 0.0106 0.4069 -0.0429 0.3781 -0.0821]
Training error (%) = 2.7357 %
Testing error (%) = 2.2633 %
Standard deviation of training error (%) = 2.1455 %
Standard deviation of testing error (%) =1.6690 %

The final MRR model is as follows:

$$\boldsymbol{MRR} = \frac{1.2256 \cdot \boldsymbol{Sc}^{0.0106} \cdot \boldsymbol{Dp}^{0.4069} \cdot \boldsymbol{Vp}^{0.3781}}{\boldsymbol{Bp}^{0.0429} \cdot \boldsymbol{T}^{0.0821}}$$

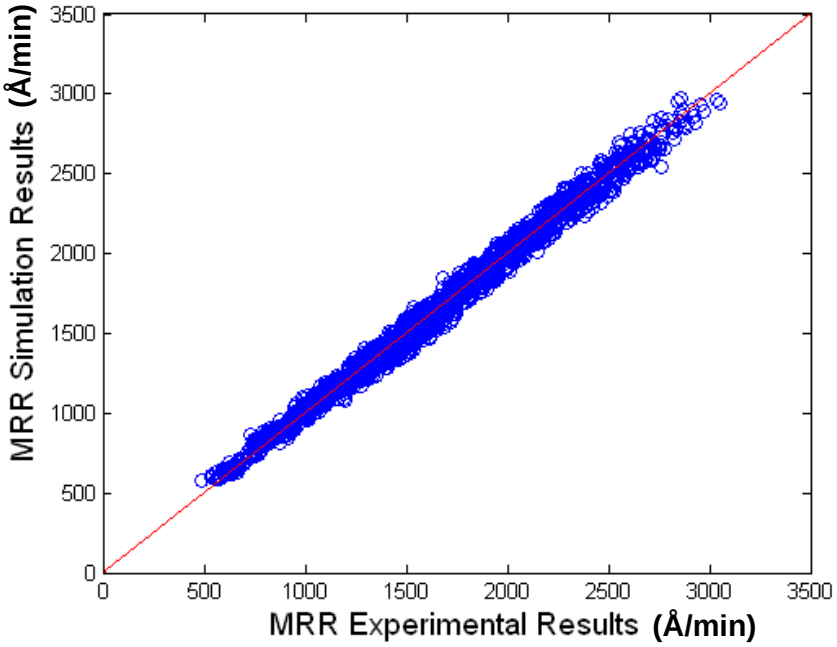


Figure 6.61 Training results for MRR using the GA model (Group I)

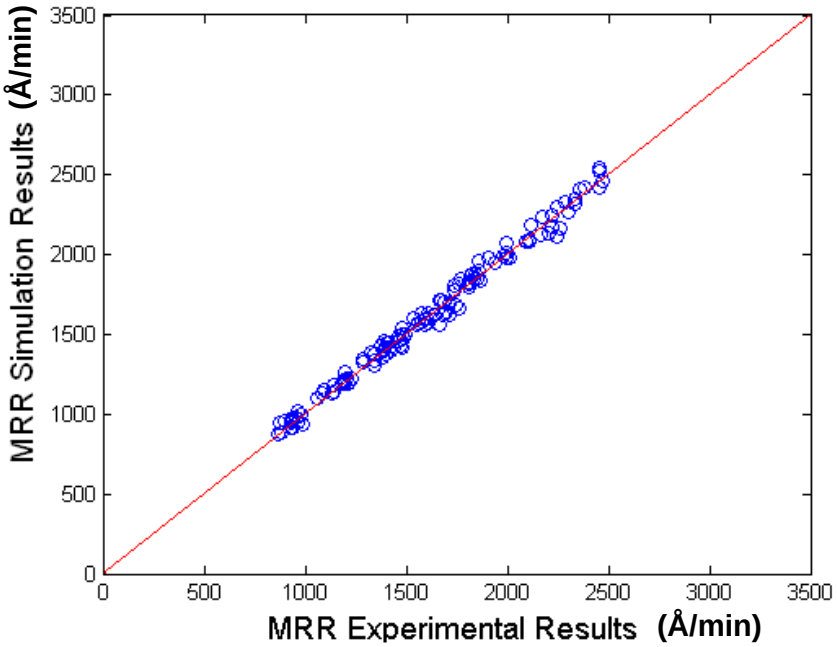


Figure 6.62 Testing Results for MRR using the GA Model (Group I)

GA Models of WIWNU with Using Group I Data

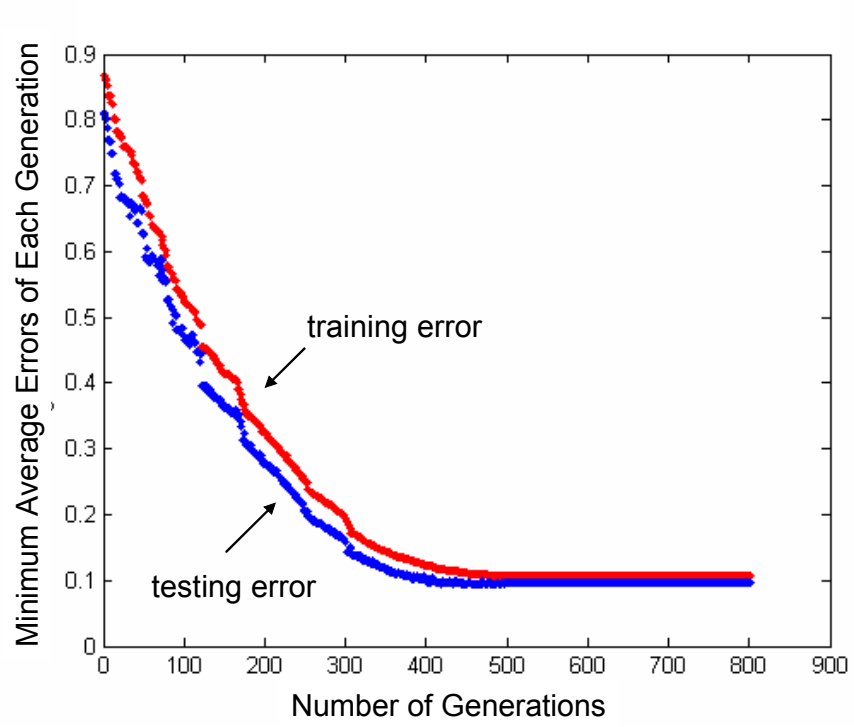


Figure 6.63 Training for GA WIWNU Model (Group I)

Initially, the WIWNU model structure was assumed to be of the form

$$WIWNU = C \cdot Sc^{u} \cdot Pd^{v} \cdot Pb^{w} \cdot Vp^{x} \cdot T^{y} \tag{6.2}$$

Here 50 chromosomes with different values for $[C\ u\ v\ w\ x\ y]$ were randomly generated to for an initial population. All the input parameters are coded between [1 , 3]. Through evolutionary operations, using a GA, a search for the best or optimal chromosomes (or solutions) was performed through a number of iterations ranging from 700 to 1200.

After 801 iterations, the training and testing errors have both converged to 0.1035 as shown in the above Figure 6.63. The best solution, and the training and testing errors are tabulated in the following. The training and testing results are given in Figures 6.63 and 6.64.

# of generations = 801

Best solution = [1.2026 0.0062 0.0210 -0.1121 0.0562 0.5219]

Training error (%) = 10.5739 %

Testing error (%) = 10.6734 %

Standard deviation of training error (%) = 8.4020 %

Standard deviation of testing error (%) = 8.3962 %

The final MRR model is as follows:

$$WIWNU = \frac{1.2026 \cdot Sc^{0.0062} \cdot Pd^{0.0210} \cdot Vp^{0.0562} \cdot T^{0.5219}}{Pb^{0.1121}}$$

Figures 6.64 and 6.65 show the training and testing results for this case.

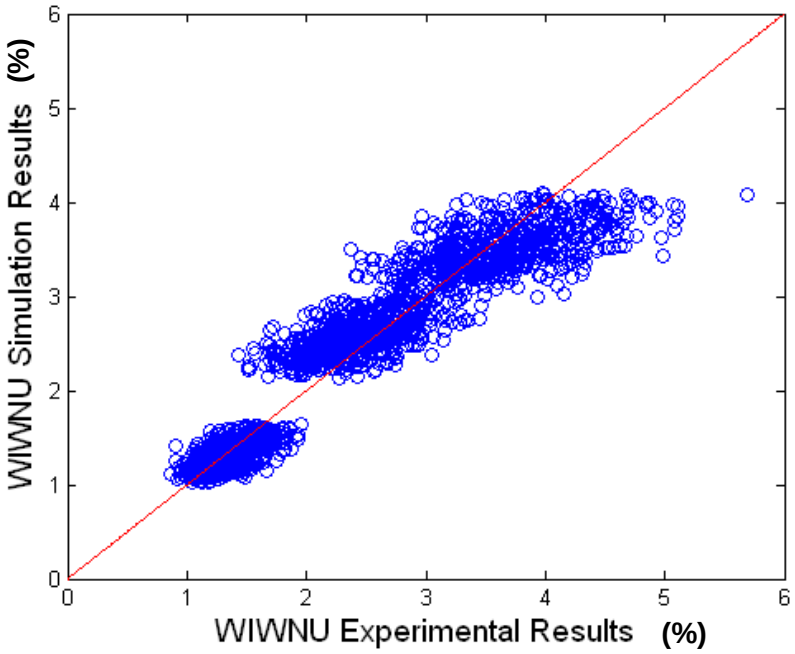


Figure 6.64 Training Results for WIWNU GA Model (Group I)

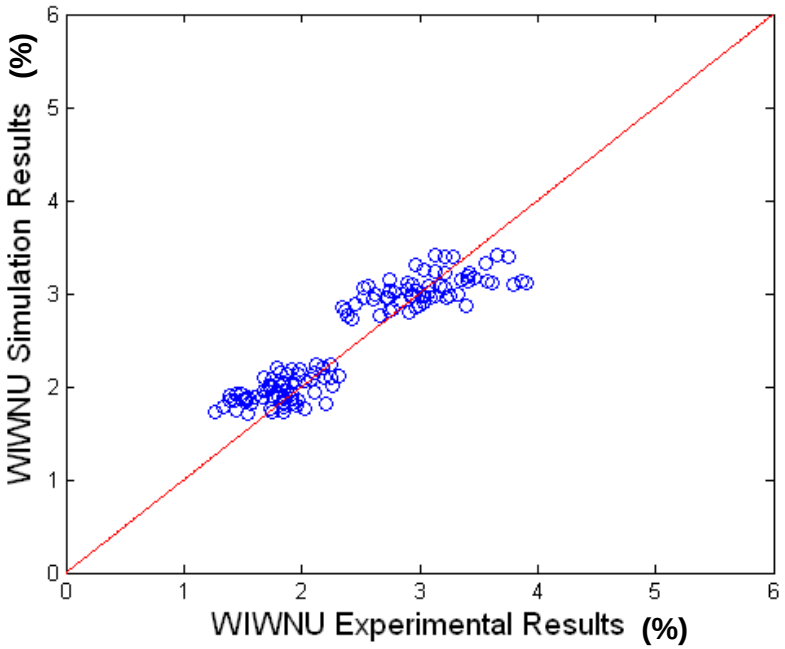


Figure 6.65 Testing Results for WIWNU GA Model (Group I)

6.3.4.2 Results with Group II Data

GA Models of MRR with Using Group II Data

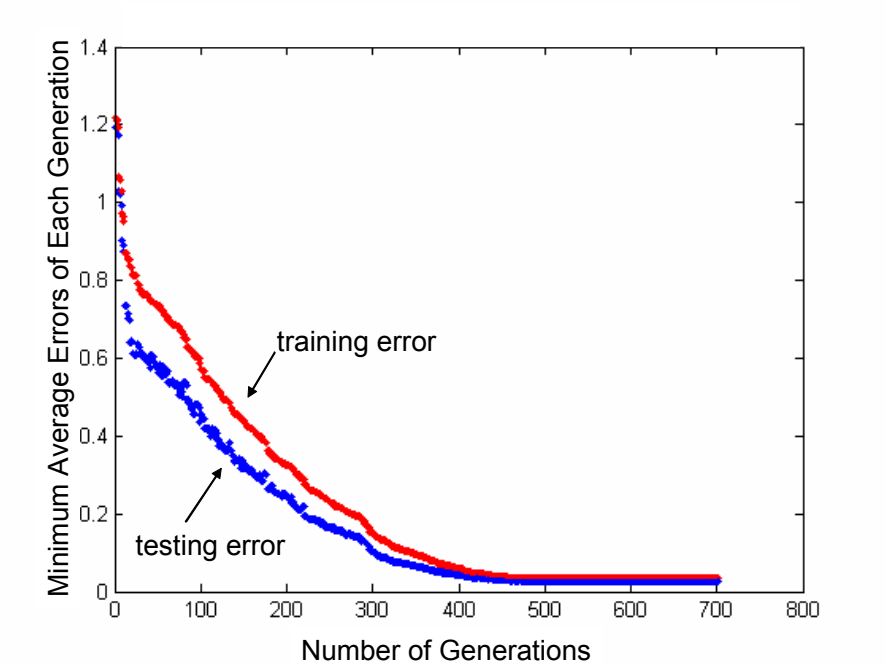


Figure 6.66 Training for GA MRR Model (Group II)

Figures 6.67 and 6.68 show the training and testing results for this case.

# of generations = 701

Best solution = [1.2282 0.0119 0.4057 -0.0462 0.3790 -0.0799]

Training error (%) = 2.8178 %

Testing error (%) = 2.3414 %

Standard deviation of training error (%) = 2.3891 %

Standard deviation of testing error (%) =1.7311 %

$$MRR = \frac{1.2282 \cdot Sc^{0.0119} \cdot Dp^{0.4057} \cdot Vp^{0.3790}}{Bp^{0.0462} \cdot T^{0.0799}}$$

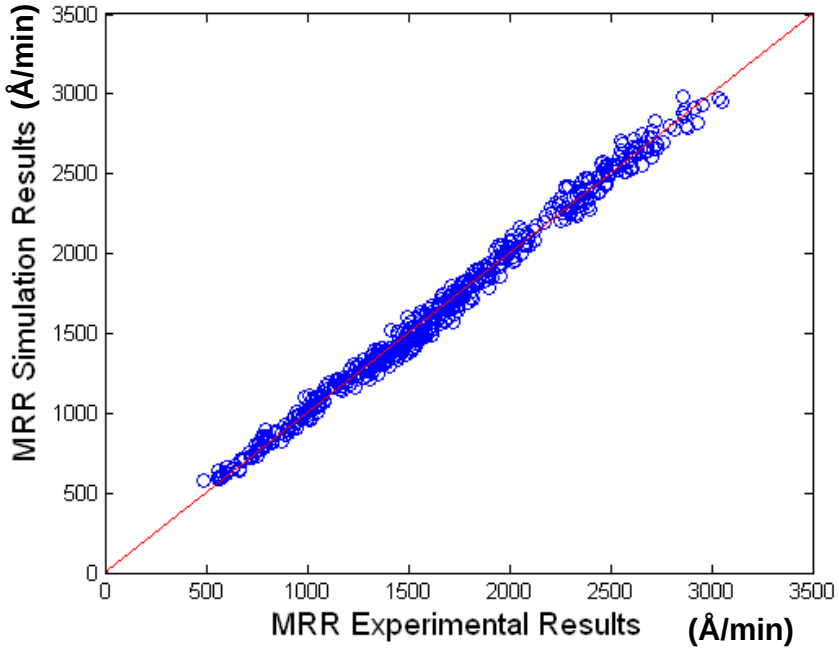


Figure 6.67 Training Results for MRR GA Model (Group II)

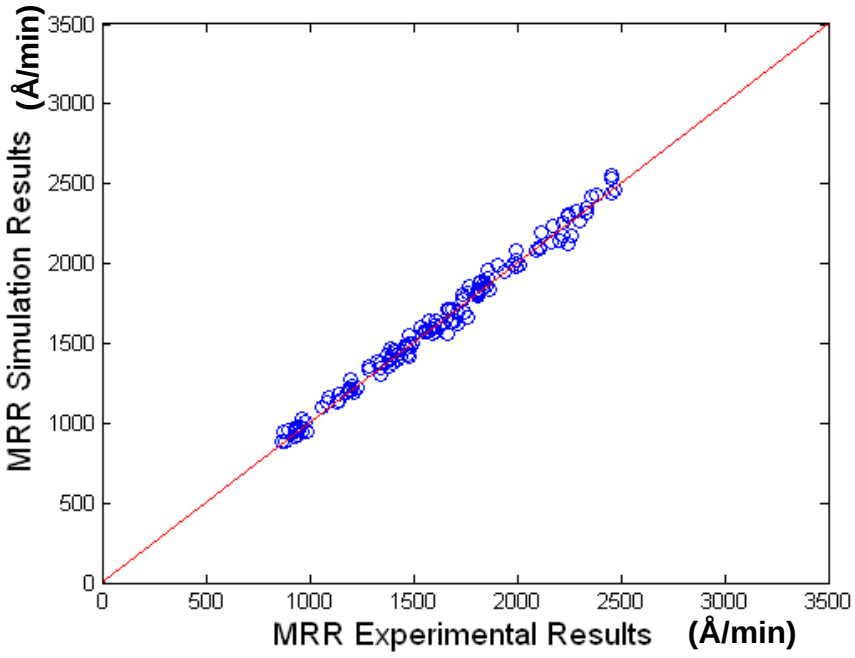


Figure 6.68 Testing Results for MRR GA Model (Group II)

GA Models of WIWNU with Using Group II Data

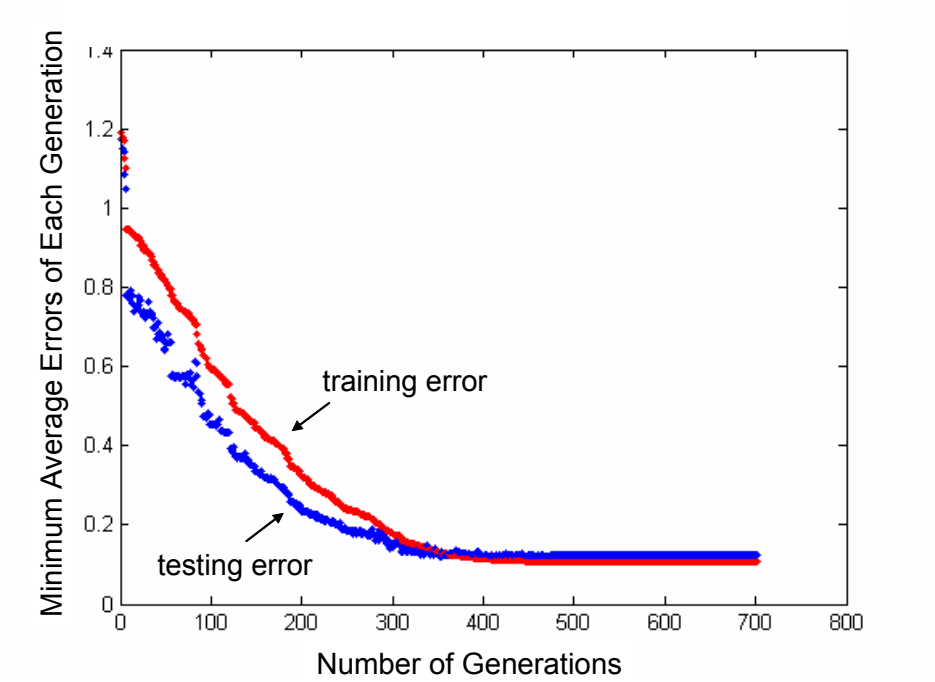


Figure 6.69 Training for GA WIWNU Model (Group II)

Figures 6.70 and 6.71 show the training and testing results for this case.

# of generations = 801
Best solution = [1.2526 0.0096 0.0164 -0.1313 0.0631 0.5691]
Training error (%) = 9.2106 %
Testing error (%) = 12.1558 %
Standard deviation of training error (%) = 7.2398 %
Standard deviation of testing error (%) = 9.7696 %

$$WIWNU = \frac{1.2526 \cdot Sc^{0.0096} \cdot Pd^{0.0164} \cdot Vp^{0.0631} \cdot T^{0.5691}}{Pb^{0.1313}}$$

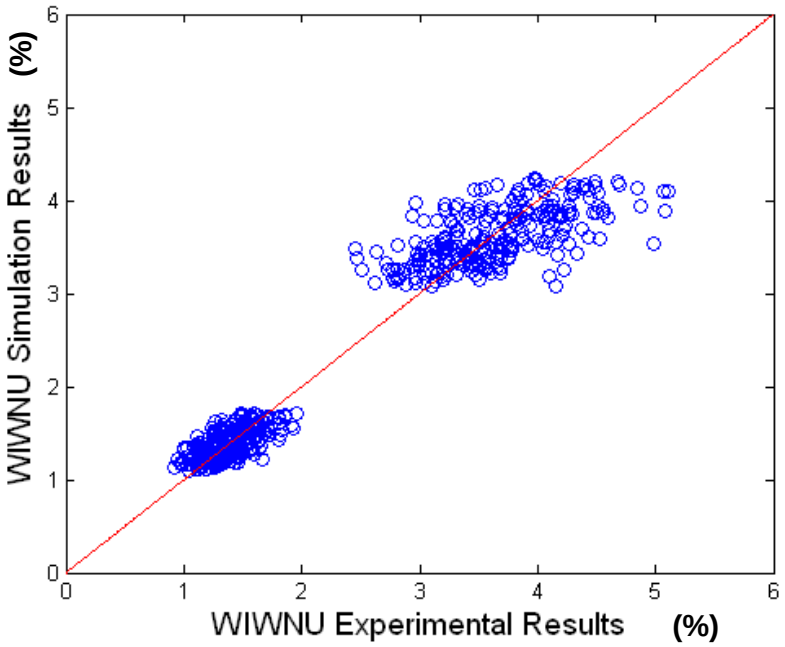


Figure 6.70 Training Results for WIWNU GA Model (Group II)

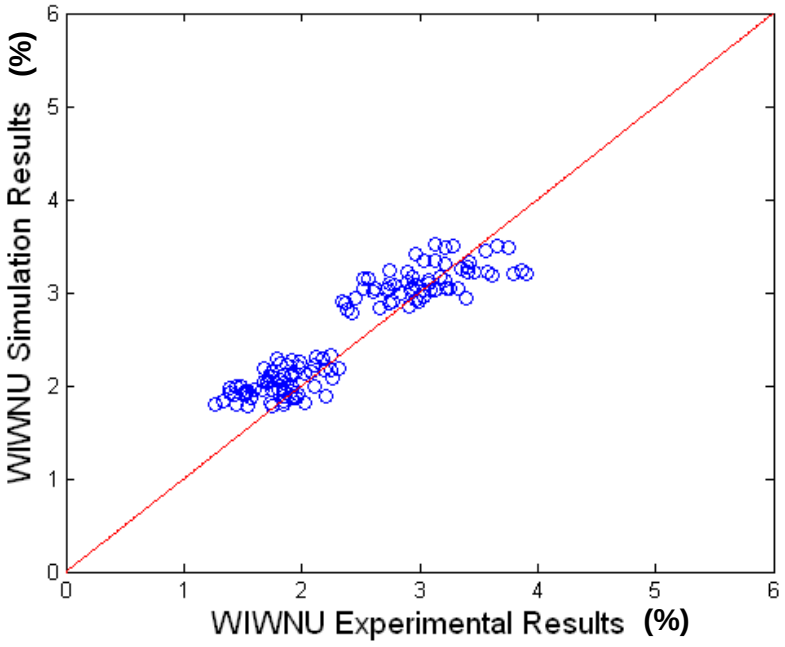


Figure 6.71 Testing Results for WIWNU GA Model (Group II)

6.3.4.3 Results of Group III

GA Models of MRR with Using Group III Data

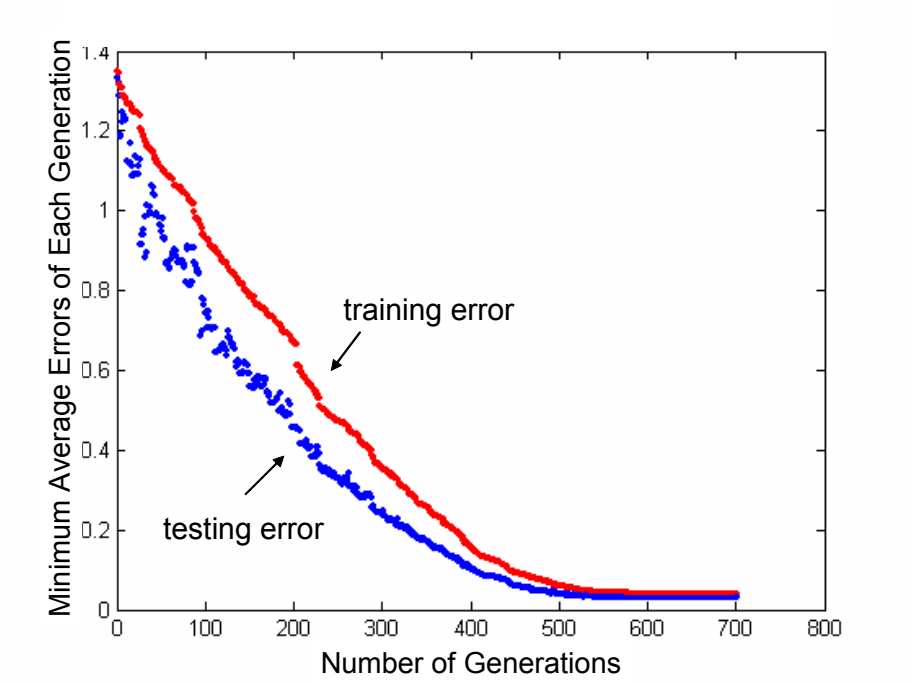


Figure 6.72 Training for GA MRR Model (Group III)

$$MRR = \frac{1.2425 \cdot Sc^{0.0067} \cdot Pd^{0.4141} \cdot Vp^{0.3883}}{Pb^{0.04} \cdot T^{0.0874}}$$

# of generations = 701
Best solution = [1.2425 0.0067 0.4141 -0.0400 0.3883 -0.0874]
Training error (%) = 3.2752 %
Testing error (%) = 2.4259 %
Standard deviation of training error (%) = 2.9372 %
Standard deviation of testing error (%) =1.8076 %

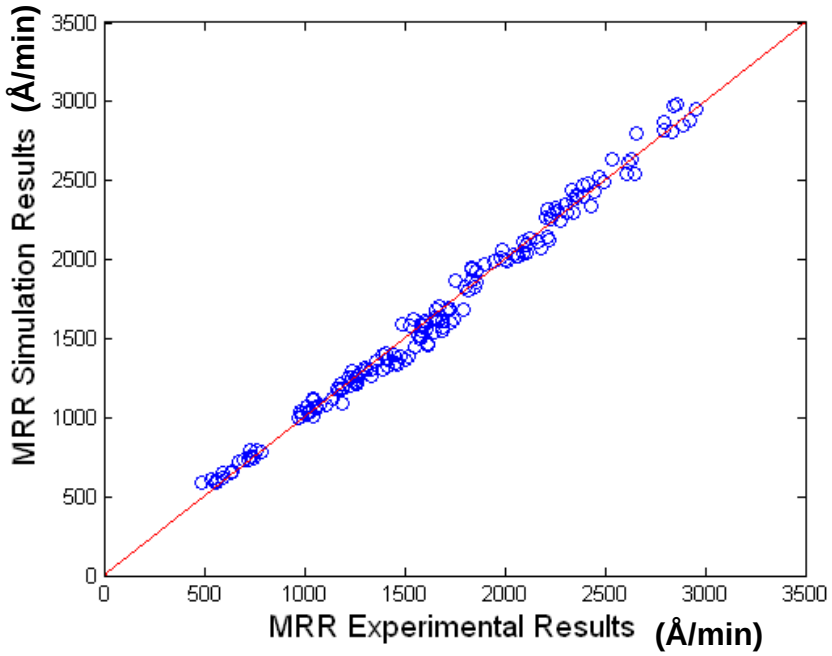


Figure 6.73 Training Results for MRR GA Model (Group III)

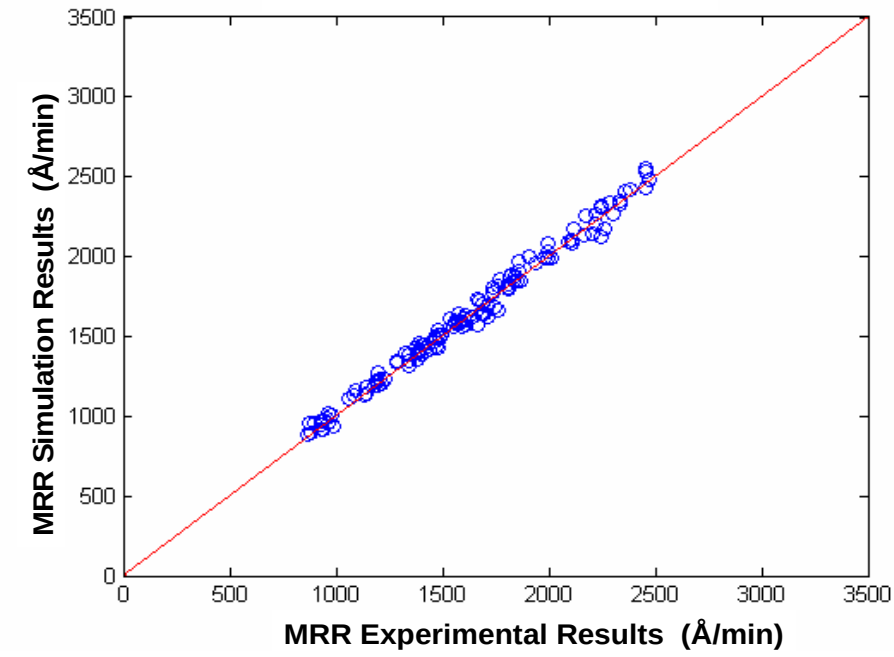


Figure 6.74 Testing Results for MRR GA Model (Group III)

GA Models of WIWNU with Using Group III Data

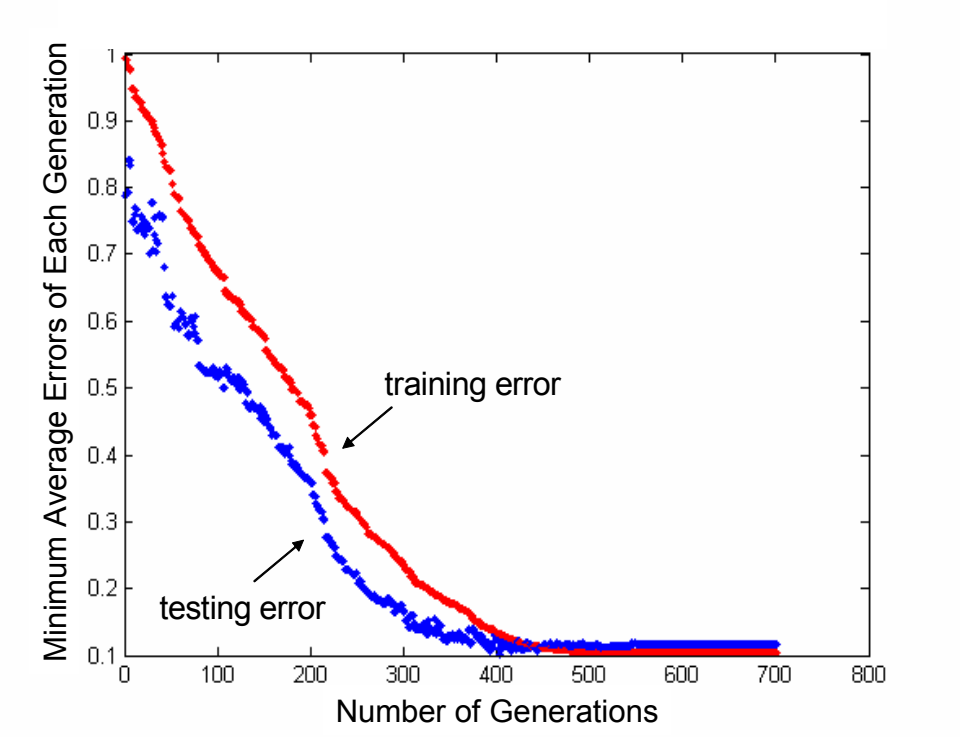


Figure 6.75 Training for GA WIWNU Model (Group III)

Figures 6.76 and 6.77 show the training and testing results for this case.

# of generations = 801
Best solution = [1.2436 -0.0204 -0.0123 -0.1074 0.0534 0.5510]
Training error (%) = 9.8708 %
Testing error (%) = 13.2084 %
Standard deviation of training error (%) = 7.9125 %
Standard deviation of testing error (%) = 10.5573 %

$$WIWNU = \frac{1.2436 \cdot Vp^{0.0534} \cdot T^{0.5510}}{Sc^{0.0204} \cdot Pd^{0.0123} \cdot Pb^{0.1074}}$$

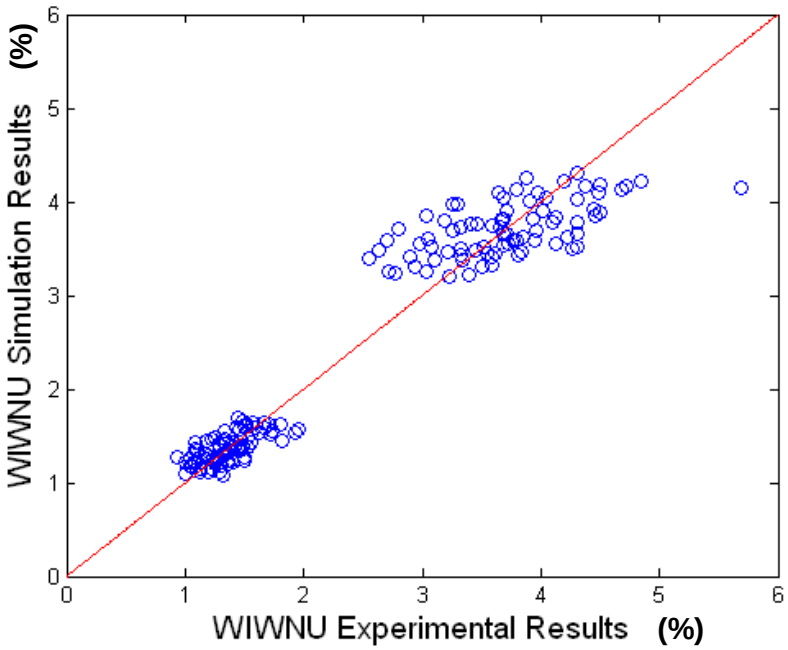


Figure 6.76 Training Results of WIWNU GA Model (Group III)

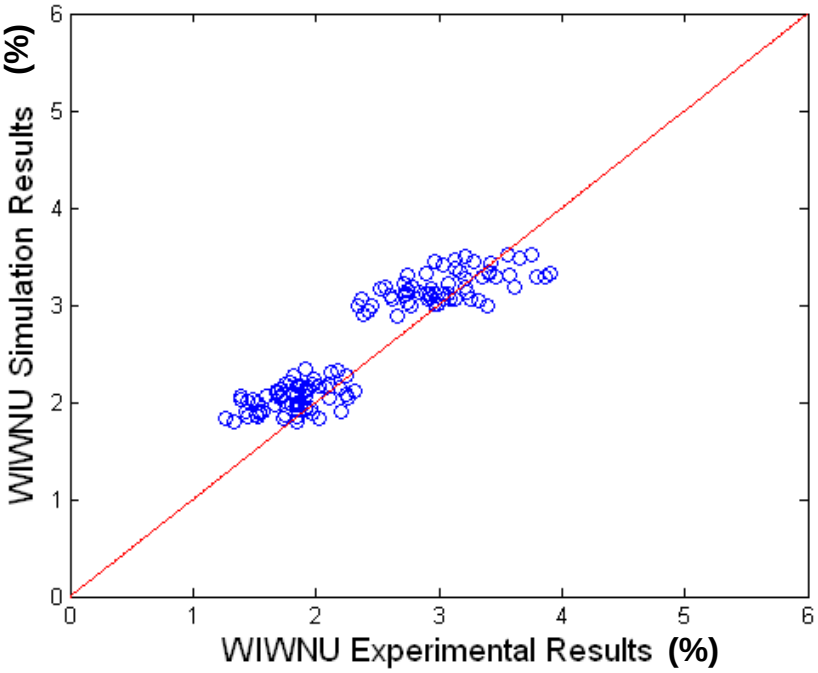


Figure 6.77 Testing Results for WIWNU GA Model (Group III)

6.3.5 Summary of Modeling Errors in Group I, II and III

Table 6.26 Summary of Modeling Errors in Group I

Experiment III - Group I (1875) MRR		ANFIS-GP 2-2-2-2-3 Linear	ANFIS-SC 12-Rule	Neural Network 5-16-5-1 tansig	Genetic Algorithms
Training error (%)	Average	2.2078	2.2104	2.1219	2.7356
	Standard deviation	1.4968	1.5837	1.5413	2.1448
Testing error (%)	Average	2.2157	2.2359	2.2389	2.2631
	Standard deviation	1.5322	1.4431	1.5272	1.6683
Experiment III - Group I (1875) WIWNU		ANFIS-GP 2-2-2-3-2 Linear	ANFIS-SC 7-Rule	Neural Network 5-16-13-1 tansig	Genetic Algorithms
Training error (%)	Average	9.1122	9.5765	9.5851	10.574
	Standard deviation	7.3286	7.7639	7.7486	8.4019
Testing error (%)	Average	9.6952	9.1558	9.06	10.6732
	Standard deviation	6.7805	6.1387	6.0803	8.3962

From Table 6.25, it is evident that the errors for all four models for MRR are very small (< 2.8%) and close to each other when adequate training data sets are available as in the Group I case. Due to higher non-linearity relationships, the average errors from the four WIWNU models are higher than those for MRR and they remain close to 10%.

Table 6.27 Summary of Modeling Errors in Group II

Experiment III - Group II (512) MRR		ANFIS-GP 2-2-2-2-2 Linear	ANFIS-SC 12-Rule	Neural Network 5-9-7-1 tansig	Genetic Algorithms
Training error (%)	Average	1.8051	2.1449	2.5199	2.8178
	Standard deviation	1.3543	1.6009	2.1664	2.3891
Testing error (%)	Average	2.3884	2.388	2.3992	2.3414
	Standard deviation	1.7385	1.6098	1.5359	1.7311
Experiment III - Group II (512) WIWNU		ANFIS-GP 2-2-2-3-2 Linear	ANFIS-SC 5-Rule	Neural Network 5-2-5-1 tansig	Genetic Algorithms
Training error (%)	Average	7.1631	9.1191	9.137	9.2106
	Standard deviation	5.5523	7.0589	7.159	12.1558
Testing error (%)	Average	10.3907	9.4168	10.9449	7.2398
	Standard deviation	7.8027	6.6441	7.7574	9.7696

Table 6.3.21 summarizes the model errors in Group II. The results are similar to the previous case (Table 6.25) in that, the average training and testing errors from all four MRR models and four WIWNU models vary from 2 to 3 % for MRR and 7 to 10 % for WIWNU, respectively. ANFIS-GP models for both MRR and WIWNU yielded less training and testing errors, compared to the other three modeling methods.

Table 6.28 Summary of Modeling Errors in Group III

Experiment III - Group III (162) MRR		ANFIS-GP 2-2-2-3-2 Linear	ANFIS-SC 24-Rule	Neural Network 5-6-2-1 tansig	Genetic Algorithms
Training error (%)	Average	1.2973	2.0228	5.4786	3.2752
	Standard deviation	1.0517	1.5624	6.1808	2.9372
Testing error (%)	Average	2.4711	2.5703	3.4925	2.4259
	Standard deviation	2.0272	1.9065	2.7797	1.8076
Experiment III - Group III (162) WIWNU		ANFIS-GP 2-2-2-3-2 Linear	ANFIS-SC 3-Rule	Neural Network 5-2-3-1 tansig	Genetic Algorithms
Training error (%)	Average	4.2443	9.3914	9.5333	9.8708
	Standard deviation	3.638	7.1057	7.0761	7.9125
Testing error (%)	Average	9.934	9.8541	11.014	13.2084
	Standard deviation	7.2834	6.6393	7.4818	10.5573

Table 6.3.22 summarizes the model errors in Group III. For the case of MRR, ANFIS-GP and –SC models the errors are about the same as previous groups (i.e., Groups I and II), but the errors from NN increases significantly due apparently to the absence of adequate data (sparse data). For the WIWNU case, ANFIS-GP model still performed better than the other three modeling methods. This is because, unlike NN, an ANFIS-GP linear model possesses more adjustable parameters in the premise part (i.e.

membership function) and consequent part, both of which provide more flexibility to learn the unknown system during the training process.

Figure 6.78 Training and Testing Errors for ANFIS-GP MRR Models

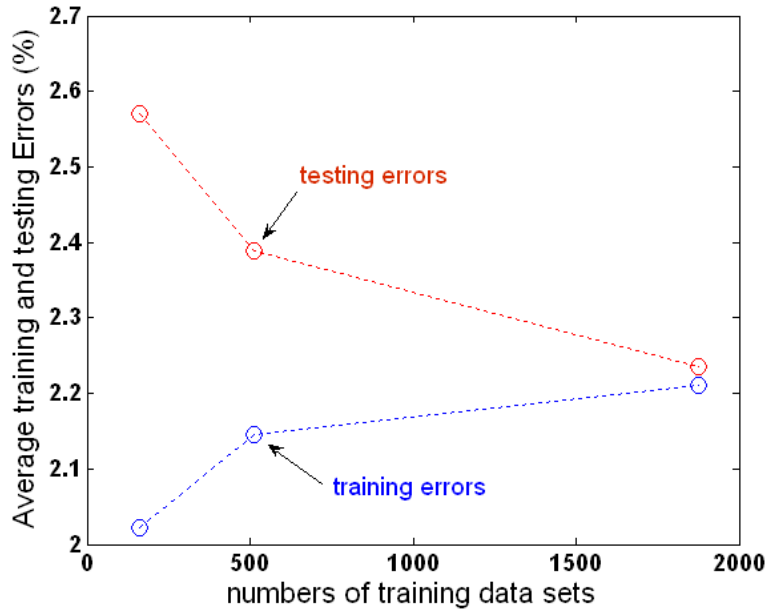


Figure 6.79 Training and Testing Errors for ANFIS-SC MRR Models

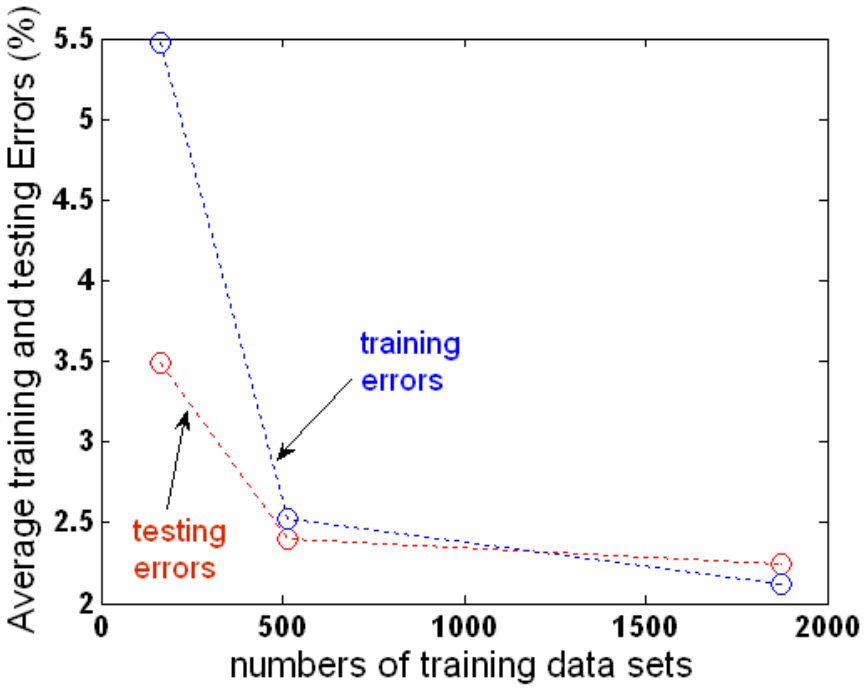


Figure 6.80 Training and Testing Errors for NN MRR Models

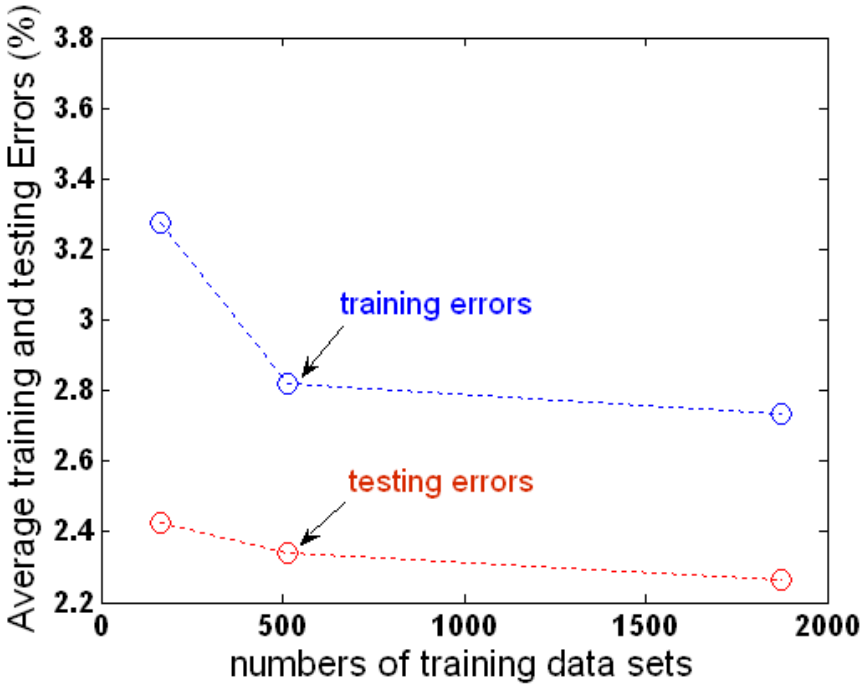


Figure 6.81 Training and Testing Errors for GA MRR Models

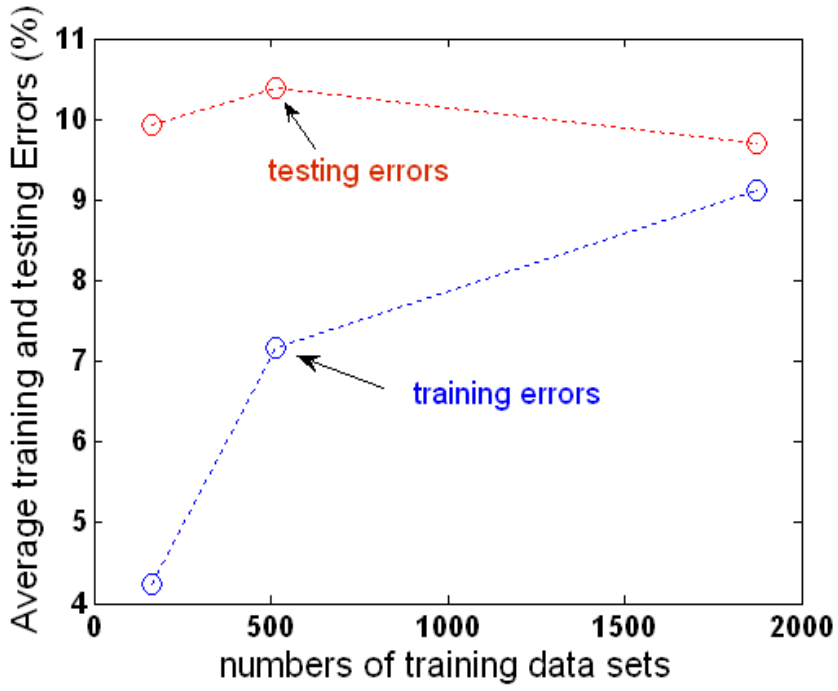


Figure 6.82 Training and Testing Errors for ANFIS-GP WIWNU Models

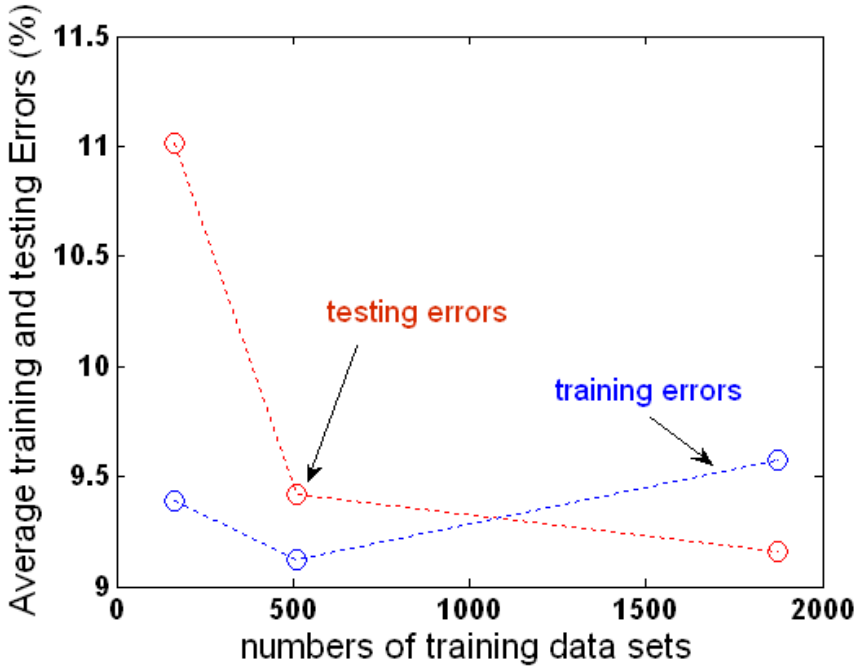


Figure 6.83 Training and Testing Errors for ANFIS-SC WIWNU Models

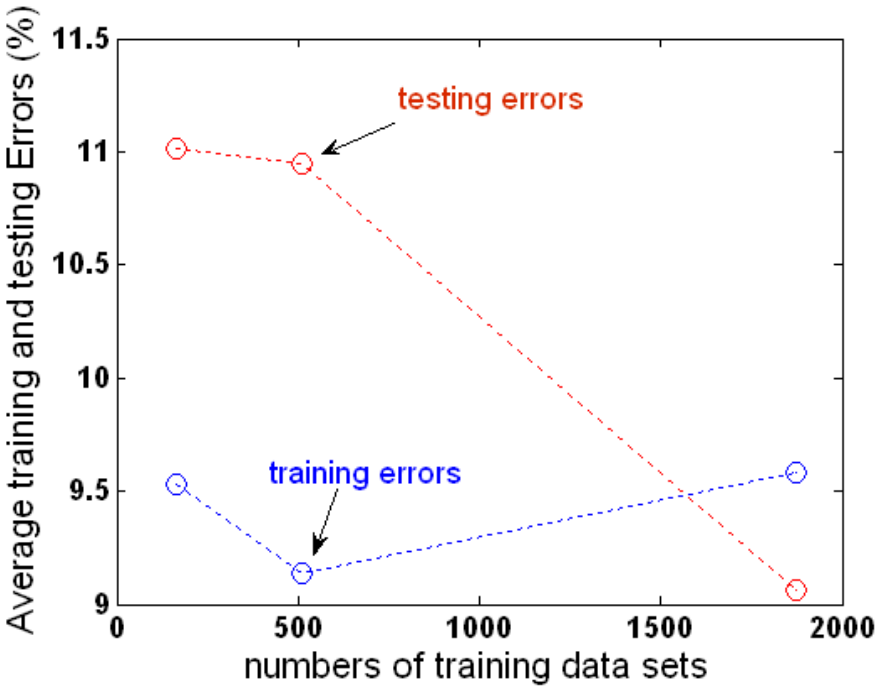


Figure 6.84 Training and Testing Errors for NN WIWNU Models

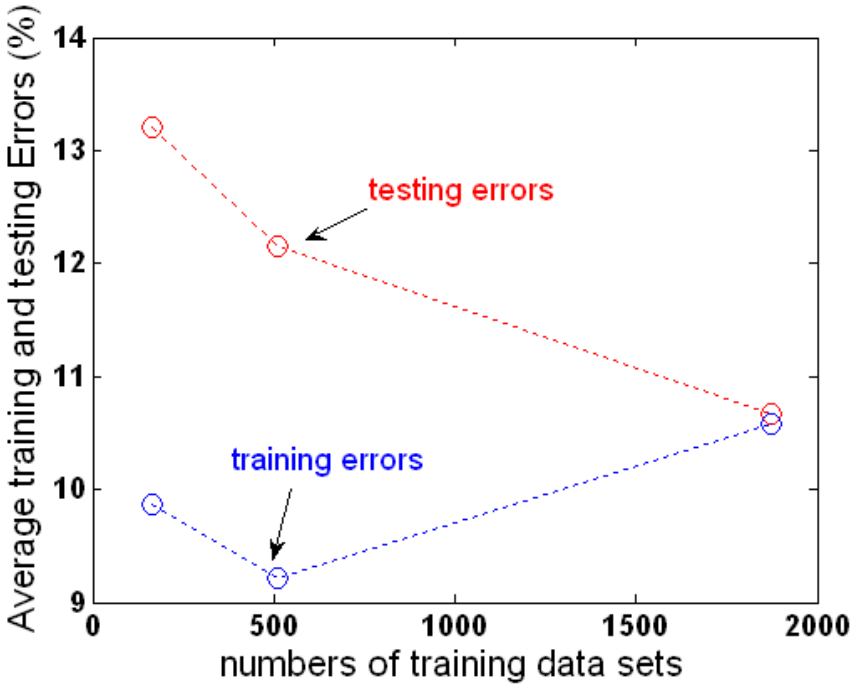


Figure 6.85 Training and Testing Errors for GA WIWNU Models

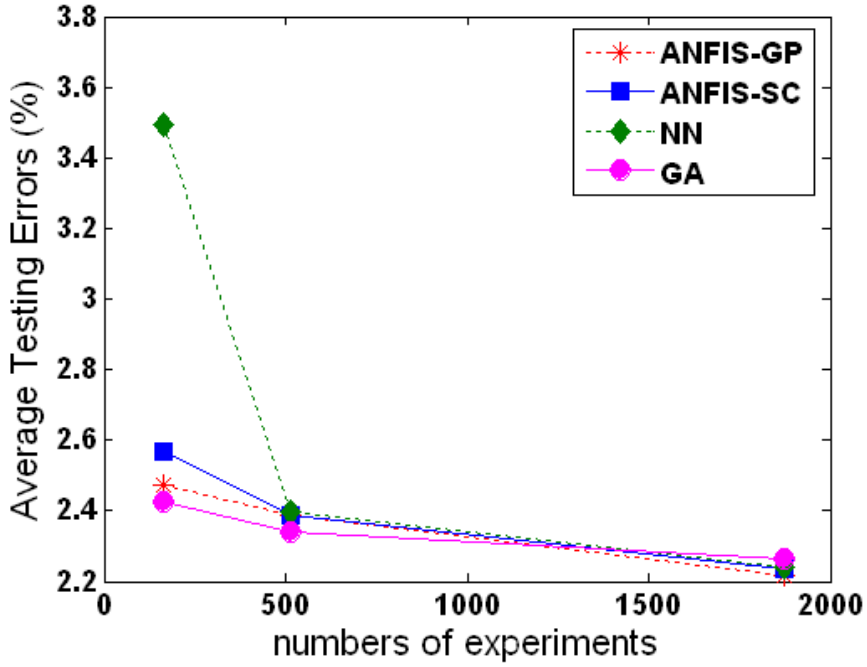


Figure 6.86 Testing Errors for ALL MRR Models

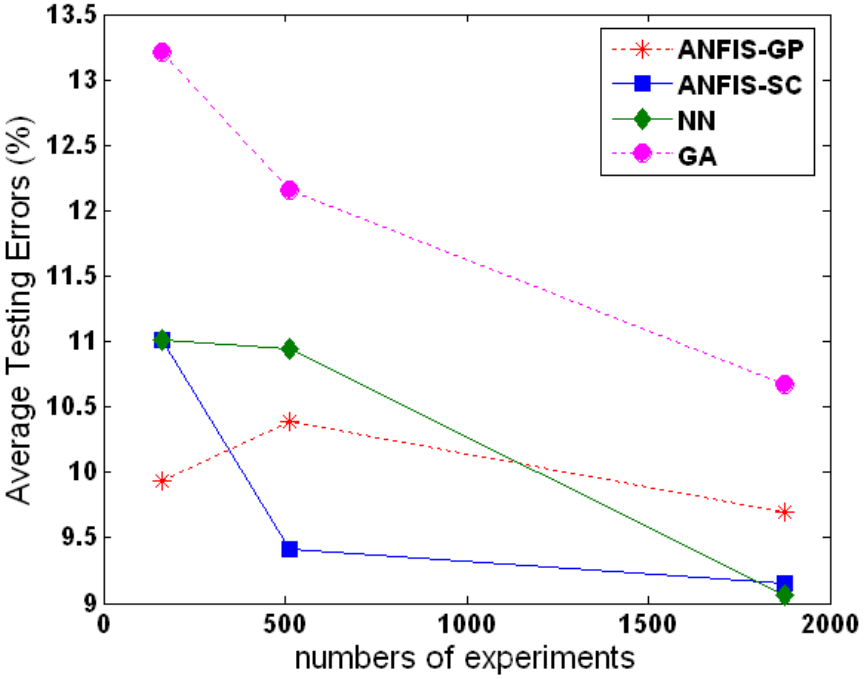


Figure 6.87 Testing Errors for ALL WIWNU Models

6.3.6 Group IV - Sparse Data for ANFIS-SC and GA Modeling

6.3.6.1 Fine-tuning Procedures for ANFIS-SC Modeling

In order to verify the modeling capability of ANFIS-SC to fit sparse-data case, 25 training data sets (Group IV) were picked based on a Taguchi array. Two fine-tuning methods (I and II) were used as described:

Method I

- Step1: Choose a suitable ANFIS-SC structure, usually starting from the number of fuzzy rules which is no more than the number of training data sets. Under the MATLAB GUI programming environment, the *range of influenc*e and *squash factor* are two main parameters to control the generated ANFIS-SC structure.
- Step2: Choose a sufficient training iteration for converging training and testing errors to some unchanged values.
- Step3: Repeatedly adjust or replace the membership functions of statistically insignificant input variables and implement re-starting the training process to reduce the testing errors to the acceptable ranges.

If the relationships between the output and input variables which are adjusted are roughly linear (usually replacing with linear-type membership functions, such as triangle and trapezoid-shaped membership functions). Likewise, replace with nonlinear-type membership functions, such as Gaussian- or bell-shaped membership functions for nonlinear relationships between an input and the output.

Method II

- Step 1: The same as Method I.

- Step 2: Perform an early stopping of the training process. (no need to check for convergence here)
- Step 3: Repeatedly adjust or replace the membership functions in both significant and insignificant input variables, and implement re-starting and early stopping the training process to simultaneously reduce the training and testing errors.

Usually, Method II can provide much greater changes (i.e., increase or decrease) in training and testing errors, but it always holds higher risk to fail. Sometimes, it will effectively shorten the training time to continuously reduce the errors to the acceptable ranges. The following example shown in Table 6.29 was fine-tuned using Method II, the training and testing errors were shrunk to around 70% and 99% in only six fine tuning steps.

Table 6.29 Fine-tuning Results of MRR Group IV ANFIS-SC 6-Rule Model

Run #	Fine-tuning Procedures	epochs	training error	testing error	reduction %
0	choosing{ range of influence = 9, squach factor = 0.2, Accept ratio = 0.5 and Reject ratio = 0.15 } ==> ANFIS_SC 6-Rule MRR Model for Group IV	0			
1	no change	10	10.00	152.04	0.00%
2	Choosing Input 1 - Solid Content Membership Function #1, [63.63 15] ==> [63.63 12]	15	9.121	109.571	27.93%
3	Choosing Input 2 - Down Pressure Membership Function #3, [63.63 50] ==> [63.63 48]	10	0.134	67.2936	55.74%
4	Choosing Input 5 - Polishing Time Membership Function #1, Gaussian Shape ==> Triangle	2	0.126	61.1827	59.76%
5	Choosing Input 5 - Polishing Time Membership Function #2, Gaussian Shape ==> Triangle	2	0.077	64.0328	57.88%
6	Choosing Input 5 - Polishing Time Membership Function #4, Gaussian Shape ==> Triangle	5	0.071	46.0745	69.70%

6.3.6.2 Results of Modeling Spare-data Case Using GA

GA Models of MRR with Using Group IV Data

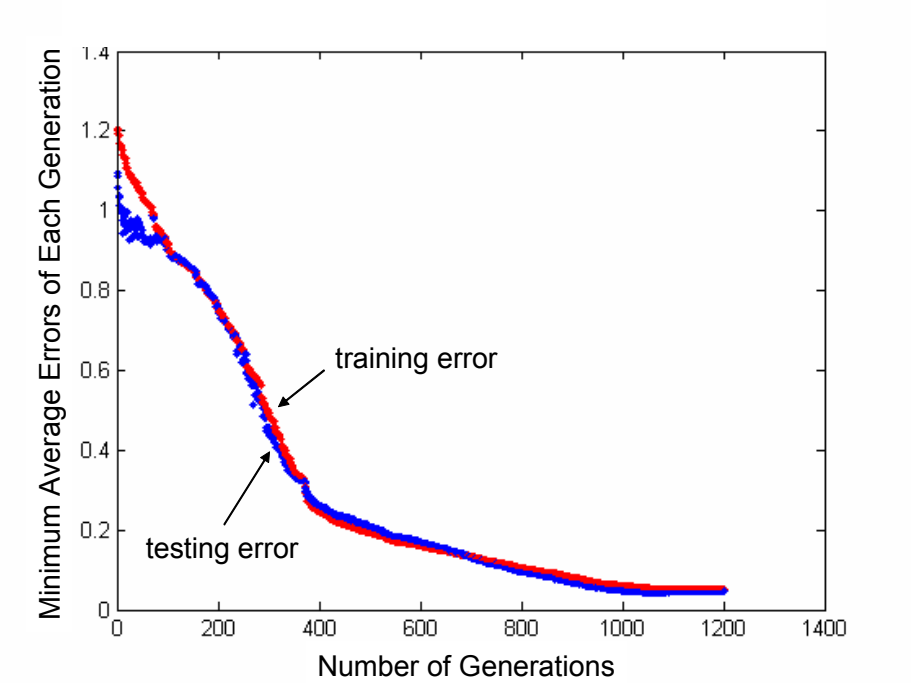


Figure 6.88 Training for GA MRR Model (Group IV)

Figures 6.89 and 6.90 show the training and testing results for this case.

```
# of generations = 1201
Best solution = [1.0283   0.0129   0.5919   -0.0195   0.4840   -0.1206]
Training error (%)  = 3.0874 %
Testing error (%) = 2.9296 %
Standard deviation of training error (%) = 3.3385 %
Standard deviation of testing error (%) =2.4735 %
```

$$MRR = \frac{1.0283 \cdot Sc^{0.0129} \cdot Pd^{0.5919} \cdot Vp^{0.4840}}{Pb^{0.0195} \cdot T^{0.1206}}$$

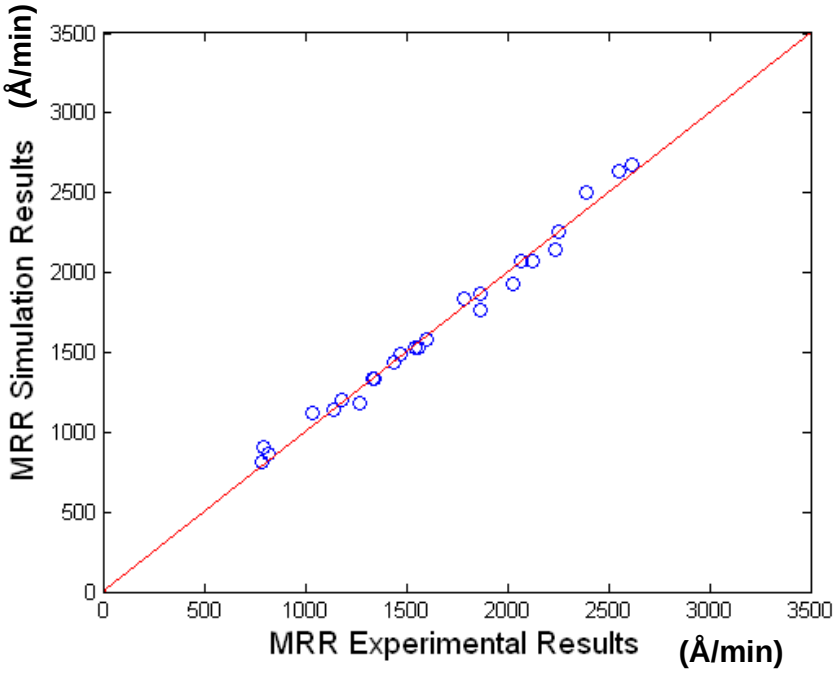


Figure 6.89 Training Results for MRR GA Model (Group IV)

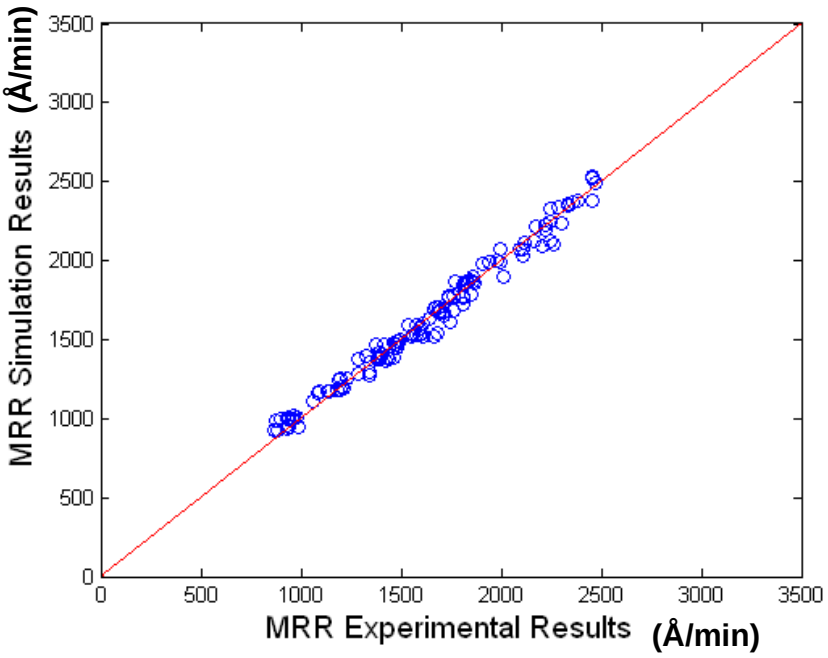


Figure 6.90 Testing Results for MRR GA Model (Group IV)

GA Model of WIWNU with Using Group IV Data

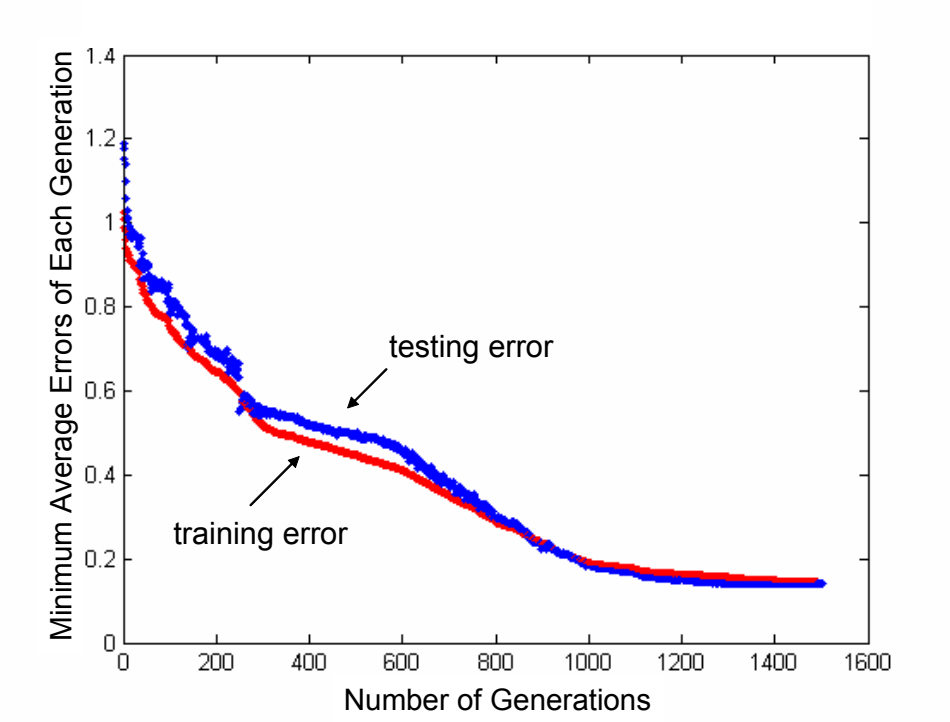


Figure 6.91 Training for GA WIWNU Model (Group IV)

Figures 6.92 and 6.93 show the training and testing results for this case.

# of generations = 1501
Best solution = [1.1039 0.0010 0.0696 -0.2665 0.1119 0.7697]
Training error (%) = 10.0224 %
Testing error (%) = 9.5152 %
Standard deviation of training error (%) = 8.6103 %
Standard deviation of testing error (%) = 6.1568 %

$$WIWNU = \frac{1.1039 \cdot Sc^{0.0010} \cdot Pd^{0.0696} \cdot Vp^{0.1119} \cdot T^{0.7697}}{Pb^{0.2665}}$$

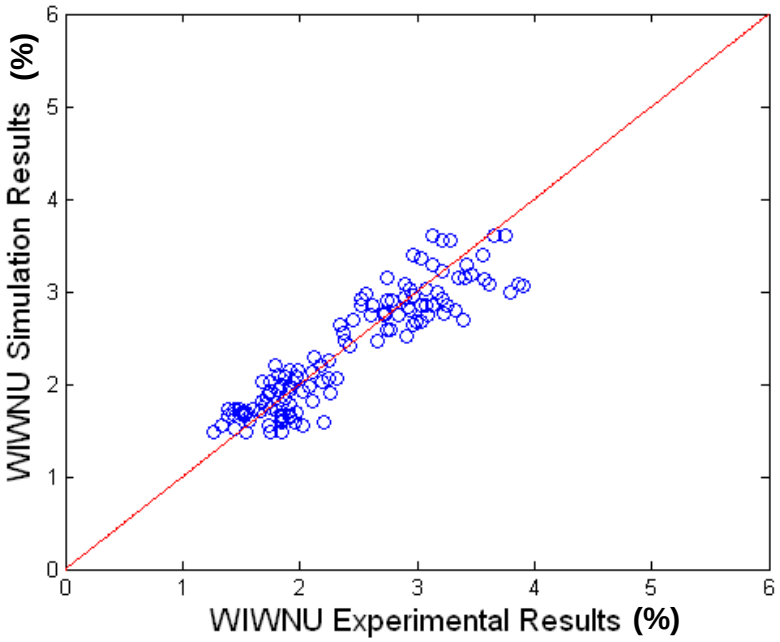


Figure 6.92 Training Results for WIWNU GA Model (Group IV)

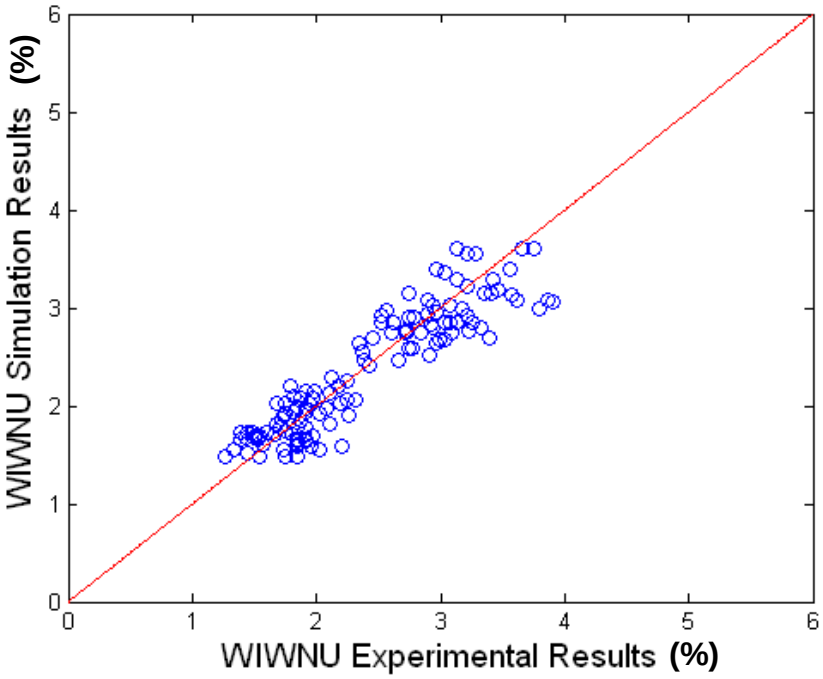


Figure 6.93 Testing Results for WIWNU GA Model (Group IV)

6.3.7 Optimization of CMP Processes Using Multi-Objective Evolutionary Algorithms (MOEA)

In this section, the ANFIS-GP models for MRR and WIWNU from Section 6.3.1.1 were used as the objective functions for the NSGA-II in MOEA to search the simulation results for optimal MRR and WIWNU. The simulation results form NSGA-II were validated using confirmation experiments (i.e., third-stage CMP experiments).

The Pareto-optimal simulation results are shown in Figure 6.94 and are listed in the Table 6.29. After 300 iterations, 16 Pareto-optimal points were searched. Theoretically, these points should be very crowded along the near-Pareto front, while the discrete settings on the CMP machine for each input variable, the optimal CMP processes (i.e., red points near the blue fitting curve) should be slightly dispersed away.

Table 6.29 shows the verification of the model with the experimental results. The MRR, the average prediction error for MRR is only 2.28% and around 12% for WIWNU. MOEA can certainly provide a very reliable direction to search for the optimal input settings for optimal MRR and WIWNU, if the process models used as objective functions were sufficiently accurate. Apart from the accurate modeling using NN, ANFIS and GA, the use of MOEA may contribute to improve the quality assurance of CMP process.

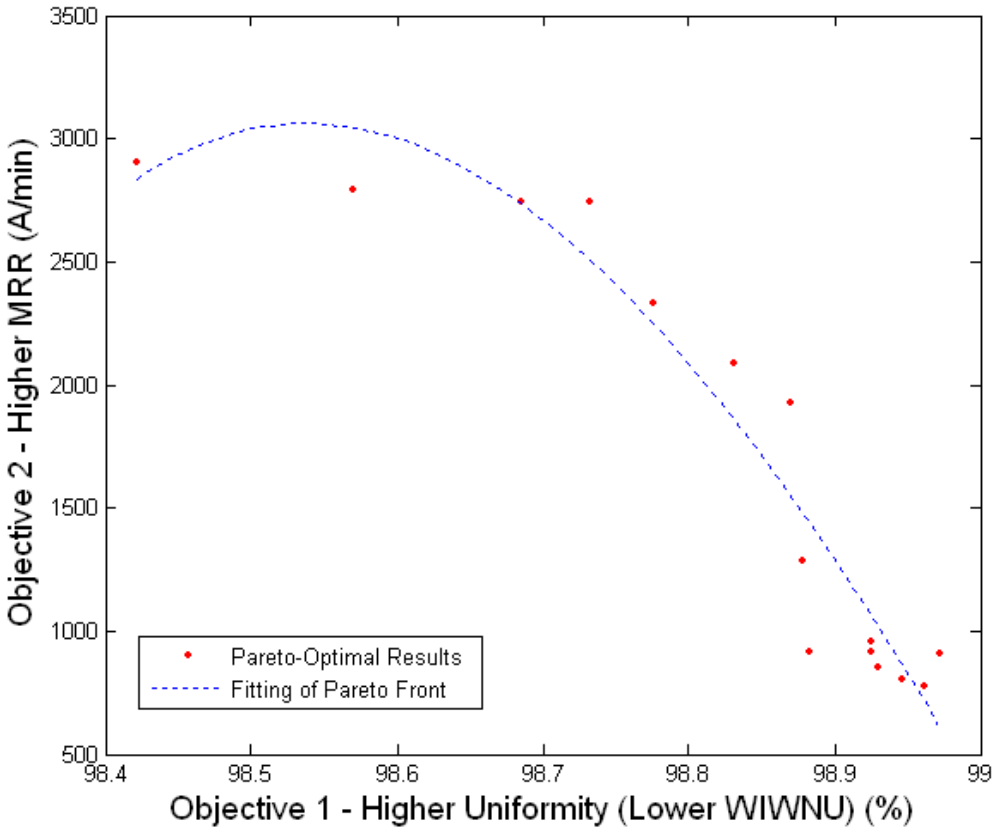


Figure 6.94 Pareto-optimal Results of MRR and WIWNU

Table 6.30 Pareto-optimal Results Simulated by NSGA-II

Inputs / Trials	*Sc*	*Pd*	*Pb*	*Vp*	*T*	Experiment MRR	Simulation MRR	error %	Experiment WIWNU	Simulation WIWNU	error %
1	7	11.9	0.4	60	40	2867.0	2905.0	1.33	1.71	1.579	7.68
2	14	11.9	2.4	59	41	2782.6	2791.4	0.32	1.61	1.430	11.18
3	12	11.7	3.8	60	42	2653.5	2743.3	3.38	1.60	1.314	17.86
4	11	11.8	3.7	59	41	2609.2	2743.1	5.13	1.64	1.267	22.73
5	22	10.7	2.2	46	40	2302.2	2329.3	1.18	1.58	1.224	22.52
6	20	8.4	3.7	50	40	2110.1	2084.0	1.24	1.35	1.169	13.41
7	18	7.6	3.9	48	40	1946.5	1926.3	1.04	1.24	1.130	8.85
8	21	4.6	3.8	25	41	869.6	915.5	5.28	1.57	1.118	28.82
9	24	4.2	3.5	22	40	801.8	805.5	0.46	1.06	1.053	0.64
10	24	4	3.7	22	40	780.1	777.0	0.40	0.95	1.038	9.29
11	22	4.6	3.9	26	40	914.4	953.9	4.32	1.29	1.075	16.69
12	20	4.3	3.9	24	40	888.6	854.7	3.81	1.02	1.070	4.92
13	5	8.2	3.9	22	40	1262.2	1286.4	1.92	1.18	1.122	4.91
14	25	4.5	3.8	25	41	919.3	912.9	0.70	1.09	1.075	1.39
15	24	4	3.7	22	40	760.5	777.0	2.17	1.06	1.038	2.05
16	25	4.2	3.9	26	40	873.9	907.4	3.83	1.20	1.027	14.41
							Average	2.28		Average	11.71
							Standard deviation	1.74		Standard deviation	8.36

Solid content = ***Sc*** (weight %), Down pressure = ***Pd*** (psi),
Back pressure = ***Pb*** (psi), Platen Speed = ***Vp*** (rpm),
Polishing time = ***T*** (sec),
MRR (Å/min), WIWNU (%)

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

7.1.1 Effect of Sparse Training Data on Model Accuracies

The results in Sections 6.1 and 6.2 for the experiments in Groups I and II provide the evidence that the sparse data can severely hamper the performance of NN and ANFIS-GP models. The sparseness of training data leave little scope for adjusting the weights in NN models, or the relative parameters in premise and consequent parts in ANFIS-GP models. Models trained by sparse data certainly cannot perform smaller testing errors, even though the training errors are very small. The high risk of overfitting situation can easily occur.

However, through effective fine-tuning steps, the deleterious effects of the sparse data can be overcome for ANFIS-SC models as explained in Fine-tuning the membership function shapes and supports of insignificant input process parameters allow for local adjustments of shapes that can significantly lower testing errors without causing many variations in the training errors. Also, from the results of GA models for Group IV (sparse data) in Sections 6.3.6.1 and 6.3.6.2, the sparseness of the data does not significantly affect GA models. This is because GA models with good choice of model structures need much fewer exemplar patterns for fitting compared to NN and ANFIS.

7.1.2 Influence of Experimental Designs

Apart from the insufficient *size* of training data sets for the sparse-data case, the *distribution* of the training data sets is an important issue. If training data sets can cover only a fraction of the process parameters space, the generalization capacities of the constructed models will be seriously affected. In addition to large testing errors, the simulation results might be continuously fluctuating between every training epoch, especially in NN model. In other words, the overfitting situations cannot be avoided. However, error fluctuations do not happen on the ANFIS cases; instead, the overfitting leads to small training errors and large testing errors, but they always keep the same (i.e., converge to some fixed values).

From the summary results shown in Tables 6.26 to 6.28, if the process is stable, the training data sets are selected by the factorial DOE and they fully cover the entire ranges of input variables, the constructed models can accurately simulate the whole polishing process, even with the fewer training-data case as seen in Group III (only 162 training data sets).

Thus, the *size* and *uniformity of distribution* of training data sets are both significant factors to construct the accurate process models, especially for NN, ANFIS and GA. The selected training data sets are picked more uniformly up from the whole input ranges, the better generalization capacities of the constructed models possess.

7.1.3 Determination of the Architectures for NN and ANFIS Models

Although for the NN and ANFIS models described in Section 6.3 are able to

perform accurate simulations, an inherent problem for these two modeling tools is "*what architecture is best for a given case*?" Definitely, different sizes and distributions of the training data sets need different NN and ANFIS architectures.

The performance of backpropagation NN used in this research is controlled by a number of design parameters, such as the network topology, type of transfer function, learning coefficients, and momentum parameter. A fixed combination of these parameters that can be applied to any specified problem, does not exist. Identifying the best combination of NN parameters for a particular application is usually done empirically and is an ongoing area of NN research. The objective in selecting the network size and training parameters is to obtain the network with the best *generalization* capabilities; where generalization is a measure of the network's performance on the data not used in training (i.e., testing data).

In this research, the optimization concepts derived from the GA were successfully employed to determine the optimal architectures for NN models. For simplicity, we only focused on emphasizing the numbers of hidden layers and the numbers of neurons in each hidden layer using GA. If we can form all NN parameters described in the previous paragraph to a chromosome and determine a suitable objective function, GA will be the one of best means for optimizing the architecture for the NN models.

In this research, all ANFIS models are determined based on traditional time-consuming trial and error methods. Surely, GA like optimization of architecture can be applied to search for the best ANFIS models. To illustrate this point, the same sizes and distribution types of training and testing data sets are used to construct the MRR and

WIWNU models in the Section 6.3.2.3, but the number of rules used for WIWNU case is 21 less than that for the MRR case. In fact, some useless fuzzy rules in MRR models may be present. These rules may not deteriorate the model performance, but certainly waste the training time. It is anticipated that our future research will focus on establishing useful algorithms for optimizing the ANFIS models.

7.1.4 Fine-tuning Techniques to Construct ANFIS-SC Models

Building on the concepts presented in Section 5.4.3, substantial improvements in predicting MRR and WIWNU characteristics in CMP are possible from the newly-developed technique based on fine-tuning the membership functions of ANFIS-SC models constructed using sparse experimental data sets used in Groups I and II. Fine-tuning the membership functions of insignificant input variables seems to be the key towards effectively minimizing the predicted errors and greatly improving the suitability of the established ANFIS-SC modeling methods for applications such as CMP where only sparse-data sets are available due to the expensive nature of experimental date collection physical and/or numerical simulations.

In reality, the large simulation errors from ANFIS-SC models shown in Tables 6.1, 6.2, 6.4 and 6.5 are resulted from the so-called overfitting-like results. Nevertheless, after useful fine-tuning operations, the testing errors tremendously decreased below 5% or lower. Because of the absence of fluctuations as in NN overfitting models, these ANFIS-SC models are still very reliable.

According to results shown in the Section 6.3.4, after appropriately fine-tuning

operations, the testing error was apparently lowered by 70%. Comparing with the results in Group III, the testing error from the ANFIS-SC fine-tuned model in Group IV has reached almost the same levels in Group III listed in Table 6.29. Obviously, this is a significant contribution on modeling highly nonlinear unknown systems with highly sparse training data sets.

Notably, one may use more number of fuzzy rules (i.e., more membership functions in each input variable) to prevent the occurrence of inconsistent models. However, this may increase the difficulty in fine-tuning the membership functions. So far, it is somewhat of an art to initially guess and re-shape the membership functions. The fine-tuning procedures of method I can be rather time-consuming. Nevertheless, the fine-tuning steps of method II can effectively reduce the adjusting time. Undoubtedly, how to substantially shorten the fine-tuning time is another significant task for research.

Theoretically, if we can clearly understand the relationships between the outputs and each individual input. It will be very worth choosing or designing the membership functions in the fuzzy rules. In other words, the human-base knowledge can be involved into the fuzzy rules for ANFIS training.

7.1.5 Contributions to GA Modeling

From the aforementioned results, it is evident that GA can be successfully applied in process modeling, even for highly nonlinear systems, such as the CMP process. The process models constructed using the sparse data from the experiment I may not be sufficiently robust to represent the real CMP process. However, from the Equation (6.3),

a different pre-determined formula for WIWNU case can tremendously reduce the modeling error. In other words, GA modeling is very flexible and reliable. Choosing a suitable pre-determined formula is a key behind excellent results for method I of GA modeling. It may usually involve several trials to find a better formula.

Comparing the simulation results of MRR and WIWNU models from the above four groups in the Case III experiments, both training and testing errors almost are very close. We can conclude that the size and distribution of the training data sets are not very important issues to GA modeling using Method I. Because the pre-determined formula provided a reliable frame work, GA operation tries to find the best solution to fit the given framework. It is quite different for the other modeling tools.

According to the procedures for constructing GA models for MRR and WIWNU, it is obviously not necessary to determine which input features are critical to the output performances, i.e., the need for tedious statistical analyses may not be necessary in GA modeling. This can be an important advantage in using GA to model highly nonlinear systems. Only if we can design evolutionary chromosomes (or solutions) to form the initial populations, they can effectively facilitate in finding the optimal chromosomes through suitable evolutional generations.

Moreover, except for keeping to trying our proposed Method I in Section 5.5.2.1, our future work will attempt to try our Method II as depicted in Section 5.5.2.2, which should potentially build more precise process models to be able to fit the any highly nonlinear systems.

From the GA models in Case III experiments, the importance of the solid content is much smaller than those of down pressure and platen speed both to MRR and WIWNU. This conclusion somewhat contradicts the observations from Experiment I. While down pressure and platen speed greatly affecting the MRR as well as back pressure, platen speed and polishing time are significant to WIWNU, all of results from these three experiments are similar.

7.1.6 Optimization of CMP Process

Lastly, on the basis of accurate models constructed using ANFIS-GP in this investigation, MOEA can be successfully applied for optimization of CMP process. From Table 6.30, the input settings for optimal CMP process can be found through the guidance of the simulation results generated from NSGA-II in MOEA. Obviously, it can definitely replace the traditional trial-and-error procedures. Even the process models are *black boxes* as NN and ANFIS models used in the Case III experiments, the optimal input settings can still be precisely found. This should be another useful contribution of this investigation.

7.2 Future Work

Currently, the experiment data sets of CMP used in this research have no real-time detecting information during the polishing process. The Ultra Precision Machining UPM experiments setup shown in Figures 7.1 and 7.2 involve three sensors (i.e., cutting forces, vibrations, and acoustic emission) for collecting the real-time processing situations. More

accurate process models (for surface roughness, or chatter detection) should be constructed by combining the real-time sensed signals with the process input settings, such as the depth of cut, feed rate, and rotational speed. Likewise, chatter detection can be achieved using the similar ideas.

Secondly, some challenges still remain for constructing and improving process models using ANFIS, NN and GA, as stated in the Sections 7.1.3 to 7.1.5. Several initial ideas were shown in the following Section 7.2.2. Our future work will attempt to determine appropriate solutions to these issues.

7.2.1 Multi-Sensor-Fusion and Monitoring Techniques for Modeling Surface Roughness for UPM Cases

Figures 7.1 and 7.2 show the front and side views of sensor setups on the UPM machine. The important features extracted from the sensing signals by the principal component analysis (PCA) will be used as extra input variables in order to increase the accuracy for process modeling. Figure 7.3 shows our approach for precisely modeling the surface roughness by involving the above-mentioned features.

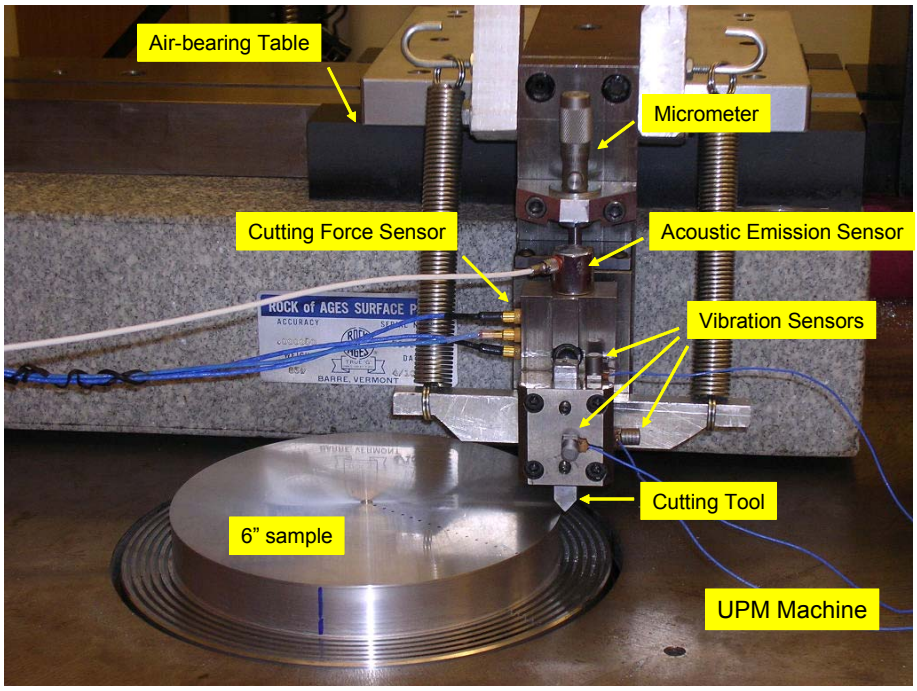


Figure 7.1 Front View of UPM Machine with Sensors System

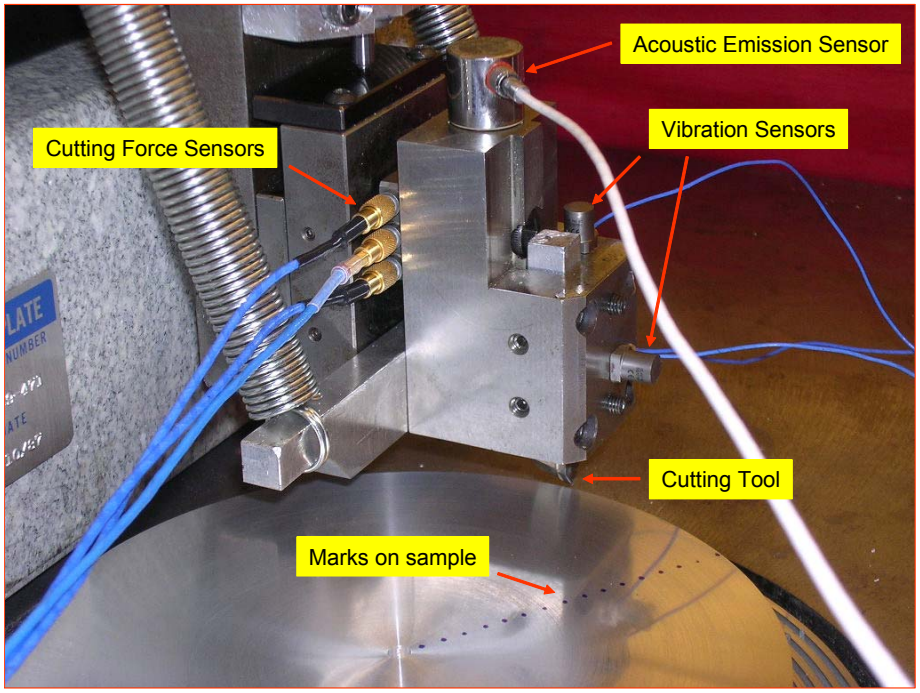


Figure 7.2 Side View of UPM Machine with Sensors System

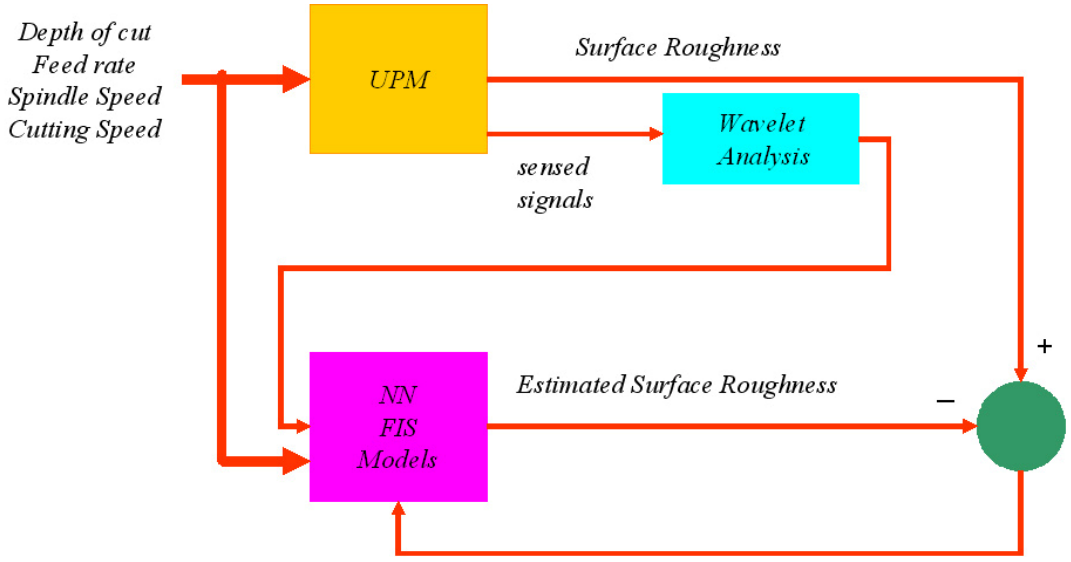


Figure 7.3 Model Design for Surface Roughness Prediction

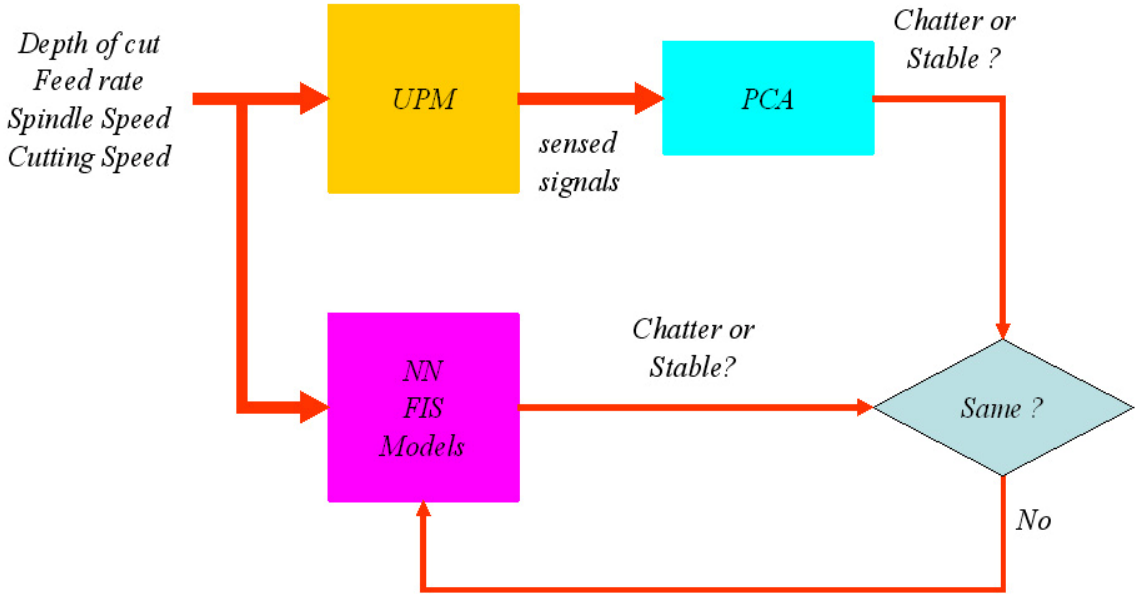


Figure 7.4 Model Design for Chatter Prediction

Figure 7.4 illustrates to the model construction a model for predicting chatter during the machining process. After PCA analysis, if there appear periodical fluctuations in some extracted features, chatter can certainly be recognized.

7.2.2 Optimization of ANFIS and NN Models

Figure 7.5 shows a schematic of the optimizing process. All control parameters, such as weights, number of layers and neurons, parameters in membership functions and consequent part, in NN or ANFIS architectures will be formed to represent chromosomes, which are gathered to be in the initial population. The appropriate evolutionary operations are chosen to find the best chromosomes which can balance the training and testing errors. Our objective is to find out the trade-off point of both training and testing error. If trade-off point cannot perform better, new architecture will replace old one and the evolutionary operations for searching trade-off point will re-start.

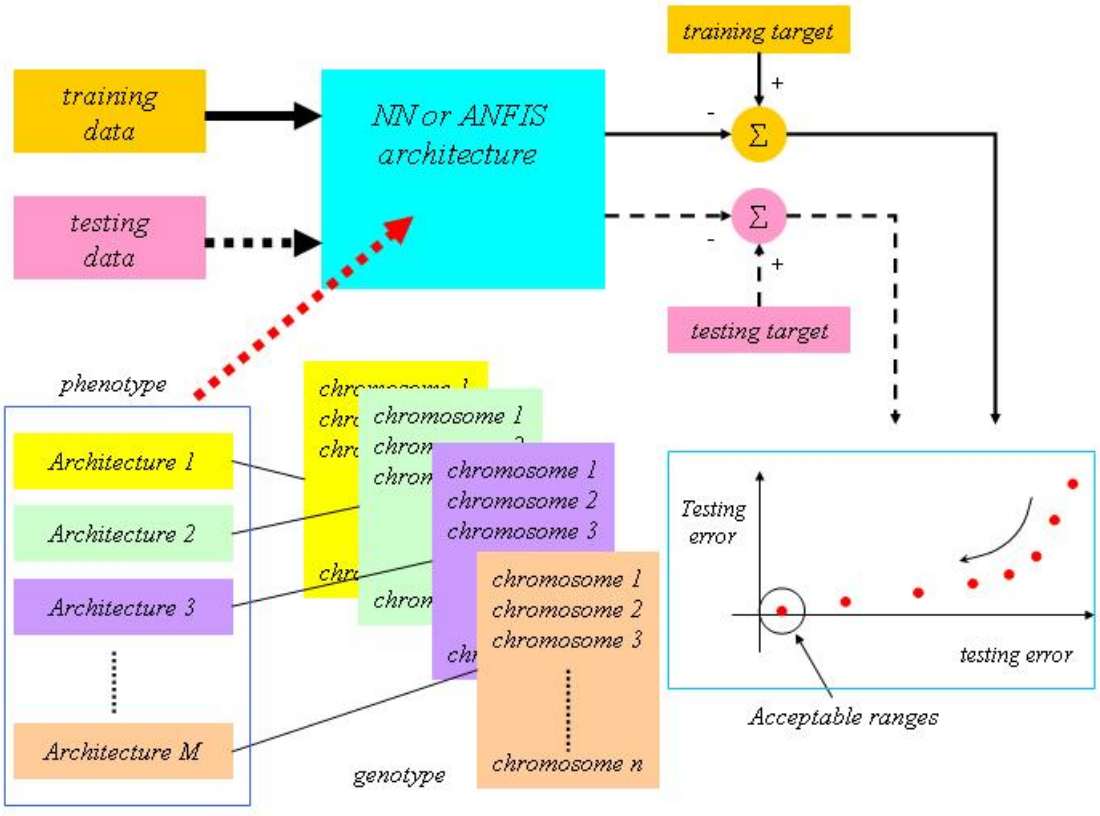


Figure 7.5 Optimization of ANFIS and NN models

The detailed steps are involved in the optimization given in the following:

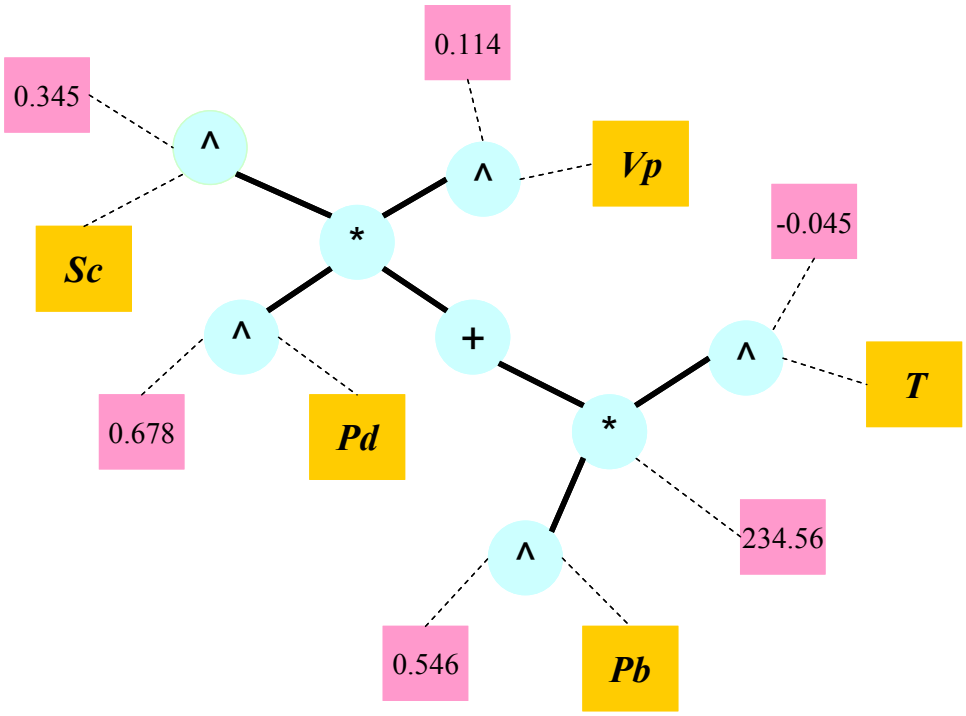
- Step1: Choose the architectures for NN or ANFIS.
- Step2: After analyzing the selected architecture, extract important features from the control parameters, such as weights, number of layers, membership functions, , to form the chromosomes.
- Step3: According to the format of constructed chromosome, randomly generate more chromosomes to form an initial population.
- Step4: Set the appropriate objective functions or rules, such as *smaller training and testing errors are better*. Further, assign better chromosomes for higher fitness values.
- Step5: Run the selected evolutionary operations to search for the best chromosome.
- Step6: Check the searched best chromosome whether performance is good or not.
- Step7: Repeat Steps 1 to Step 6 until optimal architecture is found.

7.2.3 Shortening Fine-tuning Time of ANFIS-SC Models for Sparse Data

For a given ANFIS-SC model, we can form part of control parameters into a chromosome. Use the same ideas as described in the previous subsection, but just focus on the local search, for example, only on the membership functions of insignificant variables. If any new chromosome (i.e., new architecture settings) which can reduce the testing error is found, then replace old architecture with new settings. Keep the same procedures until reaching the acceptable value of the testing error.

7.2.4 Method II for GA Modeling

Figure 7.6 shows the representation of a chromosome using the method II for GA modeling. Recombination operation can be performed through changes in the set positions, set values (or types), etc. In this method, no mutation operation will be applied for preventing from generating mathematically unreasonable chromosomes. Expectedly, this method can generate more accurate models, but it might take longer operation time and the generated models may not possess any physical meaning.



Function sets = { +, *, ^, -, …} Terminal sets = { ***Sc***, ***Pd***, ***Pb***, ***Vp***, ***T***, numbers,…}

$$MRR = Sc^{0.345} \cdot Pd^{0.678} \cdot Vp^{0.114} + 234.56 \cdot Pb^{0.546} \cdot T^{-0.045}$$

Figure 7.6 Method II for GA Modeling

There are three rules that should be followed for as modeling:

- Rule1: All circle symbol sets should connect together.
- Rule2: Any two square terminal sets can not connect together.
- Rule3: Any one circle function set can connect at most two square terminal sets.

All new generated chromosomes (i.e., the new models) should be checked if they are mathematically reasonable equations. Similarly, the appropriate evolutionary operations can be used to find the best chromosome (i.e., best model). The objective functions or rules can also be setup as *smaller training and testing errors are better*.

REFERENCES

[1] Deb, K., *Multi-Objective Optimization using Evolutionary Algorithms*. 2001: Wiley.

[2] The MathWorks, I., *Fuzzy Logic Toolbox User's Guide*. Version 2 ed. 2004.

[3] Coello, C.A.C. and A.D. Christiansen, *Multiobjective Optimization of Trusses using Genetic Algorithms.*

[4] Tsoukalas, L.H. and R.E. Uhrig, *Fuzzy and Neural Approaches in Engineering*. 1997: Wiley.

[5] Goldberg, D.E., *Genetic Algorithms for Search, Optimization, and Machine Learning*. 1989: MA : Addsion-Wesley.

[6] (SIA), S.I.A., *The International Technology Roadmap for Semiconductor (2005 Edition) _ Interconnect*. 2005.

[7] Wang, L.-X., *A Course in Fuzzy Systems and Control*. 1997, Upper Saddle River, NJ, U.S.A.: Prentice Hall, Inc.

[8] Jang, J.-S.R., C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. 1997: Prentice-Hall, Inc.

[9] Oliver, M.R., *Chemical-Mechanical Planarization of Semiconductor Materials*. 2004, Berlin: Springer-Verlag.

[10] Hagan, M., H. Demuth, and M. Beale, *Neural Network Design*. 1996, Boston, MA, U.S.A.: PWS Publishing.

[11] Nishiguchi, T., *CMP Consumable Technologies in PERL HITACHI*. 2001, Production Engineering Research Lab., Hitachi, Ltd.: Japan.

[12] Tan, K.C., E.F. Khor, and T.H. Lee, *Multiobjective Evolutionary Algorithms and Applications*. 2005, London: Springer-Verlag.

[13] Isobe, A. *A Novel Air Floating Head for Next Generation CMP*. in *10th Europe CMP Users Meeting*. 2003: Accretech/Tokyo Semitsu Co., Ltd.

[14] Lai, J.-Y., *Mechanics, Mechanisms, and Modeling of Chemical Mechanical Polishing Process*, in *Department of Mechanical Engineering*. 2001, MIT: Cambridge, MA, U.S.A.

[15] Luo, J., *Review of Chemical-Mechanical Planarization Modeling for Integrated Circuit Fabrication: From Particle Scale to Die and Wafer Scales*. 2002, Laboratory for Manufacturing Automation, University of California, Berkeley.

[16] Cook, L.M., *Chemical Process in Glass Polishing.* Jorunal of Non-Crystallin Solids, 1990. **120**: p. 152-171.

[17] Jain, A.K., J. Mao, and K. Mohiuddin, *Artificial Neural Networks: A Tutorial.* IEEE Computer, 1996. **29**(3): p. 31-44.

[18] Yu, T., C. Yu, and M. Orlowski, *A statistical polishing pad model for chemical-mechanical polishing*, in *IEEE International Electron Devices Meetings*. 1993. p.

865-868.

[19] Deb, K. and M. Goyal, *A combined genetic adaptive search (GeneAS) for engineering design.* Computer Science and Informatics, 1996. **26**(4): p. 30-45.

[20] Komanduri, R., D.A. Lucca, and Y. Tani, *Techological Advances in Fine Abrasive Process.* CIRP Annals, 1997. **46/2**: p. 545-596.

[21] Goldberg, D.E. and K. Deb, *A comparison of selection schemes used in genetic algorithms.* Foundations of Genetic Algorithms 1 (FOGA-1), 1991: p. 69-93.

[22] Moon, Y., *Mechanical aspects of the material removal mechanism in chemical mechanical plishing (CMP)*, in *Department of Mechanical Engineering*. 1999, University of Califonia at Berkeley: Berkeley, CA, U.S.A.

[23] Endo, N., S. Kondo, S. Tokiyou, B.U. Yoon, N. Ohashi, S. Sone, H.J. Shin, I. Matsumoto, and N. Kobayashi, *Challenges of CMP Technology Beyond 65 nm Node*. 2003: Tsukuba, Japan.

[24] Jang, J.-S.R., *ANFIS: Adaptive-Network-Based Fuzzy Inference System.* IEEE Transactions on System, Man and Cybernetics, 1993. **123**(3): p. 665-685.

[25] Zantye, P.B., A. Kumara, and A.K. Sikder, *Chemical Mechanical Planarization for Microelectronics Applications.* Material Science and Engineering, 2004. **R**(45): p. 89-220.

[26] Luo, J. and D.A. Dornfeld, *Material Removal Mechanism in Chemical Mechanical Polishing: Theory and Modeling.* IEEE Transactions on Semiconductor

Manufacturing, 2001. **14**: p. 112-133.

[27] Steigerwald, J.M., S.P. Murarka, and R.J. Gutmann, *Chemical Mechanical Planarization of Microelectronic Materials*. 1997, New York, U. S. A.: John Wiley & Sons.

[28] Osseo-Asare, K., *Surface chemical processes in chemical mechanical polishing: relationship between silica material removal rate and the point of zero charge of the abrasive material.* Journal of The Electrochemical Society, 2002. **149**: p. G651-G655.

[29] Fu, G., A. Chandra, S. Guha, and G. Subhash, *A plasticity-based model of material removal in chemical-mechanical polishing (CMP).* IEEE Transactions on Semiconductor Manufacturing, 2001. **14**: p. 406-417.

[30] Wang, J.F., A.R. Sethuraman, L. Cook, R.C. Kistler, and G.P. Schwartz, *Chemical-Mechanical Polishing of Dual Damascene Aluminum Interconnect Structure.* Semiconductor International, 1995(October): p. 117-120.

[31] Moore, G.E., *Cramming More Components onto Integrated Circuits.* Electronics, 1965: p. 144.

[32] Steigerwald, J.M., R. Zirpoli, S.P. Murarka, D. Price, and R.J. Gutmann, Journal of The Electrochemical Society, 1994. **141**: p. 2842.

[33] Harper, J.M.E., E.G. Colgan, C.-K. Hu, J.P. Buchwalter, and C.E. Uzoh, MRS Bulltin, 2004. **23**.

[34] Nitta, T., T. Ohmi, T. Hoshi, S. Sakai, K. Sakaibara, S. Imai, and T. Shibata, Journal of The Electrochemical Society, 1993. **140**: p. 1131.

[35] Nitta, T., T. Ohmi, M. Otsuki, T. Takewaki, and T. Shibata, Journal of The Electrochemical Society, 1992. **139**: p. 922.

[36] Steigerwald, J.M., S.P. Murarka, R.J. Gutmann, and D.J. Duquette, Journal of the Vacuum Society Technology B, 1995. **13**: p. 2215.

[37] Fayolle, M. and F. Romagna, Microelectron. Eng., 1997. **37/38**: p. 135.

[38] Stavreva, Z., D. Zeidler, M. Ploetner, and K. Drescher, Applied Surface Society, 1997. **108**: p. 39.

[39] Singer, P., *Tantalum Copper and Damascene: The Future of Interconnects.* Semiconductor International, 1998. **21**: p. 91-98.

[40] Kaufman, F.B., D.B. Thompson, R.E. Broadie, M.A. Jaso, W.L. Guthrie, D.J. Pearson, and M.B. Small, *Chemical-Mechanical Polishing for Fabricating Patterned W Metal Features as Chip Interconnects.* Journal of The Electrochemical Society, 1991. **138**(11): p. 3460-3465.

[41] Beyer, K.D., J.S. Makris, E. Mendel, K.A. Nummy, S. Ogura, J. Riseman, and N. Rovedo, *Method for removing protuberances at the surface of a semiconductor wafer using a chem-mech polishing technique*. 1987, International Bussiness Machines Corporation: U.S.A.

[42] Preston, F.W., *The Theory and Design of Plate Glass Polishing Machine.*

International Journal of Glass Science and Technology, 1927. **11**: p. 214-256.

[43] Archard, J.F., *Contact and rubbing of flat surfaces.* Journal of Applied Physics, 1953. **24**: p. 981-985.

[44] Maury, A., D. Ouma, D. Boning, and J. Chung. *A modification to Preston's equation and impact on pattern density effect modeling*. in *Advenced Metalization and Interconnect Systems for ULSI Applications*. 1997. San Diego, CA, U.S.A.

[45] Wrschka, P., J. Hernandez, Y. Hsu, T.S. Kuan, G.S. Oehrlein, H.J. Sun, D.A. Hansen, J. King, and M.A. Fury, *Polishing parameter dependencies and surface oxidation of chemical mechanical plishing of Al thin films.* Journal of the Electrochemical Society, 1999. **146**: p. 2689-2696.

[46] Johnson, K.L., *Contact Mechanics*. 1985, Cambrige, U. K.: Cambridge University Press.

[47] Liu, C.W., B.T. Dai, W.T. Tseng, and C.F. Yeh, *Modeling of the wear mechanism during chemical-mechanical polishing.* Journal of The Electrochemical Society, 1996. **143**: p. 716-721.

[48] Runnels, S.R., *Feature-scale fluid-based erosion modeling for chemical-mechanical polishing.* Journal of The Electrochemical Society, 1994. **141**: p. 1900-1904.

[49] Tseng, W.T. and Y.L. Wang, *Re-examination of pressure and speed dependence of removal rate during chemical-mechanical plishing processes.* Journal of The Electrochemical Society, 1997. **144**: p. L15-L17.

[50] Zhang, F. and A. Busnaina, *The role of particle adhension and surface deformation in chemical mechanical polishing processes.* Electrochemical and Solid-State Letters, 1998. **1**: p. 184-187.

[51] Zhang, F., A. Busnaina, and G. Ahmadi, *Particle adhesion and removal in chemical mechanical polishing and post-CMP cleaning.* Journal of The Electrochemical Society, 1999. **146**: p. 2665-2669.

[52] Ahmadi, G. and X. Xia, *A model for mechanical wear and abrasive particle adhesion during the chemical mechanical polishing process.* Journal of The Electrochemical Society, 2001. **148**: p. G99-G109.

[53] Zhang, J.Z., S.K. Huang, W.S. Toh, W.S. Tay, F. Chen, and B.-B. Zhou. *Optimization of pad conditional for stable oxide CMP process*. in *CMP-MIC Conference*. 1997. Santa Clara, CA, U. S. A.

[54] Li, W., D.W. Shin, M. Tomozawa, and S.P. Murarka, *The effect of the polishing pad treatments on the chemical-mechanical polishing of SiO_2 films.* Thin Solid Films, 1995. **270**: p. 601-606.

[55] Zhao, B. and F.G. Shi. *Chemical mechanical polishing in IC processes: new fundamental insights*. in *CMP-MIC*. 1999. Santa Clara, CA, U.S.A.

[56] Bastawros, A., A. Chandra, Y.J. Guo, and B. Yan, *Pad effects on material-removal rate in chemical-mechanical planarizaton.* Journal of The Electrochemical Society, 2002. **31**: p. 1022-1031.

[57] Seok, J., C.P. Sukam, A.T. Kim, J.A. Tichy, and T.S. Cale, *Multiscale material*

removal modeling of chemical mechanical polishing. Wear, 2003. **254**: p. 307-320.

[58] Paul, E., *A model of chemical mechanical polishing.* Journal of The Electrochemical Society, 148. **148**: p. G355-G358.

[59] Zhao, Y., L. Chang, and S.H. Kim, *A mathematical model for chemical mechanical polishing based on formation and removal of weakly bonded molecular species.* Wear, 2003. **254**: p. 332-339.

[60] Sundararajan, S., D.G. Thakurta, D.W. Schwendeman, S.P. Murarka, and W.N. Gill, *Two-dimemsiomal Wafer-scale Chemical Mechanical Planarization Models Based on Lubrication Theory and Mass Transport.* Journal of The Electrochemical Society, 1999. **146**(2): p. 761-766.

[61] Subramanian, R.S., L. Zhang, and S.V. Babu, *Transport Phenomena in Chemical Mechanical Polishing.* Journal of The Electrochemical Society, 1999. **146**(11): p. 4263-4272.

[62] Borst, C.L., D.G. Thakurta, W.N. Gill, and R. J.Gutmann, *Surface kinetic model for SiLK chemical mechanical polishing.* Journal of The Electrochemical Society, 2002. **149**: p. G118-G127.

[63] Su, Y.-T., *Investigation of removal rate properties of a floating plishing process.* Journal of The Electrochemical Society, 2000. **147**: p. 2290-2296.

[64] Luo, J. and D.A. Dornfeld, *Effects of abrasive size distribution in chemical-mechanical planarization: modeling and verification* IEEE Transactions on Semiconductor Manufacturing, 2003. **16**: p. 469-476.

[65] Luo, J. and D.A. Dornfeld, *Material removal regions in chemical mechanical planarization for sub-micron integrated circuit fabricaton: coulping effects of slurry chemicals, abrasive size distribution, and wafer-pad contact area.* IEEE Transactions on Semiconductor Manufacturing, 2003. **16**: p. 45-56.

[66] Mazaheri, A.R. and G. Ahmadi, *A model for effect of colloidal forces on chemical mechanical polishing.* Journal of The Electrochemical Society, 2003. **150**: p. G233-G239.

[67] Castillo-mejia, D., S. Gold, V. Burrows, and S. Beaudoin, *The effect of the interactions between water and polishing pads on chemical mechanical polishing removal rates.* Journal of The Electrochemical Society, 2003. **150**: p. G76-G82.

[68] Chen, P.H., H.C. Shih, B.-W. Huang, and J.-W. Hsu, *Catalytic-pad chemical kinetics model of CMP.* Electrochemical and Solid-State Letters, 2003. **6**: p. G140-G142.

[69] Zhao, Y. and L. Chang, *A micro-contact and wear model for chemical-mechanical polishing of silicon wafers.* Wear, 2002. **252**: p. 220-26.

[70] Runnels, S.R., M. Kim, J. Schleuter, C. Karlsrud, and M. Desai, *A Modeling Tool for Chemical Mechanical Polishing Design and Evaluation.* IEEE Transactions on Semiconductor Manufacturing, 1998. **11**(3): p. 501.

[71] Zhang, Y., P. Parikh, B. Stephenson, M. Bonsaver, J. Ling, and M. Li. *Effects of PETEOS Film Stress on Chemical Mechanical Planarization*. in *Proceeding of VMIC Conference*. 1996. Santa Clara, CA.

[72] Wang, D., J. Lee, K. Holland, T. Bibby, S. Beaudoin, and T. Cale, *Von Mises Stress in Chemical-mechanical Polishing Processes.* Journal of The Electrochemical Society, 1997. **144**: p. 1121-1127.

[73] Tichy, J., J.A. Levert, L. Shan, and S. Danyluk, *Contact Mechanics and Lubrication Hydrodynamics of Chemical Mechanical Polishing.* Journal of The Electrochemical Society, 1999. **146**: p. 1523-1528.

[74] Baker, A.R., *The Origin of Edge Effect in Chemical Mechanical Planarization.* Proceedings of Electro. Chem. Soc. Meeting, 1996. **96**(22): p. 228.

[75] Hocheng, H., H.-Y. Tsai, and L.-J. Chen. *A Kinematic Analysis of CMP Based on Velocity Model*. in *CMP-MIC Conference*. 1997. Santa Clara, CA, U.S.A.

[76] Yoshitomi, K., A. Une, and M. Mochida, *Ultra Precision Polishing with the Oscillation Speed Control: an Analysis of the Pressure Distribution and the Profile.* Key Engineering Materials, 2003. **238-239**: p. 219-222.

[77] Luo, Q., S. Ramarajan, and S.V. Babu, *Modification of the Preston Equation for the Chemical-Mechanical Polishing of Copper.* Thin Solid Films, 1998. **335**: p. 160-167.

[78] Bilmann, M., U. Mahajan, and R.K. Singh, *Effect of Particle Size during Tungsten Chemical Mechanical Polishing.* Electrochemical and Solid-State Letters, 1999. **16**: p. 401-403.

[79] Wang, G.-J., J.-C. Tsai, and H.-L. Chen, *Method for Determining Efficiently Parameters in Chemical-Mechanical Polishing (CMP)*. 2003: U.S.A.

[80] Wang, G.-J., J.-L. Chen, and J.-Y. Hwang, *New Optimization Strategy for Chemical Mechanical Polishing Process.* JSME International Journal, Series C, 2001. **44**(2): p. 534-543.

[81] Wang, G.-J. and M.-H. Chou. *A Neual-Taguchi Based Quasi Time-Optimal Control Strategy for Chemical-Mechanical Polishing Processes*. in *Proceedings of 2001 ASME International Mechanical Engineering Congress and Exposition (IMECE2001/EPP-24716)*. 2001.

[82] Lin, Z.-C. and C.-Y. Liu, *Application of an Adaptive Neuro-Fuzzy Inference System for Optimal Analysis of Chemical-Mechanical Polishing Process Parameters.* The International Jornal of Advanced Manufacturing Technology, 2001. **18**(20-28): p. 20-28.

[83] Zadeh, L.A., *Fuzzy Sets.* Information and Control, 1965. **8**: p. 338-353.

[84] McCulloch, W.S. and W. Pitts, *A logical calculus of ideas immanent in nervous activity.* Bulletin of Mathematical Biophysics, 1943. **5**: p. 115-133.

[85] Rosenblatt, R., *Principles of Neurodynamics*. 1962, New York, U.S.A.: Spartan Books.

[86] Minsky, M. and S. Papert, *An Introduction to Computational Geometry*. MIT Press. 1969, Cambride, MA, U.S.A.

[87] Hopfield, J.J. *Neural netwrok and physical systems with emergent collective computational abilities*. in *Proc. Natl. Acad. Sci. USA 79*. 1982.

[88] Werbos, P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, in *Applied Mathematics*. 1974, Harvard University.

[89] Rumelhart, D.E. and J.L. McClelland, *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, in *MIT Press*. 1986: Cambridge, MA, U.S.A.

[90] Eksioglu, M., J.E. Fernandez, and J.M. Twomey, *Predicting peak pinch strength: Artificial neural networks vs. regression.* International Journal of Industrial Ergonomics, 1996. **18**: p. 431-441.

[91] Cybenko, G., *Approximation by superpositions of a sigmoidal dunction.* Mathematics of Control, Signals, and Systems, 1989(2): p. 303-314.

[92] Funahashi, K., *On the Approximate Realization of Continuous Mappings by Neural Networks.* Neural Networks, 1989(2): p. 183-192.

[93] Hornik, K., M. Stinchcombe, and H. White, *Mulltilayer feedforward networks are universal approximators.* Neural Networks, 1989(2): p. 359-366.

[94] Sugeno, M. and G.T. Kang, *Structure identification of fuzzy model.* Fuzzy Sets and Systems, 1988. **28**(1): p. 15-33.

[95] Takagi, T. and M. Sugeno, *Fuzzy identification of systems and its applicationa to modeling and control.* IEEE Transactions on System, Man, and Cybern, 1985. **15**(1): p. 116-132.

[96] Takagi, T. and M. Sugeno. *Derivation of Fuzzy Control Rules from Human*

Operator's Control Actions. in *Proceedings of IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*. 1983.

[97] Jang, J.-S.R., *Self-learning Fuzzy Controller Based on Temporal Back Propagation.* IEEE Transactions on Neural Networks, 1992. **3**(5): p. 714-723.

[98] Holand, J.H., *Adaption in Natural and Artificial Systems*. 1975, The University of Michigan Press: Ann Arbor, MI, U.S.A.

[99] Rechenberg, I., *Evolutionsstretegie - Optimierung technische Systeme nach Prinzipien der biologischen Evolution*. 1973, Stuttgart: Formmann-Holzboog.

[100] Schwefel, H.P., *Numerical optimization of computer models*. 1981, Chichester: Wiley & Sons.

[101] Fogel, L.J., A.J. Owens, and M.J. Walsh, *Artificial Intelligent through Simulated Evolution*. 1966, New York: John Wiley.

[102] Antonisse, J. *A new interpretation of scheme notation that overturns the binary encoding constraint*. in *Proceedings of the Third International Conference on Genetic Algorithms*. 1989.

[103] Spears, W.M., *The Role of Mutation and Recombination in Envoltionary Algorithms*. 1998, George Mason University: Fairfax, VA, U.S.A.

[104] Herrara, F., M. Lozano, and J.L. Verdegay, *Tackling rfeal-coded genetic algorithms: Operators and tools for behavioural analysis.* Artificial Intelligent Review, 1998. **12**(4): p. 265 - 319.

[105] Wright, A.H. *Genetic Algorithms for Real Parameter Optimization*. in *Foundations of Genetic Algorithms 1 (FOGA-1)*. 1991.

[106] Eshelman, L.J. and J.D. Schaffer, *Real-codes genetic algorithms and interval schemata.* Foundations of Genetic Algorithms 2 (FOGA-2), 1993: p. 187-202.

[107] Michalewicz, Z., *Genetic Algorithms + Data Structure = Evolution Programs*. 1992: Springer-Verlag.

[108] Janikow, C.Z. and Z. Michalewicz. *An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms*. in *Proc. ICGA 4*. 1991.

[109] Tate, D.M. and A.E. Smith. *Expected Allele Convergence and the Role of Mutation in Genetic Algorithms*. in *Proc. ICGA 5*. 1993.

[110] Davis, L. *Adapting Operator Probabilities in Genetic Algorithms*. in *Proc. ICGA 3*. 1989.

[111] Fogarty, T.C. *Varying the Probability of Mutation in the Genetic Algorithm*. in *Proc. ICGA 3*. 1989.

[112] Mühlenbein, H. and D. Schlierkamp-Voosen, *Predictive Models for the Breeder Genetic Algorithm.* Evolutionary Computation, 1993. **1**(1): p. 25-49.

[113] Lucasius, C.B. and G. Kateman. *Towards Solving Subset Selection Problems with the Aid of the Genetic Algorithm*. in *Parallel Problem Solving from Nature 2*. 1992. Amsterdam, North-Holland.

[114] Fonseca, C.M. and P.J. Fleming, *An overview of evolutionary algorithms in multiobjective optimization.* Evolutionary Computation, 1995. **3**: p. 1-16.

[115] Veldhuizen, D.A.V., *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. 2001, Air Gorce Institute of Technology: Wright-Patterson AFB, OH, U.S.A.

[116] Zitzler, E. and L. Thiele, *Multiobjective evolutionary algorithms a comparative case study and the strength Pareto approach.* IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTION, 1999. **3**: p. 257-271.

[117] Zitzler, E., M. Laumanns, and L. Thiele, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, in *TIK-Report 103*. 2001: Zurich, Switzerland.

[118] Deb, K., S. Agrawal, A. Pratap, and T. Meyarivan. *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*. in *Proc. Parallel Problem Solving from Nature - PPSN VI*. 2000.

[119] Deb, K. and D.E. Goldberg. *An investigation on niche and species formation in genetic function optimization*. in *Proceedings of Third International Conference on Genetic Algorithms*. 1989. San Mateo, CA, USA.

[120] Beasley, D., D.R. Bull, and R.R. Martin, *A sequential niche technique for multimodal function optimization.* Evolutionary Computation, 1993. **1**(2): p. 101-125.

[121] Miller, B.L. and M.J. Shaw. *Genetic algorithms with dynamic niche sharing for multimodal function optimization*. in *Proceedings of the 1996 Congress on*

Evolutionary Computation. 1996.

[122] Forrest, S., B. Javornik, R.E. Smith, and A.S. Perelson, *Using genetic algorithms to explore pattern recognition in the immune system.* Evolutionary Computation, 1993. **1**(3): p. 191-211.

[123] Davidor, Y. *Epistasis Variance: A Viewpoint on GA-hardness*. 1991. Morgan Kaufmann, San Mateo, CA.

[124] Jong, K.A.D., *Analysis of the behavior of a class of genetic adaptive systems*, in *Department of Computer and Communication Sciences*. 1975, University of Michigan: Ann Arbor, MI, USA.

[125] Mahfoud, S.W., *Niching methods for genetic algorithms*. 1995, University of Illinois: Urbana-Champaign.

[126] Harik, G. *Finding multimodal solutions using restricted tournament selection*. in *Proceedings of the 6th International Conference on Genetic Algorithms*. 1995. Morgan Kaufmann, San Mateo, CA.

[127] Petrowski, A. *A clearing procedure as a niching method for genetic algorithms*. in *Proceedings of the 1996 Congress on Evolutionary Computation*. 1996. Nagoya, Japan.

[128] The MathWorks, I., *Neural Network Toolbox User's Guide*. 2003.

[129] Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan, *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II.* IEEE TRANSACTIONS ON

EVOLUTIONARY COMPUTION, 2002. **6**(2): p. 182-197.

[130] Abraham, A., L. Jain, and R. Goldberg, *Evolutionary Multiobjective Optimization*. 2005: Springer.

[131] Yen, G.G., *ECEN5773 Fall 2005 Lecture Notes*. 2005, Stillwater, Oklahoma, U.S.A.: Oklahoma State University.

APPENDIX I

Effect of Statistical Significance of CMP Process Parameters on ANFIS-GP Modeling

In this analysis, the original data used for constructing the ANFIS-GP 2-2-2-2-2-2-2 model is tabulated in Appendix III (Table A-5 from # 1 to #50). Initially, each process input has two membership functions. Next, the number of membership functions in each process input was changed to one (i.e. setting this process input as a constant, or taking this process input away from simulation model) to calculate the value of each training error one by one. Consequently, seven different values of training errors are created. Large training errors have resulted for the process inputs that have statistically significant influence on a performance variable (i.e., MRR or WIWNU). Comparisons of significance of process inputs to MRR and WIWNU are illustrated in Table A-1, and Figures A-1 and A-2. From these plots, it is evident that the carrier speed and the oscillation speed are insignificant process inputs to both MRR and WIWNU. As previously mentioned in the Section 5.6, by adjusting the premise parameters of membership functions of process inputs, such as carrier speed, one can effectively reduce the testing errors.

Table A-1 Average Training Error for MRR and WIWNU

Process Inputs	MRR Average Training Error (Å/min)	WIWNU Average Training Error $x10^{-3}$
Down Pressure (P)	196.6773	650.48
Solid Content (C)	99.9583	338.04
Platen Speed (Vp)	95.5078	332.87
Back Pressure (γ)	50.4104	311.93
Oscillation Speed (Vo)	30.1623	260.37
Polishing Time (T)	27.2822	1.9501
Carrier Speed (Vc)	10.8484	0.50728

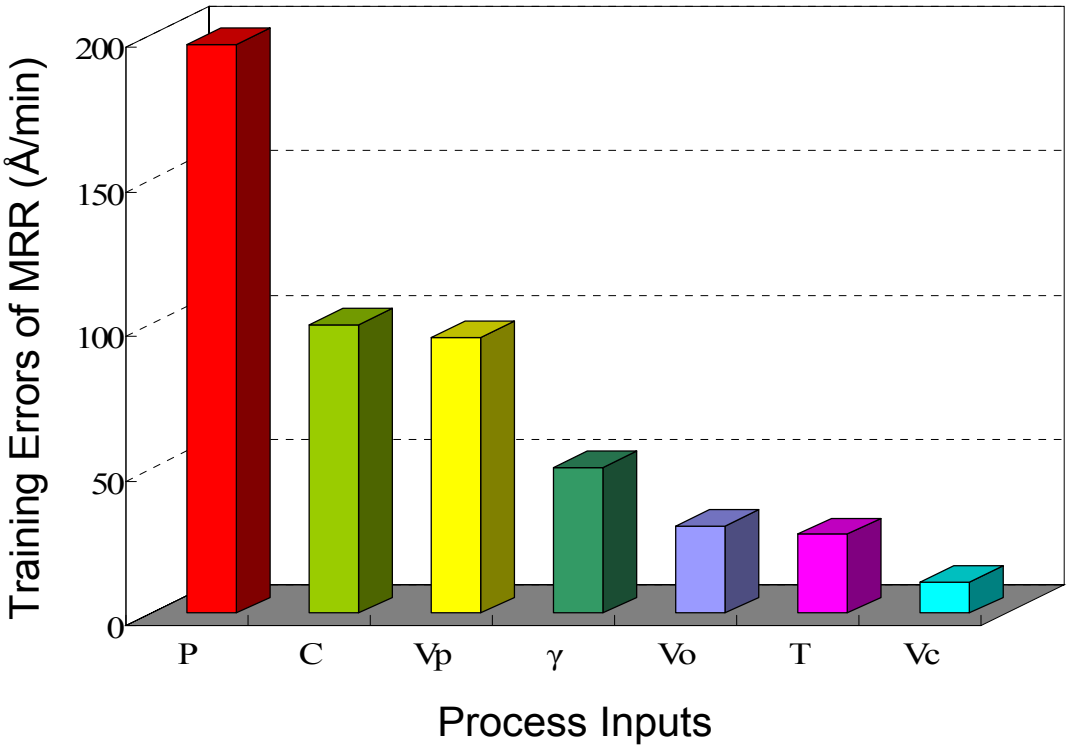


Figure A-1 Relative Effect of Process Parameters on **MRR**

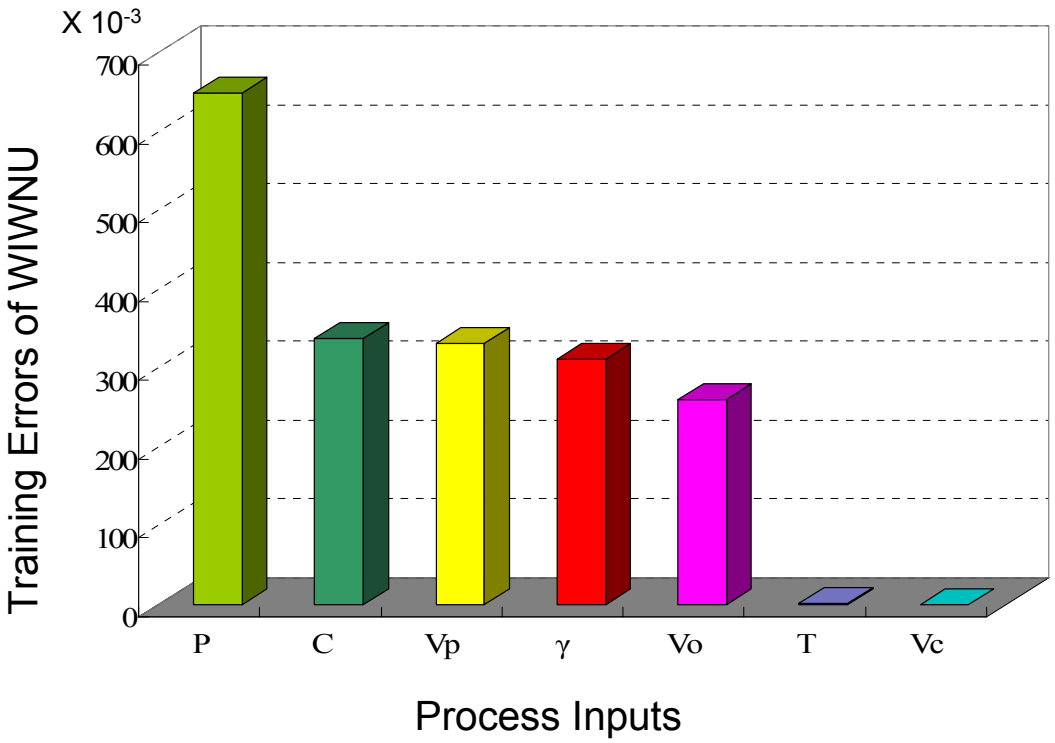


Figure A-1 Relative Effect of Process Parameters on **WIWNU**

APPENDIX II

An Example of Fragmentary Simulation Model with Insufficient Membership Functions and Fuzzy Rules

An example of ANFIS-SC Model with 7 rules is shown in Table A-2 and Figure A-3. The use of fewer rules can lead to larger training and validation errors as the model is unable to capture the complexity of the process. On closer observation, it was evident that, the results by ANFIS-SC 7 rules yield very good predictions for certain portions of the input space. However, the insufficient fuzzy rules could not fully cover the entire range of input settings, leading to undesirable results (e.g. negative value of MRR as shown in Figure A-4 and Table A-3).

Table A-2 Prediction of MRR by Neural Network and ANFIS Models

Results \ NN and ANFIS Model Types		**NN 7-4-1 purelin**	**NN 7-3-1 tansig**	**ANFIS-GP 2-2-1-3-1-1-2**	**ANFIS-SC** 7 Rules	**ANFIS-SC 45 Rules**
45 Training Error %	Means	21.004%	11.026%	13.039%	3.019%	~ 0 %
	Standard deviations	19.659%	8.420%	16.044%	4.827%	~ 0 %
9 Validation Error %	Means	17.498%	15.228%	15.157%	10.730%	3.705%
	Standard deviations	11.420%	11.519%	12.315%	10.055%	1.639%

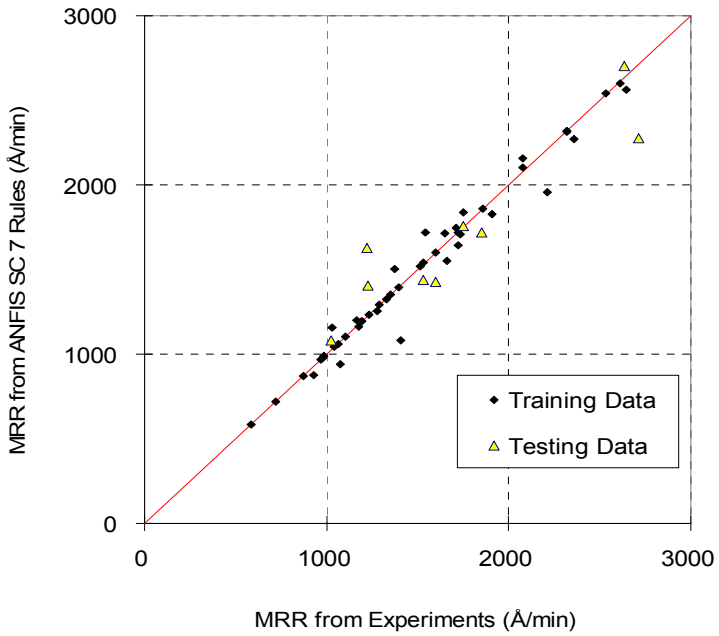


Figure A-3 Results of ANFIS-SC 7 Rules for **MRR**

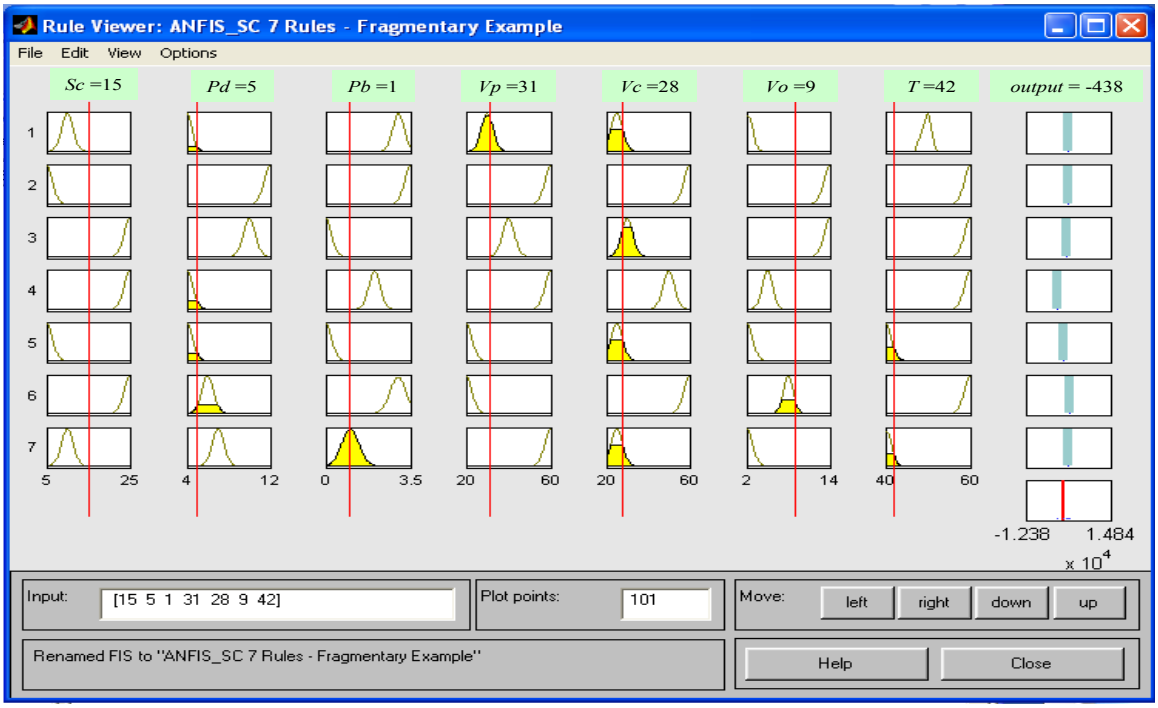


Figure A-4 Simulation of ANFIS-SC 7 Rules by Fuzzy Logic Toolbox in Matlab 6.1

Table A-3 Fragmentary Simulation Model Inputs and Output (ANFIS-SC 7 Rules)

Solid Content (wt%)	Down Pressure (psi)	Back Pressure (psi)	Platen Speed (rpm)	Carrier Speed (rpm)	Oscillation Speed (rpm)	Polishing Time (sec)	MRR (Å/min)
15.0	5.0	1.0	31.0	28.0	9.0	42	-438.0

APPENDIX III

Table A-4 27 Source Data Sets for CMP [79, 81]

	Input Data							Output Results	
	C	P	γ	V_P	T	V_C	V_O		
	Solid Content (wt%)	Down Pressure (psi)	Back Pressure (psi)	Platen Speed (rpm)	Polishing Time (sec)	Carrier Speed (rpm)	Oscillation Speed (rpm)	MRR (Å/min)	WIWNU (Variation of Thickness Std σ, Å)
1	5	4	0	20	40	25	2	548.4	137.54
2	5	5	1	30	45	25	2	870	109.591
3	5	6	2	40	50	25	2	980.4	99.567
4	5	7	3	50	55	25	2	1218.6	128.673
5	5	8	3.5	60	60	25	2	1233	98.534
6	10	4	3	30	50	25	2	965.4	51.999
7	10	5	3.5	40	55	25	2	1407	152.866
8	10	6	0	50	60	25	2	1735.2	298.643
9	10	7	1	60	40	25	2	2078.4	88.683
10	10	8	2	20	45	25	2	1350.6	112.108
11	15	4	1	40	60	25	2	1166.4	92.238
12	15	5	2	50	40	25	2	1660.8	114.126
13	15	6	3	60	45	25	2	2213.4	114.363
14	15	7	3.5	20	50	25	2	1326.6	55.401
15	15	8	0	30	55	25	2	1910.4	110.232
16	20	4	3.5	50	45	25	2	1226.4	100.058
17	20	5	0	60	50	25	2	1753.2	169.638
18	20	6	1	20	55	25	2	1102.2	167.419
19	20	7	2	30	60	25	2	1723.2	114.87
20	20	8	3	40	40	25	2	2356.8	69.075
21	25	4	2	60	55	25	2	1395	113.494
22	25	5	3	20	60	25	2	1026	46.615
23	25	6	3.5	30	40	25	2	1544.4	52.903
24	25	7	0	40	45	25	2	2075.4	139.578
25	25	8	1	50	50	25	2	2646.6	206.076
26	15	7	3	20	60	25	2	1599	112.2
27	20	8	3.2	41	42	25	2	2711.4	118

Table A-5 54 Source Data Set for CMP [79-81]

	Input Data							Output Data	
	C	P	γ	V_P	T	V_C	V_O		
	Solid Content (wt%)	Down Pressure (psi)	Back Pressure (psi)	Platen Speed (rpm)	Carrier Speed (rpm)	Oscillation Speed (rpm)	Polishing Time (sec)	MRR (Å/min)	WIWNU %
1	5	4	0	20	25	2	40	548.4	2.02
2	5	5	1	30	25	2	45	870	1.68
3	5	6	2	40	25	2	50	980.4	1.76
4	5	7	3	50	25	2	55	1218.6	1.61
5	5	8	3.5	60	25	2	60	1233	2
6	10	4	3	30	25	2	50	965.4	1.37
7	10	5	3.5	40	25	2	55	1407	0.81
8	10	6	0	50	25	2	60	1735.2	6.39
9	10	7	1	60	25	2	40	2078.4	2.56
10	10	8	2	20	25	2	45	1350.6	2.06
11	15	4	1	40	25	2	60	1166.4	6.23
12	15	5	2	50	25	2	40	1660.8	4.9
13	15	6	3	60	25	2	45	2213.4	6.49
14	15	7	3.5	20	25	2	50	1326.6	5.54
15	15	8	0	30	25	2	55	1910.4	7.27
16	20	4	3.5	50	25	2	45	1226.4	3.12
17	20	5	0	60	25	2	50	1753.2	8.78
18	20	6	1	20	25	2	55	1102.2	6.49
19	20	7	2	30	25	2	60	1723.2	3.59
20	20	8	3	40	25	2	40	2356.8	4.28
21	25	4	2	60	25	2	55	1395	6.21
22	25	5	3	20	25	2	60	1026	4.75
23	25	6	3.5	30	25	2	40	1544.4	3.81
24	25	7	0	40	25	2	45	2075.4	6.78
25	25	8	1	50	25	2	50	2646.6	5.21
26	5	4	0	20	20	2	60	1062	3.2
27	5	6	1	30	30	5	60	721	2
28	5	8	2	40	40	8	60	987	1.8

Continued from previous page

29	5	10	3	50	50	11	60	1194	2.6
30	5	12	3.5	60	60	14	60	1598	3.1
31	10	4	3	30	40	14	60	1711	5.26
32	10	6	3.5	40	50	2	60	1178	2.44
33	10	8	0	50	60	5	60	1513	4.64
34	10	10	1	60	20	8	60	1751	5.01
35	10	12	2	20	30	11	60	1371	2.52
36	15	4	1	40	60	11	60	2321	3.3
37	15	6	2	50	20	14	60	1077	2.28
38	15	8	3	60	30	2	60	1533	4.53
39	15	10	3.5	20	40	5	60	1277	2.48
40	15	12	0	30	50	8	60	1854	4.59
41	20	4	3.5	50	30	8	60	1860	4.25
42	20	6	0	60	40	11	60	1028	3.86
43	20	8	1	20	50	14	60	1040	2.19
44	20	10	2	30	60	2	60	1649	3.59
45	20	12	3	40	20	5	60	1721	3.87
46	25	4	2	60	50	5	60	2612	5.51
47	25	6	3	20	60	8	60	926	1.73
48	25	8	3.5	30	20	11	60	1288	2.26
49	25	10	0	40	30	14	60	2318	4.45
50	25	12	1	50	40	2	60	2535	5.21
51	15	7	3	20	25	2	60	1599	3.1963
52	20	8	3.2	41	25	2	42	2711.4	3.7506
53	15	8	3	20	25	2	60	1533	2.663
54	25	12	1	60	60	2	60	2633	0.96

APPENDIX IV

Table A-6 Group IV Experimental Training Data

	Solid content *Sc* (wt%)	Down Pressure *Pd* (psi)	Back Pressure *Pb* (psi)	Platen Speed *Vp* (rpm)	Polishing Time *T* (sec)	MRR (Å/min)	WIWNU (%)
1	5	4	0	20	40	786.4	1.55
2	5	6	1	30	50	1182	2.02
3	5	8	2	40	60	1601.8	3.33
4	5	10	3	50	50	2234.1	2.2
5	5	12	4	60	60	2388.8	3.77
6	10	4	1	40	40	1274	1.39
7	10	6	2	50	60	1473.3	3.92
8	10	8	3	60	40	2252.4	1.25
9	10	10	4	20	50	1335.2	2.11
10	10	12	0	30	40	2128.6	1.44
11	15	4	2	60	50	1344	3.06
12	15	6	3	20	60	796	3.1
13	15	8	4	30	50	1438.8	1.88
14	15	10	0	40	60	1870.2	4.36
15	15	12	1	50	40	2613.9	1.46
16	20	4	3	30	60	816.2	2.71
17	20	6	4	40	40	1546.5	1.16
18	20	8	0	50	50	2027.4	2.22
19	20	10	1	60	40	2549.4	1.24
20	20	12	2	20	50	1557.8	1.98
21	25	4	4	50	60	1141.5	2.96
22	25	6	0	60	50	1866.7	3.07
23	25	8	1	20	60	1034.5	3.86
24	25	10	2	30	40	1789.9	1.35
25	25	12	3	40	60	2070	3.3

VITA

Wen-Chen Lih

Candidate for the Degree of

Doctor of Philosophy

Thesis: MODELING AND OPTIMIZATION OF CHEMICAL MECHANICAL PLANARIZATION (CMP) USING NEURAL NETWORK, ANFIS AND EVOLUTIONARY ALGORITHMS

Major Field: Mechanical and Aerospace Engineering

Biographical:

Personal Data: Born in Taiwan, Republic of China, On December 31, 1967, the son of Der-Sheng Lih and Chuen-Shiou Shiue.

Education: Received Bachelor of Science degree in Mechanical Engineering from Chung Cheng Institute of Technology, National Defense University, Taiwan (R.O.C.), in July 1993. Received Master of Science degree in Mechanical Engineering from the University of Michigan, Ann Arbor, Michigan, in April 1998. Completed the requirements for the Doctor of Philosophy with a Major in Mechanical Engineering at Oklahoma State University, Stillwater, Oklahoma, in May 2006.

Experience: Teaching Assistant, Department of Mechanical Engineering, Chung Cheng Institute of Technology, National Defense University, Taiwan (R.O.C.) (07/1993-08/1996)
Lecturer, Department of Mechanical Engineering, Chung Cheng Institute of Technology, National Defense University, Taiwan (R.O.C.) (01/1999-12/2001)
Abroad Trainee Military Officer, Ministry of National Defense, Taiwan (R.O.C.) (12/2001-12/2005)

Professional Memberships: Chinese Society of Mechanical Engineers, American Society of Mechanical Engineers (ASME), The Institute of Electrical and Electronic Engineers (IEEE)

Name: Wen-Chen Lih Date of Degree: May, 2006

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: MODELING AND OPTIMIZATION OF CHEMICAL MECHANICAL PLANARIZATION (CMP) USING NEURAL NETWORKS, ANFIS AND EVOLUTIONARY ALGORITHMS

Pages in Study: 332 Candidate for the Degree of Doctor of Philosophy

Major Field: Mechanical Engineering

Scope and Method of Study: The purpose of this study is to provide new approaches to improve the current modeling and optimization of the chemical mechanical planarization or polishing (CMP) process. Neural Networks (NN), ANFIS (Adaptive-based-Network Fuzzy Inference System), and Evolutionary Algorithms (EA) were applied to construct process models of the material removal rate (MRR) and the within wafer non-uniformity (WIWNU). Furthermore, Multi-Objective Evolutionary Algorithms (MOEA) are firstly applied to search the optimal input settings for CMP process to trade-off the higher MRR and lower Non-Uniformity by using the previously constructed models.

Findings and Conclusions: The modeling approaches using NN, ANFIS and EA can fully capture the highly non-linear dynamics of complex CMP process, and successfully provide accurate models of MRR and WIWNU under sufficient training data. The uniform distribution of training data from the factorial designing experiments, providing entire information of input and output spaces, can effectively reduce the training errors for constructing more accurate process models. In addition, the fine-tuning techniques for re-modifying ANFIS models constructed by sparse data can effectively improve modeling accuracy. The size of training data can only slightly influence the generalization capabilities of EA models, if the pre-determined formulae chosen are correct enough. The results also show the simulation of MOEA optimization can certainly provide the accurate guidance to search the optimal input settings for CMP process to produce lower non-uniform wafer surfaces under higher MRR.

ADVISER'S APPROVAL: Ranga Komanduri