

國立交通大學

電控工程研究所

博士論文

內嵌粒子群優化學習演算法之類神經模糊系統  
及其應用

Neural Fuzzy System Embedded with Particle Swarm Optimizer  
and Its Applications

研究生：蘇閔財

指導教授：林進燈 教授

中華民國一〇一年二月

內嵌粒子群優化學習演算法之類神經模糊系統及其應用  
Neural Fuzzy System Embedded with Particle Swarm Optimizer  
and Its Applications

研究生：蘇閔財

Student : Miin-Tsair Su

指導教授：林進燈

Advisor : Chin-Teng Lin

國立交通大學  
電控工程研究所  
博士論文

A Dissertation

Submitted to Institute of Electrical and Control Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electrical and Control Engineering

February 2012

Hsinchu, Taiwan, Republic of China

中華民國一〇一年二月

# 內嵌粒子群優化學習演算法之類神經模糊系統及其應用

研究生：蘇閔財

指導教授：林進燈 博士

國立交通大學電控工程研究所博士班

## 摘 要

本篇論文中所提出的進化式神經模糊系統乃是採用內嵌以粒子群為基礎的學習演算法之函數鏈結類神經模糊網路(Functional-Link-Based Neuro-Fuzzy Network, FLNFN)。此一類神經模糊網路採用函數鏈結類神經網路來做為模糊法則的後件部。由於，後件部採用了非線性函數展開的方式，來形成任意複雜的決策邊界。因此，在 FLNFN 模型中，後件部的這個局部特性，可以使輸入變量的非線性組合結果，能夠更有效地近似目標輸出。本論文主要為三大部分。在第一部份，我們提出了一個高效率的免疫粒子群優化 (IPSO) 的學習方法來解決膚色檢測的問題。我們所提的免疫粒子群優化演算法主要是結合免疫演算法 (IA) 和粒子群優化 (PSO) 來進行參數學習。在第二部分中，另一種被稱為細菌覓食粒子群優化 (BFPSO) 的混合式參數學習演算法，將被介紹來解決分類的應用。BFPSO 演算法主要是透過 BFO 的趨化運動來操作執行區域性的搜索，而在整個搜索空間的全域搜索則是由 PSO 來完成。利用此一方式，便能在全域性的勘探和區域性的開採間取得最好的平衡。在第三部分中，與先前採用混合方法不同，我們引入了以距離為基礎的突變操作元，藉以用來增加粒子群的群體多樣性。此演算法包含架構學習及參數學習兩部分。架構學習是藉由熵的量測來決定所需的

模糊法則的數目。參數學習則是使用內嵌以距離為基礎的突變操作元之粒子群優化演算法 (DMPSO)，來調整歸屬函數的形狀與後件部的相對應權重。最後，我們將論文中所提出的以 PSO 為基礎之學習演算法應用到各種分類和控制問題。本論文的實驗結果證明了所提出方法的有效性。



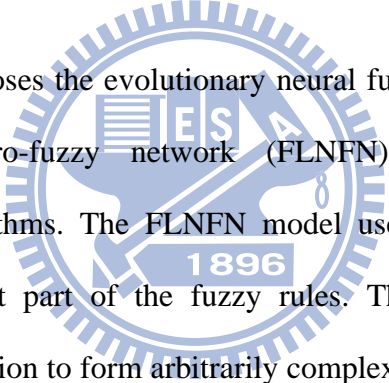
# Neural Fuzzy System Embedded with Particle Swarm Optimizer and Its Applications

Student : Miin-Tsair Su

Advisor : Dr. Chin-Teng Lin

Institute of Electrical and Control Engineering  
National Chiao Tung University

## ABSTRACT



This dissertation proposes the evolutionary neural fuzzy system, designed using functional-link-based neuro-fuzzy network (FLNFN) model embedded with PSO-based learning algorithms. The FLNFN model uses a functional link neural network to the consequent part of the fuzzy rules. The consequent part uses a nonlinear functional expansion to form arbitrarily complex decision boundaries. Thus, the local properties of the consequent part in the FLNFN model enable a nonlinear combination of input variables to be approximated more effectively. This dissertation consists of three major parts. In the first part, the efficient immune-based particle swarm optimization (IPSO) learning method is presented to solve the skin color detection problem. The proposed IPSO algorithm combines the immune algorithm (IA) and particle swarm optimization (PSO) to perform parameter learning. In the second part, another hybrid parameter learning algorithm, called bacterial foraging particle swarm optimization (BFPSO), is introduced for classification applications. The proposed BFPSO algorithm performs local search through the chemotactic movement operation of BFO whereas the global search over the entire search space is

accomplished by a PSO operator. In this way it balances between exploration and exploitation enjoying best of both the worlds. In the third part, instead of using hybrid techniques, the distance-based mutation operator is introduced to improve the population diversity. The learning algorithm consists of structure learning and parameter learning. The structure learning depends on the entropy measure to determine the number of fuzzy rules. The parameter learning, based on distance-based mutation particle swarm optimization (DMPSO), can adjust the shape of the membership function and the corresponding weights of the consequent part. Finally, the proposed PSO-based learning algorithms are applied in various classification and control problems. Results of this dissertation demonstrate the effectiveness of the proposed methods.



## 誌 謝

修習博士學位這一路走來，我需要感謝的人太多！特別是我的博士論文指導教授 林進燈 博士。感謝老師在學術研究上給予學生的啟發、鼓勵與陪伴。在老師豐富的學識、殷勤的教導及嚴謹的督促下，使我學習到許多的寶貴知識以及面對事情時應有的處理態度。在此，學生要由衷的向老師表示謝意和敬意。

感謝口試審查委員 陳文良 教授、楊谷洋 教授、張志永 教授、林正堅 教授以及 陳金聖 副教授，在百忙之中，願意撥冗參與口試，並給予論文寶貴的建議與指正，使得本篇論文更加完整。

博士的求學過程，真是一個漫長且艱辛的路程，有時看不到希望，甚至必須在絕望中持續奮鬥，不知何時光明會出現，尤其是投稿論文被拒絕，畢業遙遙無期的時候。雖然，未來前途也充滿了挑戰，希望透過如此一個真實的經驗，讓我體察到，即使未來處在困境中，仍要持續奮鬥，因為奮鬥才有希望。

在研究過程中，也要特別感謝已畢業的博士班同學 陳政宏 助理教授的一路陪伴。特別是在研究上的討論與建議，常常可以給我新的啟發與觀念的成長。

同時，要感謝交大資訊媒體實驗室的所有夥伴，尤其是肇廷、東霖、勝智、鎮宇、哲銓、耿維、…，在研究的過程中，藉由不斷的相互砥礪及針對研究內容與方法的深入探討，讓我在博士的研究路上走得更順遂，特別是您們的勤奮和努力，更是我學習的指標。

學生也要感謝碩士班的指導教授 陳文良 博士與已故的葉 莒 教授的悉心指導，使我在研究及做人處世上獲益匪淺。另外，老師為人處世的言教與身教，也深深地影響著學生。

感謝我就讀博士班期間所任職的東捷科技股份有限公司以及帆宣系統科技股份有限公司，提供了一個良好的工作環境，讓我能夠同時兼顧工作與學業，進而順利完成我的博士學業。在此，我也要感謝徐氏數學的徐清朗老師與徐陳雪美師母，在我就讀大學與研究所的寒、暑假期間，提供我一個優良的打工環境，讓

我能在學校上課期間專注於課業上，進而順利完成我的學士與碩士學業。

感謝父親與已故母親，從小到大對我們兄弟的栽培，讓我們能得到良好的教育。我也要感謝岳父與岳母對我在職進修博士學位一事，抱持著正面與肯定的態度，進而給我支持與鼓勵，讓我能夠專心於研究的工作並順利完成博士學位。

多年來，太太 怡君 在我就讀博士班期間，星期假日常陪伴我到圖書館唸書、查資料、寫文章，並協助分擔家中許多事務，讓我得以心無旁騖地專心致力於論文研究工作，更是讓我無以為報。

因篇幅有限，還有許多曾經關心我的家人親友、教導我的師長、幫助我的同仁、鼓勵我的朋友，無法一一致意，謹在此表達由衷的感謝，謝謝您們。

最後，謹將此論文獻給我已亡故的母親 蘇黃惠美 女士，以慰她在天之靈。



閱財 於交大資訊媒體實驗室

中華民國一〇一年七月十日



# Table of Contents

Chinese Abstract .....	i
English Abstract .....	iii
Acknowledgement .....	v
Table of Contents .....	vii
List of Tables.....	ix
List of Figures .....	x
Chapter 1 Introduction .....	1
1.1 Motivation.....	1
1.2 Literature Survey .....	6
1.3 Organization of Dissertation .....	11
Chapter 2 Structure of the Functional-Link-Based Neuro-Fuzzy Network .....	14
Chapter 3 Immune Algorithm Embedded with Particle Swarm Optimizer for Neuro-Fuzzy Classifier and Its Applications .....	19
3.1 Basic Concepts of the Artificial Immune System.....	20
3.2 Clonal Selection Theory .....	21
3.3 The Efficient Immune-Based PSO Learning Algorithm.....	22
3.3.1 Code fuzzy rule into antibody.....	23
3.3.2 Determine the initial parameters by self-clustering algorithm .....	25
3.3.3 Produce initial population .....	25
3.3.4 Calculate affinity values .....	26
3.3.5 Production of sub-antibodies .....	26
3.3.6 Mutation of sub-antibodies based on PSO.....	27
3.3.7 Promotion and suppression of antibodies .....	28
3.3.8 Elitism selection.....	30
3.4 Skin Color Detection.....	30
3.5 Concluding Remarks.....	34
Chapter 4 An Evolutionary Neural Fuzzy Classifier Using Bacterial Foraging Oriented by Particle Swarm Optimization Strategy.....	37
4.1 Basic Concepts of Bacterial Foraging Optimization .....	38
4.1.1 Chemotaxis .....	38
4.1.2 Swarming .....	39
4.1.3 Reproduction.....	40
4.1.4 Elimination-and-Dispersal .....	40
4.2 Learning Algorithms for the NFS Model.....	41

4.3 Illustrative Examples .....	44
Example 1: Iris Data Classification .....	45
Example 2: Wisconsin Breast Cancer Diagnostic Data Classification .....	52
Example 3: Skin Color Detection .....	55
4.4 Concluding Remarks.....	57
<b>Chapter 5 Nonlinear System Control Using Functional-Link-Based Neuro-Fuzzy Network Model Embedded with Modified Particle Swarm Optimizer .....</b>	<b>59</b>
5.1 Learning Scheme for the FLNFN Model.....	60
5.2 Structure Learning Phase .....	62
5.3 Parameter Learning Phase.....	64
5.4 Illustrative Examples .....	67
Example 1: Multi-Input Multi-Output Plant Control.....	67
Example 2: Control of Backing Up the Truck .....	70
Example 3: Control of Water Bath Temperature System.....	75
5.5 Concluding Remarks.....	81
<b>Chapter 6 Comparisons and Discussions.....</b>	<b>83</b>
6.1 Comparisons .....	83
6.1.1 Skin Color Detection Using DMPSO .....	83
6.1.2 Skin Color Detection Results Comparison with Different Approaches .....	88
6.2 Discussions .....	89
<b>Chapter 7 Conclusions and Future Works .....</b>	<b>91</b>
<b>Bibliography .....</b>	<b>95</b>
<b>Publication List .....</b>	<b>110</b>

# List of Tables

Table 3.1: The accuracy rate with different generations (%) .....	33
Table 3.2: Performance comparison with various existing models from the CIT database (Training data: 6000; Generations: 2000) .....	34
Table 4.1: Classification accuracy using various methods for the iris data. ....	51
Table 4.2: Average re-substitution accuracy comparison of various models for the iris data classification problem. ....	52
Table 4.3: Classification accuracy for the Wisconsin breast cancer diagnostic data. ..	54
Table 4.4: Average accuracy comparison of various models for Wisconsin breast cancer diagnostic data. ....	55
Table 4.5: Performance comparison with various existing models from the CIT database.....	56
Table 5.1: Performance comparison of the FLNFN-DMPSO, FLNFN-PSO, CNFC-ISEL, SEFC and MFS-SE controllers for the MIMO plant. ....	70
Table 5.2: Performance comparison of various controllers to control of backing up the truck. ....	75
Table 5.3: Performance comparison of various controllers for the water bath temperature control system. ....	81
Table 6.1: Performance comparison with PSO and DMPSO methods from the CIT database (Training data: 6000; Generations: 2000) .....	84
Table 6.2: Performance comparison with various existing models from the CIT database (Training data: 6000; Generations: 2000) .....	88
Table 6.3: The roles of IA, BFO and PSO in the proposed learning algorithm. ....	90

# List of Figures

Figure 1.1: The taxonomy of global optimization algorithms. ....	5
Figure 1.2: Aggregation chart for applications of the PSO over different years. ....	6
Figure 1.3: Taxonomy of PSO. ....	7
Figure 1.4: The variations of PSO. ....	9
Figure 2.1: Structure of the selected neuro-fuzzy system model. ....	16
Figure 3.1: The clonal selection principle. ....	22
Figure 3.2: Flowchart of the proposed IPSO algorithm. ....	24
Figure 3.3: Coding a neuro-fuzzy classifier into an antibody in the IPSO method. ....	25
Figure 3.4: The flowchart of the mutation step. ....	28
Figure 3.5: The coding of antibody population. ....	29
Figure 3.6: Flowchart of the skin color detection system. ....	31
Figure 3.7: The accuracy rate with different generations. ....	32
Figure 3.8: The learning curves of the three methods using the CIT database. ....	33
Figure 3.9: Original color images from CIT facial database. ....	35
Figure 3.10: Results of skin color detection with 3 dimension input (Y, Cb and Cr). ..	36
Figure 4.1: Flowchart of proposed BFPSO method. ....	44
Figure 4.2: Iris data: <i>iris sestosa</i> ( $\Delta$ ), <i>iris versicolor</i> ( $\circ$ ), and <i>iris virginica</i> ( $\square$ ). ....	47
Figure 4.3: The distribution of input training patterns and final assignment of three rules. ....	50
Figure 4.4: Learning curves of the NFS-BFPSO method, the NFS-BFO method, and the NFS-PSO method. ....	51
Figure 4.5: Input membership functions for breast cancer classification. ....	53
Figure 4.6: Learning curves from the NFS-BFPSO method, the NFS-BFO method and the NFS-PSO method. ....	54
Figure 4.7: The learning curves of the three methods using the CIT database. ....	56
Figure 4.8: Original face images from CIT database. ....	57
Figure 4.9: Results of skin color detection with 3 dimension input (Y, Cb, Cr). ....	57
Figure 5.1: Flowchart of the proposed learning scheme for the FLNFN model. ....	61
Figure 5.2: Learning curves of best performance of the FLNFN-DMPSO, CNFC-ISEL, SEFC and MFS-SE in MIMO plant control. ....	68
Figure 5.3: Desired (solid line) and model (dotted line) output generated by FLNFN-DMPSO in MIMO plant control. ....	69
Figure 5.4: Errors of proposed FLNFN-DMPSO in MIMO plant control. ....	69
Figure 5.5: Diagram of simulated truck and loading zone. ....	71
Figure 5.6: Learning curves of best performance of the FLNFN-DMPSO,	

CNFC-ISEL, SEFC and MFS-SE in control of backing up the truck. ....72

Figure 5.7: Trajectories of truck, starting at four initial positions under the control of the FLNFN-DMPSO after learning using training trajectories.....74

Figure 5.8: Conventional training scheme. ....76

Figure 5.9: The regulation performance of the FLNFN-DMPSO controller for the water bath system.....79

Figure 5.10: The behavior of the FLNFN-DMPSO controller under impulse noise for the water bath system. ....80

Figure 5.11: The behavior of the FLNFN-DMPSO controller when a change occurs in the water bath system dynamics. ....80

Figure 5.12: The tracking performance of the FLNFN-DMPSO controller for the water bath system.....81

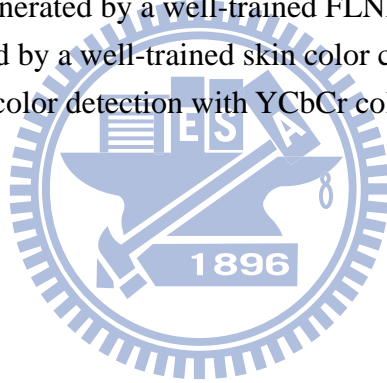
Figure 6.1: The learning curves of PSO and DMPSO methods using the CIT database. ....84

Figure 6.2: Original face images from CIT repository.....85

Figure 6.3: Fitness maps generated by a well-trained FLNFN-DMPSO.....86

Figure 6.4: Masks generated by a well-trained skin color classifier.....87

Figure 6.5: Results of skin color detection with YCbCr color space.....88



# Chapter 1

## Introduction

### 1.1 Motivation

Fuzzy systems and neural networks have attracted the growing interest of researchers in various scientific and engineering areas. The number and variety of applications of fuzzy systems and neural networks [1-6] have been increasing, ranging from consumer products and industrial process control to medical instrumentation, information systems, and decision analysis.

Fuzzy systems are *structured numerical estimators*. They start from highly formalized insights about the structure of categories found in the real world and then articulate fuzzy IF-THEN rules as a kind of expert knowledge. Fuzzy systems combine fuzzy sets with fuzzy rules to produce overall complex nonlinear behavior. Neural networks, on the other hand, are *trainable dynamical systems* whose learning, noise-tolerance, and generalization abilities grow out of their connectionist structures, their dynamics, and their distributed data representation. Neural networks have a large number of highly interconnected processing elements (nodes) which demonstrate the ability to learn and generalize from training patterns or data; these simple processing elements also collectively produce complex nonlinear behavior.

The performance of fuzzy systems critically depends on the input and output membership functions, the fuzzy rules, and the fuzzy inference mechanism. On the other hand, the performance of neural networks depends on the computational function of the neurons in the network, the structure and topology of the network, and the learning rule or the update rule of the connecting weights. The advantages and

disadvantages of fuzzy systems and neural networks are summarized as follows [7]:

**The advantages of the fuzzy systems are:**

- capacity to represent inherent uncertainties of the human knowledge with linguistic variables;
- simple interaction of the expert of the domain with the engineer designer of the system;
- easy interpretation of the results, because of the natural rules representation;
- easy extension of the base of knowledge through the addition of new rules;
- robustness in relation of the possible disturbances in the system.

**The disadvantages of the fuzzy systems are:**

- incapable to generalize, or either, it only answers to what is written in its rule base;
- not robust in relation the topological changes of the system, such changes would demand alterations in the rule base;
- depends on the existence of a expert to determine the inference logical rules;

**The advantages of the neural networks are:**

- learning capacity;
- generalization capacity;
- robustness in relation to disturbances.

**The disadvantages of the neural networks are:**

- impossible interpretation of the functionality;
- difficulty in determining the number of layers and number of neurons.

The hybrid neuro-fuzzy systems [8-34] possess the advantages of both neural networks (e.g. learning abilities, optimization abilities, and connectionist structures) and fuzzy systems (e.g. humanlike IF-THEN rules thinking and ease of incorporating expert knowledge). In this way, we can bring the low-level learning and computational power of neural networks into fuzzy systems and also high-level,

humanlike IF-THEN rule thinking and reasoning of fuzzy systems into neural networks.

There are several different ways to develop hybrid neuro-fuzzy systems; therefore, being a recent research subject, each researcher has defined its own particular models. These models are similar in its essence, but they present basic differences. The most popular neuro-fuzzy architectures include: 1) Fuzzy Adaptive Learning Control Network [8][20][21][29][35]; 2) Adaptive-Network-Based Fuzzy Inference System [24]; 3) Self-Constructing Neural Fuzzy Inference Network [25]; and 4) Functional-Link-Based Neuro-Fuzzy Network [32][33]. The advantages of a combination of neural networks and fuzzy inference systems are obvious [8][34-36]. Fusion of artificial neural networks and fuzzy inference systems have attracted the growing interest of researchers in various scientific and engineering areas due to the growing need of adaptive intelligent systems to solve the real world problems [8][9][19][20][24][25][30][33-38].

No matter which neuro-fuzzy architecture is chosen, training of the parameters is the main problem in designing a neuro-fuzzy system. Backpropagation (BP) [20][24][25][32][35][38][39] training is commonly adopted to solve this problem. It is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent approach is used in BP training to minimize the error function, the algorithms may reach the local minima very quickly and never find the global solution. The aforementioned disadvantages lead to suboptimal performance, even for a favorable neuro-fuzzy system topology. Therefore, technologies that can be used to train the system parameters and find the global solution while optimizing the overall structure are required.

Figure 1.1 sketches a rough taxonomy of global optimization methods [40]. Generally, optimization algorithms can be divided in two basic classes: deterministic



and probabilistic algorithms. Deterministic algorithms are most often used if a clear relation between the characteristics of the possible solutions and their utility for a given problem exists. Then, the search space can efficiently be explored using for example a divide and conquer scheme. If the relation between a solution candidate and its “fitness” are not so obvious or too complicated, or the dimensionality of the search space is very high, it becomes harder to solve a problem deterministically. Trying it would possibly result in exhaustive enumeration of the search space, which is not feasible even for relatively small problems. Then, probabilistic algorithms come into play.

An especially relevant family of probabilistic algorithms is the Monte Carlo-based approaches. They trade in guaranteed correctness of the solution for a shorter runtime. This does not mean that the results obtained using them are incorrect - they may just not be the global optima. An important class of probabilistic Monte Carlo metaheuristics is evolutionary computation (EC). It encompasses all algorithms that are based on a set of multiple solution candidates (called population) which are iteratively refined. This field of optimization is also a class of soft computing as well as a part of the artificial intelligence area. Some of its most important members are evolutionary algorithms (EAs) and swarm intelligence (SI).

The particle swarm optimization (PSO) developed by Kennedy and Eberhart in 1995 [41-43], is a relatively new technique. Although PSO shares many similarities with evolutionary computation techniques, the standard PSO does not use evolution operators such as crossover and mutation. PSO emulates the swarm behavior of insects, animals herding, birds flocking, and fish schooling where these swarms search for food in a collaborative manner. Each member in the swarm adapts its search patterns by learning from its own experience and other members’ experiences.

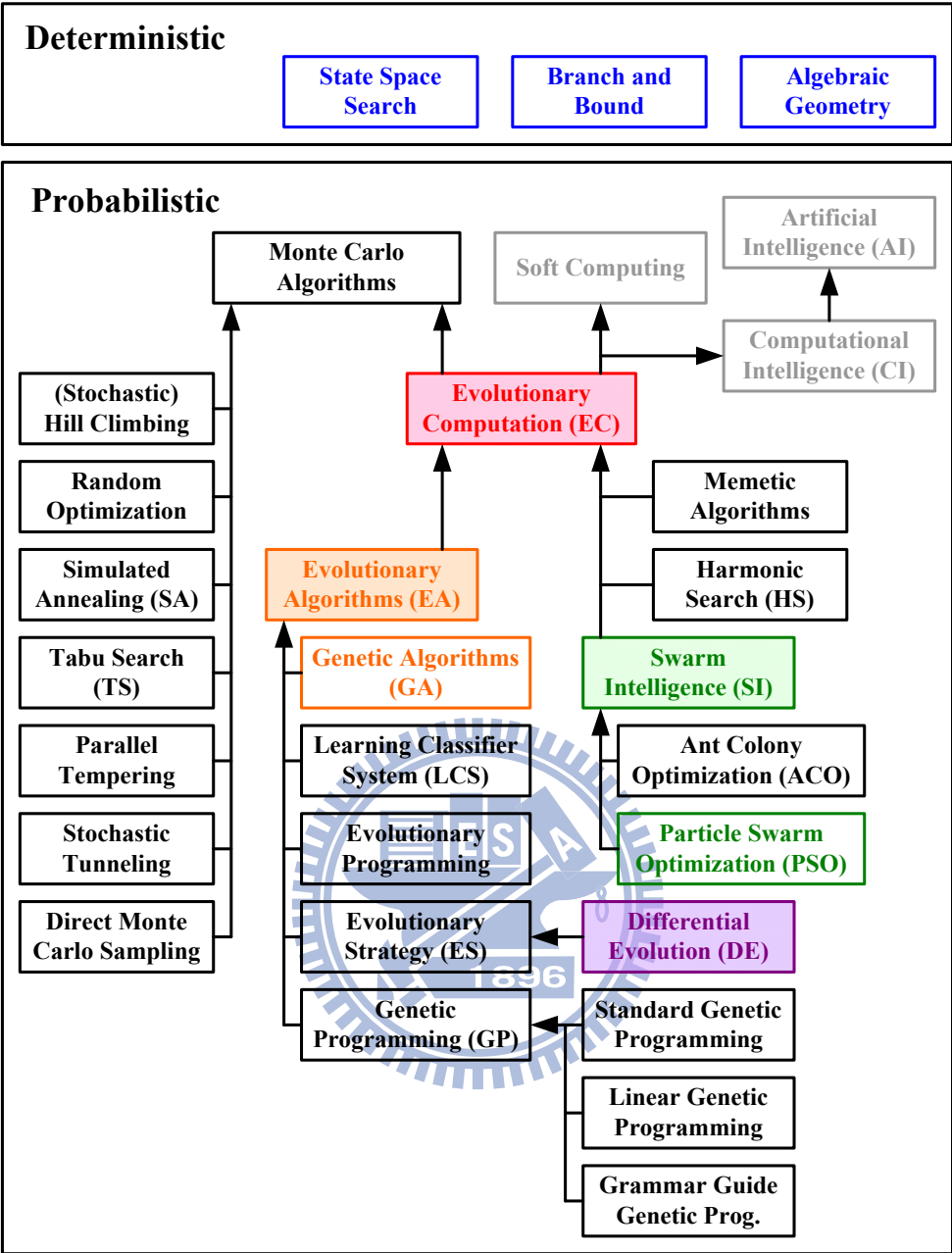


Figure 1.1: The taxonomy of global optimization algorithms.

During the past several years, PSO has been successfully applied to a diverse set of optimization problems, such as multidimensional optimization problems [44], multi-objective optimization problems [45-47], classification problems [48][49], and feedforward neural network design [39][50-53]. Aggregation chart for applications of the PSO over different years is shown in Figure 1.2 [54].

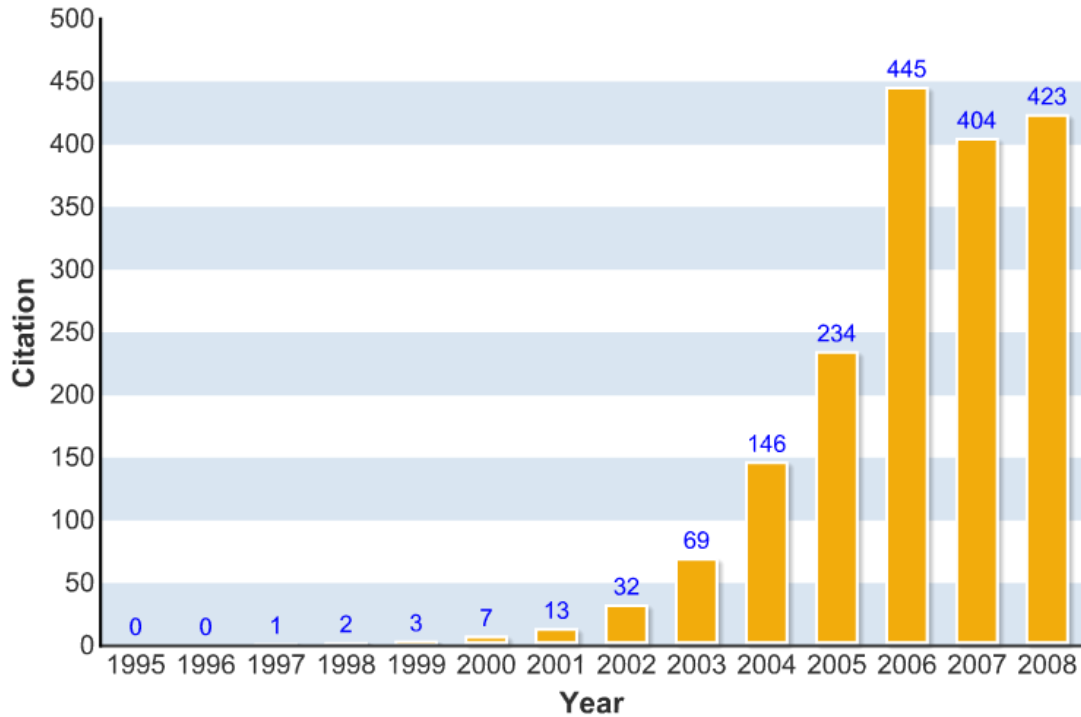


Figure 1.2: Aggregation chart for applications of the PSO over different years.

In this dissertation, we proposed the novel learning algorithms embedded with particle swarm optimizer for the neural fuzzy system in both classification and nonlinear system control applications.

## 1.2 Literature Survey

The underlying motivation for the development of PSO algorithm is the social behavior of animals, such as bird flocking, fish schooling and swarm theory. To simulate social behavior, bird flocking searches for food in an area. Each bird flies according to self-cognition and social information. Self-cognition is the generalization produced by past experience. The social information is the message that is shared by the society. The strategy of the birds is to maintain the good experiences by referring

to the knowledge of the others. A PSO's taxonomy is shown as Figure 1.3 [54].

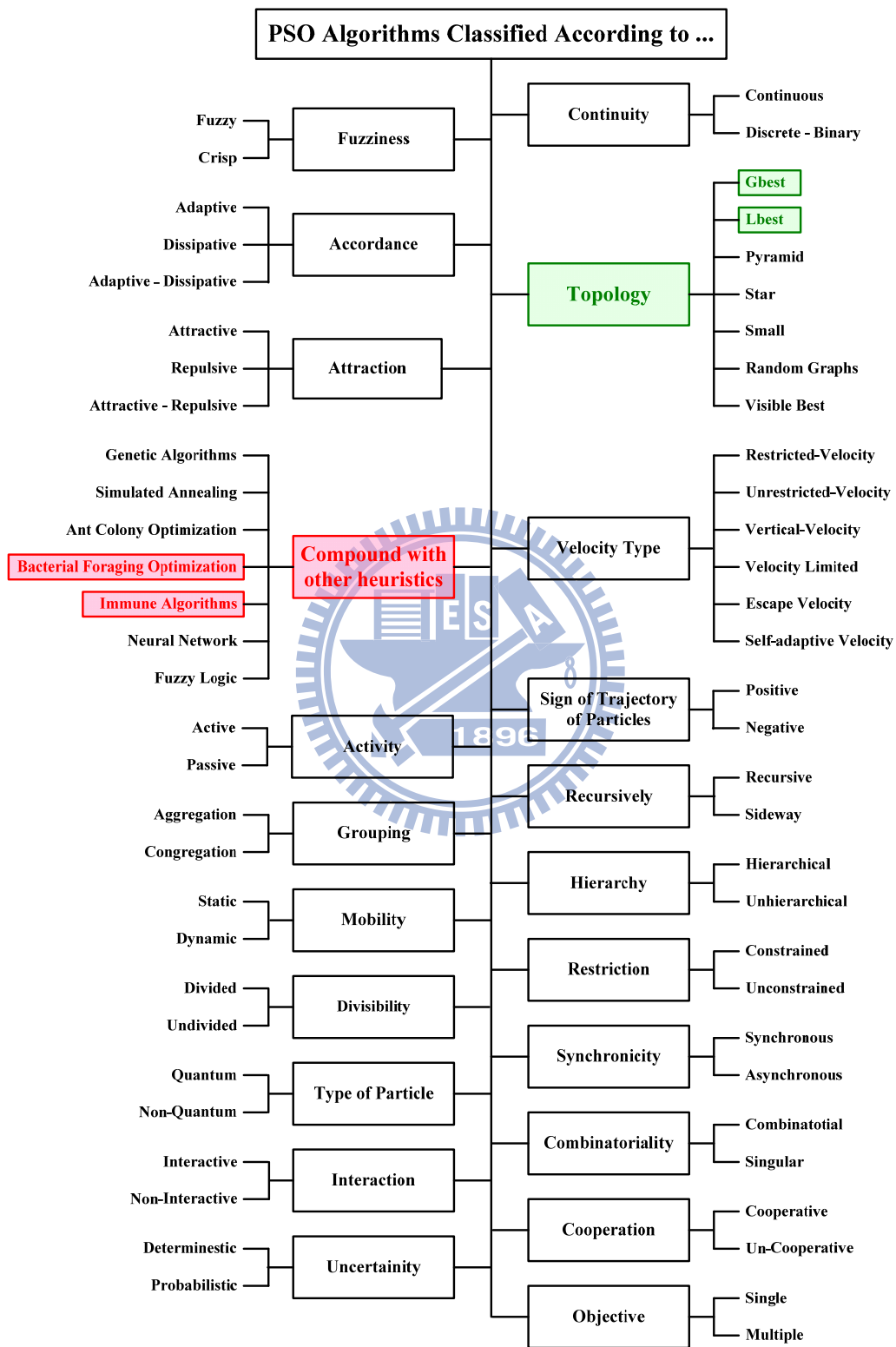


Figure 1.3: Taxonomy of PSO.

In PSO, a member in the swarm, called a *particle*, represents a potential solution which is a point in the search space. The global optimum is regarded as the location of food. Each particle has a fitness value and a velocity to adjust its flying direction according to the best experiences of the swarm to search for the global optimum in the solution space [55].

In the original PSO algorithm, the particles are manipulated according to the following equations:

$$v_{id}^t = v_{id}^{t-1} + c_1 \cdot r_1 \cdot (p_{id}^{t-1} - x_{id}^{t-1}) + c_2 \cdot r_2 \cdot (p_{gd}^{t-1} - x_{id}^{t-1}) \quad (1.1)$$

$$x_{id}^t = x_{id}^{t-1} + v_{id}^{t-1} \quad (1.2)$$

Here  $x_{id}^t$  and  $v_{id}^t$  are the  $d^{th}$  dimensional component of the position and velocity of the  $i^{th}$  particle at time step  $t$ .  $p_{id}^t$  is the  $d^{th}$  component of the best (fitness) position the  $i^{th}$  particle has achieved by time step  $t$ , and  $p_{gd}^t$  is the  $d^{th}$  component of the global best position achieved in the population by time step  $t$ . The constants  $c_1$  and  $c_2$  are known as the “cognition” and “social” factors, respectively, as they control the relative strengths of the individual behavior of each particle and collective behavior of all particles. Finally,  $r_1$  and  $r_2$  are two different random numbers in the range of 0 to 1 and are used to enhance the exploratory nature of the PSO.

The two main models of the PSO algorithm, called *gbest* (global best) and *lbest* (local best), which differ in the way of they define particle neighborhood. Kennedy and Poli [43][56] showed that the *gbest* model has a high convergence speed with a higher chance of getting stuck in local optima. On the contrary, the *lbest* model is less likely become trapped in local optima but has a slower convergence speed than *gbest*.

Many researchers have worked on improving its performance in various ways, thereby deriving many interesting variants as shown in Figure 1.4 [54].

A Parallel Vector-Based PSO (2005)	Active target PSO (2008)	Adaptive Dissipative PSO (2007)	Adaptive Mutation PSO (2008)	Adaptive PSO (2002)	Adaptive PSO Guided by acceleration information (2006)	Angle Modulated PSO (2005)	Area Extension PSO (2007)	Attractive-Repulsive PSO (2002)	Augmented Lagrangian PSO (2006)
Basic PSO (1995)	Behavior of distance PSO (2007)	Best rotation PSO (2007)	Binary PSO (1997)	Chaos PSO (2006)	Combinatorial PSO (2007)	Comprehensive Learning PSO (2006)	Constrained Optimization Via PSO (2007) C	Cooperative Co-evolutionary PSO (2006)	Cooperative Multiple PSO (2007)
Cultural Based PSO (2005)	Discrete PSO (1997)	Dissipative PSO (2002)	Divided Range PSO (2004)	Double-structure coding Binary PSO (2007)	Dual Layered PSO (2007)	Dynamic & Adjustable PSO (2007)	Dynamic Double PSO (2004)	Dynamic Neighborhood PSO (2003)	Escape Velocity PSO (2006)
Estimation of Distribution PSO (2007)	Evolutionary Iteration PSO (2007)	Evolutionary Programming PSO (2007)	Evolutionary PSO (2002)	Exploring Extended PSO (2005)	Extended PSO (2005)	Fast PSO (2007)	Fully informed PSO (2004)	Fuzzy PSO (2001)	Gaussian PSO (2003)
Genetic Binary PSO (2006)	Genetic PSO (2006)	Geometric PSO (2008)	Greedy PSO (2007)	Gregarious PSO (2006)	Heuristic PSO (2007)	Hierarchical Recursive-based PSO (2005)	Hybrid Discrete PSO (2006)	Hybrid Gradient PSO (2004)	Hybrid Recursive PSO (2007)
Hybrid Taguchi PSO (2006)	Immune PSO (2008)	Improved PSO (2006)	Interactive PSO (2005)	Map-Reduce PSO (2007)	Modified Binary PSO (2007)	Modified GPSO (2008)	Nbest PSO (2002)	Neural PSO (2005)	New PSO (2005)
New PSO (2006)	Niche PSO (2002)	Novel Hybrid PSO (2007)	Novel PSO (2008)	Optimized PSO (2006)	Orthogonal PSO (2008)	Parallel Asynchronous PSO (2006)	Perturbation PSO (2005)	Predator Prey PSO (2007)	Principal Component PSO (2005)
PSO with Craziness and Hill Climbing (2006)	PSO with Passive Congregation (2004)	Pursuit-Escape PSO (2008)	Quadratic Interpolation PSO (2007)	Quantum Delta PSO (2004)	Quantum PSO (2004)	Quantum-Inspired PSO (2004)	Restricted Velocity PSO (2006)	Self-adaptive velocity PSO (2008)	Self-Organization PSO (2006)
Simulated Annealing PSO (2004)	Spatial Extension PSO (2002)	Special Extension PSO (2006)	Species Based PSO (2004)	Sub-Swarms PSO (2007)	Trained PSO (2007)	Two-dimensional Otsu PSO (2007)	Two-Swarm PSO (2006)	Unconstrained PSO (2006)	Unified PSO (2004)
Variable Neighborhood PSO (2006)	Vector Limited PSO (2008)	Velocity Limited PSO (2006)	Velocity Mutation PSO (2008)	Vertical PSO (2007)					

Continuous
Binary
Discrete

Figure 1.4: The variations of PSO.

One of the variants introduces a parameter called inertia weight ( $w$ ) into the original PSO algorithms [56-58], and Eq. (1.1) can be rewritten as follows:

$$v_{id}^t = w \cdot v_{id}^{t-1} + c_1 \cdot r_1 \cdot (p_{id}^{t-1} - x_{id}^{t-1}) + c_2 \cdot r_2 \cdot (p_{gd}^{t-1} - x_{id}^{t-1}) \quad (1.3)$$

The inertia weight is used to balance the global and local search abilities. A large inertia weight is more appropriate for global search, and a small inertia weight facilitates local search. A linearly decreasing inertia weight over the course of search was proposed by Shi and Eberhart [58]. Parameters in PSO are discussed in [59]. Shi and Eberhart designed fuzzy methods to nonlinearly change the inertia weight [60]. In [61], inertia weight is set at zero, except at the time of re-initialization. In addition to

the time-varying inertia weight, a linearly decreasing  $v_{\max}$  is introduced in [62]. By analyzing the convergence behavior of the PSO, a PSO variant with a constriction factor was introduced by Clerc and Kennedy [63]. Constriction factor guarantees the convergence and improves the convergence velocity.

Improving PSO's performance by designing different types of topologies has been an active research direction. Kennedy [64][65] claimed that PSO with a small neighborhood might perform better on complex problems, while PSO with a large neighborhood would perform better on simple problems. Suganthan [66] applied a dynamically adjusted neighborhood where the neighborhood of a particle gradually increases until it includes all particles. In [67], Hu and Eberhart also used a dynamic neighborhood where closest particles in the performance space are selected to be its new neighborhood in each generation. Parsopoulos and Vrahatis combined the global version and local version together to construct a unified particle swarm optimizer (UPSO) [68][69]. Mendes and Kennedy introduced a fully informed PSO in [70]. Instead of using the *pbest* and *gbest* positions in the standard algorithm, all the neighbors of the particle are used to update the velocity. The influence of each particle to its neighbors is weighted based on its fitness value and the neighborhood size. Veeramachaneni *et al.* developed the fitness-distance-ratio-based PSO (FDR-PSO) with near neighbor interactions [71]. When updating each velocity dimension, the FDR-PSO algorithm selects one other particle *nbest*, which has a higher fitness value and is nearer to the particle being updated.

Some researchers investigated hybridization by combining PSO with other search techniques to improve the performance of the PSO. Evolutionary operators such as selection, crossover, and mutation have been introduced to the PSO to keep the best particles [72], to increase the diversity of the population, and to improve the ability to escape local optimum [73]. Mutation operators are also used to mutate

parameters such as the inertia weight [74]. Relocating the particles when they are too close to each other [75] or using some collision-avoiding mechanisms [76] to prevent particles from moving too close to each other in order to maintain the diversity and to escape from local optima has also been used. In [73], the swarm is divided into subpopulations, and a breeding operator is used within a subpopulation or between the subpopulations to increase the diversity of the population. Negative entropy is used to discourage premature convergence in [77]. In [78], deflection, stretching, and repulsion techniques are used to find as many minima as possible by preventing particles from moving to a previously discovered minimal region. Recently, a cooperative particle swarm optimizer (CPSO-H) [79] was proposed. Although CPSO-H uses one-dimensional (1-D) swarms to search each dimension separately, the results of these searches are integrated by a global swarm to significantly improve the performance of the original PSO on multimodal problems.

From our review of the state-of-the-art, we noticed two tendencies: 1) PSO variants are mostly added with further operators (e.g. mutation operator) and mechanisms (e.g. “fly-back”, multi-swarms, co-evolution), and 2) PSO variants are merged into one in order to improve its performance. Therefore, in this dissertation, we present three novel PSO-based learning algorithms for the neuro-fuzzy systems according to these two tendencies.

### **1.3 Organization of Dissertation**

The overall objective of this dissertation is to develop the novel learning algorithms embedded with particle swarm optimizer for the neuro-fuzzy systems. The proposed learning algorithms are suitable for any neuro-fuzzy architecture. In this



research, we take the functional-link-based neuro-fuzzy network (FLNFN) model for example to demonstrate the performance of the proposed learning algorithms. Organization and objectives of each chapter in this dissertation are as follows.

In Chapter 2, we describe the structure of FLNFN model. The FLNFN model is based on our laboratory's previous research [32]. Each fuzzy rule corresponds to a sub-FLNN [80-82] comprising a functional expansion of input variables. The functional link neural network (FLNN) is a single layer neural structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries with nonlinear functional expansion. Therefore, the consequent part of the FLNFN model is a nonlinear combination of input variables, which differs from the other existing models [20][24][25].

In Chapter 3, we propose an efficient immune-based particle swarm optimization (IPSO) algorithm for neuro-fuzzy classifiers to solve the skin color detection problem. The proposed IPSO algorithm combines the immune algorithm (IA) and PSO to perform parameter learning. The IA uses the clonal selection principle, such that antibodies between others of high similar degree are affected, and these antibodies, after the process, will have higher quality, accelerating the search and increasing the global search capacity. On the other hand, we employed the advantages of PSO to improve the mutation mechanism of IA. Simulations have conducted to show the performance and applicability of the proposed method.

In Chapter 4, we present an evolutionary neural fuzzy classifier, designed using the neural fuzzy system (NFS) and a new evolutionary learning algorithm. This new evolutionary learning algorithm is based on a hybrid of bacterial foraging optimization (BFO) and PSO. It is thus called bacterial foraging particle swarm optimization (BFPSO). The proposed BFPSO method performs local search through the chemotactic movement operation of bacterial foraging whereas the global search

over the entire search space is accomplished by a particle swarm operator. The proposed NFS with BFPSO learning algorithm (NFS-BFPSO) is adopted in several classification applications. Experimental results have demonstrated that the proposed NFS-BFPSO method can outperform other methods.

In Chapter 5, we present an evolutionary NFS for nonlinear system control. A supervised learning algorithm, which consists of structure learning and parameter learning, is presented. The structure learning depends on the entropy measure to determine the number of fuzzy rules. The parameter learning, based on the PSO algorithm, can adjust the shape of the membership function and the corresponding weighting of the FLNN. The distance-based mutation operator, which strongly encourages a global search giving the particles more chance of converging to the global optimum, is introduced. The simulation results have shown the proposed method can improve the searching ability and is very suitable for the nonlinear system control applications.

In Chapter 6, we compare the performance of the proposed learning algorithms using skin color detection problem. In addition, a brief discussion of the proposed learning methods is also made.

Finally, Chapter 7 draws conclusions and future works.

## Chapter 2

# Structure of the Functional-Link-Based Neuro-Fuzzy Network

In the field of artificial intelligence, neural networks are essentially low-level computational structures and algorithms that offer good performance when they deal with sensory data. However, it is difficult to understand the meaning of each neuron and each weight in the networks. Generally, fuzzy systems are easy to appreciate because they use linguistic terms and IF-THEN rules. However, they lack the learning capacity to fine-tune fuzzy rules and membership functions. Therefore, neuro-fuzzy networks combine the benefits of neural networks and fuzzy systems to solve many engineering problems.

In [83], the definition of hybrid neuro-fuzzy system is as follows: “A hybrid neuro-fuzzy system is a fuzzy system that uses a learning algorithm based on gradients or inspired by the neural networks theory (heuristic learning strategies) to determine its parameters (fuzzy sets and fuzzy rules) through the patterns processing (input and output)”. In other words, neuro-fuzzy networks bring the low-level learning and computational power of neural networks into fuzzy systems and give the high-level human-like thinking and reasoning of fuzzy systems to neural networks.

Recently, neuro-fuzzy networks have become popular topics of research. The advantages of a combination of neural networks and fuzzy inference systems are obvious [8][34-36]. They not only have attracted considerable attention due to their diverse applications in fields such as pattern recognition, image processing, prediction, and control, but they can also handle imprecise information through linguistic

expressions. The most popular neuro-fuzzy architectures include: 1) Fuzzy Adaptive Learning Control Network (FALCON) [8][20][21][29][35]; 2) Adaptive-Network-Based Fuzzy Inference System (ANFIS) [24]; 3) Self-Constructing Neural Fuzzy Inference Network (SONFIN) [25]; and 4) Functional-Link-Based Neuro-Fuzzy Network (FLNFN) [32][33].

In this dissertation, the selected NFS model is based on our laboratory's previous research [32][33], called FLNFN. Figure 2.1 presents the structure of the FLNFN model, which combines a neuro-fuzzy network with a FLNN [80-82]. The FLNN [81][84] is a single layer neural structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries with nonlinear functional expansion. Moreover, the FLNN was conveniently used for function approximation and pattern classification with faster convergence rate and less computational loading than a multilayer neural network. In the selected FLNFN model, each fuzzy rule that corresponds to a FLNN consists of a functional expansion of input variables, which differs from the other existing models [20][24][25].

The FLNFN model realizes a fuzzy IF-THEN rule in the following form.

Rule  $j$  :

$$\begin{aligned} \text{IF } \hat{x}_1 \text{ is } A_{1j} \text{ and } \hat{x}_2 \text{ is } A_{2j} \dots \text{ and } \hat{x}_i \text{ is } A_{ij} \dots \text{ and } \hat{x}_N \text{ is } A_{Nj} \\ \text{THEN } \hat{y}_j = \sum_{k=1}^M w_{kj} \phi_k = w_{1j} \phi_1 + w_{2j} \phi_2 + \dots + w_{Mj} \phi_M \end{aligned} \quad (2.1)$$

where  $\hat{x}_i$  and  $\hat{y}_j$  are the input and local output variables, respectively;  $A_{ij}$  is the linguistic term of the precondition part with a Gaussian membership function;  $N$  is the number of input variables;  $w_{kj}$  is the link weight of the local output;  $\phi_k$  is the basis trigonometric function of input variables;  $M$  is the number of basis functions, and Rule  $j$  is the  $j^{\text{th}}$  fuzzy rule.

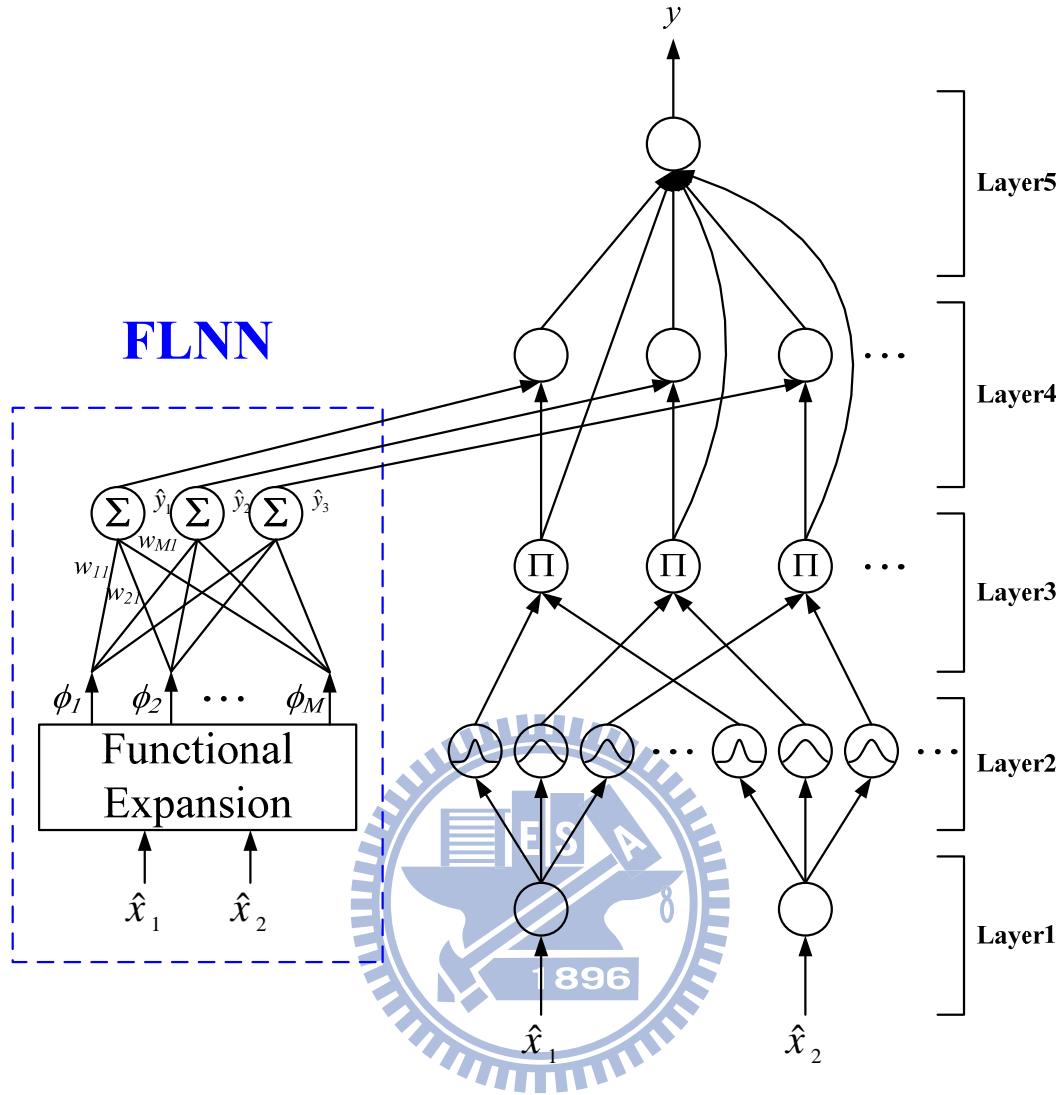


Figure 2.1: Structure of the selected neuro-fuzzy system model.

The operation functions of the nodes in each layer of the FLNFN model are now described. In the following description,  $u^{(l)}$  denotes the output of a node in the  $l^{th}$  layer.

**Layer 1 (Input node):** No computation is performed in this layer. Each node in this layer is an input node, which corresponds to one input variable, and only transmits input values to the next layer directly:

$$u_i^{(1)} = \hat{x}_i \quad (2.2)$$

**Layer 2 (Membership function node):** Nodes in this layer correspond to a single

linguistic label of input variables in layer 1. Therefore, the calculated membership value specifies the degree to which an input value belongs to a fuzzy set in layer 2. The implemented Gaussian membership function in layer 2 is

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \quad (2.3)$$

where  $m_{ij}$  and  $\sigma_{ij}$  are the mean and standard deviation of the Gaussian membership function, respectively, of the  $j^{th}$  term of the  $i^{th}$  input variable  $\hat{x}_i$ .

**Layer 3 (Rule Node):** Nodes in this layer represent the preconditioned part of a fuzzy logic rule. They receive one-dimensional membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described above is adopted to perform the IF-condition matching of the fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \quad (2.4)$$

where the  $\prod_i u_{ij}^{(2)}$  of a rule node represents the firing strength of its corresponding rule.

**Layer 4 (Consequent Node):** Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are nonlinear combinations of input variables from a FLNN, as shown in Figure 2.1. For such a node,

$$u_j^{(4)} = u_j^{(3)} \cdot \sum_{k=1}^M w_{kj} \phi_k \quad (2.5)$$

where  $w_{kj}$  is the corresponding link weight of the FLNN and  $\phi_k$  is the functional expansion of input variables. Considering the computational efficiency, the functional expansion uses a trigonometric polynomial basis function, given by

$[\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6] = [\hat{x}_1, \sin(\pi\hat{x}_1), \cos(\pi\hat{x}_1), \hat{x}_2, \sin(\pi\hat{x}_2), \cos(\pi\hat{x}_2)]$  for the two-dimensional input variables  $[\hat{x}_1, \hat{x}_2]$ . Therefore,  $M$  is the number of basis functions,  $M = 3 \cdot N$ , where  $N$  is the number of input variables. Moreover, the output nodes of FLNN depend on the number of fuzzy rules of the FLNFN model.

**Layer 5 (Output Node):** Each node in this layer corresponds to a single output variable. The node integrates all of the actions recommended by layers 3 and 4 and acts as a *center of area* (COA) defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^R u_j^{(4)}}{\sum_{j=1}^R u_j^{(3)}} = \frac{\sum_{j=1}^R u_j^{(3)} \left( \sum_{k=1}^M w_{kj} \phi_k \right)}{\sum_{j=1}^R u_j^{(3)}} = \frac{\sum_{j=1}^R u_j^{(3)} \hat{y}_j}{\sum_{j=1}^R u_j^{(3)}} \quad (2.6)$$

where  $R$  is the number of fuzzy rules, and  $y$  is the output of the FLNFN model.

As described above, the number of tuning parameters for the FLNFN model is known to be  $(2+3P) \cdot N \cdot R$ , where  $N$ ,  $R$ , and  $P$  denote the number of inputs, existing rules, and outputs, respectively.

## Chapter 3

# Immune Algorithm Embedded with Particle Swarm Optimizer for Neuro-Fuzzy Classifier and Its Applications

Skin color detection is the process of finding skin-colored pixels and regions in an image or a video. This process is typically used as a preprocessing step to find regions that potentially have human faces and limbs in images. Several computer vision approaches have been developed for skin color detection. A skin color detector typically transforms a given pixel into an appropriate color space and then use a skin color classifier to label the pixel whether it is a skin or a non-skin pixel. A skin color classifier defines a decision boundary of the skin color class in the color space based on a training database of skin-colored pixels.

This chapter presents the efficient immune-based particle swarm optimization (IPSO) for neuro-fuzzy classifiers to solve the skin color detection problem. The proposed IPSO algorithm combines the immune algorithm (IA) and particle swarm optimization (PSO) to perform parameter learning. The IA uses the clonal selection principle to affect antibodies between others of high similar degree, and these antibodies, after the process, will be of higher quality, accelerating the search, and increasing the global search capacity. The PSO algorithm, proposed by Kennedy and Eberhart [41-43], has proved to be very effective for solving global optimization. It is not only a recently invented high-performance optimizer that is easy to understand



and implement, but it also requires little computational bookkeeping and generally only a few lines of code [85]. In order to avoid trapping in a local optimal solution and to ensure the search capability of a near global optimal solution, mutation plays an important role in IPSO. Therefore, we employ the advantages of PSO to improve mutation mechanism of IA. The proposed method can improve the searching ability and greatly increase the converging speed that we can observe in the simulations.

### **3.1 Basic Concepts of the Artificial Immune System**

The biological immune system is successful at protecting living bodies from the invasion of various foreign substances, such as viruses, bacteria, and other parasites (called antigens), and eliminating debris and malfunctioning cells. Over the last few years, a growing number of computer scientists have carefully studied the success of this competent natural mechanism and proposed computer immune models, named artificial immune systems (AIS), for solving various problems [86-94]. AIS aim at using ideas gleaned from immunology in order to develop adaptive systems capable of performing a wide range of tasks in various areas of research.

In this research, we review the clonal selection concept, together with the affinity maturation process, and demonstrate that these biological principles can lead to the development of powerful computational tools. The algorithm to be presented focuses on a systemic view of the immune system and does not take into account cell-cell interactions. It is not our concern to model exactly any phenomenon, but to show that some basic immune principles can help us not only to better understand the immune system itself, but also to solve complex engineering tasks.

## 3.2 Clonal Selection Theory

Any molecule that can be recognized by the adaptive immune system is known as an antigen (Ag). When an animal is exposed to an Ag, some subpopulation of its bone-marrow-derived cells (B lymphocytes) responds by producing antibodies (Ab's). Ab's are molecules attached primarily to the surface of B cells whose aim is to recognize and bind to Ag's. Each B cell secretes a single type of antibody (Ab), which is relatively specific for the Ag. By binding to these Ab's (cell receptors) and with a second signal from accessory cells, such as the T-helper cell, the Ag stimulates the B cell to proliferate (divide) and mature into terminal (non-dividing) Ab secreting cells, called plasma cells. The process of cell division (mitosis) generates a clone, i.e., a cell or set of cells that are the progenies of a single cell. While plasma cells are the most active Ab secretors, large B lymphocytes, which divide rapidly, also secrete Ab's, albeit at a lower rate. On the other hand, T cells play a central role in the regulation of the B cell response and are preeminent in cell mediated immune responses, but will not be explicitly accounted for the development of our model.

Lymphocytes, in addition to proliferating and/or differentiating into plasma cells, can differentiate into long-lived B memory cells. Memory cells circulate through the blood, lymph and tissues, and when exposed to a second antigenic stimulus commence to differentiate into large lymphocytes capable of producing high affinity antibodies, pre-selected for the specific antigen that had stimulated the primary response [95]. In this study, we treat the long-lived B memory cells as the better antibodies by elitism selection. Figure 3.1 depicts the clonal selection principle [95].

The main features of the clonal selection theory [96][97] that will be explored in this study are:

- Proliferation and differentiation on stimulation of cells with Ag's;

- Generation of new random genetic changes, subsequently expressed as diverse Ab patterns, by a form of accelerated somatic mutation (a process called affinity maturation);
- Elimination of newly differentiated lymphocytes carrying low affinity antigenic receptors.

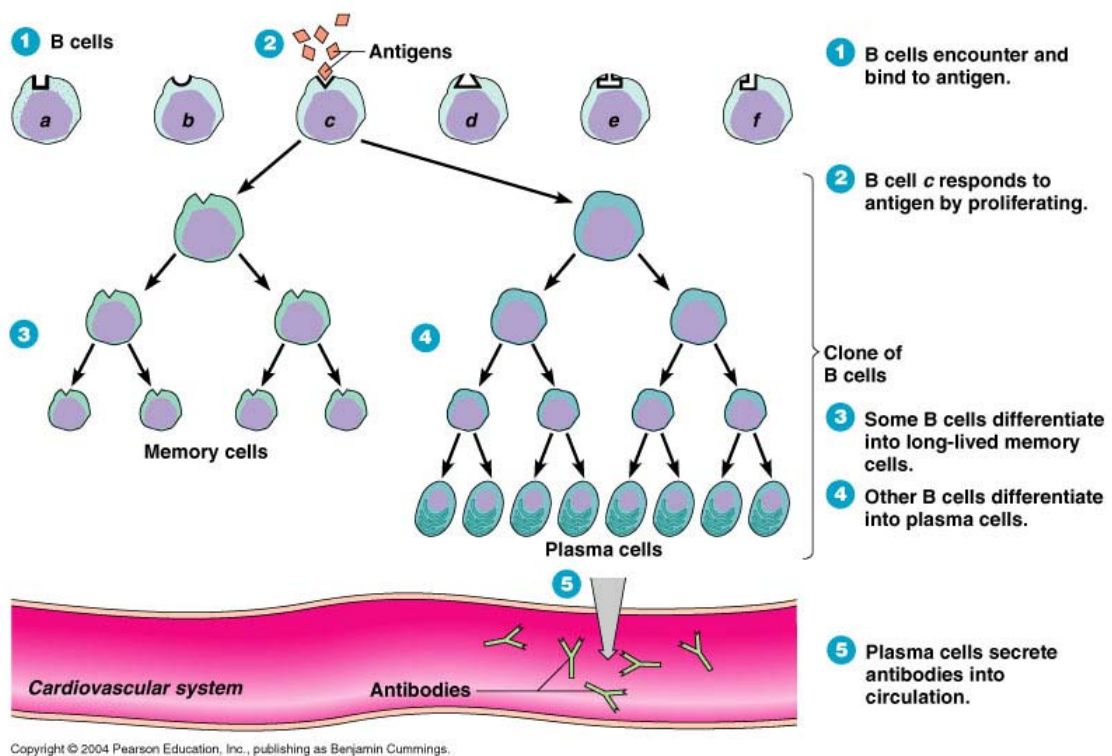


Figure 3.1: The clonal selection principle.

### 3.3 The Efficient Immune-Based PSO Learning Algorithm

This section describes the efficient immune-based PSO (IPSO) learning algorithm for use in the neuro-fuzzy classifier. Analogous to the biological immune system, the proposed algorithm has the capability of seeking feasible solutions while maintaining diversity. The proposed IPSO combines the immune algorithm (IA) and particle swarm optimization (PSO) to perform parameter learning. The IA uses the

clonal selection principle to accelerate the search and increase global search capacity. The PSO algorithm has proved to be very effective for solving global optimization. It is not only a recently invented high-performance optimizer that is very easy to understand and implement, but it also requires little computational bookkeeping and generally only a few lines of code. In order to avoid trapping in a local optimal solution and to ensure the search capability of a near global optimal solution, mutation plays an important role in IPSO. Moreover, the PSO adopted in evolution algorithm yields high diversity to increase the global search capacity, as well as the mutation scheme. Therefore, we employed the advantages of PSO to improve the mutation mechanism of IA. A detailed IPSO of the neuro-fuzzy classifier is presented in Figure 3.2. The whole learning process is described step-by-step below.

### 3.3.1 Code fuzzy rule into antibody

The coding step is concerned with the membership functions and the corresponding parameters of the consequent part of a fuzzy rule that represent Ab's suitable for IPSO. This step codes a rule of a neuro-fuzzy classifier into an Ab. Figure 3.3 shows an example of a neuro-fuzzy classifier coded into an Ab (i.e. an Ab represents a rule set), where  $i$  and  $j$  represent the  $i^{th}$  dimension and the  $j^{th}$  rule, respectively. In this research, a Gaussian membership function is used with variables representing the mean and standard deviation of the membership function. Each fuzzy rule has the form in Figure 2.1, where  $m_{ij}$  and  $\sigma_{ij}$  represent a Gaussian membership function with mean and standard deviation of the  $i^{th}$  dimension and  $j^{th}$  rule node and  $w_{ij}$  represents the corresponding parameters of consequent part.

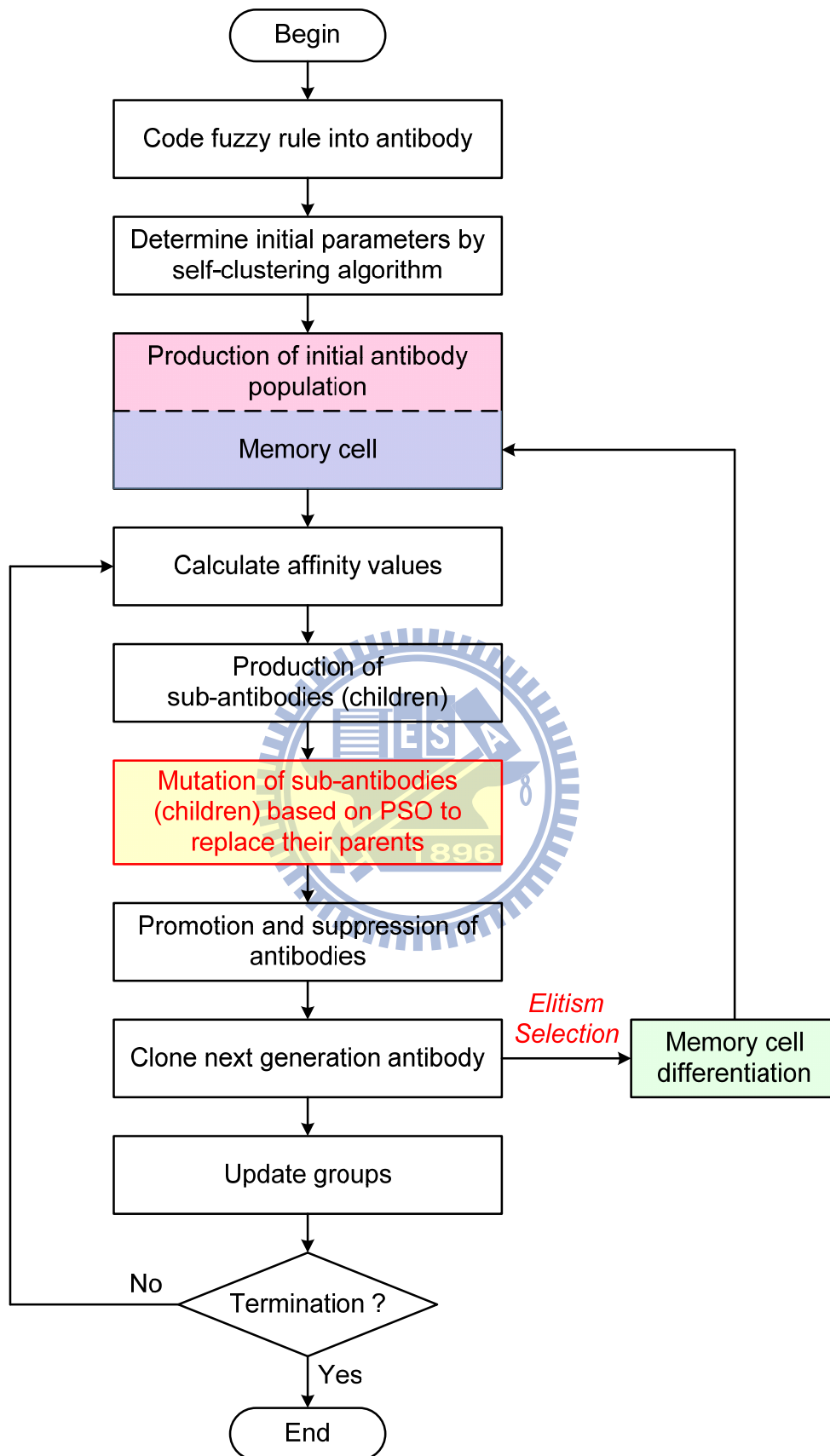


Figure 3.2: Flowchart of the proposed IPSO algorithm.

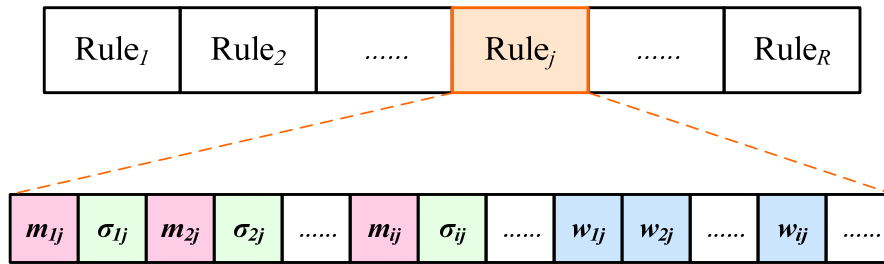


Figure 3.3: Coding a neuro-fuzzy classifier into an antibody in the IPSO method.

### 3.3.2 Determine the initial parameters by self-clustering algorithm

Before the IPSO method is designed, the initial Ab's in the populations are generated according to the initial parameters of the antecedent part and the consequent part. In this study, the initial parameters of a neuro-fuzzy classifier were computed by the self-clustering algorithm (SCA) method [52][98][99]. That is, we used SCA method to determine the initial mean and standard deviation of the antecedent part. On the other hand, the initial link weight of the consequent part is a random number in the range of 0 to 1.

SCA is a distance-based connectionist clustering algorithm. In any cluster, the maximum distance between an example point and the cluster center is less than a threshold value. This clustering algorithm sets clustering parameters and affects the number of clusters to be estimated. In the clustering process, the data examples come from a data stream. The clustering process starts with an empty set of clusters. The clusters will be updated and changed depending on the position of the current example in the input space.

### 3.3.3 Produce initial population

In the immune system, the Ab's are produced in order to cope with the Ag's. In other words, the Ag's are recognized by a few of high affinity Ab's (i.e. the Ag's are optimal solutions). The first initial Ab utilizing a real variable string is generated by

SCA, and the other Ab's of population are generated based on the first initial Ab by adding some random value.

### 3.3.4 Calculate affinity values

For the large number of various Ag's, the immune system has to recognize them for their posterior influence. In biological immune system, *affinity* refers to the binding strength between a single antigenic determinants and an individual antibody-combining site. The process of recognizing Ag's is to search for Ab's with the maximum affinity with Ag's. Moreover, every Ab in the population is applied to problem solving, and the affinity value is a performance measure of an Ab which is obtained according to the error function. In this study, the affinity value is designed according to the follow formulation:

$$Affinity\ value = \frac{1}{1 + \sqrt{\frac{1}{N_D} \sum_{k=1}^{N_D} (y_k - y_k^d)^2}} \quad (3.1)$$

where  $y_k$  represents the  $k^{th}$  model output,  $y_k^d$  represents the desired output, and  $N_D$  represents the number of the training data. In the problems, the higher affinity refers to the better Ab.

### 3.3.5 Production of sub-antibodies

In this step, we will generate several neighborhoods to maintain solution variation. This strategy can prevent the search process from becoming premature. We can generate several clones for each Ab on feasible space by Eqs. (3.2), (3.3) and (3.4). Each Ab regarded as parent while the clones regarded as children (sub-antibodies). In other words, children regarded as several neighborhoods of near parent.

$$\text{mean: } clones[children_{i\_c}] = antibody[parent_i] + \alpha \quad (3.2)$$

$$\text{deviation: } clones[children_{i_c}] = antibody[parent_i] + \alpha \quad (3.3)$$

$$\text{weight : } clones[children_{i_c}] = antibody[parent_i] + \beta \quad (3.4)$$

where  $parent_i$  represents the  $i^{th}$  Ab from the Ab population;  $children_{i_c}$  represents clones number  $c$  from the  $i^{th}$  Ab;  $\alpha$  and  $\beta$  are parameters that control the distance between parent. In this scheme,  $\alpha$  and  $\beta$  are important parameters. The large values lead to the speed of convergence slowly and the search of optimal solution difficulty, whereas the small values lead to fall in a local optimal solution easily. Therefore, the selection of the  $\alpha$  and  $\beta$  will critically affect the learning results, and their values will be based on practical experimentation or on trial-and-error tests.

### 3.3.6 Mutation of sub-antibodies based on PSO

In order to avoid trapping in a local optimal solution and to ensure the search capability of near global optimal solution, mutation plays an important role in IPSO. Moreover, the PSO adopted in evolution algorithm yields high diversity to increase the global search capacity, as well as the mutation step. Hence, we employed the advantages of PSO to improve mutation mechanism. Through the mutation step, only one best child can survive to replace its parent and enter the next generation.

PSO is a recently invented high-performance optimizer that is very easy to understand and implement. Each particle has a velocity vector  $v_i$  and a position vector  $x_i$  to represent a possible solution. In this research, the velocity for each particle is updated by Eq. (1.3). The parameter  $w \in (0, 1]$  is the inertia of the particle, and controls the exploratory properties of the algorithm. The constants  $c_1$  and  $c_2$  are known as the “cognition” and “social” factors, respectively.  $r_1$  and  $r_2$  are uniformly distributed random numbers in  $[0, 1]$ . The term  $v_i$  is limited to the range



$\pm v_{\max}$ . If the velocity violates this limit, it will be set at its proper limit. Changing velocity enables every particle to search around its individual best position and global best position. Based on the updated velocities, each particle changes its position according to Eq. (1.2).

When every particle is updated, the affinity value of each particle is calculated again. If the affinity value of the new particle is higher than those of local best, then the local best will be replaced with the new particle. Moreover, in the mutation step, each Ab (or particle) in the population must be mutated only one time by PSO in each generation. The mutation step flowchart is presented in Figure 3.4.

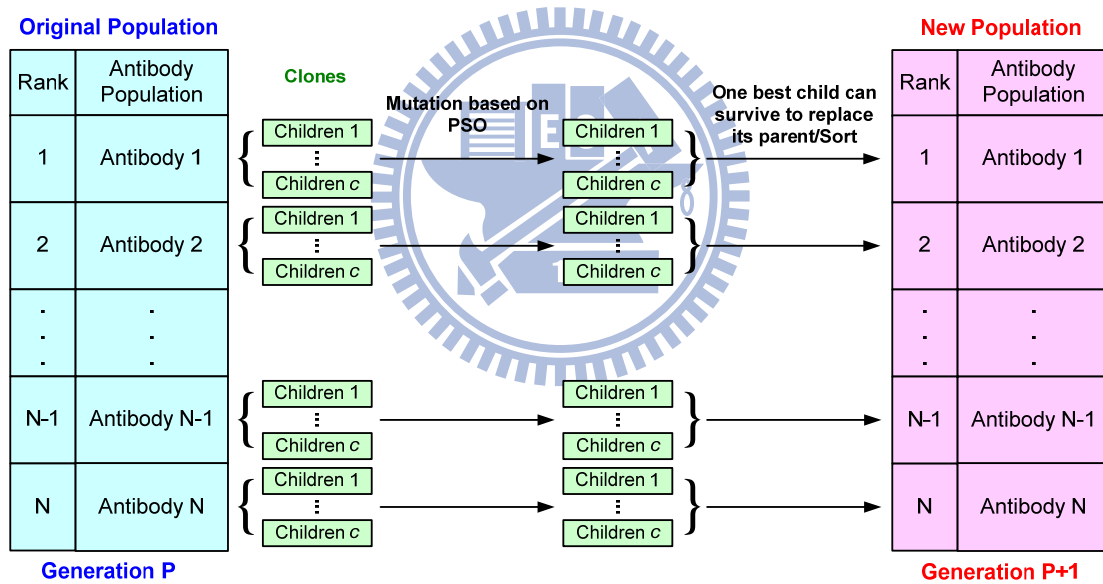


Figure 3.4: The flowchart of the mutation step.

### 3.3.7 Promotion and suppression of antibodies

In order to affect Ag's and keep diversity to a certain degree, we use information entropy theory to measure the diversity of Ab's. If the affinity between two Ab's is greater than the suppression threshold  $Th_{aff}$ , these two Ab's are similar, and the Ab of lower affinity value is reduced a small amount of value  $\lambda$ . Figure 3.5 shows the

immune algorithm composed of  $N$  Ab's having  $L$  genes.

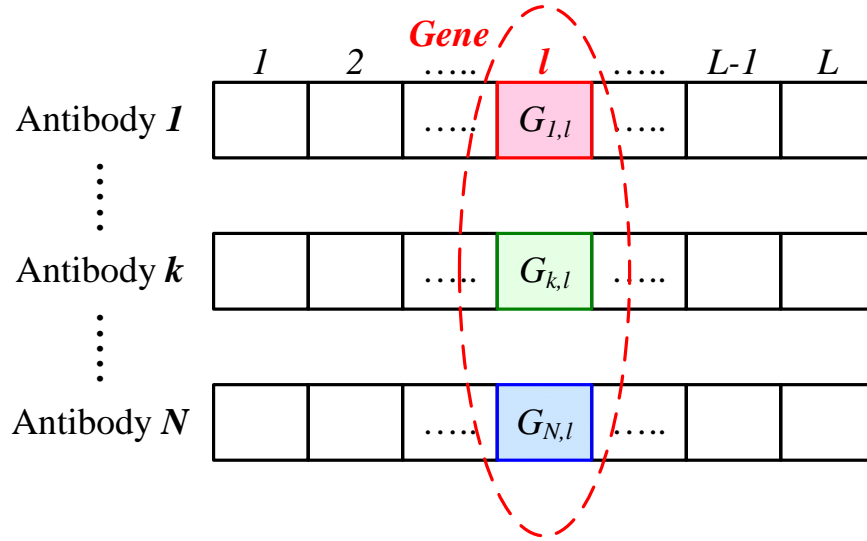


Figure 3.5: The coding of antibody population.

From information entropy theory, we get

$$IE_l(N) = \sum_{i=1}^N -P_{il} \log P_{il} \quad (3.5)$$

where  $P_{il}$  is the probability that the  $i^{th}$  allele comes out at the  $l^{th}$  gene. The diversity of the genes is calculated using Eq. (3.5). The average entropy value  $IE(N)$  of diversity can be also computed as follows:

$$IE(N) = \frac{1}{L} \sum_{l=1}^L IE_l(N) \quad (3.6)$$

where  $L$  is the size of the gene in an Ab. Equation (3.6) yields the diversity of the Ab pool in terms of the entropy. There are two kinds of affinities in IPSO. One explains the relationship between an Ab and an Ag using Eq. (3.1). The other accounts for the degree of association between the  $j^{th}$  Ab and the  $k^{th}$  Ab and measures how similar these two Ab's are. It can be calculated by using

$$Affinity_{-Ab_{jk}} = \frac{1}{1 + IE(2)} \quad (3.7)$$

where  $Affinity_{Ab_{jk}}$  is the affinity between two Ab's  $j$  and  $k$ , and  $IE(2)$  is the entropy of only the Ab's  $j$  and  $k$ . This affinity is constrained from zero to one. When  $IE(2)$  is zero, the genes of the  $j^{th}$  Ab and the  $k^{th}$  Ab are the same.

### 3.3.8 Elitism selection

When a new generation is created, the risk of losing the best Ab is always existent. In this study, we adopt elitism selection to overcome the above-mentioned problem. Therefore, the Ab's are ranked in ascending order of their affinity values. The best Ab is kept as the parent for the next generation. Moreover, the best Ab and Ab's with high antigenic affinity are transformed into long-lived B memory cells. Elitism selection improves the efficient of IPSO considerably, as it prevents losing the best result.



## 3.4 Skin Color Detection

Detecting skin-colored pixels, although seems a straightforward easy task, has proven quite challenging for many reasons. The appearance of skin in an image depends on the illumination conditions where the image was captured. Therefore, an important challenge in skin detection is to represent the color in a way that is invariant or at least insensitive to changes in illumination. The choice of the color space affects greatly the performance of any skin detector and its sensitivity to change in illumination conditions. Another challenge comes from the fact that many objects in the real world might have skin-tone colors. This causes any skin detector to have much false detection in the background if the environment is not controlled.

Figure 3.6 shows a flowchart of a skin color detection system. Skin detection

process has two phases: a training phase and a detection phase. Training a skin detector involves three basic steps:

1. Collecting a database of skin patches from different images. Such a database typically contains skin-colored patches from a variety of people under different illumination conditions.
2. Choosing a suitable color space.
3. Learning the parameters of a skin classifier.

Given a trained skin detector, identifying skin pixels in a given image or video frame involves:

1. Converting the image into the same color space that was used in the training phase.
2. Classifying each pixel using the skin classifier to either a skin or non-skin.
3. Typically post processing is needed using morphology to impose spatial homogeneity on the detected regions.

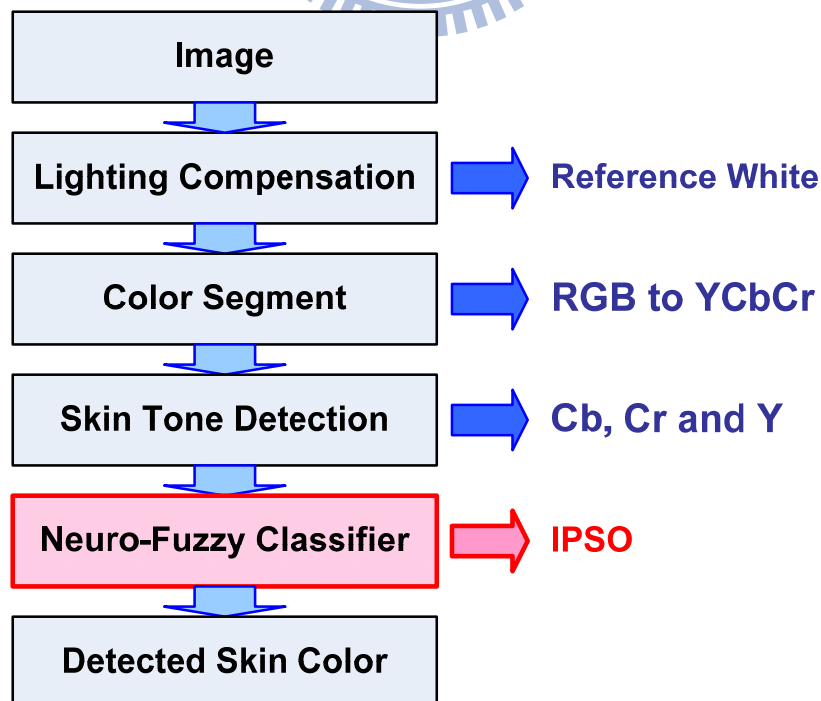


Figure 3.6: Flowchart of the skin color detection system.

In this research, we used the California Institute of Technology (CIT) facial database (on [http://www.vision.caltech.edu/Image\\_Datasets/faces/](http://www.vision.caltech.edu/Image_Datasets/faces/).) The database has 450 color images, the size of each being 320×240 pixels, and contains 27 different people and a variety of lighting, backgrounds, and facial expressions.

Three input dimensions (Y, Cb and Cr) were used in this experiment. We chose 6000 training data and 6000 testing data. We used the CIT database to produce both the training data and the testing data. We chose 3000 skin and 3000 non-skin pixels as the training data in the color images. Also, we chose other 3000 skin and 3000 non-skin pixels as the testing set. We set four rules constituting a neuro-fuzzy classifier.

The number of Ab's for a swarm was set to 100. With the same initial condition, the accuracy rate with different generations for 50 runs is shown in Figure 3.7 and tabulated in Table 3.1. It seems a good choice to terminate the training phase after 2000 generations process.

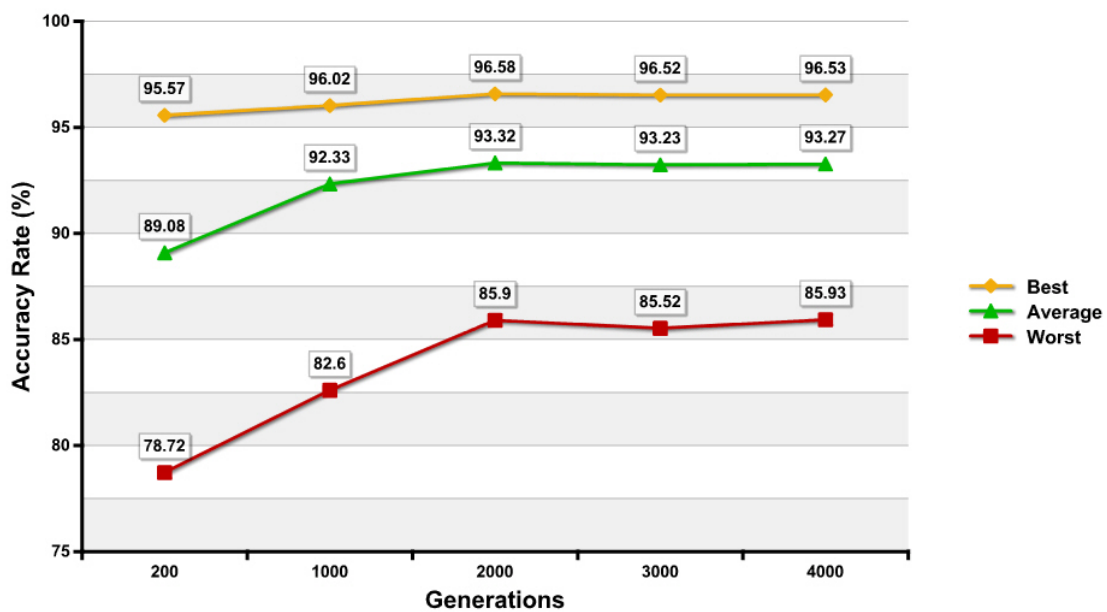


Figure 3.7: The accuracy rate with different generations.

Table 3.1: The accuracy rate with different generations (%)

	Generations				
	200	1000	<b>2000</b>	3000	4000
Best accuracy rate (training)	95.57%	96.02%	<b>96.58%</b>	96.52%	96.53%
Worst accuracy rate (training)	78.72%	82.6%	<b>85.9%</b>	85.52%	85.93%
Average accuracy rate (training)	89.08%	92.33%	<b>93.32%</b>	93.23%	93.27%

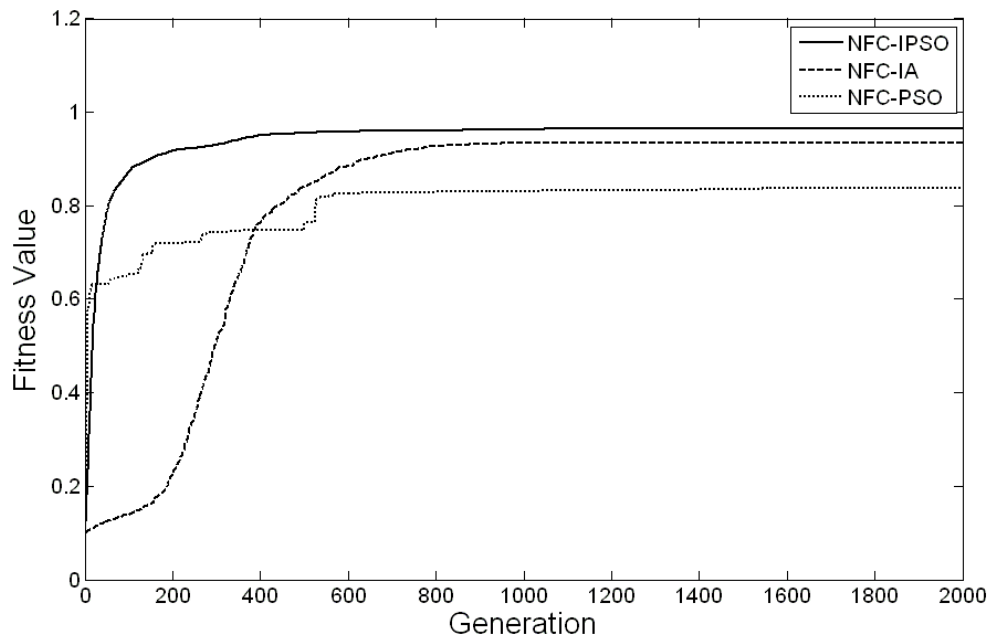


Figure 3.8: The learning curves of the three methods using the CIT database.

In this example, the performance of the IPSO method is compared with the IA method [94], and the PSO method [41]. First, the learning curves of IA, PSO and IPSO methods are shown in Figure 3.8. In Figure 3.8, we find that the performance of the proposed IPSO method is superior to the other methods. Furthermore, the comparison items include the training and testing accuracy rates are tabulated in Table 3.2.

Table 3.2: Performance comparison with various existing models from the CIT database (Training data: 6000; Generations: 2000)

Method		IPSO	IA [94]	PSO [41]
Accuracy rate (Training data)	Best	96.58%	93.5%	83.72%
	Worst	85.9%	82.53%	73.25%
	<b>Average</b>	<b>93.32%</b>	<b>88.1%</b>	<b>79.05%</b>
Accuracy rate (Testing data)	Best	95.43%	87.4%	79.77%
	Worst	82.1%	76.15%	67.3%
	<b>Average</b>	<b>90.18%</b>	<b>82.63%</b>	<b>74.32%</b>

The CIT facial database consists of complex backgrounds and diverse lighting. Hence, from the comparison data listed in Table 3.2, the average of the test accuracy rate is 74.32% for PSO, 82.63% for IA and 90.18% for the proposed IPSO. The proposed IPSO method still maintains a superior test accuracy rate. To demonstrate the skin color detection result, the color images from the CIT database are shown in Figure 3.9. A well-trained classifier can generate binary outputs (1/0 for skin/non-skin) to detect a facial region. Figure 3.10 shows that our approach accurately determines a facial region.

### 3.5 Concluding Remarks

In this chapter, the efficient immune-based particle swarm optimization (IPSO) is proposed to improve the searching ability and the converge speed. We proposed the IPSO for a neuro-fuzzy classifier to solve the skin color detection problem. The advantages of the proposed IPSO method are summarized as follows: 1) We employed the advantages of PSO to improve the mutation mechanism; 2) The experimental results show that our method is more efficient than IA and PSO in accuracy rate and convergence speed.



Figure 3.9: Original color images from CIT facial database.





Figure 3.10: Results of skin color detection with 3 dimension input (Y, Cb and Cr).

## Chapter 4

# **An Evolutionary Neural Fuzzy Classifier Using Bacterial Foraging Oriented by Particle Swarm Optimization Strategy**

Classification is one of the most important tasks for different application such as text categorization, tone recognition, image classification, micro-array gene expression, proteins structure predictions, data classification etc. There are many methods to construct classifiers, such as statistical models [100], neural networks [37][39][101], and fuzzy systems [6][16][17][102]. Most of the existing supervised classification methods are based on traditional statistics, which can provide ideal results when sample size is tending to infinity. However, only finite samples can be acquired in practice.

In this chapter, an evolutionary neural fuzzy classifier, using bacterial foraging oriented by particle swarm optimization strategy (BFPSO), is applied on different data sets which have two or multi class. The proposed BFPSO is a hybrid method which combines bacterial foraging optimization (BFO) and particle swarm optimization (PSO). The proposed algorithm performs local search through the chemotactic movement operation of BFO whereas the global search over the entire search space is accomplished by a PSO operator. In this way it balances between exploration and exploitation enjoying best of both the worlds.

## 4.1 Basic Concepts of Bacterial Foraging Optimization

Passino [103] proposed the BFO in 2002. The idea of the BFO is based on the fact that natural selection tends to eliminate animals with poor “foraging strategies” and favor the propagation of genes of those animals that have successful foraging strategies. After many generations, poor foraging strategies are either eliminated or shaped into good ones. Logically, such evolutionary principles have led scientists in the field of “foraging theory” to hypothesize that it is appropriate to model the activity of foraging as an optimization process. Take the *E. coli* bacteria (the ones that are living in our intestines) foraging strategy for instance, their foraging strategy is governed by four processes, namely, chemotaxis, swarming, reproduction, and elimination-and-dispersal.

### 4.1.1 Chemotaxis

Chemotaxis is achieved through swimming and tumbling. Depending upon the rotation of the flagella in each bacterium, it decides whether it should move in a predefined direction (swimming) or in an altogether different direction (tumbling), over the entire lifetime of the bacterium.

Let  $S$  denote the bacterial population size and  $N_c$  be the length of the lifetime of the bacteria as measured by the number of chemotactic steps they take during their life. Let  $C(i) > 0, i = 1, 2, \dots, S$  denote a basic chemotactic step size that we will use to define the lengths of steps during runs. To represent a tumble, a unit-length random direction, say  $\phi(j)$ , is generated; this will be used to define the direction of movement after a tumble. In particular, we let

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i)\phi(j) \quad (4.1)$$

where  $\theta^i(j, k, l)$  represents the location of the  $i^{\text{th}}$  bacterium at the  $j^{\text{th}}$

chemotactic step,  $k^{th}$  reproduction step, and  $l^{th}$  elimination-dispersal event.  $C(i)$  is the size of the step taken in the random direction specified by the tumble.

Then, the movement of the  $i^{th}$  bacterium at  $j^{th}$  chemotactic step, can be represented as

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i) \cdot \frac{\Delta(j)}{\sqrt{\Delta^T(j) \cdot \Delta(j)}} \quad (4.2)$$

where  $\Delta(j)$  is the direction vector of the  $j^{th}$  chemotactic step.

With the activity of run or tumble taken at each step of the chemotaxis process, a step fitness, denoted as  $J(i, j, k, l)$ , will be evaluated. If at  $\theta^i(j+1, k, l)$  the cost  $J(i, j+1, k, l)$  is better (lower) than at  $\theta^i(j, k, l)$ , then another step of size  $C(i)$  in this same direction will be taken, and again, if that step resulted in a position with a better cost value than at the previous step, another step is taken. This swim is continued as long as it continues to reduce the cost, but only up to a maximum number of steps,  $N_s$ . This represents that the cell will tend to keep moving if it is headed in the direction of increasingly favorable environments.

#### 4.1.2 Swarming

It is always desired for the bacterium that has searched out the optimum path of food should try to attract other bacteria, so that they reach the desired place more rapidly. Swarming makes the bacteria congregate into groups, and hence move as concentric patterns of groups with high bacterial density. Mathematically, swarming can be represented as

$$\begin{aligned}
J_{cc}[\theta, P(j, k, l)] &= \sum_{i=1}^S J_{cc}^i[\theta, \theta^i(j, k, l)] \\
&= \sum_{i=1}^S \left\{ -d_{attract} \exp \left[ -\omega_{attract} \sum_{m=1}^p (\theta_m - \theta_m^i)^2 \right] \right\} \\
&\quad + \sum_{i=1}^S \left\{ h_{repellant} \exp \left[ -\omega_{repellant} \sum_{m=1}^p (\theta_m - \theta_m^i)^2 \right] \right\}
\end{aligned} \tag{4.3}$$

where  $J_{cc}[\theta, P(j, k, l)]$  is the value of the cost function to be added to the actual cost function to minimize a time-varying cost function;  $S$  is the total number of bacteria;  $p$  is the number of parameters to be optimized that are present in each bacterium; and  $d_{attract}$ ,  $\omega_{attract}$ ,  $h_{repellant}$  and  $\omega_{repellant}$  are different coefficients that are to be judiciously chosen.

### 4.1.3 Reproduction

After  $N_c$  chemotactic steps, a reproduction step is taken. Let  $N_{re}$  be the number of reproduction steps to be taken. The health cost of each bacterium is calculated as the sum of the step fitness during its life, that is,  $J_{health}^i = \sum_{j=1}^{N_c+1} J(i, j, k, l)$ , where  $N_c$  is the maximum step in a chemotaxis process. For convenience, we assume that  $S = 2 \cdot S_r$  is a positive even integer. The population is sorted in order of ascending accumulated cost (higher accumulated cost represents that a bacterium did not get as many nutrients during its lifetime of foraging and hence is not as “healthy” and thus unlikely to reproduce); then the  $S_r$  least healthy bacteria die and the other  $S_r$  healthiest bacteria each split into two bacteria, which are placed at the same location. Thus, the population of bacteria keeps constant which is very convenient in coding the algorithm.

### 4.1.4 Elimination-and-Dispersal

Let  $N_{ed}$  be the number of elimination-dispersal events. The chemotaxis

provides a basis for local search, and the reproduction process speeds up the convergence which has been simulated by the classical BFO. While to a large extent, only chemotaxis and reproduction are not enough for global optima searching. Since bacteria may get stuck around the initial positions or local optima, it is possible for the diversity of BFO to change either gradually or suddenly to eliminate the accidents of being trapped into the local optima. In BFO, the dispersion event happens after a certain number of reproduction processes. Then some bacteria are chosen, according to a preset probability  $p_{ed}$ , to be killed and moved to another position within the environment.

## 4.2 Learning Algorithms for the NFS Model

BFO is based on the foraging behavior of *Escherichia Coli* (*E. Coli*) bacteria present in the human intestine and already been in use to many engineering problems, such as optimal control [104][105], and machine learning [106]. However, bacteria foraging strategies with fixed step size suffers from two main problems. If the step size is very large, then the precision becomes low, although the bacterium quickly reaches the vicinity of the optimum point. It moves around the maximum for the remaining chemotactic steps. If the step size is very small, then it takes many chemotactic steps to reach the optimum point. The rate of convergence thus decreases [107].

In PSO, a *particle* represents a potential solution which is a point in the search space. Each particle has a fitness value and a velocity to adjust its flying direction according to the best experiences of the swarm to search for the global optimum in the solution space. In Eq. (1.3), the inertia weight is used to balance the global and local

search abilities. A large inertia weight is more appropriate for global search, and a small inertia weight facilitates local search.

The proposed BFPSO algorithm, a new algorithm that combines BFO with PSO algorithm, is endowed with high convergence speed and commendable accuracy. This can be otherwise stated as the PSO performing a global search and providing a near optimal solution very quickly which is followed by a local search by BFO which fine-tunes the solution and gives an optimum solution of high accuracy. PSO has an inherent disability of trapping in the local optima but high convergence speed whereas BFO has the drawback of having a very poor convergence speed but the ability to not trap in the local optima. Figure 4.1 is the flowchart of proposed BFPSO algorithm. The brief pseudo code of the proposed BFPSO method has been provided below:



**Step 1: Initialization**

- $P$  : Dimension of the search space.
- $S$  : The number of bacteria in the population.
- $N_c$  : The number of chemotactic steps.
- $N_s$  : The number of swimming steps.
- $N_{re}$  : The number of reproduction steps.
- $N_{ed}$  : The number of elimination-dispersal events.
- $p_{ed}$  : The probability that each bacterium will be eliminated-dispersed.
- $C$  : The size of the step taken in the random direction specified by the tumble.
- $c_1$  : The cognitive learning rates.
- $c_2$  : The social learning rates.
- $w$  : The coefficient of the inertia term to control exploratory properties.

**Step 2: Elimination-dispersal loop:**  $l = l + 1$ .

**Step 3: Reproduction loop:**  $k = k + 1$ .

**Step 4: Chemotaxis loop:**  $j = j + 1$ .

[Step 4.1] For  $i = 1, 2, \dots, S$ , take a chemotactic step for bacterium  $i$  as follows.

[Step 4.2] Evaluate the cost function  $J(i, j, k, l)$ , then let  $J_{last} = J(i, j, k, l)$ .

[Step 4.3] **Tumble:** let

$$\begin{aligned} \phi^i(j+1) = & w \cdot \phi^i(j) + c_1 \cdot r_1 \cdot (\theta^{pbest}(j, k, l) - \theta^i(j, k, l)) \\ & + c_2 \cdot r_2 \cdot (\theta^{gbest}(j, k, l) - \theta^i(j, k, l)) \end{aligned}$$

[Step 4.4] **Move:** let  $\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i) \cdot \phi^i(j)$

Compute fitness function:  $J(i, j+1, k, l)$ , and then let

$$J(i, j+1, k, l) = J(i, j+1, k, l) + J_{cc}(\theta^i(j+1, k, l), P(j+1, k, l))$$

[Step 4.5] **Swim:** Let  $m = 0$ ;

while ( $m < N_s$ )

- let  $m = m + 1$ ;

if  $J(i, j+1, k, l) < J_{last}$ , let  $J_{last} = J(i, j+1, k, l)$  and let

$$\theta^i(j+1, k, l) = \theta^i(j+1, k, l) + C(i) \phi^i(j);$$

Compute fitness function:  $J(i, j+1, k, l)$ . Let

$$J(i, j+1, k, l) = J(i, j+1, k, l) + J_{cc}(\theta^i(j+1, k, l), P(j+1, k, l))$$

- else let  $m = N_s$ ;

[Step 4.6] Go to next bacterium.

**Step 5:** If ( $j < N_c$ ), go to **Step 4**. Since the life of the bacteria is not over.

**Step 6: Reproduction:** Compute the health of the bacterium  $i$ :

$$J_{health}^i = \sum_{j=1}^{N_c+1} J(i, j, k, l)$$

Sort bacteria and chemotactic parameters  $C(i)$  in order of ascending cost  $J_{health}$  (higher cost means lower health). The  $S_r$  bacteria with the highest  $J_{health}$  values die and the other  $S_r$  bacteria with the best values split (and the copies that are made are placed at the same location as their parent).

**Step 7:** If ( $k < N_{re}$ ), go to **Step 3**.

**Step 8: Elimination-dispersal:** Eliminate and disperse bacteria with probability  $p_{ed}$ .

**Step 9:** If ( $l < N_{ed}$ ), go to **Step 2**; otherwise end and output the results.



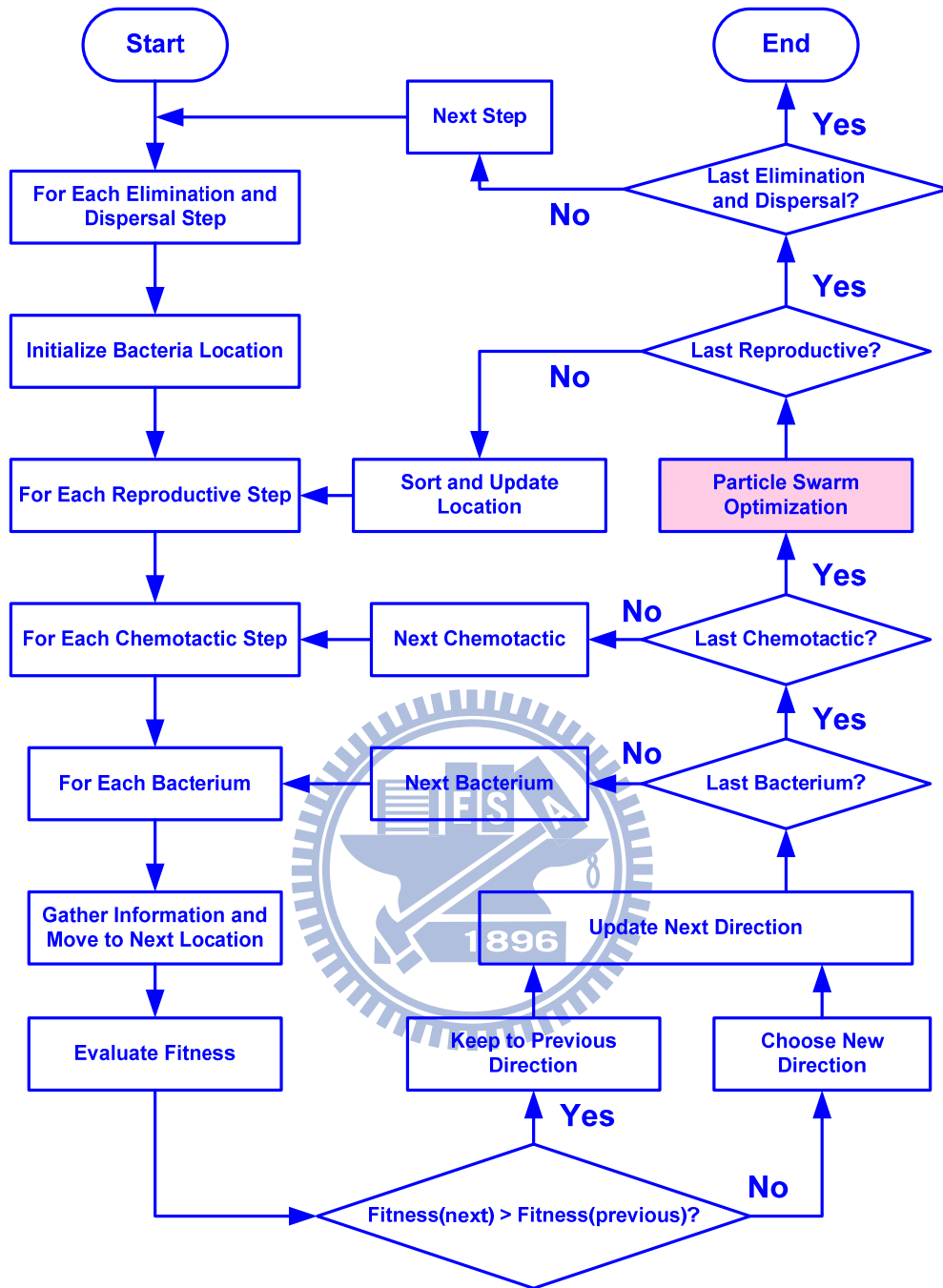


Figure 4.1: Flowchart of proposed BFPSO method.

### 4.3 Illustrative Examples

In this section, we evaluate the classification performance of the proposed NFS-BFPSO method using two better-known benchmark data sets and one skin color

detection problem. The first example uses the iris data and the second example uses the Wisconsin breast cancer data. The two benchmark data sets are available from the University of California, Irvine, via an anonymous ftp address <ftp://ftp.ics.uci.edu/pub/machine-learning-databases>. In the following simulations, the parameters and number of training epochs were based on the desired accuracy. In short, the trained NFS with BFPSO was stopped once its high learning efficiency was demonstrated.

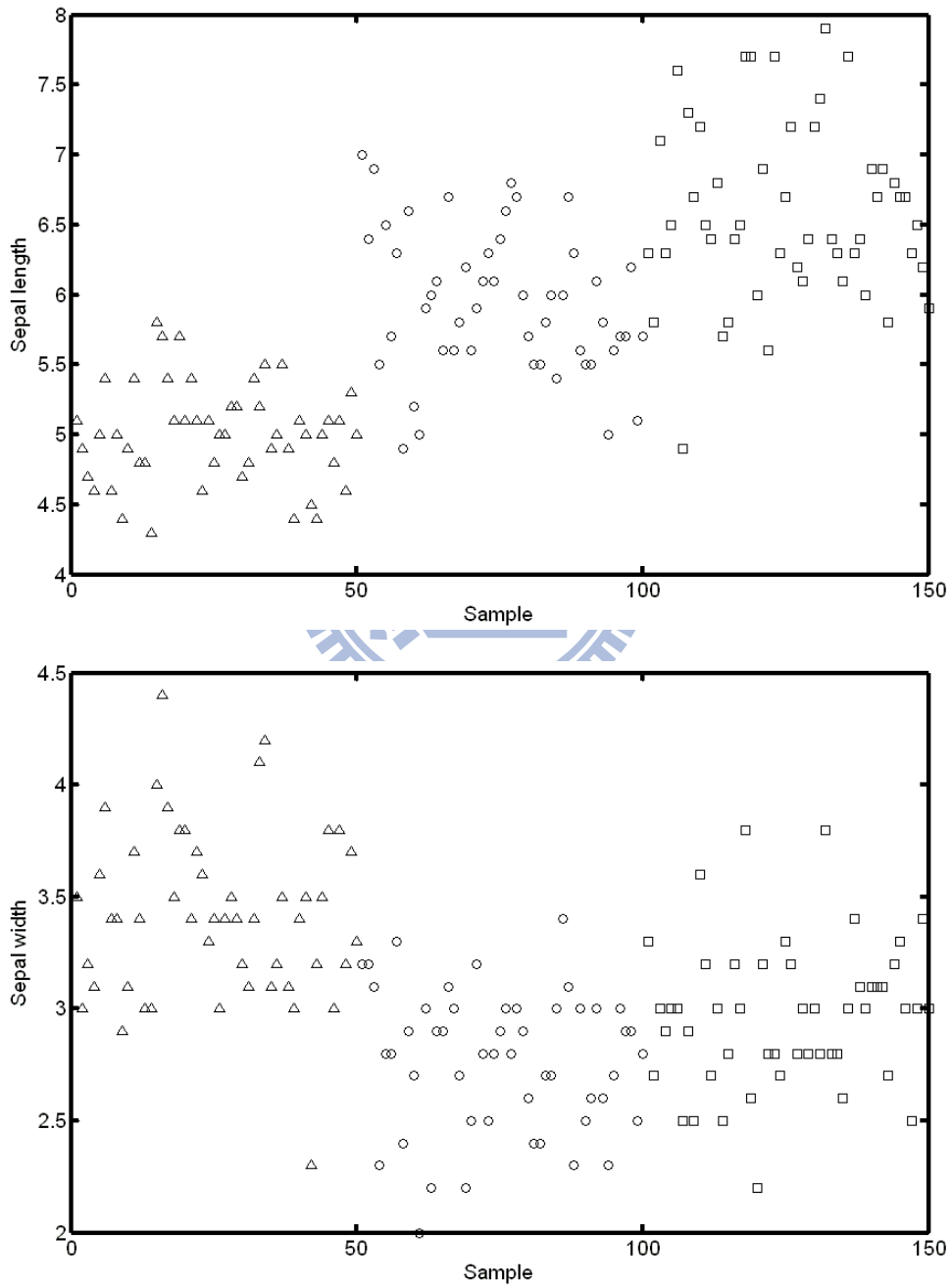
### **Example 1: Iris Data Classification**

The Fisher-Anderson iris data consists of four input measurements, sepal length ( $sl$ ), sepal width ( $sw$ ), petal length ( $pl$ ), and petal width ( $pw$ ), on 150 specimens of the iris plant. Three species of iris were involved, *Iris Sestosa*, *Iris Versicolor* and *Iris Virginica*, and each species contains 50 instances. The measurements are shown in Figure 4.2.

In the iris data experiments, 25 instances with four features from each species were randomly selected as the training set (i.e., a total of 75 training patterns were used as the training data set) and the remaining instances were used as the testing set. Once the NFS was trained, all 150 test patterns of the iris data were presented to the trained NFS, and the re-substitution error was computed. In this example, three fuzzy rules are adopted. After 4000 generations, the final fitness value was 0.9278.

Figure 4.3 (a)-(f) show the distribution of the training pattern and the final assignment of the fuzzy rules (i.e., distribution of input membership functions). Since the region covered by a Gaussian membership function is unbounded, in Figure 4.3 (a)-(f), the boundary of each ellipse represent a rule with a firing strength of 0.5. We compared the testing accuracy of our proposed method with that of other methods – the neural fuzzy system with bacterial foraging optimization (NFS-BFO) and the

neural fuzzy system with particle swarm optimization (NFS-PSO). The experiments calculated the classification accuracy and the values of the average produced on the testing set using the NFS-BFO method, the NFS-PSO method, and the proposed NFS-BFPSO method.



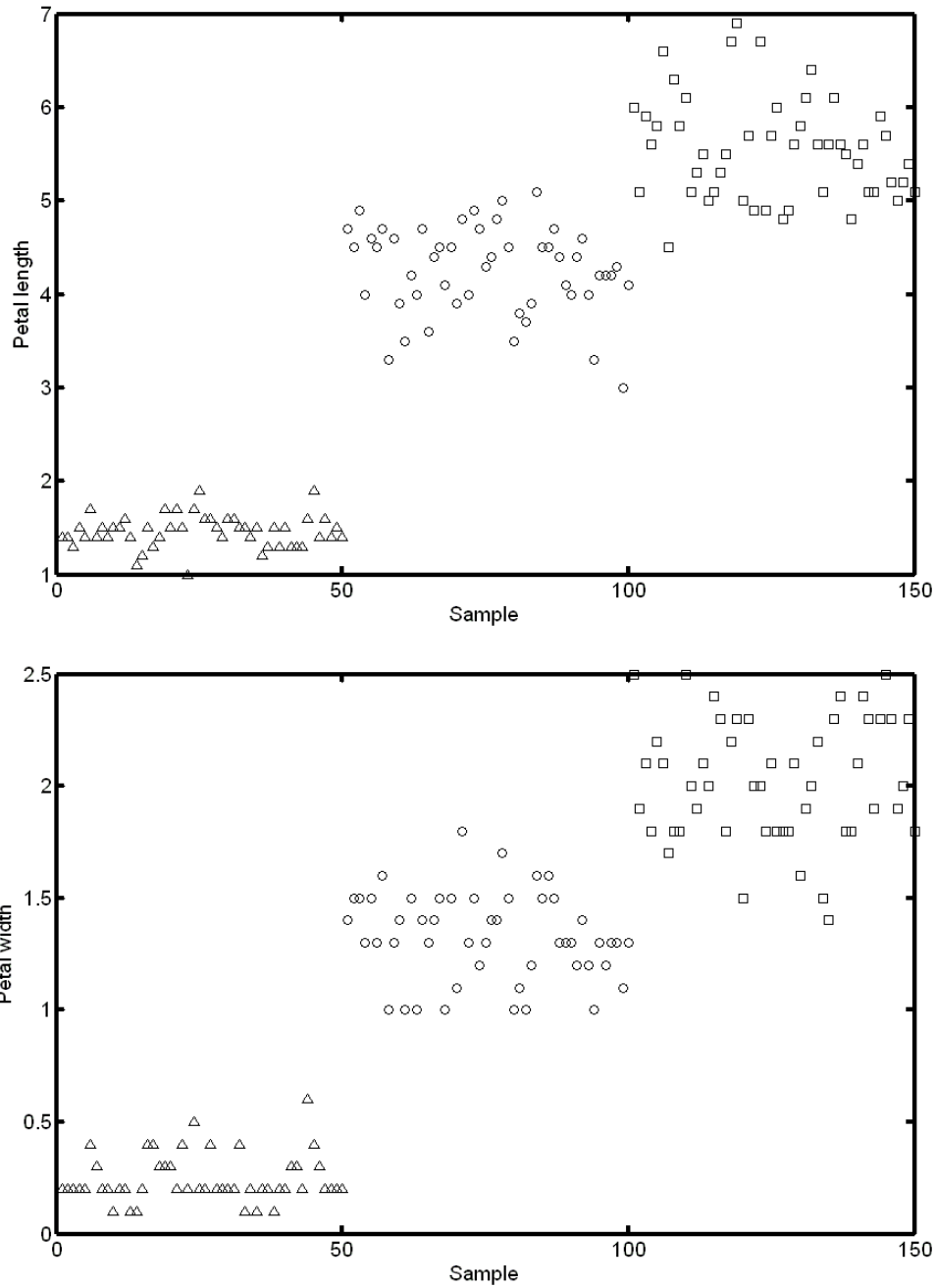
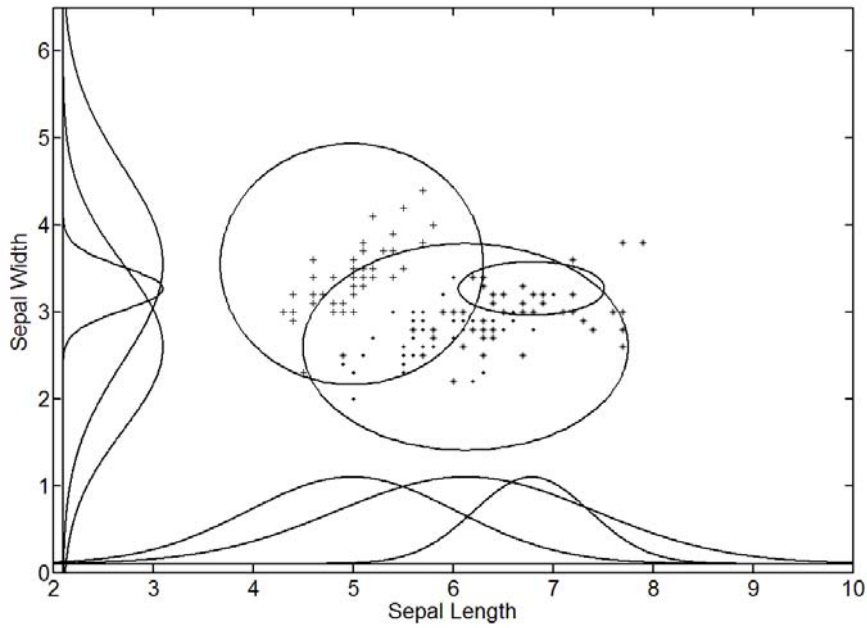


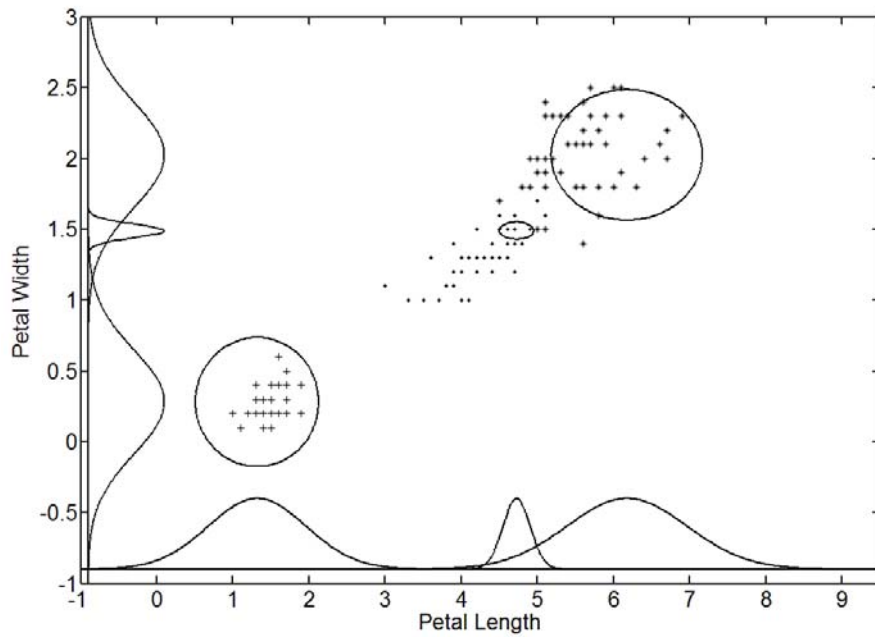
Figure 4.2: Iris data: *iris setosa* ( $\Delta$ ), *iris versicolor* ( $\circ$ ), and *iris virginica* ( $\square$ ).

During the learning phase, the learning curves from the proposed NFS-BFPSO method, the NFS-BFO method, and the NFS-PSO method are shown in Figure 4.4. Table 4.1 shows that the experiments with the NFS-BFPSO method result in high accuracy, with an accuracy percentage ranging from 96% to 98.67%. The means of re-substitution accuracy was 97.6%. The average classification accuracy of the

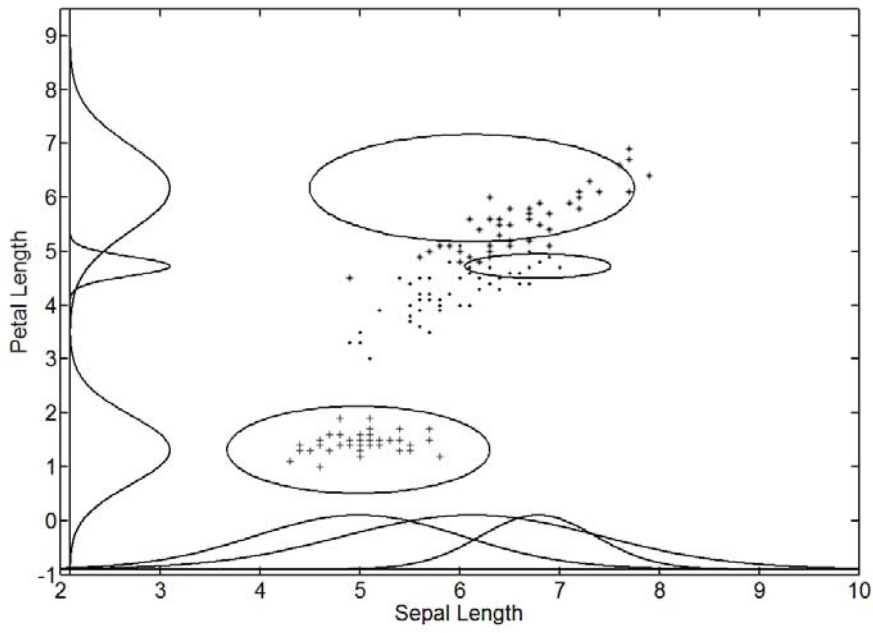
NFS-BFPSO method was better than that of other methods. Table 4.2 shows the comparison of the classification results of the NFS-BFPSO method with other methods [28][102][108-110] on the iris data. The results show that the proposed NFS-BFPSO method is able to keep similar average substitution accuracy.



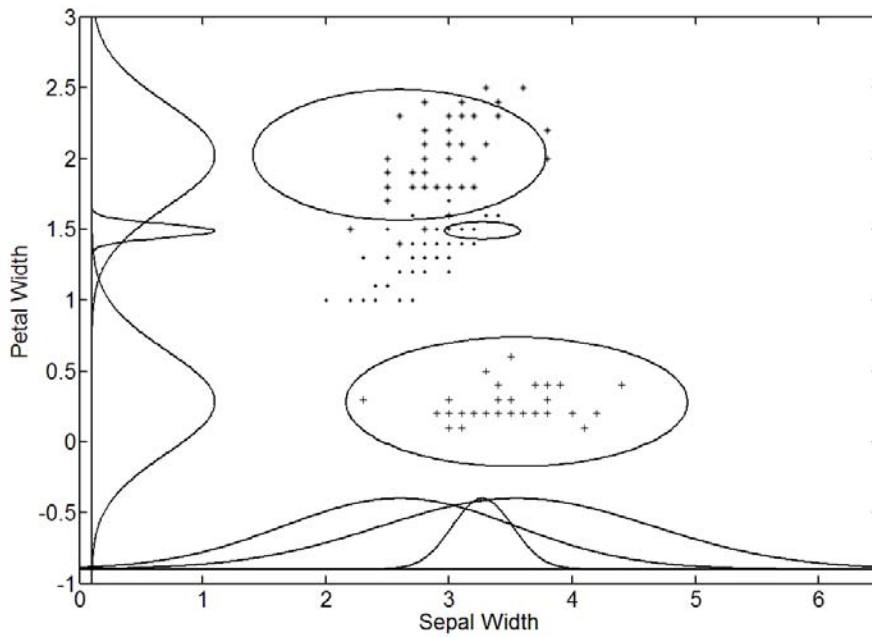
(a) For the Sepal Length and Sepal Width dimensions.



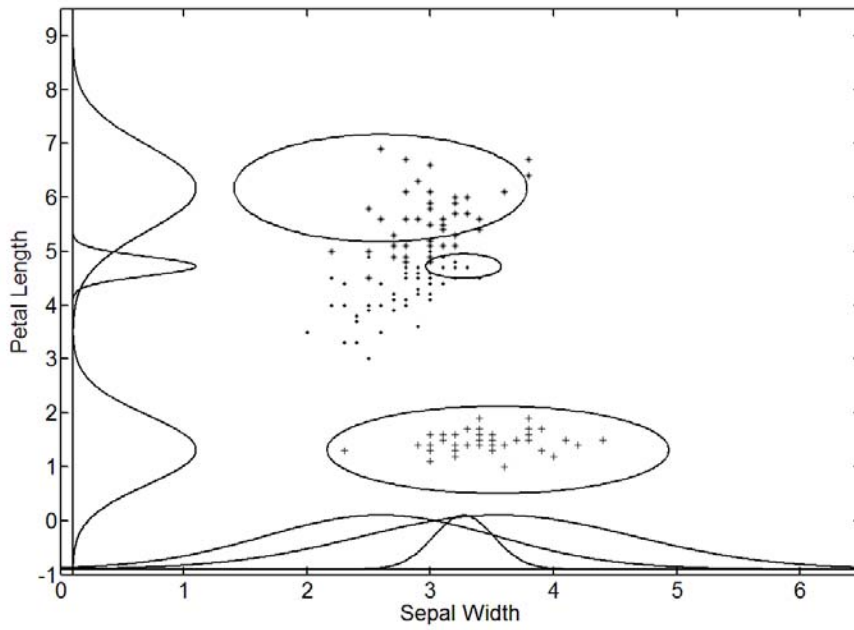
(b) For the Petal Length and Petal Width dimensions.



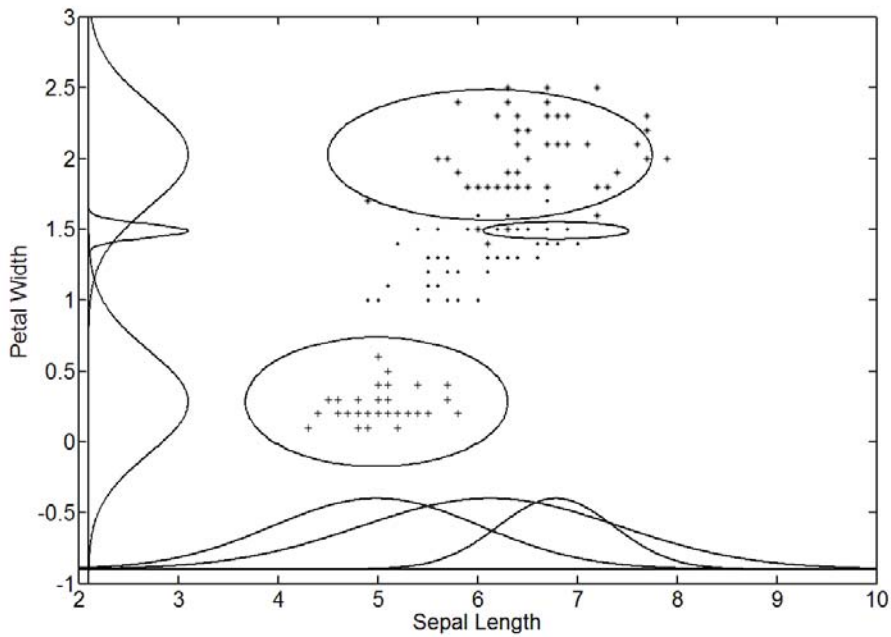
(c) For the Sepal Length and Petal Length dimensions.



(d) For the Sepal Width and Petal Width dimensions.



(e) For the Sepal Width and Petal Length dimensions.



(f) For the Sepal Length and Petal Width dimensions.

Figure 4.3: The distribution of input training patterns and final assignment of three rules.

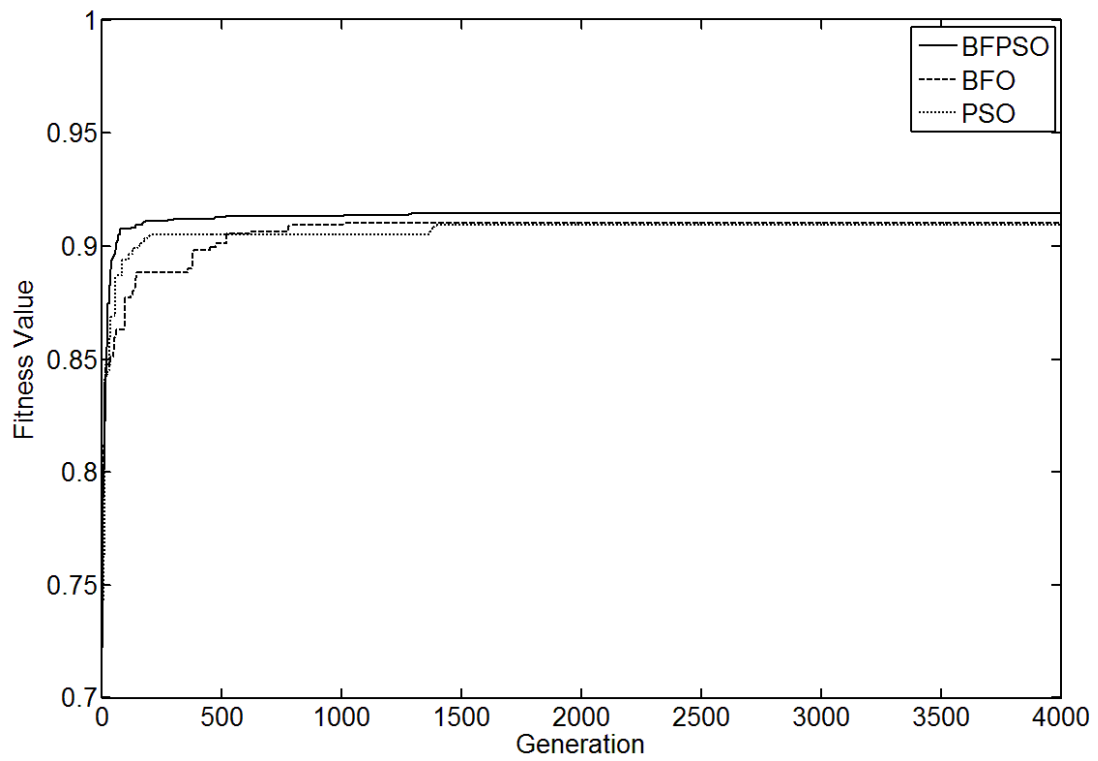


Figure 4.4: Learning curves of the NFS-BFPSO method, the NFS-BFO method, and the NFS-PSO method.

Table 4.1: Classification accuracy using various methods for the iris data.

Model \ Experiment #	NFS-BFO	NFS-PSO	<b>NFS-BFPSO</b>
1	96	98.67	<b>98.67</b>
2	92	93.33	<b>96</b>
3	97.33	94.67	<b>98.67</b>
4	97.33	98.67	<b>97.33</b>
5	94.67	94.67	<b>97.33</b>
<b>Average (%)</b>	<b>95.47</b>	<b>96</b>	<b>97.6</b>



Table 4.2: Average re-substitution accuracy comparison of various models for the iris data classification problem.

Models	Average re-substitution accuracy (%)
FEBFC [102]	96.91
SANFIS [28]	97.33
FMMC [108]	97.3
FUNLVQ+GFENCE [109]	96.3
Wu-and-Chen's [110]	96.21
<b>NFS-BFPSO</b>	<b>97.6</b>

### Example 2: Wisconsin Breast Cancer Diagnostic Data Classification

The Wisconsin breast cancer diagnostic data set contains 699 patterns distributed into two output classes, “benign” and “malignant.” Each pattern consists of nine input features: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. 458 patterns are in the benign class and the other 241 patterns are in the malignant class. Since there were 16 patterns containing missing values, we used 683 patterns to evaluate the performance of the proposed NFS-BFPSO method. To compare the performance with other models, we used half of the 683 patterns as the training set and the remaining patterns as the testing set.

Experimental conditions were the same as the previous experiment. The training patterns were randomly chosen, and the remaining patterns were used for testing. The experiments calculated the classification accuracy and the values of the average produced on the testing set by the NFS-BFO method, the NFS-PSO method, and the proposed NFS-BFPSO method.

During the supervised learning phase, 4000 epochs of training were performed. Figure 4.5 shows the membership functions for each input feature. The learning curves from the proposed NFS-BFPSO method, the NFS-BFO method, and the

NFS-PSO method are shown in Figure 4.6. The performance of the NFS-BFPSO method is better than the performance of all other models.

Table 4.3 shows that the experiments with the NFS-BFPSO method result in high accuracy, with an accuracy percentage ranging from 97.66% to 98.54%. The means of re-substitution accuracy was 97.95%. The average classification accuracy of the NFS-BFPSO method was better than that of other methods. We compared the testing accuracy of our model with that of other methods [26][28][101][102][111]. Table 4.4 shows the comparison between the learned NFS-BFPSO method and other fuzzy, neural networks, and neural fuzzy systems. The average classification accuracy of the NFS-BFPSO method is better than that of other methods.

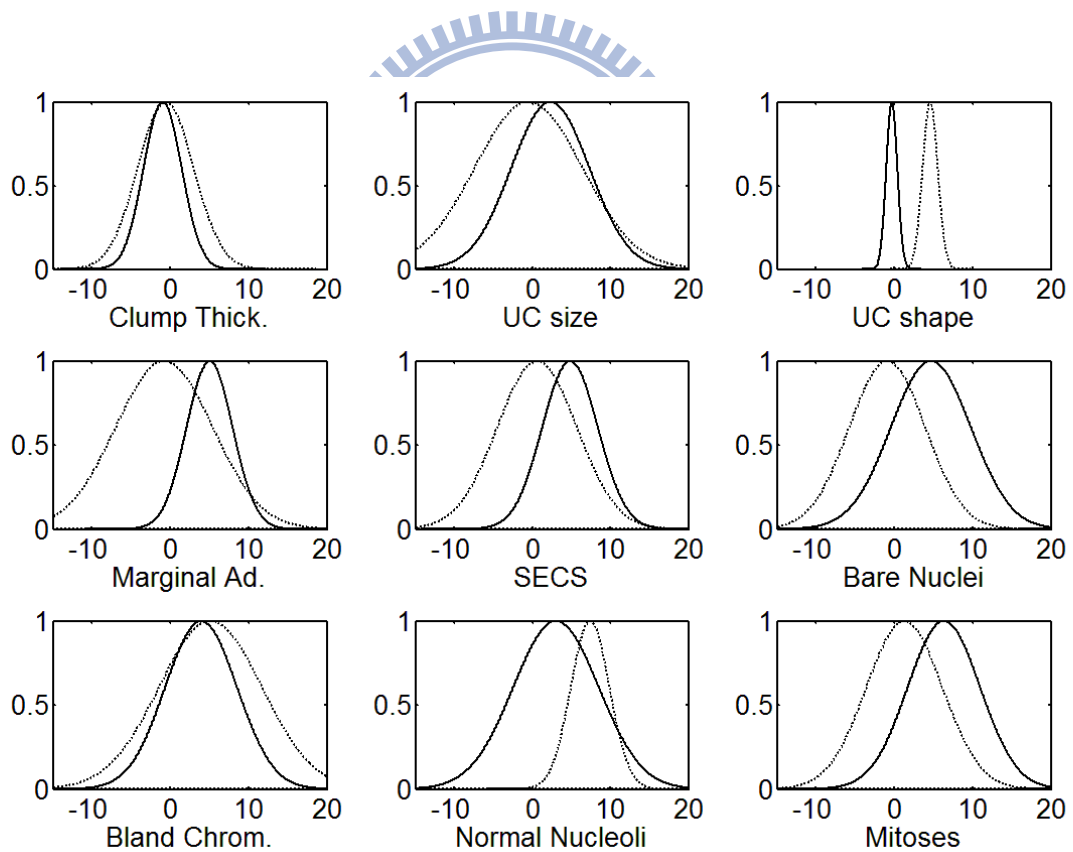


Figure 4.5: Input membership functions for breast cancer classification.

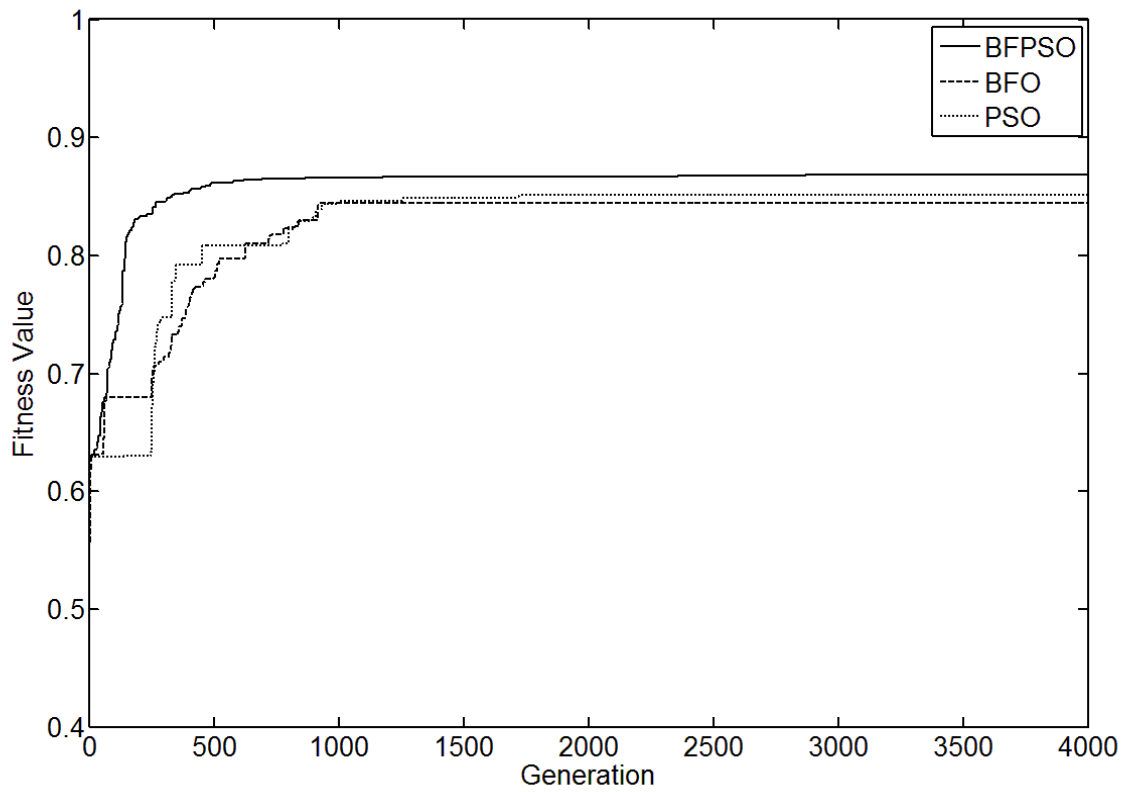


Figure 4.6: Learning curves from the NFS-BFPSO method, the NFS-BFO method and the NFS-PSO method.

Table 4.3: Classification accuracy for the Wisconsin breast cancer diagnostic data.

Model Experiment #	NFS-BFO	NFS-PSO	<b>NFS-BFPSO</b>
1	95.32	96.49	97.66
2	95.61	97.08	98.54
3	93.86	94.44	97.66
4	94.74	97.37	97.95
5	94.74	96.49	97.95
<b>Average (%)</b>	<b>94.85</b>	<b>96.37</b>	<b>97.95</b>

Table 4.4: Average accuracy comparison of various models for Wisconsin breast cancer diagnostic data.

Models	Average re-substitution accuracy (%)
NNFS [101]	94.15
FEBFC [102]	95.14
SANFIS [28]	96.3
NEFCLASS [26]	92.7
MSC [111]	94.9
<b>NFS-BFPSO</b>	<b>97.95</b>

### Example 3: Skin Color Detection

The description of the system is the same as Section 3.4. Unlike the previous chapter set four rules to constitute the neuro-fuzzy classifier, we set three fuzzy rules in this example. In addition, the parameter learning method is change to be BFPSO method.

In this example, the performance of the NFS-BFPSO method is compared with the NFS-BFO method, and the NFS-PSO method. The learning curves are shown in Figure 4.7. In Figure 4.7, we find that the performance of the proposed NFS-BFPSO method is superior to the other methods. In addition, the comparison items include the training and testing accuracy rates with various existing models are tabulated in Table 4.5.

The CIT facial database consists of complex backgrounds and diverse lighting. Hence, from the comparison data listed in Table 4.5, the average of the test accuracy rate is 82.39% for the NFS-BFO method, 83.64% for the NFS-PSO method and 85.82% for the proposed NFS-BFPSO method. This demonstrates that the CIT database is more complex and does not lead to a decrease in the accuracy rate. The proposed NFS-BFPSO method maintains a superior accuracy rate. The color images from the CIT database are shown in Figure 4.8. A well-trained network can generate

binary outputs (1/0 for skin/non-skin) to detect a facial region. Figure 4.9 shows that our model accurately determines a facial region.

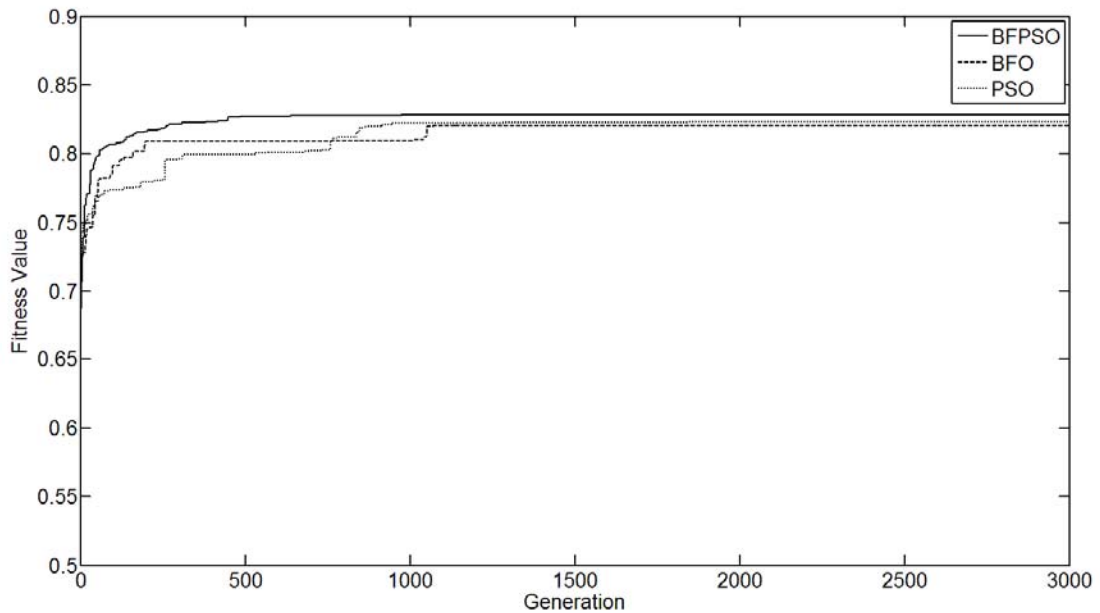


Figure 4.7: The learning curves of the three methods using the CIT database.

Table 4.5: Performance comparison with various existing models from the CIT database.

Method	NFS-BFPSO	NFS-PSO	NFS-BFO
Average training accuracy rate	<b>97.63%</b>	96.77%	96.5%
Average testing accuracy rate	<b>85.82%</b>	83.64%	82.39%





Figure 4.8: Original face images from CIT database.

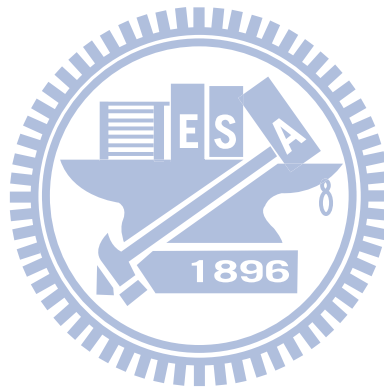


Figure 4.9: Results of skin color detection with 3 dimension input (Y, Cb, Cr).

#### 4.4 Concluding Remarks

This chapter proposes an efficient evolutionary learning method, using bacterial

foraging oriented by particle swarm optimization strategy (BFPSO), for the neural fuzzy system (NFS) in classification applications. The proposed BFPSO method attempts to make a judicious use of exploration and exploitation abilities of the search space and therefore likely to avoid false and premature convergence in many cases. The advantages of the proposed BFPSO method are summarized as follows: 1) BFPSO involves the elite-selection mechanism to gain a chance to reproduce near optimal solutions. 2) BFPSO records the best previous solution and the global best solution to evolve. 3) BFPSO can balance the exploration and exploitation abilities of the search space. Three examples showed that the proposed NFS-BFPSO method improves the system performance in terms of a fast learning convergence, and a high correct classification rate.



## **Chapter 5**

# **Nonlinear System Control Using Functional-Link-Based Neuro-Fuzzy Network Model Embedded with Modified Particle Swarm Optimizer**

Nonlinear system control is an important tool that is adopted to improve control performance and achieve robust fault-tolerant behavior. Among nonlinear control techniques, those based on artificial neural networks and fuzzy systems have become popular topics of research in recent years [112-114] because classical control theory usually requires a mathematical model to design the controller. However, the inaccuracy of the mathematical modeling of plants usually degrades the performance of the controller, especially for nonlinear and complex control problems [115]. On the contrary, both the fuzzy system controller and the artificial neural network controller provide key advantages over traditional adaptive control systems. Although traditional neural networks can learn from data and feedback, the meaning associated with each neuron and each weight in the network is not easily interpreted. Alternatively, the fuzzy logical models are easily appreciated, because they use linguistic terms and the structure of IF-THEN rules. However, fuzzy systems have a lack of an effective learning algorithm to refine the membership functions to minimize output errors. According to the literature review mentioned before, it can be said that, in contrast to pure neural or fuzzy methods, neural fuzzy networks (NFNs) systems [8-34] possess the advantages of both neural networks and fuzzy systems. NFNs bring the low-level learning and computational power of neural networks into fuzzy systems and give the



high-level human-like thinking and reasoning of fuzzy systems to neural networks.

This chapter presents a PSO-based learning algorithm for the neural fuzzy system (NFS) in nonlinear system control applications. PSO is an efficient tool for optimization and search problems. However, it is easy to become trapped in local optima due to its information sharing mechanism. Many research works have shown that mutation operators can help PSO prevent premature convergence [116-118]. To prevent basic PSO from becoming trapped in local optima, we modified the basic PSO by adding a diversity scheme, called the distance-based mutation operator, which strongly encourages a global search giving the particles more chance of converging to the global optimum. Therefore, the proposed learning algorithm is so called distance-based mutation particle swarm optimization (DMPSO).

The idea behind the proposed DMPSO learning algorithm is that there are only two kinds of convergence: 1) local optimum convergence and 2) global optimum convergence. If local optimum convergence occurred, meaning that the basic PSO is trapped in a local optimum, this is a good time to apply the mutation operator to help the PSO to escape from the local optimum. If global optimum convergence occurred, applying the mutation operator will cause the PSO to naturally converge again at the global optimum.

## **5.1 Learning Scheme for the FLNFN Model**

This section presents the learning scheme for constructing the FLNFN model. The proposed learning scheme comprises a structure learning phase and a parameter learning phase.

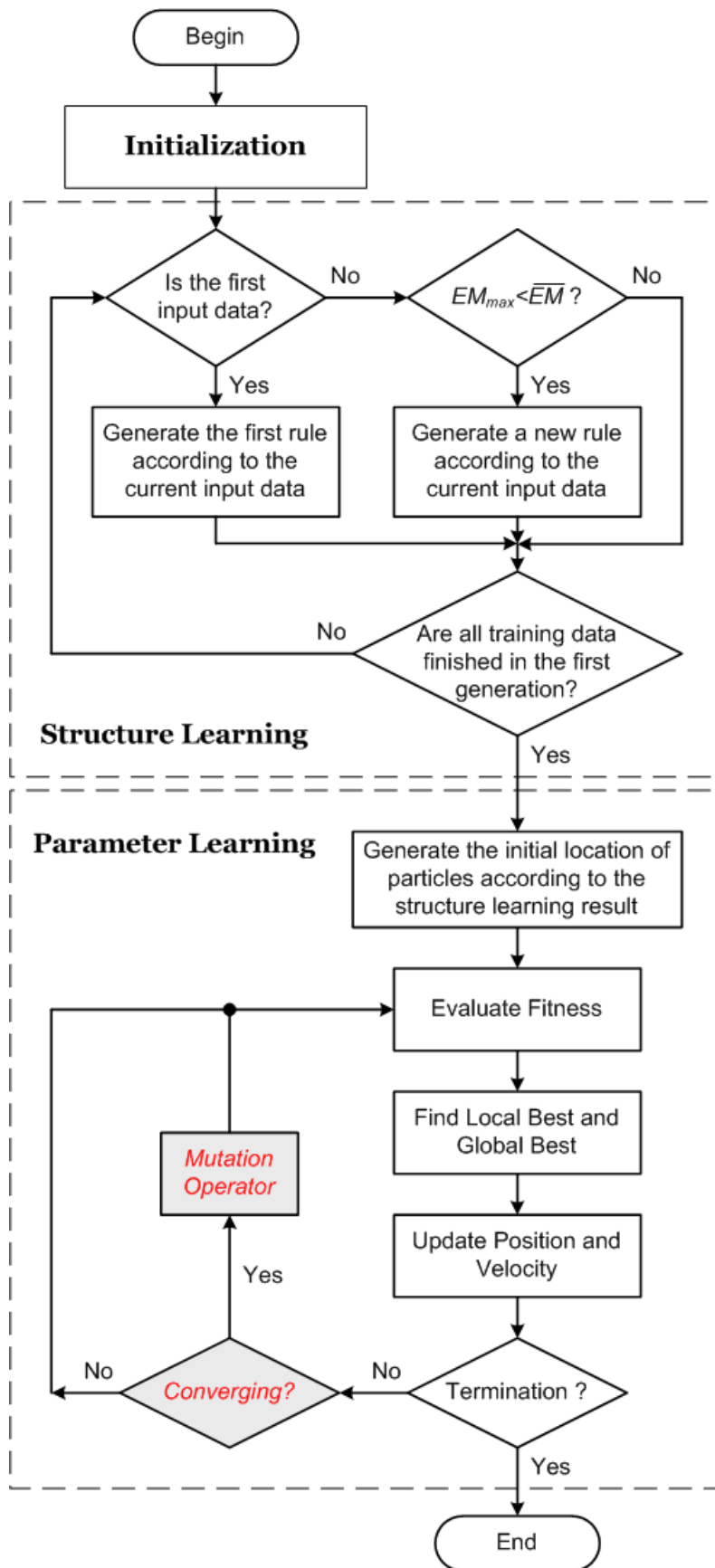
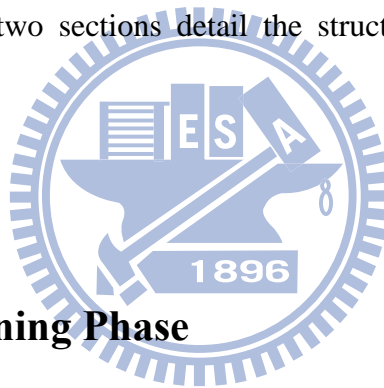


Figure 5.1: Flowchart of the proposed learning scheme for the FLNFN model.

Figure 5.1 presents flowchart of the learning scheme for the FLNFN model. Structure learning is based on the entropy measure used to determine whether a new rule should be added to satisfy the fuzzy partitioning of input variables. Parameter learning is based on the proposed evolutionary learning algorithm, which minimizes a given cost function by adjusting the link weights in the consequent part and the parameters of the membership functions. Initially, there are no nodes in the network except the input–output nodes, i.e., there are no nodes in the FLNFN model. The nodes are created automatically as learning proceeds, upon the reception of incoming training data in the structure and parameter learning processes. In this research, once the learning process is completed, the trained-FLNFN can act as the nonlinear system controller. The following two sections detail the structure learning phase and the parameter learning phase.



## 5.2 Structure Learning Phase

The foremost step in structure learning is to determine whether a new rule should be extracted from the training data and to determine the number of fuzzy sets in the universe of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, in which  $m_{ij}$  and  $\sigma_{ij}$  represent the mean and standard deviation of that cluster, respectively. For each incoming pattern  $x_i$ , the rule firing strength can be regarded as the degree to which the incoming pattern belongs to the corresponding cluster. The entropy measure between each data point and each membership function is calculated based on a similarity measure. A data point of closed mean will have lower entropy. Therefore, the entropy values between data points and current membership functions are calculated to determine

whether or not to add a new rule. For computational efficiency, the entropy measure can be calculated using the firing strength from  $u_{ij}^{(2)}$  as

$$EM_j = -\sum_{i=1}^N D_{ij} \log_2 D_{ij} \quad (5.1)$$

where  $D_{ij} = \exp(-1/u_{ij}^{(2)})$  and  $EM_j \in [0, 1]$ . According to Eq. (5.1), the measure is used to generate a new fuzzy rule and new functional link bases for new incoming data are described as follows. The maximum entropy measure

$$EM_{\max} = \max_{1 \leq j \leq R(t)} EM_j \quad (5.2)$$

is determined, where  $R(t)$  is the number of existing rules at time  $t$ . If  $EM_{\max} \leq \overline{EM}$ , then a new rule is generated, where  $\overline{EM} \in [0, 1]$  is a prespecified threshold that decays during the learning process.

In the structure learning phase, the threshold parameter  $\overline{EM}$  is an important parameter. The threshold is set between zero and one. A low threshold leads to the learning of coarse clusters (i.e., fewer rules are generated), whereas a high threshold leads to the learning of fine clusters (i.e., more rules are generated). If the threshold value equals zero, then all the training data belong to the same cluster in the input space. Therefore, the selection of the threshold value  $\overline{EM}$  will critically affect the simulation results. As a result of our extensive experiments and by carefully examining the threshold value  $\overline{EM}$ , which uses the range  $[0, 1]$ , we concluded that there was a relationship between threshold value  $\overline{EM}$  and the number of input variables ( $N$ ). Accordingly,  $\overline{EM} = \tau N$ , where  $\tau$  belongs to the range  $[0.26, 0.3]$ .

Once a new rule has been generated, the next step is to assign the initial mean and standard deviation to the new membership function and the corresponding link weight for the consequent part. Since the goal is to minimize an objective function,

the mean, standard deviation, and weight are all adjustable later in the parameter learning phase. Hence, the mean, standard deviation, and weight for the new rule are set as

$$\begin{aligned}
 m_{ij}^{(R(t+1))} &= x_i \\
 \sigma_{ij}^{(R(t+1))} &= \sigma_{init} \\
 w_{kj}^{(R(t+1))} &= \text{random}[-1, 1]
 \end{aligned} \tag{5.3}$$

where  $x_i$  is the current input data and  $\sigma_{init}$  is a prespecified constant.

After the network structure has been adjusted according to the current training data, the network enters the parameter learning phase to adjust the parameters of the network optimally based on the same training data.

### 5.3 Parameter Learning Phase

Ratnaweera *et al.* [61] stated that the lack of population diversity in PSO algorithms is understood to be a factor in their convergence on local optima. Therefore, the addition of a mutation operator to PSO should enhance its global search capacity and thus improve its performance. There are mainly two types of mutation operators: one type is based on particle position [118] and the other type is based on particle velocity [117]. The former method is the most common technique, and the mutation operator we proposed in this research is also based on particle position.

In [116], Li, Yang, and Korejo modified the PSO by adding a mutation operator; the mutation operator provides a chance to escape from local optima. They focused on determining which random generator of the mutation operator is good for improving the population. However, the timing of application of the mutation operator is the most important thing. If mutation operator is applied too early, when the particles are

not nearly convergent, the local search ability of PSO is destroyed. If the mutation operator is applied too late, the parameter learning algorithm will be very inefficient. Hence, it is an important issue to consider when to apply mutation operator. In our study, we used the distances between each particle as a measure to determine whether the mutation operator needed to be applied or not, and the modified PSO we used is the so called distance-based mutation particle swarm optimization (DMPSO). Comparing the basic PSO with DMPSO, a convergent detection unit used to detect the particle convergent status is introduced. If the particles are convergent, the mutation operator will be processed. Otherwise, the mutation operator will be skipped.

The convergent detection unit computes the average distance from every particle to the particle that has global best value using Eq. (5.4)

$$\omega(t) = \frac{\sum_{i=1}^S \|P_i^t - G_{best}^t\|}{S} \quad (5.4)$$

where  $P_i^t$  and  $G_{best}^t$  indicate the  $i^{th}$  particle and the particle that has the global best value at the  $t^{th}$  iteration, respectively, and  $S$  is the population size.

After the average distance is computed, the threshold  $Th_{conv}$  is used to determine whether the particles are close enough or not according to Eq. (5.5). If all particles are close enough, meaning that all particles are converging to the same position, the mutation operator will be applied. Otherwise, the mutation operator will be skipped.

$$\omega(t) \leq Th_{conv} \quad (5.5)$$

In this study, every particle has its own mutation probability. If the average distance is greater than  $Th_{conv}$ , implying that the majority of particles are not convergent, the mutation probability is set to zero, meaning that every particle does

not mutate and the behavior of every particle is like a generic PSO. If the average distance is less than  $Th_{conv}$ , meaning that all particles are converging to the same position, named  $G_{best}^t$ , the mutation probability ( $MP$ ) of each particle is computed by Eq. (5.6).

$$\begin{aligned}
 success_i(t) &= \begin{cases} 1, & \text{if } F(P_i^t) > F(P_{best}^{t-1}) \\ 0, & \text{otherwise} \end{cases} \\
 progress(t) &= \sum_{i=1}^S success_i(t) \\
 MP &= \exp\left(-\left(\frac{progress(t)}{S}\right)\right)
 \end{aligned} \tag{5.6}$$

where  $F(\cdot)$  denotes the fitness value of the particle. The value of  $success_i(t)$  is set to 1 only when the  $i^{th}$  particle is successfully evolved at the  $t^{th}$  iteration, meaning that the local best fitness value is improved at the  $t^{th}$  iteration, and  $progress(t)$  is the number of successful evolution particles at time step  $t$ .

The design of mutation probability is based on the ratio of improved population. If the ratio of the improved population is higher, the mutation probability becomes smaller. Most particles are moving toward to the best value that they have currently found. The lower probability guarantees the direction of the moving group will not be destroyed by the mutation operator. On the other hand, if most particles do not improve their fitness value, the population is in the stable status. There are two possibilities: the first possibility is that the particles have converged to the global optimum (or near global optimum). The application of the mutation operator at the moment will not destroy the moving group, because the particles still remember the global optimum, and the mutated particles will move toward the global optimum in the next iteration. The second possibility is that the particles have converged to the local optimum, or in other words, they have fallen into a trap. The mutation operator provides a chance to escape from the trap. If some particles mutate and the new

position the particle reaches has a better fitness value than the local optima, the other particles that are trapped will fly to the new position in the next iteration according to the PSO, meaning that the trapped particles can escape from the local optimum.

## 5.4 Illustrative Examples

In this section, we demonstrate the performance of the proposed FLNFN model using DMPSO algorithm (FLNFN-DMPSO) for nonlinear system control. The FLNFN-DMPSO is adopted to design controllers in three simulations of nonlinear system control problems: multi-input multi-output (MIMO) plant control [114], control of the truck backing system [119], and a water bath temperature control system [120].

### Example 1: Multi-Input Multi-Output Plant Control

In this example, the MIMO plants [114] to be controlled are described by the equations

$$\begin{bmatrix} y_{p1}(k+1) \\ y_{p2}(k+1) \end{bmatrix} = \begin{bmatrix} 0.5 \frac{y_{p1}(k)}{1+y_{p2}^2(k)} \\ 0.5 \frac{y_{p1}(k)y_{p2}(k)}{1+y_{p2}^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \quad (5.7)$$

The controlled outputs should follow the desired output  $y_{r1}$  and  $y_{r2}$ , as specified by the following 250 pieces of data;

$$\begin{bmatrix} y_{r1}(k) \\ y_{r2}(k) \end{bmatrix} = \begin{bmatrix} \sin(k\pi/45) \\ \cos(k\pi/45) \end{bmatrix} \quad (5.8)$$

The inputs of the FLNFN-DMPSO are  $y_{p1}(k)$ ,  $y_{p2}(k)$ ,  $y_{r1}(k)$ , and  $y_{r2}(k)$ , and the outputs are  $u_1(k)$  and  $u_2(k)$ .



Figure 5.2 plots the learning curves of the best performance of the FLNFN-DMPSO model for the affinity/fitness value, the CNFC-ISEL [121], the SEFC [122], and the Mamdani-type fuzzy system using symbiotic evolution algorithm (MFS-SE) [123], after the learning process of 600 generations. To demonstrate the performance of the proposed controller, Figure 5.3 plots the control results of the desired output (solid line) and the model output (dotted line) after the learning process of 600 generations, and Figure 5.4 shows the errors of the proposed method. Table 5.1 presents the best and averaged affinity/fitness values after 600 generations of training. The comparison indicates that the best and averaged affinity/fitness values of FLNFN-DMPSO are better than those of other methods.

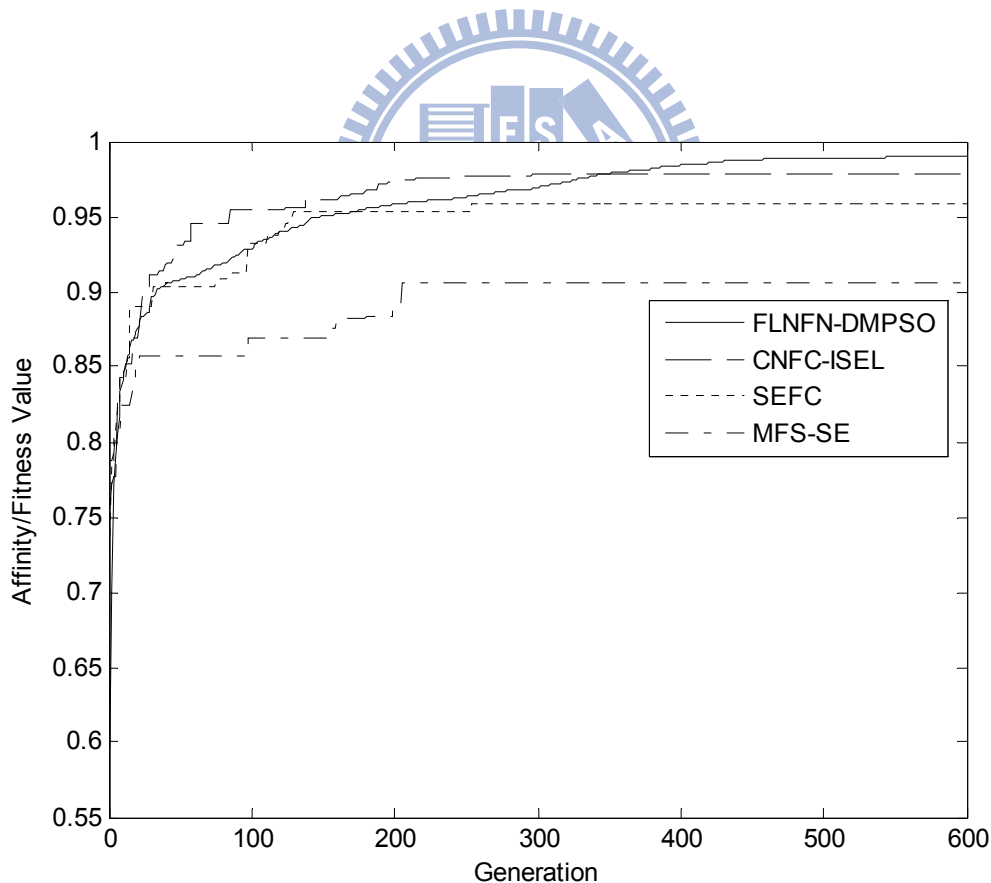


Figure 5.2: Learning curves of best performance of the FLNFN-DMPSO, CNFC-ISEL, SEFC and MFS-SE in MIMO plant control.

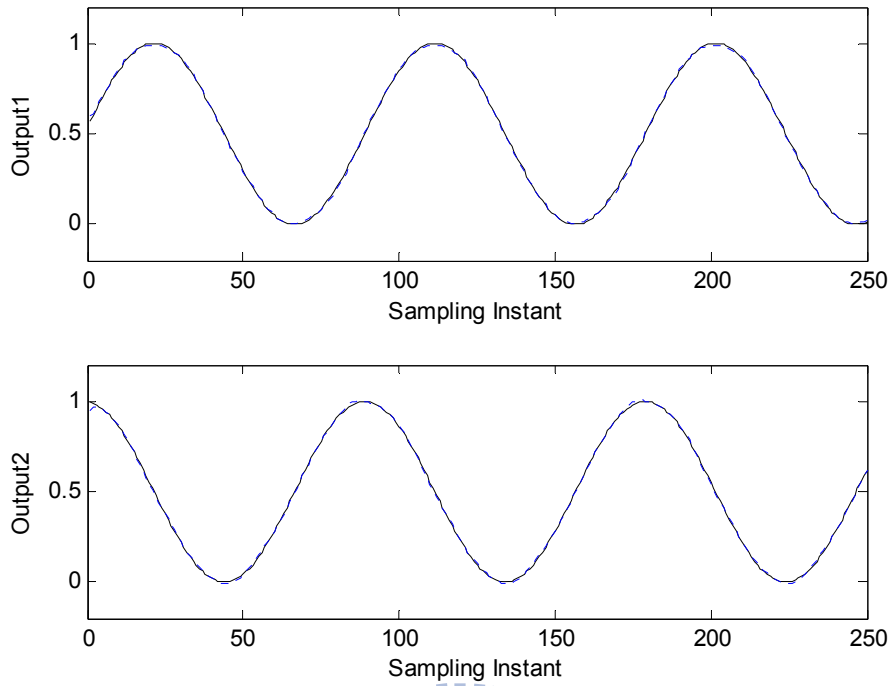


Figure 5.3: Desired (solid line) and model (dotted line) output generated by FLNFN-DMPSO in MIMO plant control.

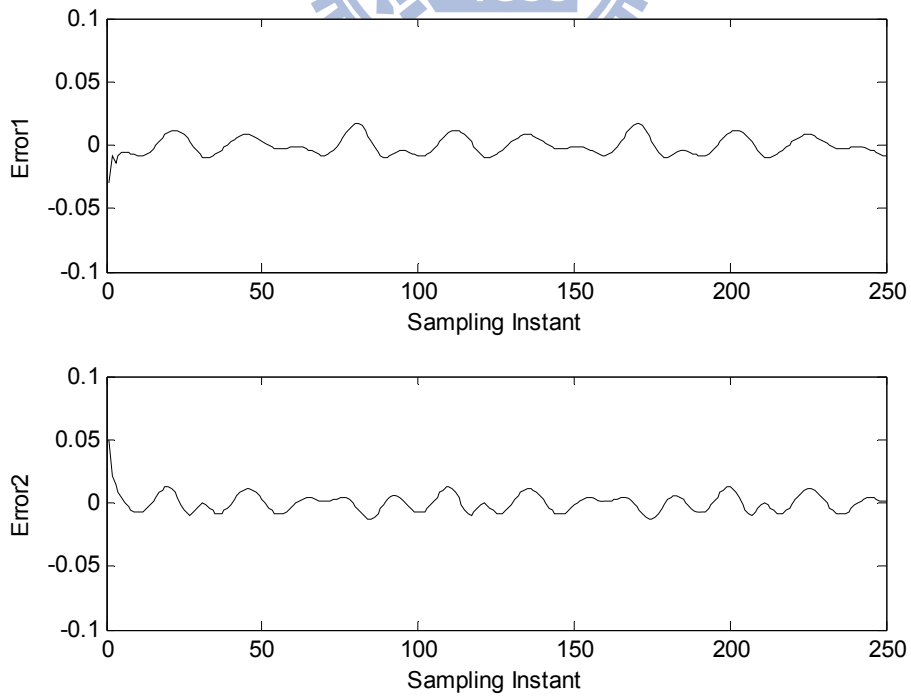


Figure 5.4: Errors of proposed FLNFN-DMPSO in MIMO plant control.

Table 5.1: Performance comparison of the FLNFN-DMPSO, FLNFN-PSO, CNFC-ISEL, SEFC and MFS-SE controllers for the MIMO plant.

Method	Affinity/Fitness Value (Best)	Affinity/Fitness Value (Avg.)
<b>FLNFN-DMPSO</b>	<b>0.9898</b>	<b>0.9856</b>
FLNFN-PSO	0.9506	0.9149
CNFC-ISEL [121]	0.9786	0.9721
SEFC [122]	0.9581	0.9553
MFS-SE [123]	0.8560	0.8503

### Example 2: Control of Backing Up the Truck

Backing a truck into a loading dock is difficult. It is a nonlinear control problem for which no traditional control method exists [119]. Figure 5.5 shows the simulated truck and loading zone. The truck's position is exactly determined by three state variables  $\phi$ ,  $x$  and  $y$ , where  $\phi$  is the angle between the truck and the horizontal, and the coordinate pair  $(x, y)$  specifies the position of the center of the rear of the truck in the plane. The steering angle  $\theta$  of the truck is the controlled variable. Positive values of  $\theta$  represent clockwise rotations of the steering wheel and negative values represent counterclockwise rotations. The truck is placed at some initial position and is backed up while being steered by the controller. The objective of this control problem is to use backward only motion of the truck to make it arrive at the desired loading dock  $(x_{desired}, y_{desired})$  at a right angle ( $\phi_{desired} = 90^\circ$ ). The truck moves backward as the steering wheel moves through a fixed distance ( $d_f$ ) in each step. The loading region is limited to the plane  $[0 \ 100] \times [0 \ 100]$ .

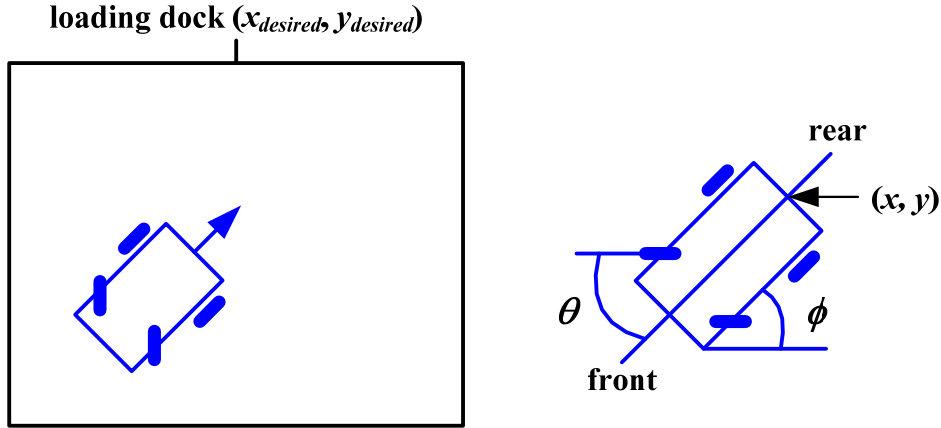


Figure 5.5: Diagram of simulated truck and loading zone.

The input and output variables of the FLNFN-DMPSO controller must be specified. The controller has two inputs: truck angle  $\phi$  and cross position  $x$ . When the clearance between the truck and the loading dock is assumed to be sufficient, the  $y$  coordinate is not considered to be an input variable. The output of the controller is the steering angle  $\theta$ . The ranges of the variables  $x$ ,  $\phi$ , and  $\theta$  are as follows:

$$\begin{aligned} 0 &\leq x \leq 100 \\ -90^\circ &\leq \phi \leq 270^\circ \\ -30^\circ &\leq \theta \leq 30^\circ \end{aligned} \quad (5.9)$$

The equations of backward motion of the truck are

$$\begin{aligned} x(k+1) &= x(k) + d_f \cos \theta(k) + \cos \phi(k) \\ y(k+1) &= y(k) + d_f \cos \theta(k) + \sin \phi(k) \\ \phi(k+1) &= \tan^{-1} \left[ \frac{l \sin \phi(k) + d_f \cos \phi(k) \sin \theta(k)}{l \cos \phi(k) - d_f \sin \phi(k) \sin \theta(k)} \right] \end{aligned} \quad (5.10)$$

where  $l$  is the length of the truck. Equation (5.10) yields the next state from the present state.

Learning involves several attempts, each starting from an initial state and terminating when the desired state is reached; the FLNFN-DMPSO is thus trained. The training process continues for 2000 generations. The affinity/fitness value of the

FLNFN-DMPSO is approximately 0.9637, and the learning curve of FLNFN-DMPSO is compared with those obtained using various existing models [121-123], as shown in Figure 5.6. Figure 5.7 plots the trajectories of the moving truck controlled by the FLNFN-DMPSO, starting at initial positions  $(x, y, \phi) = (40, 20, -30^\circ)$ ,  $(10, 20, 150^\circ)$ ,  $(70, 20, -30^\circ)$  and  $(80, 20, 150^\circ)$ , after the training process has been terminated. The considered performance indices include the best affinity/fitness and the average affinity/fitness value. Table 5.2 compares the results. According to these results, the proposed FLNFN-DMPSO outperforms various existing models.

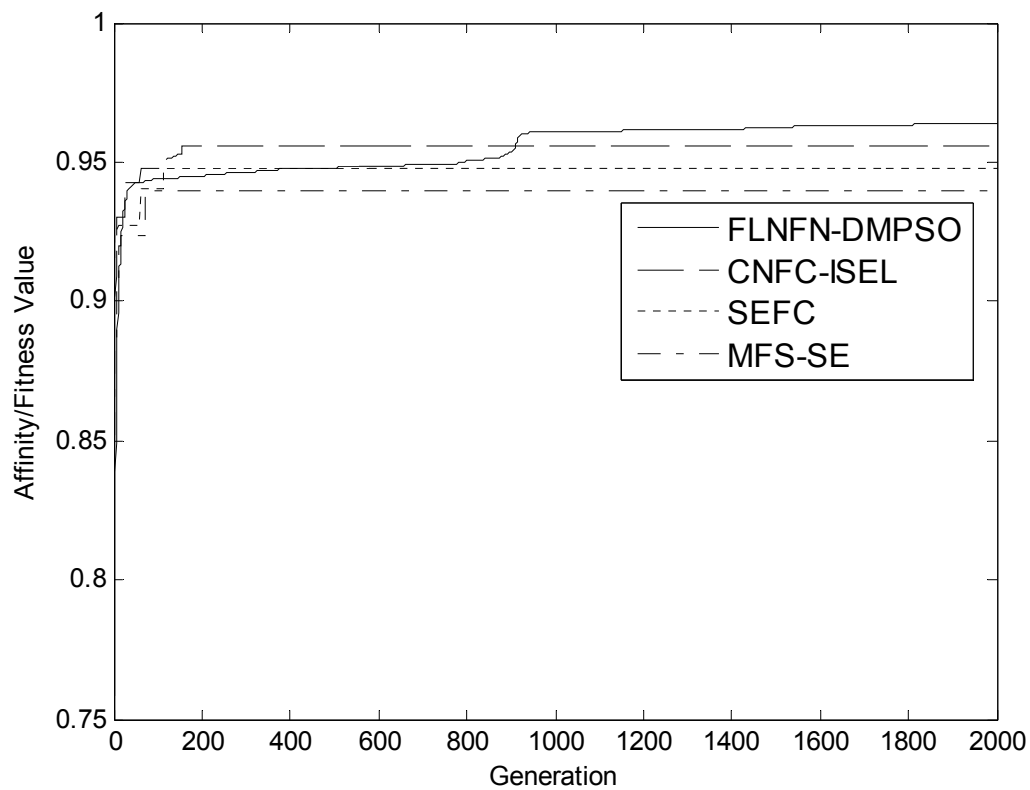
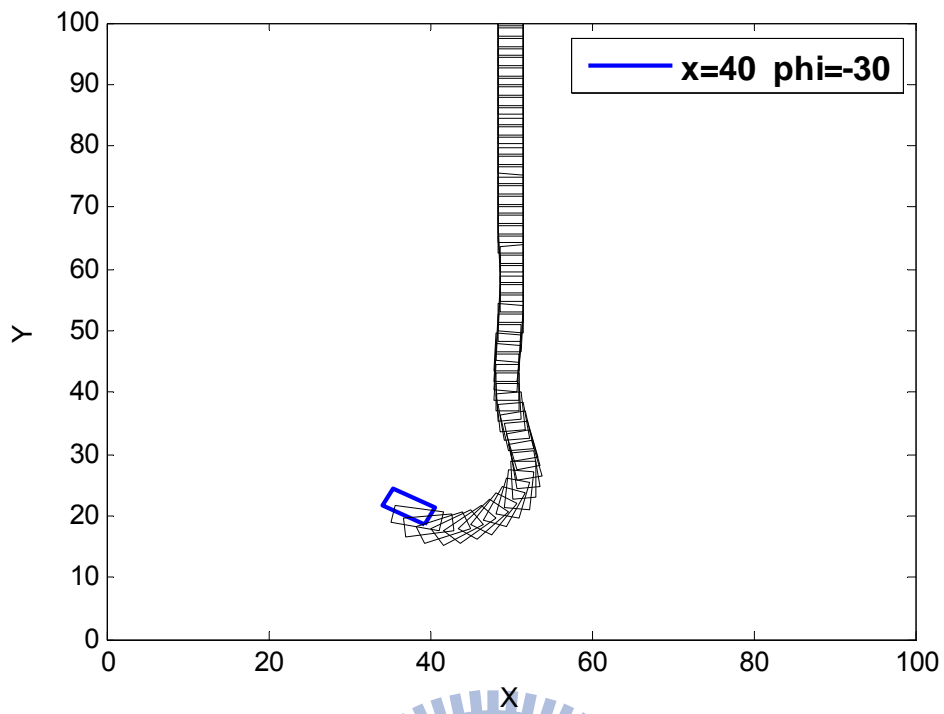
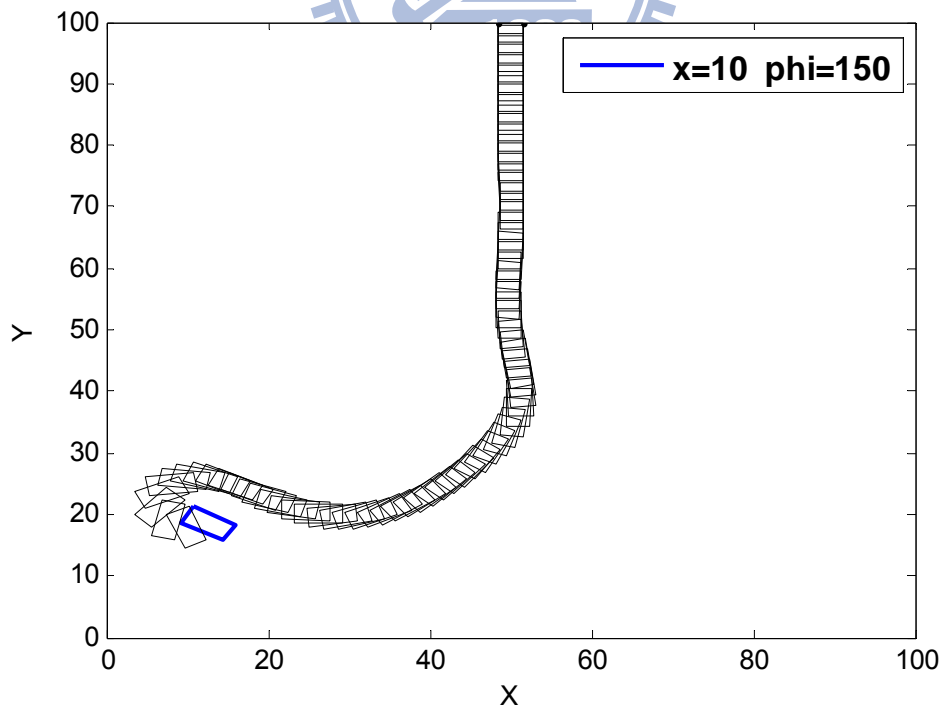


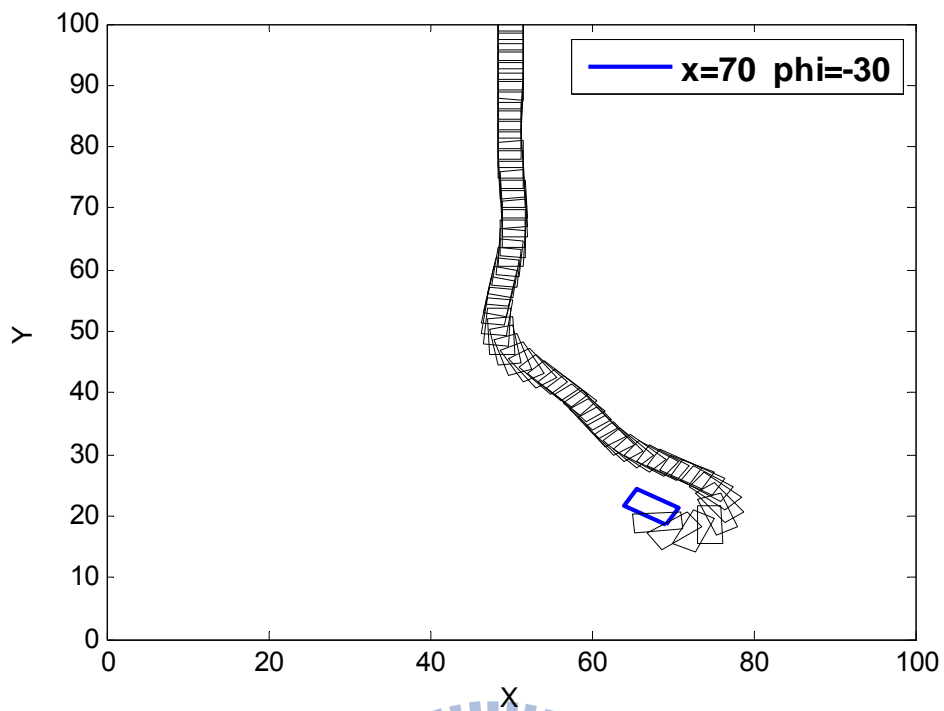
Figure 5.6: Learning curves of best performance of the FLNFN-DMPSO, CNFC-ISEL, SEFC and MFS-SE in control of backing up the truck.



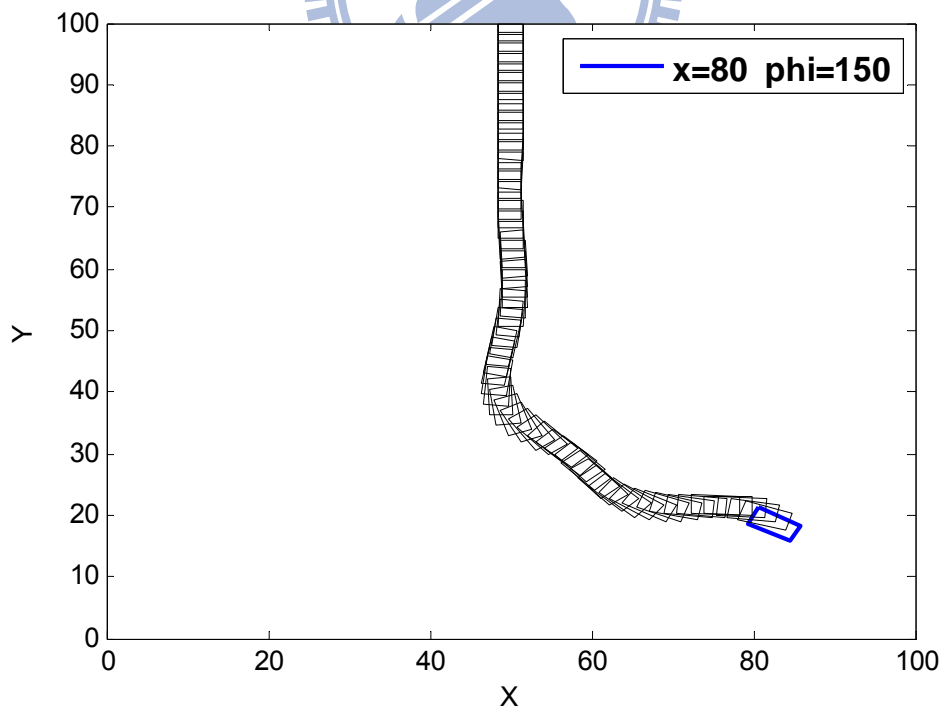
(a) initial positions  $(x, y, \phi) = (40, 20, -30^\circ)$



(b) initial positions  $(x, y, \phi) = (10, 20, 150^\circ)$



(c) initial positions  $(x, y, \phi) = (70, 20, -30^\circ)$



(d) initial positions  $(x, y, \phi) = (80, 20, 150^\circ)$

Figure 5.7: Trajectories of truck, starting at four initial positions under the control of the FLNFN-DMPSO after learning using training trajectories.

Table 5.2: Performance comparison of various controllers to control of backing up the truck.

Method	Affinity/Fitness Value (Best)	Affinity/Fitness Value (Avg.)
<b>FLNFN-DMPSO</b>	<b>0.9637</b>	<b>0.9502</b>
FLNFN-PSO	0.9423	0.9355
CNFC-ISEL [121]	0.9558	0.9511
SEFC [122]	0.9516	0.9451
MFS-SE [123]	0.9398	0.9332

### Example 3: Control of Water Bath Temperature System

The goal of this example is to elucidate the control of the temperature of a water bath system according to

$$\frac{dy(t)}{dt} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{T_R C} \quad (5.11)$$

where  $y(t)$  is the output temperature of the system in degrees Celsius ( $^{\circ}C$ );  $u(t)$  is the heat flowing into the system;  $Y_0$  is the room temperature;  $C$  is the equivalent thermal capacity of the system and  $T_R$  is the equivalent thermal resistance between the borders of the system and the surroundings.

$T_R$  and  $C$  are assumed to be essentially constant, and the system in Eq. (5.11) is rewritten in discrete-time form to some reasonable approximation. The system

$$y(k+1) = e^{-\alpha T_s} y(k) + \frac{\delta/\alpha(1-e^{-\alpha T_s})}{1+e^{0.5y(k)-40}} u(k) + [1-e^{-\alpha T_s}] y_0 \quad (5.12)$$

is obtained, where  $\alpha$  and  $\delta$  are some constant values of  $T_R$  and  $C$ . The system parameters used in this example are  $\alpha = 1.00151 \times 10^{-4}$ ,  $\delta = 8.67973 \times 10^{-3}$  and  $y_0 = 25.0$  ( $^{\circ}C$ ), which were obtained from a real water bath plant considered elsewhere [120]. The plant input  $u(k)$  is limited between 0V and 5V where V represents the voltage unit. The sampling period is  $T_s = 30$  second.

Figure 5.8 presents a block diagram for the conventional training scheme. This block diagram has two phases – the training phase and the control phase. In the



training phase, the switches S1 and S2 are connected to nodes 1 and 2, respectively, to form a training loop. In this loop, training data with input vector  $I(k)=[y_p(k+1) \ y_p(k)]$  and desired output  $u(k)$  can be defined, where the input vector of the FLNFN controller is the same as that used in the general inverse modeling [124] training scheme. In the control phase, the switches S1 and S2 are connected to nodes 3 and 4, respectively, forming a control loop. In this loop, the control signal  $\hat{u}(k)$  is generated according to the input vectors  $I'(k)=[y_{ref}(k+1) \ y_p(k)]$ , where  $y_p$  is the plant output and  $y_{ref}$  is the reference model output.

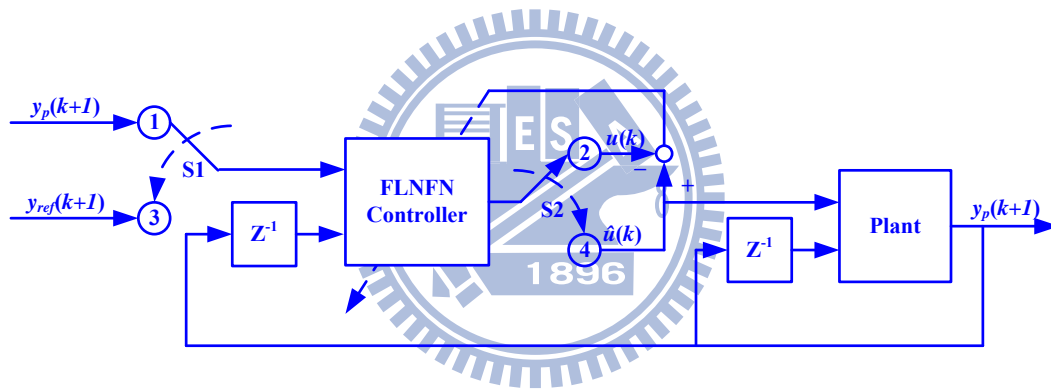


Figure 5.8: Conventional training scheme.

A sequence of random input signals  $u_{rd}(k)$  limited between 0V and 5V is injected directly into the simulated system described in Eq. (5.12), using the training scheme for the FLNFN-DMPSO controller. The 120 training patterns are selected based on the input–outputs characteristics to cover the entire reference output. The temperature of the water is initially 25 °C, and rises progressively when random input signals are injected.

This dissertation compares the FLNFN-DMPSO controller to the FLNFN controller [32], the proportional-integral-derivative (PID) controller [125], the

manually designed fuzzy controller [8], the FLNN [80], and the TSK-type neuro-fuzzy network (TSK-type NFN) [24]. Each of these controllers is applied to the water bath temperature control system. The performance measures include the set points regulation, the influence of impulse noise, large parameter variations in the system and the tracking capability of the controllers.

The first task is to control the simulated system to follow three set points

$$y_{\text{ref}}(k) = \begin{cases} 35^{\circ}\text{C}, & \text{for } k \leq 40 \\ 55^{\circ}\text{C}, & \text{for } 40 < k \leq 80 \\ 75^{\circ}\text{C}, & \text{for } 80 < k \leq 120 \end{cases} \quad (5.13)$$

Figure 5.9 presents the regulation performance of the FLNFN-DMPSO controller. The regulation performance was also tested using the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller and the TSK-type NFN controller. To test their regulation performance, a performance index, the sum of absolute error (SAE), is defined by

$$\text{SAE} = \sum_k |y_{\text{ref}}(k) - y(k)| \quad (5.14)$$

where  $y_{\text{ref}}(k)$  and  $y(k)$  are the reference output and the actual output of the simulated system, respectively. The SAE values of the FLNFN-DMPSO, the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller and the TKS-type NFN controller are 352.32, 352.84, 418.5, 401.5, 379.22 and 361.96, which values are given in the second column of Table 5.3. The proposed FLNFN-DMPSO controller has a much better SAE value of regulation performance than the other controllers.

The second set of simulations is performed to elucidate the noise rejection ability of the six controllers when some unknown impulse noise is imposed on the process. One impulse noise value of  $-5^{\circ}\text{C}$  is added to the plant output at the 60<sup>th</sup> sampling instant. A set point of  $50^{\circ}\text{C}$  is adopted in this set of simulations. For the

FLNFN-DMPSO controller, the same training scheme, training data and learning parameters were used as in the first set of simulations. Figure 5.10 presents the behaviors of the FLNFN-DMPSO controller under the influence of impulse noise. The SAE values of the FLNFN-DMPSO controller, the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller and the TSK-type NFN controller are 270.29, 270.41, 311.5, 275.8, 324.51 and 274.75, which values are shown in the third column of Table 5.3. The FLNFN-DMPSO controller performs quite well. It recovers very quickly and steadily after the occurrence of the impulse noise.

One common characteristic of many industrial control processes is that their parameters tend to change in an unpredictable way. The value of  $0.7u(k-2)$  is added to the plant input after the 60<sup>th</sup> sample in the third set of simulations to test the robustness of the six controllers. A set point of  $50^{\circ}\text{C}$  is adopted in this set of simulations. Figure 5.11 presents the behaviors of the FLNFN-DMPSO controller when the plant dynamics change. The SAE values of the FLNFN-DMPSO controller, the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller and the TSK-type NFN controller are 262.91, 263.35, 322.2, 273.5, 311.54 and 265.48, which values are shown in the fourth column of Table 5.3. The results present the favorable control and disturbance rejection capabilities of the trained FLNFN-DMPSO controller in the water bath system.

In the final set of simulations, the tracking capability of the FLNFN-DMPSO controller with respect to ramp-reference signals is studied. Define

$$y_{\text{ref}}(k) = \begin{cases} 34^{\circ}\text{C}, & \text{for } k \leq 30 \\ (34 + 0.5(k - 30))^{\circ}\text{C}, & \text{for } 30 < k \leq 50 \\ (44 + 0.8(k - 50))^{\circ}\text{C}, & \text{for } 50 < k \leq 70 \\ (60 + 0.5(k - 70))^{\circ}\text{C}, & \text{for } 70 < k \leq 90 \\ 70^{\circ}\text{C}, & \text{for } 90 < k \leq 120 \end{cases} \quad (5.15)$$

Figure 5.12 presents the tracking performance of the FLNFN-DMPSO controller.

The SAE values of the FLNFN-DMPSO controller, the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller and the TSK-type NFN controller are 42.45, 44.28, 100.6, 88.1, 98.43 and 54.28, which values are shown in the fifth column of Table 5.3. The results present the favorable control and tracking capabilities of the trained FLNFN-DMPSO controller in the water bath system. The aforementioned simulation results, presented in Table 5.3, demonstrate that the proposed FLNFN-DMPSO controller outperforms other controllers.

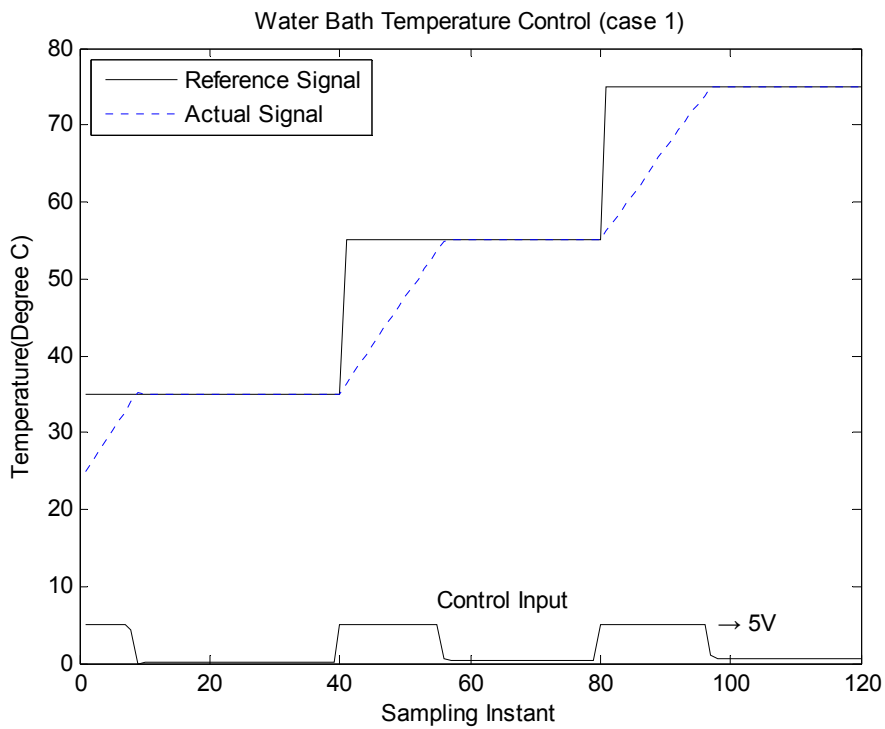


Figure 5.9: The regulation performance of the FLNFN-DMPSO controller for the water bath system.

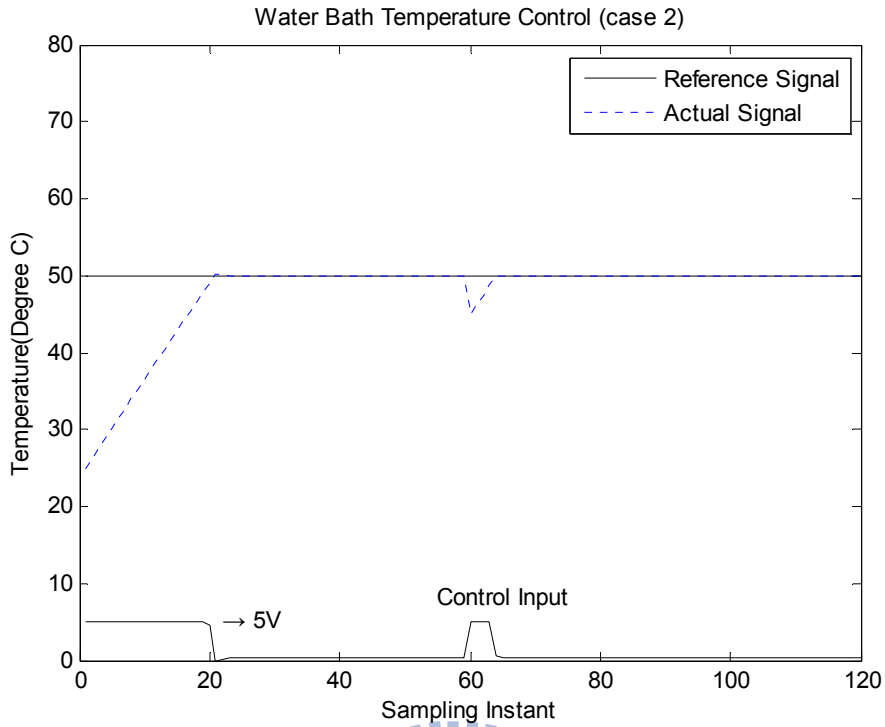


Figure 5.10: The behavior of the FLNFN-DMPSO controller under impulse noise for the water bath system.

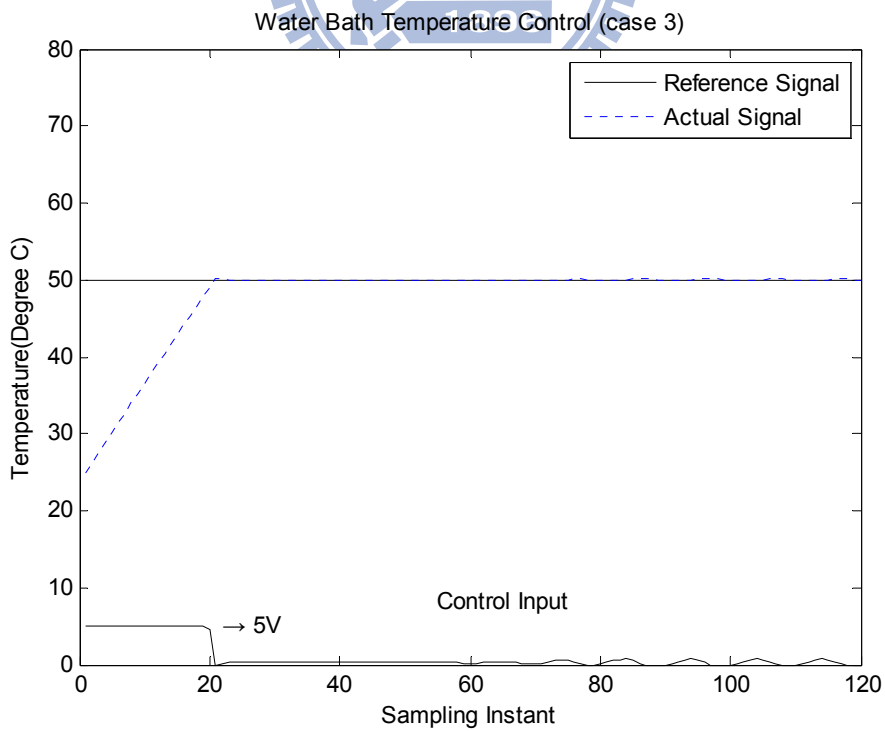


Figure 5.11: The behavior of the FLNFN-DMPSO controller when a change occurs in the water bath system dynamics.

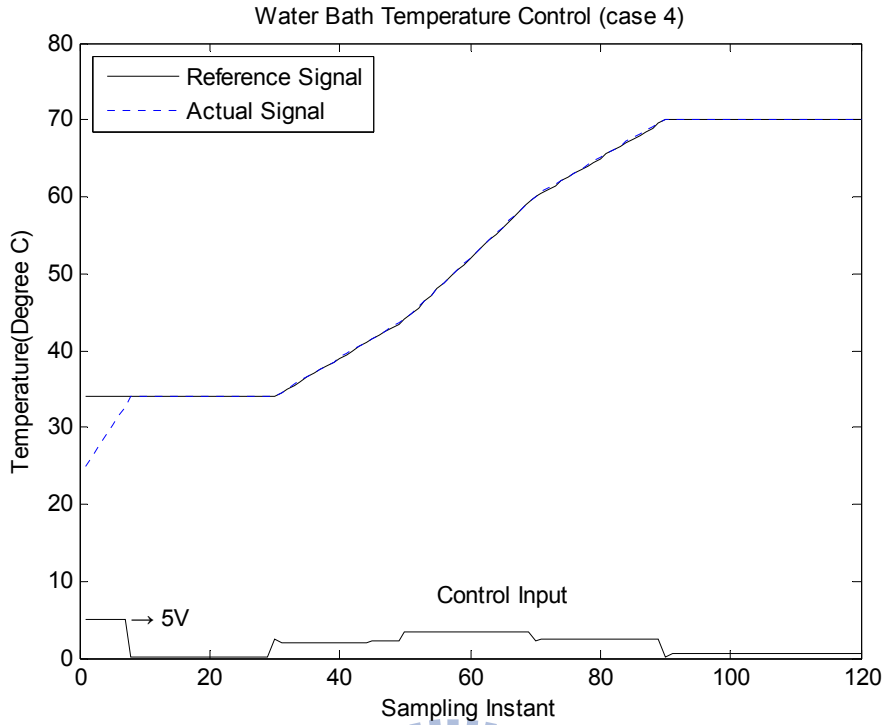


Figure 5.12: The tracking performance of the FLNFN-DMPSO controller for the water bath system.

Table 5.3: Performance comparison of various controllers for the water bath temperature control system.

$SAE = \sum_{k=1}^{120}  y_{ref}(k) - y(k) $	Regulation Performance	Influence of Impulse Noise	Effect of Change in Plant Dynamics	Tracking Performance
<b>FLNFN-DMPSO</b>	<b>352.32</b>	<b>270.29</b>	<b>262.91</b>	<b>42.45</b>
FLNFN [32]	352.84	270.41	263.35	44.28
PID [125]	418.5	311.5	322.2	100.6
Fuzzy [8]	401.5	275.8	273.5	88.1
FLNN [80]	379.22	324.51	311.54	98.43
TSK-type NFN [24]	361.96	274.75	265.48	54.28

## 5.5 Concluding Remarks

This chapter proposes an evolutionary neural fuzzy system, designed using

FLNFN model embedded with DMPSO algorithm. The proposed learning scheme consists of structure learning and parameter learning for the FLNFN model. The structure learning depends on the entropy measure to determine the number of fuzzy rules. The proposed DMPSO parameter learning method can adjust the shape of fuzzy rule's membership function and the corresponding weighting of FLNN. The simulation results have shown the proposed FLNFN-DMPSO method has more chance of converging to the global optimum and yields better performance than other existing models under some circumstances.



# Chapter 6

## Comparisons and Discussions

PSO is an efficient tool for optimization and search problems. However, it is easy to be trapped into local optima due to its information sharing mechanism. Many researchers have worked on improving its performance in various ways, thereby deriving many interesting variants. This dissertation develops three novel learning algorithms embedded with particle swarm optimizer, named IPSO, BFPSO and DMPSO for the neuro-fuzzy systems.

### 6.1 Comparisons

In this section, skin color detection problem is performed to evaluate the performance of the proposed IPSO, BFPSO and DMPSO methods.

The skin color detection experimental results of the IPSO and BFPSO methods are given in Section 3.4 and Section 4.3, respectively. In the following subsection, the skin color detection problem is performed to assess the performance of the DMPSO approach in classification application.

#### 6.1.1 Skin Color Detection Using DMPSO

The description of the system is the same as Section 3.4. We set three rules constituting a neuro-fuzzy classifier. In this example, the performance of the DMPSO method is compared with the PSO method [41]. First, the learning curves of DMPSO and PSO methods are shown in Figure 6.1. In Figure 6.1, we find that the performance of the proposed DMPSO method is superior to the PSO method.



Furthermore, the comparison items include the training and testing accuracy rates are tabulated in Table 6.1.

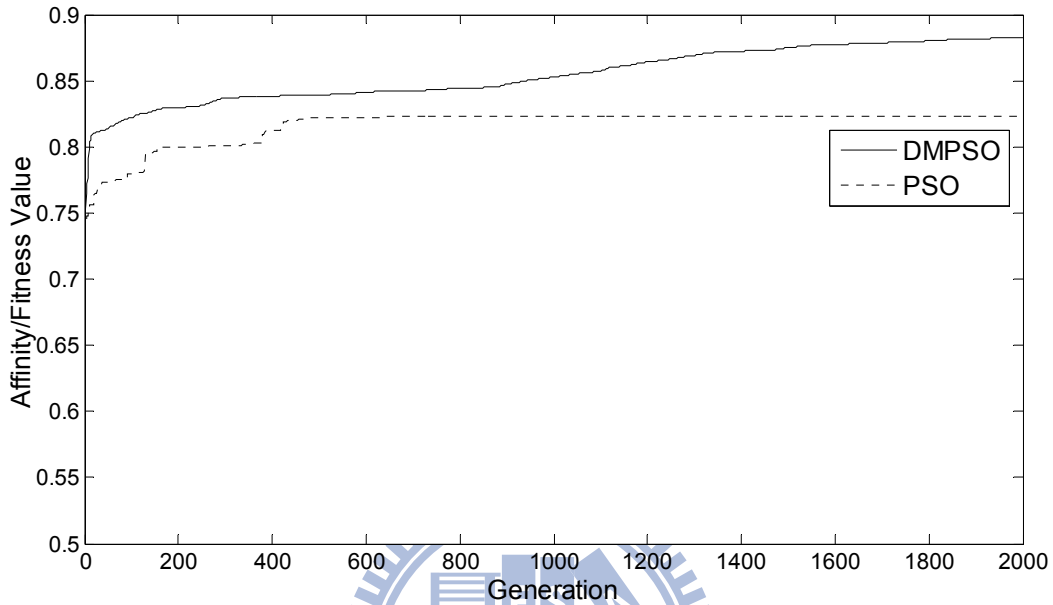


Figure 6.1: The learning curves of PSO and DMPSO methods using the CIT database.

Table 6.1: Performance comparison with PSO and DMPSO methods from the CIT database (Training data: 6000; Generations: 2000)

Method	DMPSO	PSO
Average training accuracy rate	<b>98.05%</b>	96.77%
Average testing accuracy rate	<b>87.26%</b>	83.64%

The CIT facial database consists of complex backgrounds and diverse lighting. Hence, from the comparison data listed in Table 6.1, the average of the test accuracy rate is 83.64% for the PSO method and 87.26% for the proposed DMPSO method. This demonstrates that the CIT database is more complex and does not lead to a decrease in the accuracy rate. The proposed DMPSO method maintains a superior accuracy rate. The color images from the CIT facial database are shown in Figure 6.2.

The corresponding fitness maps generated by well-trained network using the proposed DMPSO method are shown in Figure 6.3. With proper selection of the threshold value, a well-trained network can generate binary outputs (1/0 for skin/non-skin) to detect a facial region. Figure 6.4 shows the masks generated by the proposed skin color classifier. Figure 6.5 shows that the proposed approach determines a facial region accurately.

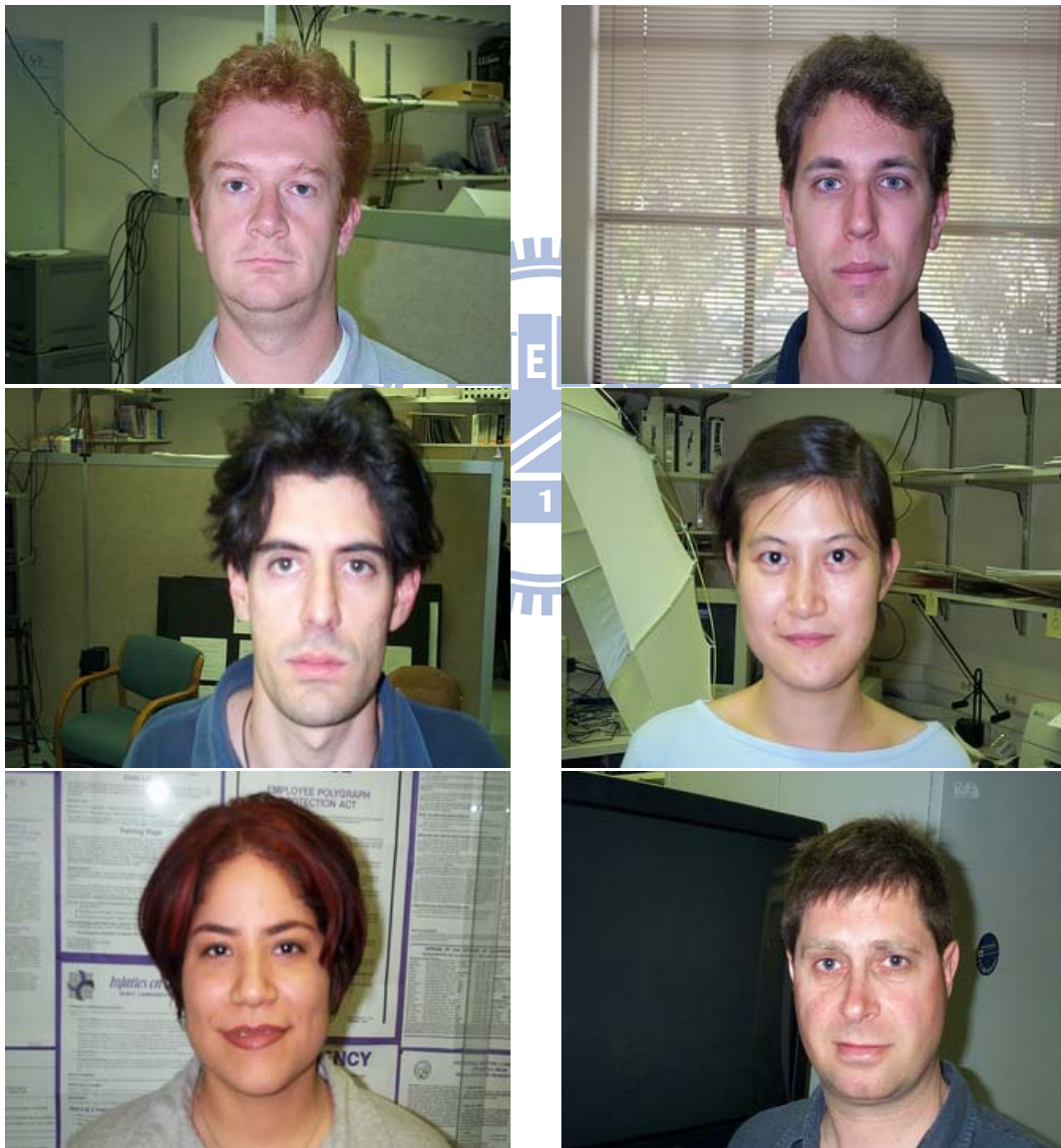


Figure 6.2: Original face images from CIT repository.



Figure 6.3: Fitness maps generated by a well-trained FLNFN-DMPSO



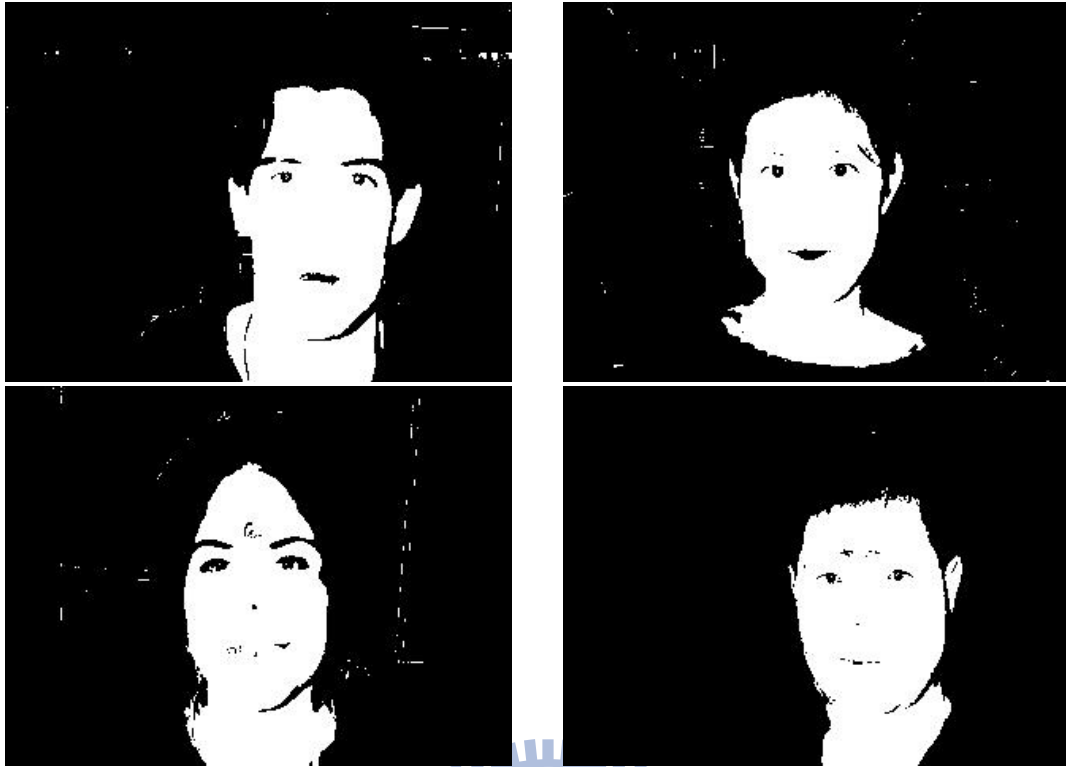


Figure 6.4: Masks generated by a well-trained skin color classifier.



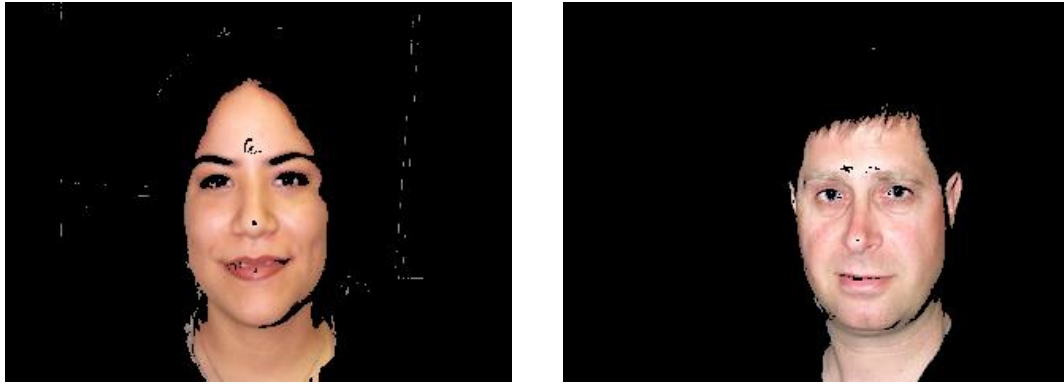


Figure 6.5: Results of skin color detection with YCbCr color space

### 6.1.2 Skin Color Detection Results Comparison with Different Approaches

In this subsection, the skin color detection experimental results of neuro-fuzzy classifier embedded with different parameter learning algorithms demonstrated. In this research, we select the FLNFN model as our neuro-fuzzy architecture to develop the skin color classifier. The aim of the skin color detection is to distinguish between skin and non-skin pixels based on the Y, Cb and Cr information. Table 6.2 summarized the average accuracy rates of testing and training data with different approaches.

Table 6.2: Performance comparison with various existing models from the CIT database (Training data: 6000; Generations: 2000)

Method	No. of fuzzy rules	Average accuracy rate (training data)	Average accuracy rate (testing data)
<b>IPSO</b>	<b>4</b>	<b>93.32%</b>	<b>90.18%</b>
IA [94]	4	88.1%	82.63%
PSO [41]	4	79.05%	74.32%
	3	96.77%	83.64%
BFO	3	96.5%	82.39%
<b>BFPSO</b>	<b>3</b>	<b>97.63%</b>	<b>85.82%</b>
<b>DMPSO</b>	<b>3</b>	<b>98.05%</b>	<b>87.26%</b>

## 6.2 Discussions

In the IPSO and BFPSO approaches, we investigated hybridization by combining PSO with IA and BFO, respectively. In IPSO method, the major parameter learning process is achieved by IA. In order to avoid trapping in a local optimal solution and ensure the search capability of near global optimal solution, we employ the advantages of the PSO to improve mutation mechanism of IA. In addition, the balance between exploration of the search space and exploitation of potentially good solutions is considered as a fundamental problem in nature-inspired systems. Too much stress on exploration results in a pure random search whereas too much exploitation results in a pure local search. Clearly, intelligent search must self-adaptively combine exploration of the new regions of the space with evaluation of potential solutions already identified. The BFPSO combines both algorithms BFO and PSO to balance the exploration and exploitation abilities of the search space.

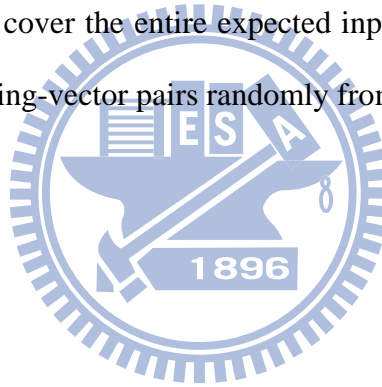
Unlike IPSO and BFPSO approaches that use PSO as the enhance mechanism to improve the performance of basic IA and BFO. In DMPSO approach, the parameter learning method is based on the PSO algorithm and the distance-based mutation operator is introduced to increase the population diversity, which strongly encourages a global search giving the particles more chance of escaping from local optimum and converging to the global optimum.

It should be notice that due to the PSO plays different role between the proposed IPSO, BFPSO and DMPSO methods, the parameters of PSO are not totally the same for these three parameter learning algorithms. The functions of IA, BFO and PSO are summarized in Table 6.3. Furthermore, the predefined fuzzy rule number in IPSO method is set to be 4 which were different from others.

Table 6.3: The roles of IA, BFO and PSO in the proposed learning algorithm.

Method		IPSO	BFPSO	DMPSO
Fuzzy Rule Numbers		<b>4</b>	<b>3</b>	<b>3</b>
Basic/Main Algorithm		IA	BFO	<b>PSO</b>
Enhancement	Mechanism	<b>PSO</b>	<b>PSO</b>	Mutation operator
	Function	Increase population diversity	Improve global search ability	Increase population diversity

Moreover, in order to obtain better simulation results, the proposed learning algorithms always require training data to be sufficient and proper. However, there is no procedure or rule suitable for all cases in choosing training data. One rule of thumb is that training data should cover the entire expected input space and then during the training process select training-vector pairs randomly from the set.



# Chapter 7

## Conclusions and Future Works

Fuzzy logic and artificial neural networks are complementary technologies in the design of intelligent systems. The combination of these two technologies into an integrated system appears to be a promising path toward the development of intelligent systems capable of capturing qualities characterizing the human brain. Both neural networks and fuzzy logic are powerful design techniques that have their strengths and weaknesses. The integrated neuro-fuzzy systems possess the advantages of both neural networks (e.g. learning abilities, optimization abilities and connectionist structures) and fuzzy systems (e.g. humanlike IF-THEN rules thinking and ease of incorporating expert knowledge). In this way, it is possible to bring the low-level learning and computational power of neural networks into fuzzy systems and also high-level humanlike IF-THEN thinking and reasoning of fuzzy systems into neural networks.

A neuro-fuzzy system is a fuzzy system, whose parameters are learned by a learning algorithm. It has a neural network architecture constructed from fuzzy reasoning, and can always be interpreted as a system of fuzzy rules. Learning is used to adaptively adjust the rules in the rule base, and to produce or optimize the membership functions of a fuzzy system. Structured knowledge is codified as fuzzy rules. Modern neuro-fuzzy systems are usually represented as special multilayer feedforward neural networks. Hayashi *et al.* [126] showed that a feedforward neural network could approximate any fuzzy rule based system and any feedforward neural network may be approximated by a rule based fuzzy inference system.

In this dissertation, the neuro-fuzzy architecture we used is called



functional-link-based neuro-fuzzy network (FLNFN) model. The FLNFN model uses a functional link neural network to the consequent part of the fuzzy rules. FLNFN is a multilayer feedforward network in which each node performs a particular function (node function) based on incoming signals and a set of parameters pertaining to this node. The FLNFN model can automatically be constructed and the FLNFN parameters can be adjusted by performing structure/parameter learning schemes.

In Chapter 3, the proposed IPSO method combines the IA and PSO to perform parameter learning. The advantages of the proposed IPSO method are summarized as follows: 1) We employed the advantages of PSO to improve the mutation mechanism; 2) The complicated problems can be better solved than IA and PSO; 3) There is more of a likelihood to get a global optimum compared to heuristic methods; 4) The experimental results have shown that our method obtains better results than other existing methods in accuracy rate and convergence speed.

In Chapter 4, an innovative BFPSO algorithm is applied for the design of neuro-fuzzy classifier. Conventional BFO depends on random search directions which may lead to delay in reaching global solution while PSO is prone to be trapped in local optima. In order to get better optimization, the new algorithm combines advantages of both the algorithms i.e. PSO's ability to exchange social information and BFO's ability in finding new solutions by elimination and dispersal. The BFPSO algorithm combines PSO-based mutation operator with bacterial chemotaxis in order to make judicious use of exploration and exploitation abilities of search space and to avoid false and premature convergence. The simulation results showed that the overall performance of the hybrid algorithm outperforms conventional BFO and PSO.

Unlike IPSO and BFPSO approaches that use PSO as the enhance mechanism to improve the performance of basic IA and BFO. In chapter 5, the PSO-based learning algorithm, called DMPSO, for the neural fuzzy system is presented. In DMPSO

approach, the parameter learning method is based on the PSO algorithm and the distance-based mutation operator is introduced to increase the population diversity, which strongly encourages a global search giving the particles more chance of escaping from local optimum and converging to the global optimum. The simulation results have shown the proposed DMPSO method yields better performance than other existing models under some circumstances in the nonlinear system control application fields.

In Chapter 6, the well-known skin color detection problem is used as the benchmark to demonstrate the performance and efficiency of the proposed IPSO, BFPSO and DMPSO method. The aim of the skin color detection is to distinguish between skin and non-skin pixels based on the Y, Cb and Cr information. The average accuracy rates of testing and training data with different approaches were depicted in Table 6.2. Since the predefined rule number is not identical, we cannot make the comparison fairly. From the simulation results, we can only conclude that DMPSO outperforms BFPSO and IPSO seems to be over-trained.

Although the proposed algorithms yield better performance in the classification and nonlinear system control applications, but there still some advanced topics should be addressed in future research.

In general, synthesizing a neuro-fuzzy system, two major types of learning are required: structure learning algorithms to find appropriate fuzzy logic rules; and parameter learning algorithms to fine-tune the membership functions and other parameters. There are several ways that structure learning and parameter learning can be combined in a neuro-fuzzy system. They can be performed sequentially: structure learning is used first to find the appropriate structure of a neuro-fuzzy system; and parameter learning is then used to fine-tune the parameters. In some situations, only parameter learning or structure learning is necessary when structure (fuzzy rules) or

parameters (membership functions) are provided by experts, and the structure in some neuro-fuzzy systems is fixed. Identification of fuzzy rules has been one of the most important aspects in the design of neuro-fuzzy systems. Identified rules and concise rules can provide an initial structure of networks so that learning processes can be fast, reliable and highly intuitive. To overcome the limitations of using expert knowledge in defining the fuzzy rules, data driven methods to create fuzzy systems are needed. Therefore, the first advanced research topic is to generate fuzzy rules from numerical data more efficiently.

The choice of the model's structure is very important, as it determines the flexibility of the model in the approximation of (unknown) systems. Despite of the research that has already been done in the area of neuro-fuzzy systems the recurrent variants of this architecture are still rarely studied. In contrast to pure feed-forward architectures, that have a static input-output behavior, recurrent models are able to store information of the past (e.g. prior system states) and are thus more appropriate for the analysis of dynamic systems. The second advanced research topic is to apply the proposed IPSO, BFPSO and DMPSO into the recurrent neural network to learn and optimize a hierarchical fuzzy rule base with feedback connections.

In this dissertation, a systematic method was not used to determine the initial parameters. The initial parameters are determined by practical experimentation or by trial-and-error. In future works, we will try to develop a well-defined method to automatically determine the initial parameters, and thus inexperienced users could design a neuro-fuzzy system with ease.

# Bibliography

1. H. Takagi, N. Suzuki, T. Koda and Y. Kojima, "Neural Networks Designed on Approximate Reasoning Architecture and Their Applications," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 752-760, 1992.
2. E. Sanchez, T. Shibata and L. A. Zadeh, *Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives*. World Scientific, 1997.
3. O. Cordon, F. Gomide, F. Herrera, F. Hoffmann and L. Magdalena, "Ten Years of Genetic Fuzzy Systems: Current Framework and New Trends," *Fuzzy Sets and Systems*, vol. 141, no. 1, pp. 5-31, 2004.
4. A. Homaifar and E. McCormick, "Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 2, pp. 129-139, 1995.
5. J. R. Velasco, "Genetic-Based On-Line Learning for Fuzzy Process Control," *International Journal of Intelligent Systems*, vol. 13, no. 10-11, pp. 891-903, 1998.
6. H. Ishibuchi, T. Nakashima and T. Murata, "Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 29, no. 5, pp. 601-618, 1999.
7. J. Vieira, F. M. Dias, A. Mota, "Neuro-Fuzzy Systems: A Survey," *WSEAS Transactions on Systems*, vol. 3, no. 2, pp. 414-419, 2004.
8. C.-T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice-Hall, 1996.
9. S. Mitra and Y. Hayashi, "Neuro-Fuzzy Rule Generation: Survey in Soft

- Computing Framework,” *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 748-768, 2000.
10. A. V. Nandedkar and P. K. Biswas, “A Granular Reflex Fuzzy Min-Max Neural Network for Classification,” *IEEE Transactions on Neural Networks*, vol. 20, no. 7, pp. 1117-1134, 2009.
  11. G.-D. Wu and P.-H. Huang, “A Maximizing-Discriminability-Based Self-Organizing Fuzzy Network for Classification Problems,” *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 2, pp. 362-373, 2010.
  12. O. Cordon, F. Herrera, F. Hoffmann and L. Magdalena, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. World Scientific, 2001.
  13. P. P. Angelov, *Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems*. Physica-Verlag, 2002.
  14. A. Gonzalez and R. Perez, “SLAVE: A Genetic Learning System Based on an Iterative Approach,” *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 2, pp. 176-191, 1999.
  15. M. Russo, “FuGeNeSys – A Fuzzy Genetic Neural System for Fuzzy Modeling,” *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 3, pp. 373-388, 1998.
  16. H. Ishibuchi, K. Nozaki, N. Yamamoto and H. Tanaka, “Selecting Fuzzy If-Then Rules for Classification Problems Using Genetic Algorithms,” *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 260-270, 1995.
  17. H. Ishibuchi, T. Murata and I. B. Turksen, “Single-Objective and Two-Objective Genetic Algorithms for Selecting Linguistic Rules for Pattern Classification Problems,” *Fuzzy Sets and Systems*, vol. 89, no. 2, pp. 135-150, 1997.
  18. H. Ishibuchi and Y. Nojima, “Analysis of Interpretability-Accuracy Tradeoff of Fuzzy Systems by Multiobjective Fuzzy Genetics-Based Machine Learning,”

- International Journal of Approximate Reasoning*, vol. 44, no. 1, pp. 4-31, 2007.
19. L.-X. Wang and J. M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414-1427, 1992.
  20. C.-J. Lin and C.-T. Lin, "An ART-Based Fuzzy Adaptive Learning Control Network," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 4, pp. 477-496, 1997.
  21. C.-T. Lin, C.-J. Lin and C. S. G. Lee, "Fuzzy Adaptive Learning Control Network with On-Line Neural Learning," *Fuzzy Sets and Systems*, vol. 71, no. 1, pp. 25-45, 1995.
  22. W.-S. Lin, C.-H. Tsai and J.-S. Liu, "Robust Neuro-Fuzzy Control of Multivariable Systems by Tuning Consequent Membership Functions," *Fuzzy Sets and Systems*, vol. 124, no. 2, pp. 181-195, 2001.
  23. T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116-132, 1985.
  24. J.-S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665-685, 1993.
  25. C.-F. Juang and C.-T. Lin, "An On-Line Self-Constructing Neural Fuzzy Inference Network and Its Applications," *IEEE Transactions on Fuzzy Systems*, vol. 6, no.1, pp. 12-32, 1998.
  26. D. Nauck and R. Kruse, "A Neuro-Fuzzy Method to Learn Fuzzy Classification Rules from Data," *Fuzzy Sets and Systems*, vol. 89, no.3, pp. 277-288, 1997.
  27. S. Paul and S. Kumar, "Subsethood-Product Fuzzy Neural Inference System (SuPFuNIS)," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp.

- 578-599, 2002.
28. J.-S. Wang and C. S. G. Lee, "Self-Adaptive Neuro-Fuzzy Inference Systems for Classification Applications," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 6, pp. 790-802, 2002.
  29. C.-J. Lin and C.-T. Lin, "Reinforcement Learning for an ART-Based Fuzzy Adaptive Learning Control Network," *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 709-731, 1996.
  30. F.-J. Lin, C.-H. Lin and P.-H. Shen, "Self-Constructing Fuzzy Neural Network Speed Controller for Permanent-Magnet Synchronous Motor Drive," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 5, pp. 751-759, 2001.
  31. C.-J. Lin and C.-H. Chen, "Identification and Prediction Using Recurrent Compensatory Neuro-Fuzzy Systems," *Fuzzy Sets and Systems*, vol. 150, no. 2, pp. 307-330, 2005.
  32. C.-H. Chen, C.-J. Lin and C.-T. Lin, "A Functional-Link-Based Neurofuzzy Network for Nonlinear System Control," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 5, pp. 1362-1378, 2008.
  33. M.-T. Su, C.-H. Chen, C.-J. Lin and C.-T. Lin, "A Rule-Based Symbiotic Modified Differential Evolution for Self-Organizing Neuro-Fuzzy Systems," *Applied Soft Computing*, vol. 11, no. 8, pp. 4847-4858, 2011.
  34. R. Fuller, *Introduction to Neuro-Fuzzy Systems, Studies in Fuzziness and Soft Computing*. Physica-Verlag, 2000.
  35. C.-T. Lin and C. S. G. Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System," *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1320-1336, 1991.
  36. H. Bunke and A. Kandel, *Neuro-Fuzzy Pattern Recognition*. World Scientific, 2000.

37. S. K. Pal and S. Mitra, *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing*. John Wiley & Sons, 1999.
38. L. Chen, D. H. Cooley and J. Zhang, "Possibility-Based Fuzzy Neural Networks and Their Application to Image Processing," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 29, no. 1, pp. 119-126, 1999.
39. S.-W. Lin, S.-C. Chen, W.-J. Wu and C.-H. Chen, "Parameter Determination and Feature Selection for Back-Propagation Network by Particle Swarm Optimization," *Knowledge and Information Systems*, vol. 21, no. 2, pp. 249-266, 2009.
40. T. Weise, *Global Optimization Algorithms – Theory and Application*. it-weise.de (self-published), 2009. (<http://www.it-weise.de/projects/book.pdf>)
41. J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, pp. 1942-1948, 1995.
42. R. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43, 1995.
43. J. Kennedy, R. C. Eberhart and Y. Shi, *Swarm Intelligence*. Morgan Kaufmann, 2001.
44. M. P. Wachowiak, R. Smolikova, Y. Zheng, J. M. Zurada and A. S. Elmaghraby, "An Approach to Multimodal Biomedical Image Registration Utilizing Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 289-301, 2004.
45. W.-F. Leong and G. G. Yen, "PSO-Based Multiobjective Optimization with Dynamic Population Size and Adaptive Local Archives," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 38, no. 5, pp.



- 1270-1293, 2008.
46. E. Mezura-Montes and C. A. C. Coello, "A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 1-17, 2005.
  47. C. A. C. Coello, G. T. Pulido and M. S. Lechuga, "Handling Multiple Objectives with Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256-279, 2004.
  48. R. Xu, G. C. Anagnostopoulos and D. C. Wunsch II, "Multiclass Cancer Classification Using Semisupervised Ellipsoid ARTMAP and Particle Swarm Optimization with Gene Expression Data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 1, pp. 65-77, 2007.
  49. F. Melgani and Y. Bazi, "Classification of Electrocardiogram Signals with Support Vector Machines and Particle Swarm Optimization," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 5, pp. 667-677, 2008.
  50. M. Sugisaka and X. Fan, "An Effective Search Method for Neural Network Based Face Detection Using Particle Swarm Optimization," *IEICE Transactions on Information and Systems*, vol. E88-D, no. 2, pp. 214-222, 2005.
  51. C.-F. Juang, "A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 34, no. 2, pp. 997-1006, 2004.
  52. C.-T. Lin, C.-T. Yang and M.-T. Su, "A Hybridization of Immune Algorithm with Particle Swarm Optimization for Neuro-Fuzzy Classifiers," *International Journal of Fuzzy Systems*, vol. 10, no. 3, pp. 139-149, 2008.
  53. M.-T. Su and C.-T. Lin, "Nonlinear System Control Using Functional-Link-Based Neuro-Fuzzy Network Model Embedded with Modified Particle Swarm Optimizer," in *Proceedings of the 19th National Conference on*

*Fuzzy Theory and Its Application*, 2011.

54. D. Sedighizadeh and E. Masehian, "Particle Swarm Optimization Methods, Taxonomy and Applications," *International Journal of Computer Theory and Engineering*, vol. 1, no. 5, pp. 486-502, 2009.
55. J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281-295, 2006.
56. R. Poli, J. Kennedy and T. Blackwell, "Particle Swarm Optimization: An Overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33-57, 2007.
57. A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.
58. Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 69-73, 1998.
59. Y. Shi and R. C. Eberhart, "Parameter Selection in Particle Swarm Optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming VII*, vol. 160, no. 4, pp. 591-600, 1998.
60. Y. Shi and R. Eberhart, "Particle Swarm Optimization with Fuzzy Adaptive Inertia Weight," in *Proceedings of the Workshop on Particle Swarm Optimization*, pp. 101-106, 2001.
61. A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240-255, 2004.
62. H.-Y. Fan and Y. Shi, "Study on Vmax of Particle Swarm Optimization," in

- Proceedings of the Workshop on Particle Swarm Optimization*, 2001.
63. M. Clerc and J. Kennedy, "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.
  64. J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3, pp. 1931-1938, 1999.
  65. J. Kennedy and R. Mendes, "Population Structure and Particle Swarm Performance," in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1671-1676, 2002.
  66. P. N. Suganthan, "Particle Swarm Optimizer with Neighborhood Operator," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3, pp. 1958-1962, 1999.
  67. X. Hu and R. Eberhart, "Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1677-1681, 2002.
  68. K. E. Parsopoulos and M. N. Vrahatis, "UPSO: A Unified Particle Swarm Optimization Scheme," in *Lecture Series on Computer and Computational Sciences*, vol. 1, pp. 868-873, 2004.
  69. K. E. Parsopoulos and M. N. Vrahatis, "Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems," in *Lecture Notes in Computer Science*, vol. 3612, pp. 582-591, 2005.
  70. R. Mendes, J. Kennedy and J. Neves, "The Fully Informed Particle Swarm: Simpler, Maybe Better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204-210, 2004.
  71. T. Peram, K. Veeramachaneni and C. K. Mohan, "Fitness-Distance-Ratio Based

- Particle Swarm Optimization,” in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 174-181, 2003.
72. P. J. Angeline, “Using Selection to Improve Particle Swarm Optimization,” in *Proceedings of the 1998 IEEE Congress on Evolutionary Computation*, pp. 84-89, 1998.
73. M. Lovbjerg, T. K. Rasmussen and T. Krink, “Hybrid Particle Swarm Optimizer with Breeding and Subpopulations,” in *Proceedings of the Third Genetic and Evolutionary Computation Conference*, vol. 1, pp. 469-476, 2001.
74. V. Miranda and N. Fonseca, “New Evolutionary Particle Swarm Algorithm (EPSO) Applied to Voltage/VAR Control,” in *Proceedings of the 14th Power Systems Computation Conference*, 2002.
75. M. Lovbjerg and T. Krink, “Extending Particle Swarm Optimizers with Self-Organized Criticality,” in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1588-1593, 2002.
76. T. M. Blackwell and P. Bentley, “Don’t Push Me! Collision-Avoiding Swarms,” in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1691-1696, 2002.
77. X.-F. Xie, W.-J. Zhang and Z.-L. Yang, “A Dissipative Particle Swarm Optimization,” in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1456-1461, 2002.
78. K. E. Parsopoulos and M. N. Vrahatis, “On the Computation of All Global Minimizers Through Particle Swarm Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 211-224, 2004.
79. F. van den Bergh and A. P. Engelbrecht, “A Cooperative Approach to Particle Swarm Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, 2004.

80. J. C. Patra, R. N. Pal, B. N. Chatterji and G. Panda, "Identification of Nonlinear Dynamic Systems Using Functional Link Artificial Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 29, no. 2, pp. 254-262, 1999.
81. Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley, 1989.
82. J. C. Patra and R. N. Pal, "A Functional Link Artificial Neural Network for Adaptive Channel Equalization," *Signal Processing*, vol. 43, no. 2, pp. 181-195, 1995.
83. D. Nauck, F. Klawonn and R. Kruse, *Foundations of Neuro-Fuzzy Systems*. John Wiley & Sons, 1997.
84. Y.-H. Pao, S. M. Phillips and D. J. Sobajic, "Neural-Net Computing and Intelligent Control Systems," *International Journal of Control*, vol. 56, no. 2, pp. 263-289, 1992.
85. D. W. Boeringer and D. H. Werner, "Particle Swarm Optimization versus Genetic Algorithms for Phased Array Synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 3, pp. 771-779, 2004.
86. L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, 1996.
87. J. E. Hunt and D. E. Cooke, "Learning Using an Artificial Immune System," *Journal of Network and Computer Applications*, vol. 19, no. 2, pp. 189-212, 1996.
88. D. Dasgupta, *Artificial Immune Systems and Their Applications*. Springer-Verlag, 1999.
89. S. A. Hofmeyr and S. Forrest, "Immunity by Design: An Artificial Immune System," in *Proceedings of the Genetic and Evolutionary Computation*

*Conference*, vol. 2, pp. 1289-1296, 1999.

90. L. N. de Castro and F. J. Von Zuben, "Artificial Immune Systems: Part I - Basic Theory and Applications," *Technical Report – TR-DCA 01/99*, School of Electrical and Computing Engineering, State University of Campinas, Brazil, 1999. ([ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/tr\\_dca/trdca0199.pdf](ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/tr_dca/trdca0199.pdf))
91. L. N. de Castro and F. J. Von Zuben, "Artificial Immune Systems: Part II - A Survey of Applications," *Technical Report DCA-RT 02/00*, School of Electrical and Computing Engineering, State University of Campinas, Brazil, 2000. ([ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/tr\\_dca/trdca0200.pdf](ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/tr_dca/trdca0200.pdf))
92. A. Kalinli and N. Karaboga, "Artificial Immune Algorithm for IIR Filter Design," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 8, pp. 919-929, 2005.
93. X. Wen and A. Song, "An Immune Evolutionary Algorithm for Sphericity Error Evaluation," *International Journal of Machine Tools & Manufacture*, vol. 44, no. 10, pp. 1077-1084, 2004.
94. J.-S. Chun, M.-K. Kim and H.-K. Jung, "Shape Optimization of Electromagnetic Devices Using Immune Algorithm," *IEEE Transactions on Magnetics*, vol. 33, no. 2, pp. 1876-1879, 1997.
95. L. N. de Castro and F. J. Von Zuben, "Learning and Optimization Using the Clonal Selection Principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239-251, 2002.
96. F. M. Burnet, "Clonal Selection and After," in *Theoretical Immunology*, G. I. Bell, A. S. Perelson, and G. H. Pimbley Jr., Eds., pp. 63-85, Marcel Dekker, 1978.
97. F. M. Burnet, *The Clonal Selection Theory of Acquired Immunity*, Cambridge University Press, 1959.
98. C.-J. Lin, I-F. Chung and C.-H. Chen, "An Entropy-Based Quantum

- Neuro-Fuzzy Inference System for Classification Applications,” *Neurocomputing*, vol. 70, no. 13-15, pp. 2502-2516, 2007.
99. C.-H. Chen, C.-J. Lin and C.-T. Lin, “An Efficient Quantum Neuro-Fuzzy Classifier Based on Fuzzy Entropy and Compensatory Operation,” *Soft Computing*, vol. 12, no. 6, pp. 567-583, 2008.
100. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.
101. R. Setiono and H. Liu, “Neural-Network Feature Selector,” *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 654-662, 1997.
102. H.-M. Lee, C.-M. Chen, J.-M. Chen and Y.-L. Jou, “An Efficient Fuzzy Classifier with Feature Selection Based on Fuzzy Entropy,” *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 31, no. 3, pp. 426-432, 2001.
103. K. M. Passino, “Biomimicry of Bacterial Foraging for Distributed Optimization and Control,” *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52-67, 2002.
104. S. D. Muller, J. Marchetto, S. Airaghi and P. Koumoutsakos, “Optimization Based on Bacterial Chemotaxis,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 16-29, 2002.
105. D. H. Kim and J. H. Cho, “Adaptive Tuning of PID Controller for Multivariable System Using Bacterial Foraging Based Optimization,” in *Proceedings of the Third International Atlantic Web Intelligence Conference*, vol. 3528 of *Lecture Notes in Computer Science*, pp. 231-235, 2005.
106. D. H. Kim and C. H. Cho, “Bacterial Foraging Based Neural Network Fuzzy Learning,” in *Proceedings of the Indian International Conference on Artificial Intelligence*, pp. 2030-2036, 2005.
107. T. Datta and I. S. Misra, “A Comparative Study of Optimization Techniques in

- Adaptive Antenna Array Processing: The Bacteria-Foraging Algorithm and Particle-Swarm Optimization,” *IEEE Antennas and Propagation Magazine*, vol. 51, no. 6, pp. 69-81, 2009.
108. P. K. Simpson, “Fuzzy Min-Max Neural Networks – Part 1: Classification,” *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 776-786, 1992.
109. H.-M. Lee, K.-H. Chen and I-F. Jiang, “A Neural Network Classifier with Disjunctive Fuzzy Information,” *Neural Networks*, vol. 11, no. 6, pp. 1113-1125, 1998.
110. T.-P. Wu and S.-M. Chen, “A New Method for Constructing Membership Functions and Fuzzy Rules from Training Examples,” *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 29, no. 1, pp. 25-40, 1999.
111. B. C. Lovell and A. P. Bradley, “The Multiscale Classifier,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp. 124-137, 1996.
112. C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404-418, 1990.
113. C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part II,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 419-435, 1990.
114. K. S. Narendra and K. Parthasarathy, “Identification and Control of Dynamical Systems Using Neural Networks,” *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.
115. K. J. Astrom and B. Wittenmark, *Adaptive Control*. Addison-Wesley, 1989.
116. C. Li, S. Yang and I. Korejo, “An Adaptive Mutation Operator for Particle Swarm Optimization,” in *Proceedings of the 2008 UK Workshop on*



- Computational Intelligence*, pp. 165-170, 2008.
117. K. V. Deligkaris, Z. D. Zaharis, D. G. Kampitaki, S. K. Goudos, I. T. Rekanos and M. N. Spasos, "Thinned Planar Array Design Using Boolean PSO with Velocity Mutation," *IEEE Transactions on Magnetics*, vol. 45, no. 3, pp. 1490-1493, 2009.
118. S. H. Ling, H. H. C. Iu, K. Y. Chan, H. K. Lam, B. C. W. Yeung and F. H. Leung, "Hybrid Particle Swarm Optimization with Wavelet Mutation and Its Industrial Applications," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 38, no. 3, pp. 743-763, 2008.
119. D. Nguyen and B. Widrow, "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, pp. 357-363, 1989.
120. J. Tanomaru and S. Omatu, "Process Control by On-Line Trained Neural Controllers," *IEEE Transactions on Industrial Electronics*, vol. 39, no. 6, pp. 511-521, 1992.
121. C.-H. Chen, C.-J. Lin and C.-T. Lin, "Using an Efficient Immune Symbiotic Evolution Learning for Compensatory Neuro-Fuzzy Controller," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 3, pp. 668-682, 2009.
122. C.-F. Juang, J.-Y. Lin and C.-T. Lin, "Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design," *IEEE Transactions on Systems, Man, and Cybernetic – Part B: Cybernetics*, vol. 30, no. 2, pp. 290-302, 2000.
123. M. Jamei, M. Mahfouf and D. A. Linkens, "Elicitation and Fine-Tuning of Fuzzy Control Rules Using Symbiotic Evolution," *Fuzzy Sets and Systems*, vol. 147, no. 1, pp. 57-74, 2004.
124. D. Psaltis, A. Sideris and A. A. Yamamura, "A Multilayered Neural Network

- Controller,” *IEEE Control Systems Magazine*, vol. 8, no. 2, pp. 17-21, 1988.
125. C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design, 3rd Edition*. Prentice-Hall, 1995.
126. Y. Hayashi and J.J. Buckley, “Approximations Between Fuzzy Expert Systems and Neural Networks,” *International Journal of Approximate Reasoning*, vol. 10, pp. 63-73, 1994.



# Publication List

## Journal :

- [1] Chin-Teng Lin, Chien-Ting Yang and **Miin-Tsair Su**, “A Hybridization of Immune Algorithm with Particle Swarm Optimization for Neuro-Fuzzy Classifiers,” *International Journal of Fuzzy Systems*, vol. 10, no. 3, pp. 139-149, 2008. (Full paper, 1.0 點)
- [2] **Miin-Tsair Su**, Cheng-Hung Chen, Cheng-Jian Lin and Chin-Teng Lin, “A Rule-based Symbiotic Modified Differential Evolution for Self-Organizing Neuro-Fuzzy Systems,” *Applied Soft Computing*, vol. 11, no. 8, pp. 4847-4858, 2011. (Full paper, 1.2 點)
- [3] **Miin-Tsair Su**, Chin-Teng Lin and Keng-Wei Hsu, “A Novel Method for Locating Solder Joints based on Modified Binary Potential Function,” *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 1(B), pp. 911–932, 2012. (Full paper, 1.4 點)
- [4] Chen-Yu Lee, Chin-Teng Lin, Chao-Ting Hong and **Miin-Tsair Su**, “Smoke Detection Using Spatial and Temporal Analyses,” *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 7(A), pp. 4749–4770, 2012. (Full paper, 0.6 點).
- [5] **Miin-Tsair Su**, Chin-Teng Lin, Sheng-Chih Hsu, Dong-Lin Li, Cheng-Jian Lin and Cheng-Hung Chen, “Nonlinear System Control Using Functional-Link-Based Neuro-Fuzzy Network Model Embedded with Modified Particle Swarm Optimizer,” *International Journal of Fuzzy Systems*, vol. 14, no. 1, pp. 97-109, 2012. (Full paper, 1.2 點)

## Conference :

- [1] **Miin-Tsair Su**, Keng-Wei Hsu and Chin-Teng Lin, “A New Method for Locating Solder Joints Based on Potential Function,” *The 22th IPPR Conference on Computer Vision, Graphics and Image Processing*, Nantou, Taiwan, R.O.C., Aug. 23-25, 2009.
- [2] **Miin-Tsair Su** and Chin-Teng Lin, “Nonlinear System Control Using Functional-Link-Based Neuro-Fuzzy Network Model Embedded with Modified Particle Swarm Optimizer,” *The 19th National Conference on Fuzzy Theory and Its Application*, Yunlin, Taiwan, R.O.C., Nov. 18-19, 2011.