國立交通大學

電控工程研究所

博 士 論 文

分組式差分進化演算法及其應用於模糊系統最佳化設計

Group-Based Differential Evolution Algorithm and Its Application to Fuzzy System Optimization

研 究 生：韓明峰

指導教授：林進燈　張志永

中 華 民 國 一百零二 年 一 月

分組式差分進化演算法及其應用於模糊系統最佳化設計
Group-Based Differential Evolution Algorithm and Its Application to Fuzzy System Optimization

研 究 生：韓明峰　　Student：Ming-Feng Han

指導教授：林進燈 博士　　Advisor：Dr. Chin-Teng Lin

張志永 博士　　Dr. Jyh-Yeong Chang

國 立 交 通 大 學

電 控 工 程 研 究 所

博 士 論 文

A Dissertation

Submitted to Institute of Electrical Control Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Control Engineering

Jan. 2013

Hsinchu, Taiwan, Republic of China

中華民國一百零二年一月

分組式差分進化演算法及其應用於模糊系統最佳化設計

研究生：韓明峰　　指導教授：林進燈 博士

張志永 博士

國立交通大學電控工程研究所　　博士班

摘　要

本篇論文主要分為兩個部分，第一部分，我們提出分組式差分進化演算法解決函數最佳化問題。該進化演算法使用兩種不同類型的突變運算，可解決傳統演算法常遇到的停滯問題，進而達到良好的演化搜尋能力。在演化程序中，依照個體之適應值，所有個體被區分為優等組與劣等組。優等組進行區域性的突變運算，劣等組進行全域性的突變運算。再藉由交配和選擇運算以產生新的子代。我們也提出一個新的適應學習策略為了避免人為設定參數問題，該策略能自動的找到最佳設定參數。在模擬中，我們測試 13 個函數最佳化問題。本論文所提出的演算法皆呈現良好的搜尋效能。第二部分，我們將分組式差分進化演算法應用在函數聯結之模糊系統最佳化設計上。在架構學習中，使用凝聚分群演算法自動地給予模糊系統最適合的模糊規則數。在參數學習中，群體將被拆善成數個子群體且每個子群體各自進化，最後可獲得最佳化的函數聯結之模糊系統。我們將與其他方法比較，以證實所提出的網路架構及其相關演算法之有效性。

Group-Based Differential Evolution Algorithm and Its Application to Fuzzy System Optimization

Student：Ming-Feng Han Advisor：Dr. Chin-Teng Lin

Dr. Jyh-Yeong Chang

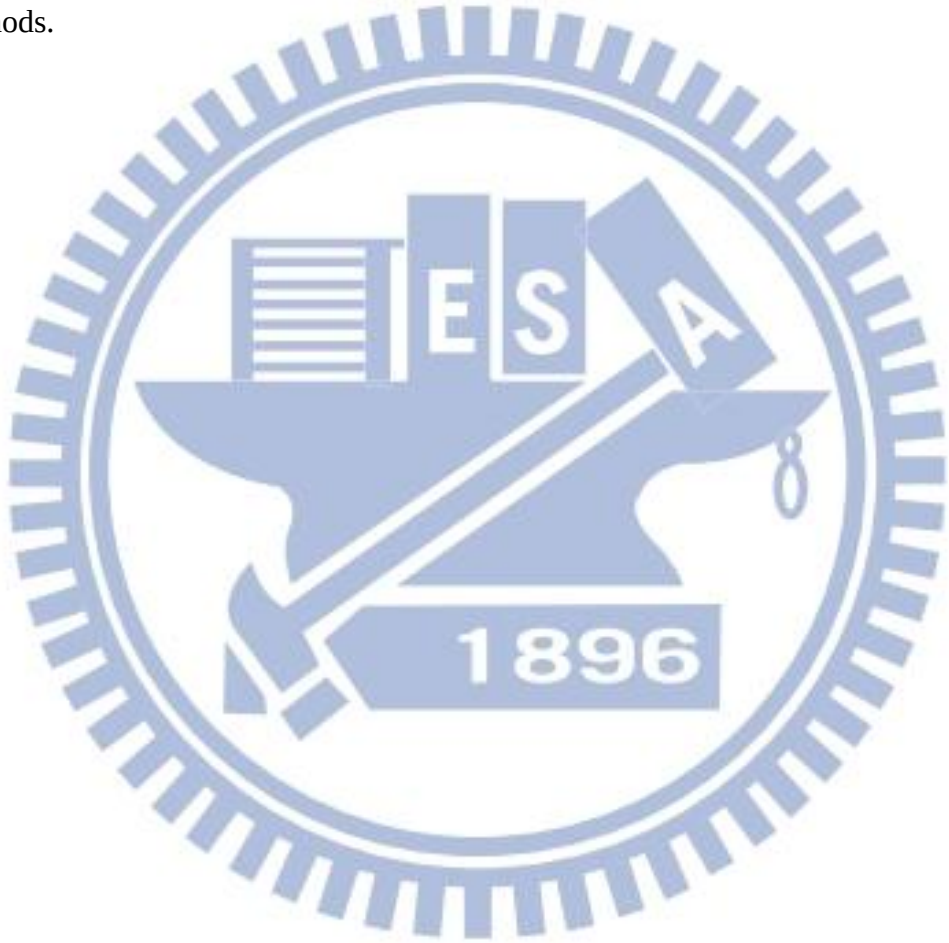
Institute of Electrical Control Engineering

National Chiao-Tung University

ABSTRACT

This dissertation consists of two major parts. In the first part, we propose a group-based differential evolution (GDE) algorithm for numerical optimization problems. The proposed GDE algorithm employs two different mutation operations to solve the stagnation problem and achieve good performance. Initially, all individuals in population are grouped into an inferior group and a superior group based on their fitness value. The inferior group uses the global mutation model. The superior group employs the local mutation model. Subsequently, crossover and selection operations are employed for the next generation. An adaptive strategy is also proposed to automatically find good parameters in the GDE algorithm. To validate the performance of the GDE algorithm, 13 numerical benchmark functions are tested. The simulation results indicate that the approach is effective and efficient. In the second part, we apply the GDE algorithm to function-link fuzzy system (FLFS) optimization.

For structure learning, an agglomerative clustering algorithm is proposed to find the optimal number of fuzzy rules. For parameter learning, we use symbiotic learning method and GDE algorithm. The population is separated as subpopulations. Each subpopulation performs GDE algorithm to search the optimal parameter. The FLFS model with GDE learning algorithm (FLFS-GDE) is applied in real world prediction problems. Results of this dissertation demonstrate the effectiveness of the proposed methods.



Acknowledgement

本篇論文能夠順利完成，首先要感謝兩位指導教授 – 林進燈老師與張志永老師。在二位教授豐富的學識、殷勤的教導及嚴謹的督促下，使我學習到許多的寶貴知識及在面對事情中應有的處理態度、方法，並且在研究與投稿論文的過程中，二位教授有許多深入的見解及看法且對於斟酌字句、思慮周延，更是我該學習的目標。師恩好蕩，指導提攜，銘感於心。同時也要感謝陶金旺教授、林正堅教授、蘇木春教授、楊谷洋教授等，在電機資訊工程領域各執牛耳的論文口試委員，能於百忙之中蒞臨指導，給予最寶貴的意見，使得本論文內容上更加完善。

在艱辛的求學路上，謝謝這一路伴隨的學長和學弟們。感謝仕宇學長、東霖學長、肇廷學長及君玲學姐，因為你們的提點，讓我在研究上體悟更深成的道理，也感謝 Bio-CI Group 的成員：洋印與時慧，因為你們的存在，讓博士四年半的生活更加多采多姿，也祝福你們在未來的博士班口試順利。

特別要感謝我的父親、母親、大姐、二姐、三姐，在這段日子中不斷的給予支持及鼓勵，讓我能夠專心於研究的工作並完成博士學位。最後誠摯地以本論文研究成果獻給我的師長、父母、家人及所有的朋友們。

韓 明 峰

民國 102 年 1 月 21 日

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Evolutionary algorithms (EAs)[1-6] are population based stochastic optimization methods that are inspired by Darwin's Theory of Evolution. EAs are able to deal with difficult objective functions which are, e.g., discontinuous, non-convex, multi-modal, non-linear and non-differentiable functions. Since engineering, economic and scientific problems include such difficult objectives, EAs have become popular optimization tools during the last couple of decades.

The optimization process of the EAs usually adopt stochastic search techniques that work with a set of individuals instead of a single individual, and use some evolution operators to naturally produce offsprings for the next generation. These algorithms include genetic algorithm (GA)[7-8], evolutionary programming (EP)[9-10], evolution strategies (ES)[11], particle swarm optimization (PSO)[12-13] and differential evolution (DE)[14-15] which are famous, effectual and classical search techniques.

The GA is a powerful optimization tool based on biological evolution mechanism and natural selection. This algorithm was first proposed and investigated by John Holland in 1973. The main idea of the GA follows the natural selection principle of selecting fittest individuals for the next generation and explores the relevant search space according to the evolutionary computing strategies. In the GA, chromosome is represented by a binary

bit-string. Generally, the initial population of GA is generated code “1” or“0” randomly for each design variable. Offsprings (new population) are produced by Reproduction. The Reproduction usually involves crossover and mutation. Crossover is the process of combining genetic building blocks from two or more parent vectors to form one or more new offspring. Mutation is the process of injecting random noise into offspring vectors to form a slightly different offspring individual, thereby increasing the genetic diversity of the population.

Evolution strategies (ES) were developed by Rechenberg and Schwefel[11]. This algorithm is an effective continuous function optimizer. In evolution process, ESs perform mutation operator as main operator to produce offspring. After mutating and evaluating all λ children, the (μ, λ)-ES selects the best μ children to become the next generation’s parents. Alternatively, the $(\mu + \lambda)$-ES populates the next generation with the best μ vectors from the combined parent and child populations. The special case $(\mu + 1)$ is also referred to as steady-state ES.

A new population-based evolutionary algorithm, called particle swarm optimization (PSO), was proposed by Kennedy and Eberhart [12] in 1995. The population in PSO is referred to as a swarm. The PSO is based on simulations of social behaviors such as fish in a school, birds in a flock etc. A swarm in PSO consists of a number of particles. Each particle represents a potential solution of the optimization task. All of the particles iteratively discover a probable solution. Each particle moves to a new position according to the new velocity and the previous positions of the particle. The PSO has faster convergence than GA and ES to over a small number of generations.

In recent years, the DE algorithm is interested by researchers [14-35] among the EAs. The DE algorithm, proposed by Storn and Price [14-15] in 1998, is an efficient and effective global optimizer in the continuous search domain. It has been shown to perform better than the GA, ES and PSO over several numerical benchmarks [14-15, 19, 30, 34]. The DE

algorithm employs the difference of two randomly selected individuals as the source of random variations for the mutation operation. Subsequently, crossover and selection operations are used for generating offsprings. Many studies have applied the DE algorithm to difficult optimization problems and achieved better solutions[19, 28, 31]. However, the stagnation problem has been identified that the DE algorithm occasionally stops proceeding toward the global optimum [18-19]. The reason for stagnation problem is the limitation of the mutation operation model. In the DE algorithm, the mutation operation model always favors the exploration ability (DE/rand strategy) or the exploitation ability (DE/best strategy), which easily results in the blind search in individual space or the insufficient diversity in population. In order to deal with this problem, previous studies have proposed ideas to improve the mutation operation model. In [28, 31], the researchers have proposed a modified differential evolution (MODE) algorithm for an adaptive neural fuzzy network and locally recurrent neuro-fuzzy system optimazation. This MODE algorithm provides a convex type mutation model and cluster-based scheme to increase the diversity of the population. The concept of the tradeoff between the exploration ability and exploitation ability was proposed by Das et al.[18]. They designed a novel mutation model, called neighborhood-based mutation operation, to handle stagnation problem. In their paper, they utilized new mutation strategy and ring topology of neighborhood to find potential individuals in population. However, a single evolution model may not be suitable for various problems [21, 24, 27]. Therefore, other researchers which combine with other learning methods have proposed for solving the stagnation problem. Rahnamayan et al. [27] combined a opposition-based learning method and the DE algorithm, called opposition-based differential evolution (ODE). The ODE employs opposition-based optimization to choose the better solutions by simultaneously checking fitness of the opposite solution in the current population. The ODE possesses successfully increases diversity of the population. A combination of one-step k-Means clustering and multi-parent crossover operation in the DE algorithm was proposed

by Cai et al. [21]. Their method enhances the performance of the DE algorithm and balances the exploration ability and the exploitation ability in the evolutionary process. Noman and Iba [24] proposed an adaptive local search (ALS) algorithm to increase exploitation ability in the DE algorithm. The ALS algorithm uses a simple hill-climbing algorithm to adaptively determine the search length and effectively explore the neighborhood of each individual. Ali and Pant [36] applied a Cauchy mutation to improve the performance of the DE algorithm. The Cauchy mutation using Cauchy distribution randomly forces solutions to move to some other position. This method efficiently increases the probability of searching potential solutions in the DE algorithm. A combination of the fuzzy adaptive PSO algorithm and the DE algorithm, called FAPSO-DE model, was proposed by Niknam et al. [37]. They utilize two evolution processes to balance the exploration ability and exploitation ability for economic dispatch problems.

Unlike above mentioned studies, this dissertation proposes a new idea to solve the stagnation problem. This idea employs the inherent properties of the DE algorithm without depending on other learning algorithms. The idea combines two classical mutation strategies instead of a single mutation model. The two mutation strategies are composed of the DE/rand/bin operation and the DE/best/bin operation. The DE/rand/bin has powerful exploitation ability; and the DE/best/bin has efficient exploration ability. This dissertation uses the two operations to tradeoff between the exploration ability and the exploitation ability for solving the stagnation problem.

In this dissertation, a group-based differential evolution (GDE) algorithm is proposed for numerical optimization problems. The GDE algorithm provides a new process using the DE/rand/bin model and the DE/best/bin model in mutation operation. Initially, all individuals in population are grouped into an inferior group and a superior group based on their fitness value. The inferior group uses the DE/rand/bin mutation model for globally searching potential solutions and for maintaining the diversity of the population. The superior group

employs the DE/best/bin mutation model to efficiently search the neighborhood of the current best solution. Subsequently, crossover and selection operations are employed for the next generation. An adaptive strategy is also proposed in this dissertation. This strategy uses successful information to automatically tend to good parameters (factor F and crossover rate CR). It is thus helpful to enhance the robustness of the GDE algorithm. In order to validate the performance of the GDE algorithm, 13 well-known numerical benchmark functions with low dimensional problems and high dimensional problems are tested. Simulation results indicate that our approach is efficient. Comparison with other advance evolutionary algorithms, the proposed GDE algorithm performs better performance.

In addition, we also apply the proposed GDE algorithm to practical problems based on functional-link fuzzy systems (FLFS) optimization. Initially, the FLFS has no rules. The fuzzy rules are automatically generated by an agglomerative clustering algorithm. The agglomerative clustering algorithm (ACA) determines the optimal number of fuzzy rules for the FLFS. Subsequently, all free parameters are learned by the GDE algorithm for the FLFS optimization. During evolution process, the scale fact and crossover are adjusted by adaptive parameter tuning strategy. In the simulation, five prediction problems are tested to validate the performance of the proposed functional-link fuzzy system with the GDE algorithm (FLFS-GDE). The proposed FLFS-GDE model shows better prediction performance than other methods.

The overall objective of this dissertation is to develop a novel evolutionary algorithm and its related application. Organization and objectives of each chapter in this dissertation are as follows.

In Chapter 2, we introduce a basic DE algorithm and its evolution process. The DE algorithm employs the difference of two randomly selected individuals as the source of random variations for the mutation operation. Subsequently, crossover and selection operations are used for the next generation.

In Chapter 3, we present a new differential evolution algorithm, called group-based differential evolution algorithm for global optimization problems. This algorithm employs two mutation strategies instead of a single mutation model to tradeoff between the exploration ability and the exploitation ability for solving the stagnation problem. Furthermore, an adaptive strategy is also proposed to enhance the robustness of the GDE algorithm by an automatic process for finding good parameters. 13 well-known numerical benchmark functions are tested for simulations. The result shows significant differences between the proposed GDE algorithm and other methods.

In Chapter 4, the proposed GDE algorithm is applied to FLFS optimization for prediction problems. The learning process consists of rule generation phase and parameter learning phase. The rule generation phase can determine the optimal number of fuzzy rules using the agglomerative clustering algorithm. The parameter learning phase combines a subpopulation symbiotic evolution and a GDE algorithm. Initially, population is separated as many subpopulations according to the number of fuzzy rules. Each subpopulation performs the GDE algorithm for parameter learning. We also compare our method and other methods in simulations. Finally, conclusions and future works are summarized in Chapter 5.

Chapter 2

Differential Evolution

This section introduces a complete DE algorithm. The process of the DE algorithm, likes other EAs, produces offsprings for next generation by the mutation operation, the crossover operation and the selection operation. Figure 2.1 shows a standard flow chart of the DE algorithm.

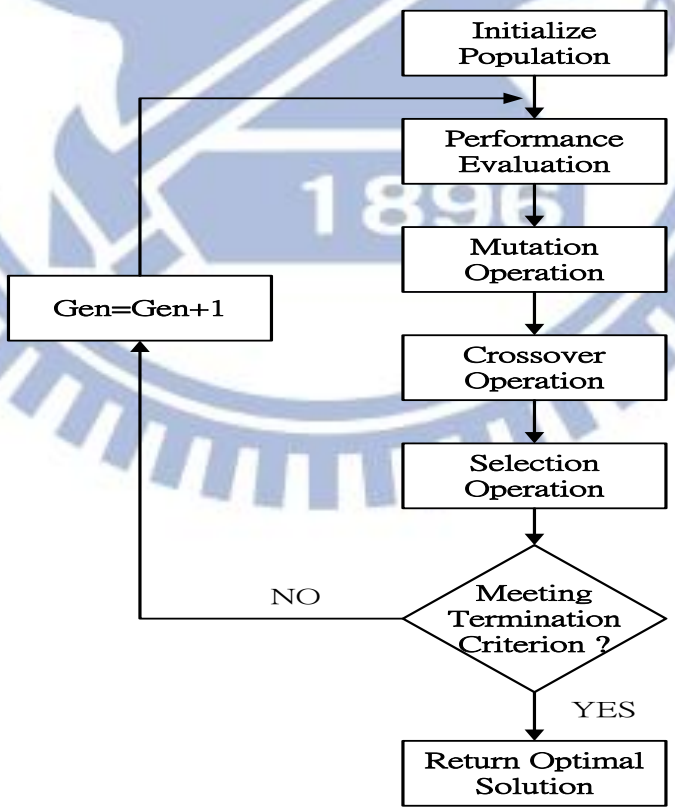


Figure 2.1 : The flow chart of the DE algorithm. Gen is the generation counter.

Initially, a population of *NP* *D*-dimensional parameter vectors which represents the candidate solutions (individuals) is generated by uniformly random process. All individuals and search space are constrained by the prescribed minimum $\mathbf{X}_{\min} = (x_{1,\min}, x_{2,\min}, ..., x_{D,\min})$ and maximum $\mathbf{X}_{\max} = (x_{1,\max}, x_{2,\max}, ..., x_{D,\max})$ parameter bounds. A simple representation of *i*-th individual at the current generation *Gen* is shown as follows：

$$\mathbf{X}_{i,Gen} = (x_{i,1,Gen}, x_{i,2,Gen}, x_{i,3,Gen}, ..., x_{i,D-1,Gen}, x_{i,D,Gen}) \quad . \tag{1}$$

After Initial population production with *NP* individuals, fitness evaluation process measures quality of individuals to calculate the performance. The succeeding steps include the mutation operation, the crossover operation and the selection operation are explained in the following.

Mutation Operation

Each individual in the current generation is allowed to breed through mating with other randomly selected individuals from the population. This process randomly selected a parent pool of three individuals is formed to produce an offspring. Specifically, for each individual $\mathbf{X}_{i,gen}$, $i = 1, 2, ..., NP$, where *gen* denotes the current generation, *NP* is population size, three random individuals, $\mathbf{X}_{r1,gen}$, $\mathbf{X}_{r2,gen}$, $\mathbf{X}_{r3,gen}$, $\mathbf{X}_{r4,gen}$ and $\mathbf{X}_{r5,gen}$ are selected from the population such that *r*1, *r*2, *r*3, *r*4 and *r*5 ∈ { 1, 2, . . . , *NP* } and $i \neq r1 \neq r2 \neq r3 \neq r4 \neq r5$. This way, a parent pool of four individuals is formed to produce an offspring. The following are different mutation strategies frequently used in the literature:

$$\text{DE/rand/bin:} \quad \mathbf{V}_{i,gen} = \mathbf{X}_{r1,gen} + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) \tag{2}$$

$$\text{DE/best/bin:} \quad \mathbf{V}_{i,gen} = \mathbf{X}_{gbest,gen} + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) \tag{3}$$

$$\text{DE/target-to-best/bin:}\quad \mathbf{V}_{i,gen} = \mathbf{X}_{r1,gen} + F(\mathbf{X}_{gbest,gen} - \mathbf{X}_{r1,gen}) + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) \tag{4}$$

$$\text{DE/rand/bin/2:}\quad \mathbf{V}_{i,gen} = \mathbf{X}_{r1,gen} + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) + F(\mathbf{X}_{r4,gen} - \mathbf{X}_{r5,gen}) \tag{5}$$

$$\text{DE/best/bin/2:}\quad \mathbf{V}_{i,gen} = \mathbf{X}_{gbest,gen} + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) + F(\mathbf{X}_{r4,gen} - \mathbf{X}_{r5,gen}) \tag{6}$$

where F is scaling factors $\in [0,1]$, $\mathbf{X}_{gbest,gen}$ is the best-so-far individual (i.e., $\mathbf{X}_{gbest,gen}$ keeps best fitness value up to now in the population). For various problems, the DE algorithm usually employs different mutation strategy. The DE/rand/bin/ mutation and DE/rand/bin/2 mutation which have more exploration ability are suitable for multimodal problems. The "DE/best/bin", "DE/best/bin/2" "DE/target-to-best" mutations which consider the current best information in generation are more suitable for unimodal problems.

Crossover Operation

After the mutation operation, The DE algorithm uses a crossover operation, often referred to as discrete recombination, in which the mutated individual $\mathbf{V}_{i,gen}$ is mated with $\mathbf{X}_{i,gen}$ and generates the offspring $\mathbf{U}_{i,gen}$. The elements of an individual $\mathbf{U}_{i,gen}$ are inherited from $\mathbf{X}_{i,gen}$ and $\mathbf{V}_{i,gen}$, which are determined by a parameter called crossover probability (CR ∈ [0, 1]), as follows:

$$\mathbf{U}_{i,d,gen} = \begin{cases} \mathbf{V}_{i,d,gen}, & \text{if rand}(d) \le \text{CR} \\ \mathbf{X}_{i,d,gen}, & \text{if rand}(d) > \text{CR} \end{cases} \tag{5}$$

where d = 1, 2, . . . , D denotes the dth element of individual vectors, D is total element of

individual vector, r (d) ∈ [0, 1] is the dth evaluation of a random number generator. Figure 2.2 gives an example of the crossover mechanism for 7-dimensional vectors.

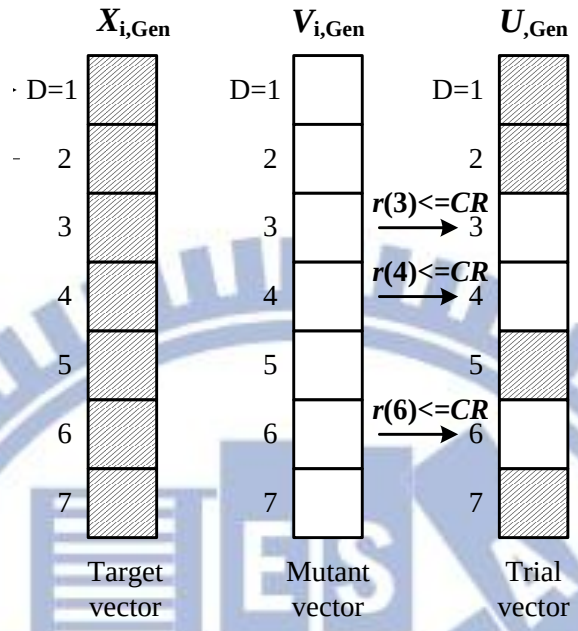


Figure 2.1 : Illustration of the crossover process for NP=7 parameters.

Selection Operation

The DE algorithm applies selection operation to determine whether the individual survives to the next generation. A knockout competition is played between each individual $\mathbf{X}_{i,gen}$ and its offspring $\mathbf{U}_{i,gen}$, and the winner is selected deterministically based on objective function values and is then promoted to the next generation. The selection operation is described as

$$\mathbf{X}_{i,gen+1} = \begin{cases} \mathbf{X}_{i,gen}, & \text{if fitness}(\mathbf{X}_{i,gen}) < \text{fitness}(\mathbf{U}_{i,gen}) \\ \mathbf{U}_{i,gen}, & \text{otherwise} \end{cases} \tag{6}$$

where fitness(z) is the fitness value of individual z. After the selection operation, the population obtains better fitness value or remains the same fitness value, but never deteriorates.

Chapter 3

Group-Based Differential Evolution

3.1 A GDE Algorithm

In the DE algorithm, mutation operation which leads a successful evolution performance is a principal operator. For various problems, we often employ different mutation strategy in the DE algorithm. However, choosing suitable mutation strategy which deals with a practical problem is difficult. Therefore, we propose the GDE algorithm with the exploration ability and the exploitation ability, which combines two mutation strategies to solve practical problems. A flow chart of the GDE algorithm is shown in Figure 3.1.

In first step of the GDE algorithm, a population of NP D-dimensional individuals is generated by uniformly random process, and evaluated for the fitness value of all individuals. A sorting process arranges all individuals based on their fitness value as $fitness_1 < fitness_2 < ... < fitness_{NP-1} < fitness_{NP}$ for minimum objective problems. According to fitness value, all individuals are partitioned into an inferior group and a superior group, called the Group A and the Group B. The Group A, including *NP/2* worse individuals, performs global search to increase the diversity of the population and widely find potential solutions. Other *NP/2* individuals for the Group B perform local search to actively detect better solutions nearby current best solution. A complete mutation operation is shown for the Group A and the Group B as follows.

$$\text{Group A: } \mathbf{V}_{i,gen} = \mathbf{X}_{i,gen} + F_a(\mathbf{X}_{r1,gen} - \mathbf{X}_{r2,gen}) \tag{7}$$

$$\text{Group B: } \mathbf{V}_{i,gen} = \mathbf{X}_{gbest,gen} + F_b(\mathbf{X}_{r3,gen} - \mathbf{X}_{r4,gen}) \tag{8}$$

Where F_a and F_b are scale factors, $\mathbf{X}_{r1,gen}$, $\mathbf{X}_{r2,gen}$, $\mathbf{X}_{r3,gen}$ and $\mathbf{X}_{r4,gen}$ are random selected from the population, and $i \neq r1 \neq r2 \neq r3 \neq r4$, the $\mathbf{X}_{gbest,gen}$ is the best-so-far individual in the population.

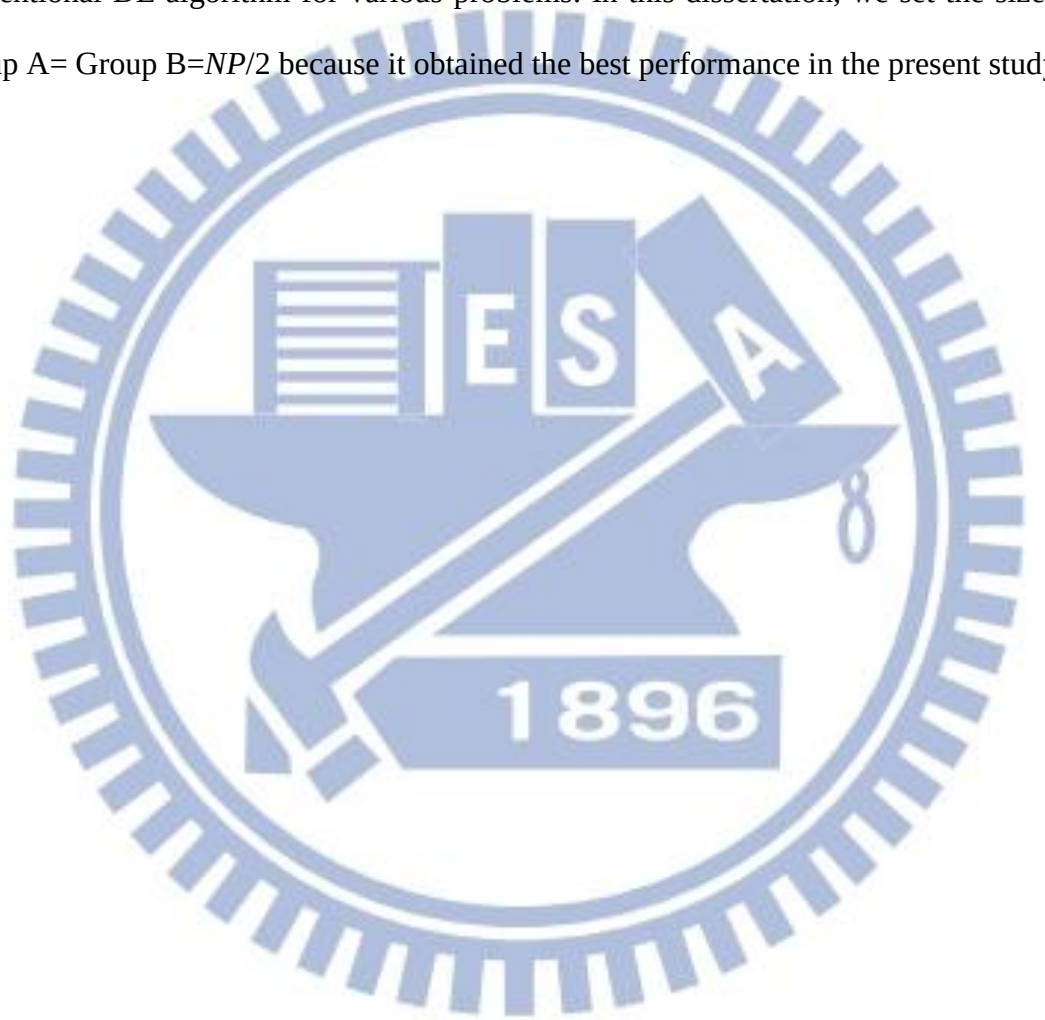
After mutation operation, The GDE algorithm uses a crossover operation, often referred to as discrete recombination, in which the mutated individual $\mathbf{V}_{i,gen}$ is mated with $\mathbf{X}_{i,gen}$ and generates the offspring $\mathbf{U}_{i,gen}$. Equation 9 presents the crossover operation for the Group A and the Group B. If the random number rand(d) is smaller than the CR value, the variable of the mutated individual $\mathbf{V}_{i,d,gen}$ is chosen to the variable of the trial vector $\mathbf{U}_{i,d,gen}$. Otherwise, the variable of the target vector $\mathbf{X}_{i,d,gen}$ is selected to the variable of the trial vector $\mathbf{U}_{i,d,gen}$.

$$\mathbf{U}_{i,d,gen} = \begin{cases} \mathbf{V}_{i,d,gen}, & \text{if rand}(d) \le \text{CR} \\ \mathbf{X}_{i,d,gen}, & \text{if rand}(d) > \text{CR} \end{cases} \tag{9}$$

where d = 1, 2, . . . , D denotes the dth element of individual vectors, D is total element of individual vector, CR ∈ [0, 1],rand(d) ∈ [0, 1] is the dth evaluation of a random number generator. The mutation and crossover operators are used to diversify the search space in terms of the optimization problems.

Selection operation is used to determine whether the individual survives to the next generation. A knockout competition is played between each individual $\mathbf{X}_{i,gen}$ and its offspring $\mathbf{U}_{i,gen}$, and the winner is selected deterministically based on objective function values and is then promoted to the next phase. After the selection operation, the population gets better or remains the same in fitness value, but never deteriorates.

The conventional DE only utilizes DE/best or DE/rand mutation to deal with problems. The proposed GDE algorithm employed two mutation operations to maintain useful diversity in the population and increase the search capability. It is worth noting that DE/best/bin and DE/rand/bin are a special case in the proposed GDE algorithm when population = Group A and population = Group B. Thereby, the proposed GDE algorithm has more variety than conventional DE algorithm for various problems. In this dissertation, we set the size of the Group A= Group B=*NP*/2 because it obtained the best performance in the present study.



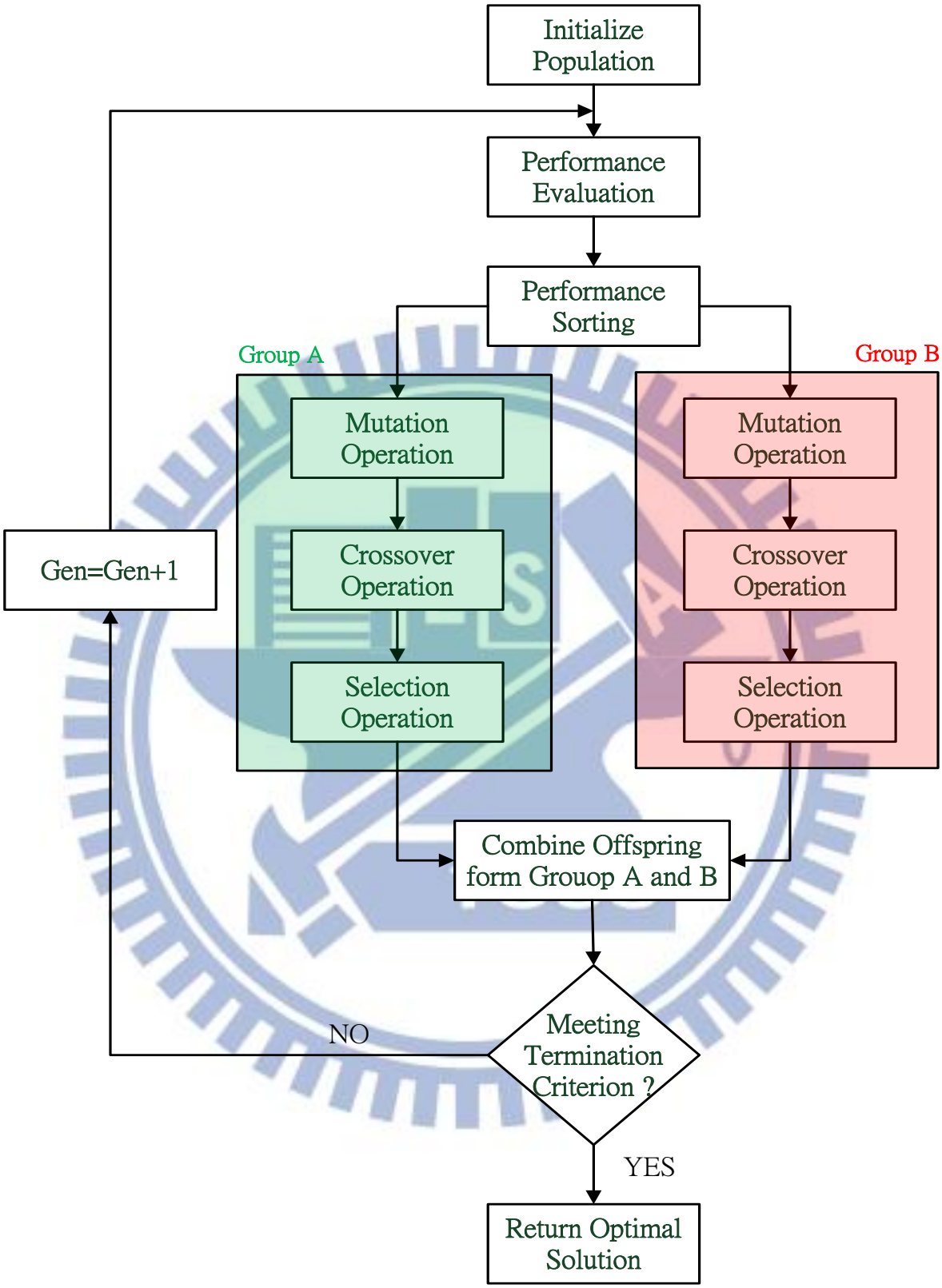


Figure 3.1: The flow chart of the proposed GDE algorithm. GEN is the generation counter.

3.2 A Self-Adaptive Parameter Tuning Strategy

Parameter control which can directly influence the convergence speed and search capability is an important task in the EAs [12, 19]. However, conventional DE algorithm always used trial-and-error method for choosing suitable parameter requires multiple optimization runs. Based on this consideration, different adaptive or self-adaptive mechanisms [16, 18, 22, 25, 32] have been recently introduced to dynamically update the control parameters without a user's prior knowledge of the relationship between the parameter setting and the characteristics of optimization problems. In this section, we propose a generalized self-adaptive approach to control parameter the F and the CR for the Group A (inferior) and the Group B (superior). The concept of the proposed parameter tuning strategy is shown in Figure 3.2. The generalized scheme is designed as follows：

(1) Assume new parameters $G_i \in [G_{\min}, G_{\max}]$, $1 \le i \le NP$. The G_i is composed of F_i and CR_i for individual x_i. We set a initial center G_{center}

(2) Set $g = g + 1$ and randomly generate G_i by Gaussian distribution (G_{center} , 0.2) for every individual x_i.

(3) After evolution process, the G_i that is able to make the offspring $x_{i,g+1}$ of x_i to successfully enter the next generation. That is, a good parameter value G_i will be marked and recorded in our algorithm. The successful parameter value $G_{\text{success}}(k)$ and fitness improvements $\delta(k)$, where k=1,2,...,N_{g+1}

$$G_{\text{sucess}}(k) = [F_{\text{sucess},k}, CR_{\text{sucess},k}] \tag{10}$$

$$\delta(k) = \left(fitness(x_{k,g+1}) - fitness(x_k) \right)^2 \tag{11}$$

(4) Update the parameter center according to

$$G_{center} = (1-w)\cdot G_{center} + w\cdot G_{center,g+1} \tag{12}$$

where the weight w is determined by

$$w = \frac{N_{g+1}}{N_g + N_{g+1}} \tag{13}$$

and $G_{center,g+1}$ is the weighted mean of values in G_{sucess} :

$$G_{center,g+1} = \sum_{k=1}^{N_g} G_{\text{sucess}}(k)\cdot \frac{\delta(k)}{\sum_{k=1}^{N_g}\delta(k)} \tag{14}$$

(5) If $N_{g+1} \le N_g$, then update the N_{g+1} as follows

$$N_{g+1} = 0.9N_g \tag{15}$$

(6) Go to Step 2 for the next generation until a stopping criterion is satisfied.

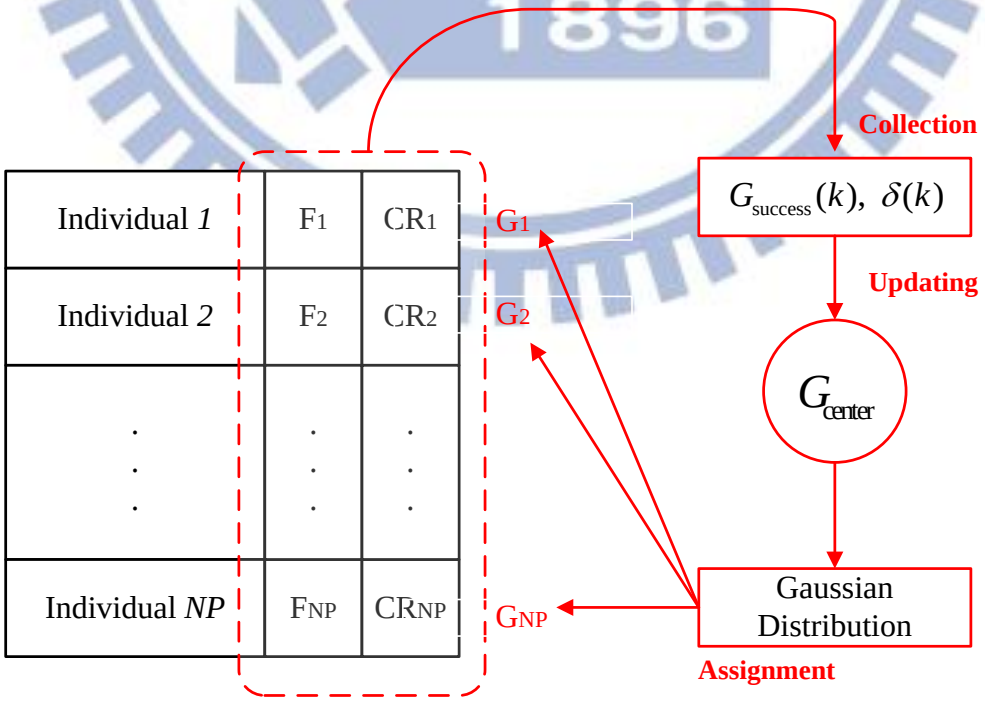


Figure 3.2 : A concept of the self-adaptive parameter tuning strategy.

3.3 Simulation

In order to verify the performance of proposed algorithm, a set of thirteen classical benchmark test functions [38-39] is used in this simulation. The analytical form of these functions is given in section 3.3.1. The GDE algorithm is compared with three classic DE algorithms, including the DE/rand/bin, the DE/best/bin and the DE/target-to-best/bin algorithms. In all simulations, we set the parameters of the GDE algorithm to be fixed, initial F_a= 0.5, initial F_b =0.8, initial CR_a=0.9 ,initial CR_b =0.9. The parameter setting for three classic DE algorithms is recommended by other papers as follows. For DE/rand/bin model, the F=0.5 and the CR=0.9 [15, 22, 32]；For DE/bes/bin model, the F=0.8 and the CR=0.9 [18]；For DE/target-to-best/bin, the F=0.8 and the CR=0.9[20].

Many papers have used the same parameter setting to solve their problems. In this simulation, we set the population size NP to be 100 and 400 in the case of D = 30, and D = 100, respectively. All results reported in this section are obtained based on 50 independent runs. In addition, Section 3.3.3 demonstrates significant difference results based on statistical comparison process. A complete comparison with other evolutionary algorithms, such as RMEA[45],CEP[30], ALEP[43], BestLevy[43], NSDE[40], RTEP[44], jDE[33,47], SaDE[26], ODE[27], SaCDE[16], DEGL[18] and JADE[22,46], is presented in Section 3.3.4.

3.3.1 Test Functions

In this section, we introduce thirteen numerical functions for verifying the performance of proposed GDE algorithm. Based on their properties, the functions can be divided into two problems as unimodal function problem and multimodal function problem. f_1– f_4 are continuous unimodal functions. f_5 is a discontinuous step function, and f_6 is a noisy quartic function. f_7 is the Rosenbrock function which is multimodal function problem for $D > 3$ [39]. f_8– f_{13} are multimodal and the number of their local minima increases exponentially with the problem dimension [40]. In addition, f_8 is the only bound-constrained function investigated in this paper. All these functions have an optimal value at zero. Completed functions are described as follows:

(1) Function 1 ：Sphere function

$$f_1 = \sum_{i=1}^{D} (x_i)^2 \quad , \quad -100 \le x_i \le 100$$

(2) Function 2 ：Schwefel’s problem_a

$$f_2 = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i| \quad , \quad -10 \le x_i \le 10$$

(3) Function 3 ：Schwefel’s problem_b

$$f_3 = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_i \right)^2 \quad , \quad -100 \le x_i \le 100$$

(4) Function 4 ：Schwefel’s problem_c

$$f_4 = \max_i |x_i| \quad , \quad -100 \le x_i \le 100$$

(5) Function 5 ：Schwefel’ s problem_d

$$f_5 = \sum_{i=1}^{D} (x_i + 0.5)^2 , \quad -100 \le x_i \le 100$$

(6) Function 6 ：Schwefel’ s problem_e

$$f_6 = \sum_{i=1}^{D} i x_i^4 + rand[0,1) , \quad -1.28 \le x_i \le 1.28$$

(7) Function 7 ：Rosenbrock’ s function

$$f_7 = \sum_{i=1}^{D} \left[100(x_{i+1} - x_i^2) + (x_i - 1)^2 \right] , \quad -30 \le x_i \le 30$$

(8) Function 8 ：Schwefel’ s function

$$f_8 = \sum_{i=1}^{D} -x_i \sin \sqrt{|x_i|} + D \cdot 418.98288727243369 , \quad -500 \le x_i \le 500$$

(9) Function 9 ：Rastrigin’ s function

$$f_9 = \sum_{i=1}^{D} [x_i - 10\cos(2\pi x_i) + 10] , \quad -5.12 \le x_i \le 5.12$$

(10) Function 10 ：Ackley’ s function

$$f_{10} = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e ,$$

$$-32 \le x_i \le 32$$

(11) Function 11 ：Griewank’ s function

$$f_{11}=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1\ ,\qquad -600\le x_i\le 600$$

(12) Function 12 ：Generalized penalized function_1

$$f_{12}=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_D-1)^2\right\}$$
$$+\sum_{i=1}^{D}u(x_i,10,100,4)$$

where

$$y_i=1+\frac{1}{4}(x_i+1),$$

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & ,\ \text{if}\ x_i>a\\ k(-x_i-a)^m & ,\ \text{if}\ x_i<-a\\ 0 & ,\ \text{otherwise}\end{cases}$$

$$-50\le x_i\le 50$$

(13) Function 13 ：Generalized penalized function_2

$$f_{13}=\frac{1}{10}\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_D-1)^2\left[1+\sin^2(2\pi x_D)\right]\right\}$$
$$+\sum_{i=1}^{D}u(x_i,10,100,4)$$

where

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & ,\ \text{if}\ x_i>a\\ k(-x_i-a)^m & ,\ \text{if}\ x_i<-a\\ 0 & ,\ \text{otherwise}\end{cases}$$

$$-50\le x_i\le 50$$

3.3.2 Low-Dimensional Problems

In this simulation, the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms are applied to low dimensional problems on 13 benchmark test functions. Table 3.1 and Table 3.2 show the detailed performance of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms, including the mean, best and worst performance over 50 independent runs. This table indicates that the GDE algorithm obviously achieves better performance than the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms on 13 benchmark test functions. Especially, the GDE algorithm searches the global optimal solution at zero on the Function 5 and the Function 11. Focus on three classical DE algorithms, the DE/target-to-best/bin algorithm often obtains a better performance than the DE/rand/bin and DE/best/bin algorithms on 13 benchmark test functions. The DE/rand/bin obtains obvious difference on the Function 11 and the Function 13 among three classical DE algorithms.

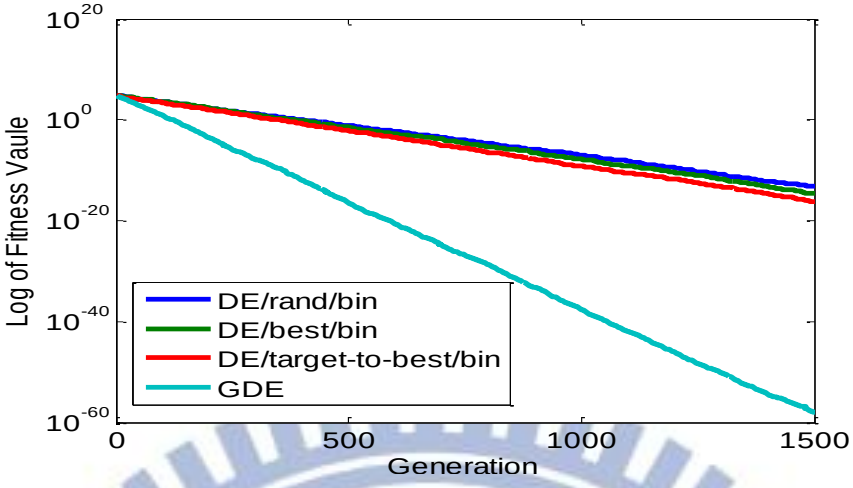
The learning curve of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms on 13 test function for low dimensional (D=30) problems is shown in Figure 3.3. This Figure presented that the GDE algorithm possesses speedier convergence than the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms on 13 benchmark test functions. An interesting case is shown in Figure 3(h) and Figure 3(i). The DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms are stopped at locally optimal solutions on the Function 9 and the Function 10. The GDE algorithm maintains a continued convergence to find the optimal solutions. It is shown that the proposed GDE algorithm successfully overcomes the stagnation problem for low dimensional problems.

Table 3.1: Experimental results (Function 1－Function 8)of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin for low dimensional problems (D=30), averaged over 50 independent runs.

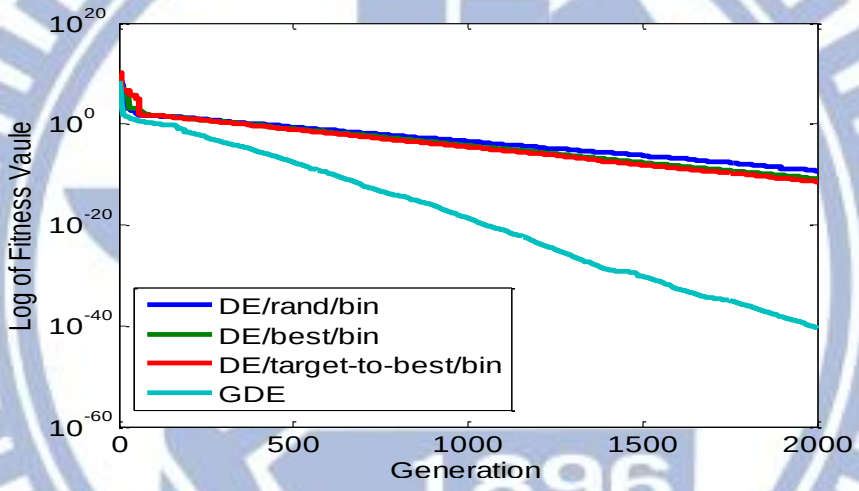
Function	Gen.	GDE	DE/rand/bin	DE/best/bin	DE/target－to－best/bin
		Mean (Best, Worst)			
f1	1500	1.83E–42 (9.61E–59, 9.15E–41)	2.53E–13 (5.37E–14, 1.16E–12)	4.51E–14 (2.30E–15, 1.56E–13)	4.84E–16 (7.17E–17, 1.76E–15)
f2	2000	4.02E–30 (3.86E–41, 1.37E–28)	2.93E–09 (5.42E–10, 8.45E–09)	7.82E–11 (1.75E–11, 3.00E–10)	2.11E–11 (3.84E–12, 6.81E–11)
f3	5000	1.13E–25 (9.22E–38, 5.53E–24)	3.78E–10 (3.72E–11, 1.93E–09)	3.77E–11 (3.43E–13, 7.58E–10)	3.18E–14 (1.96E–16, 1.60E–13)
f4	5000	6.67E–11 (2.43E–14, 2.59E–10)	2.17 E–02 (4.15E–13, 5.25E–01)	1.93E–09 (2.48E–11, 1.95E–08)	8.34E–11 (4.04E–14, 6.83E–10)
f5	1500	0.0E+00 (0.0E+00, 0.0E+00)	2.98E–13 (6.03E–14, 8.50E–13)	3.97E–14 (4.03E–15, 1.82E–13)	5.55E–16 (3.87E–17, 5.20E–15)
f6	3000	2.08E–03 (6.02E–04, 9.43E–03)	1.74E–01 (3.60E–03, 7.77E–01)	7.12E–03 (3.00E–03, 1.23E–02)	5.79E–03 (2.16E–03, 1.14E–02)
f7	3000	3.73E–07 (1.27E–19, 1.12E–05)	1.17E+00 (1.67E–05, 3.06E+00)	7.97E–01 (1.83E–11, 3.98E+00)	5.58E–01 (1.04E–13, 3.98E+00)
f8	1500	2.52E+00 (1.18E+02, 8.58E–04)	6.80E+03 (4.71E+03, 7.27E+03)	2.94E+03 (1.78E+03, 4.88E+03)	3.12E+03 (9.49E+02, 6.89E+03)

Table 3.2: Experimental results (Function 9 ─ Function 13)of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin for low dimensional problems (D=30), averaged over 50 independent runs.

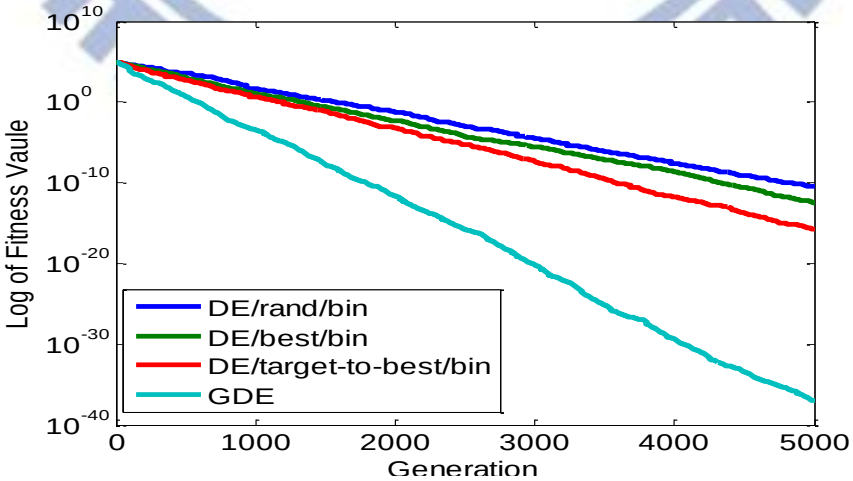
Function	Gen.	GDE	DE/rand/bin	DE/best/bin	DE/target‐to‐best/bin
		Mean (Best, Worst)			
f9	1500	5.68E–13 (0.0E+00, 6.86 E–12)	7.62E+01 (7.88E+00, 1.67 E+02)	4.55E+01 (2.28E+01, 7.36E+01)	1.71E+02 (1.33E+02, 2.13E+02)
f10	1500	9.69E–15 (7.99E–15, 3.28E–14)	1.68E–07 (7.25E–08, 3.31E–07)	5.59E–08 (2.08E–08, 2.16E–07)	6.64E–09 (2.47E–09, 1.67E–08)
f11	1500	0.0E+00 (0.0E+00, 0.0E+00)	1.08E–12 (5.87E–14, 1.38E–11)	8.31E–03 (6.32E–15, 5.65E–02)	5.86E–03 (0.0E+00, 2.21E–02)
f12	1500	1.50E–32 (1.34E–32, 4.06E–32)	3.81E–14 (1.66E–15, 2.84E–13)	1.03E–01 (8.03E–16, 2.06E+00)	2.69E–02 (3.60E–18, 5.19E–01)
f13	1500	1.70E–32 (1.57E–32, 6.8E–32)	3.17E–13 (2.82E–14, 1.76E–12)	2.63E–03 (2.49E–15, 1.09E–02)	1.08E–08 (2.86E–17, 5.41E–07)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

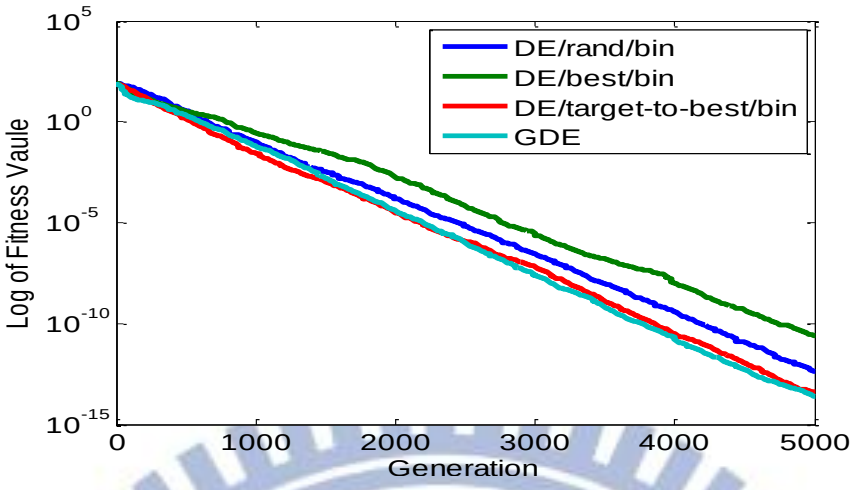
(a)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

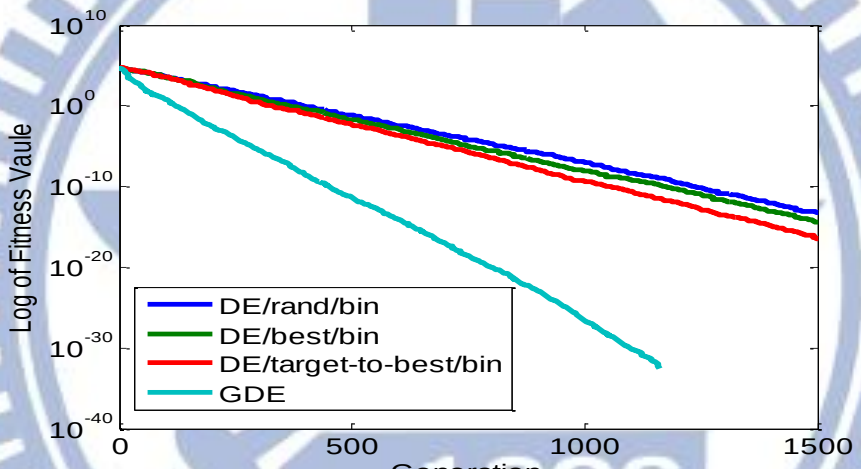
(b)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

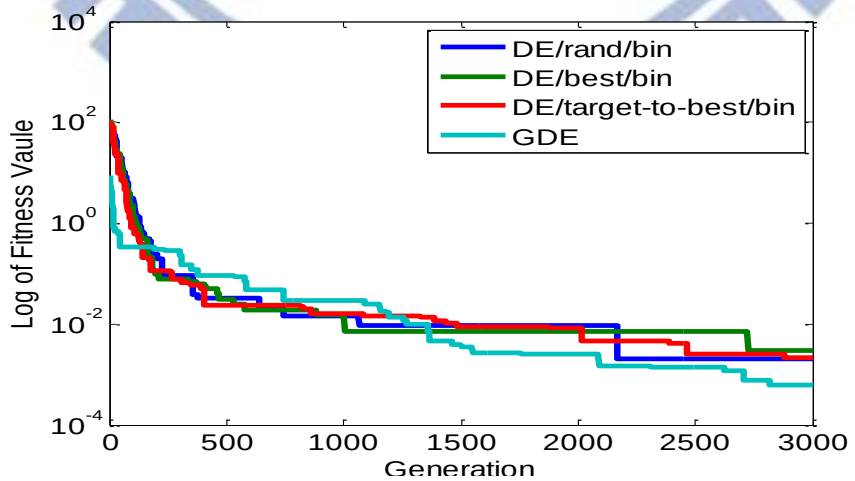
(c)


DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE
Log of Fitness Vaule
Generation

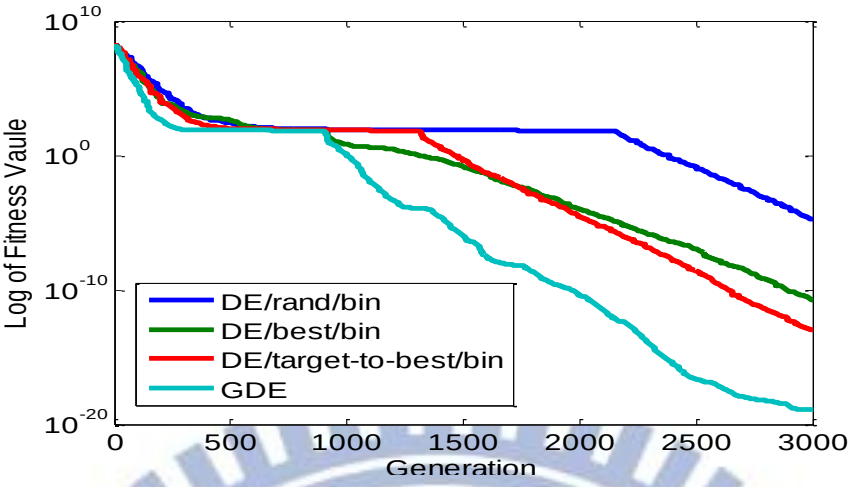
(d)


DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE
Log of Fitness Vaule
Generation

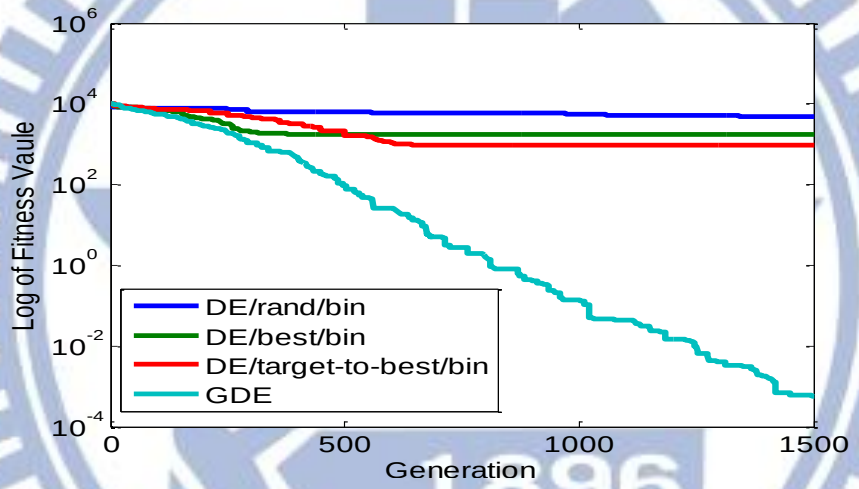
(e)


DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE
Log of Fitness Vaule
Generation

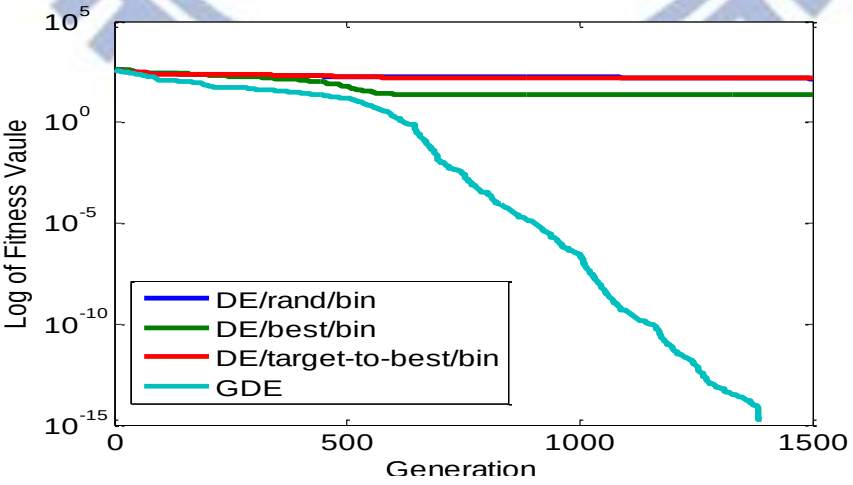
(f)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

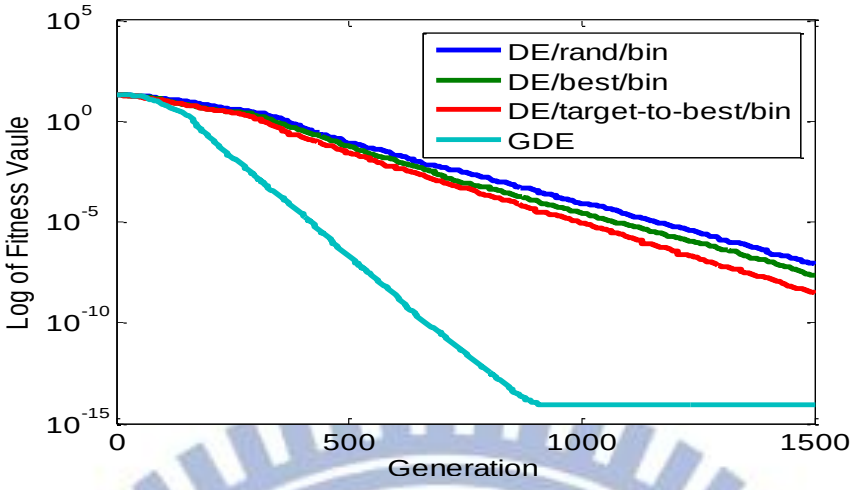
(g)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

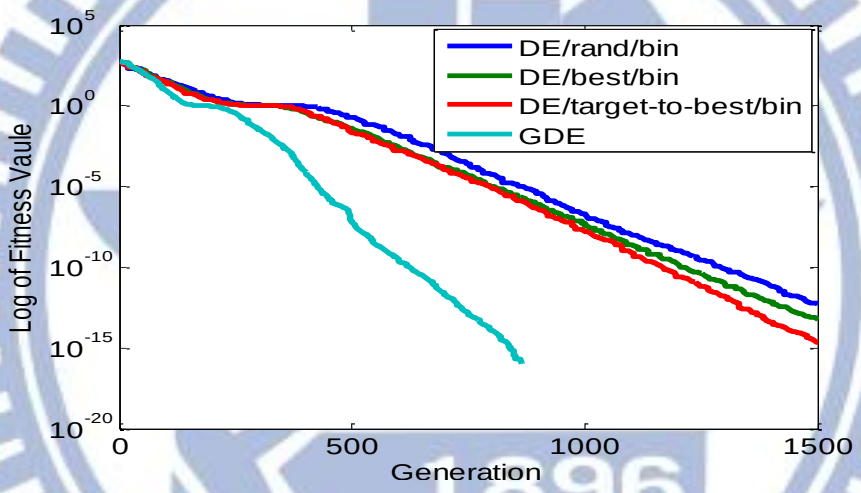
(h)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

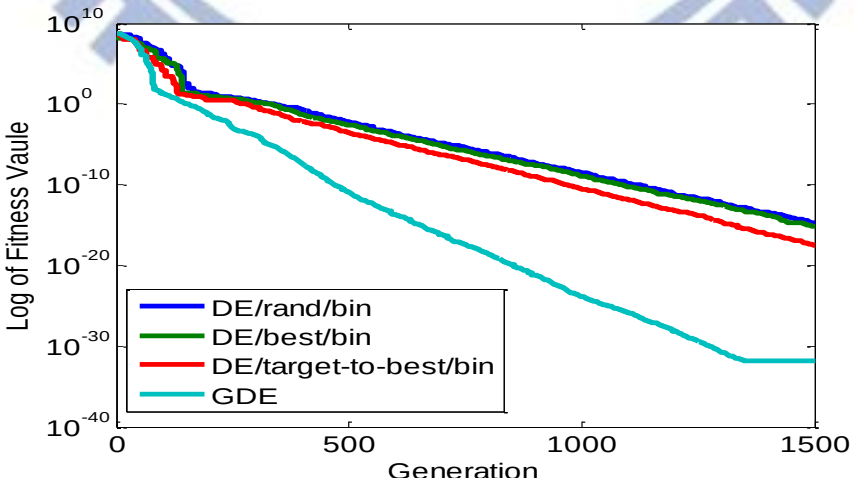
(i)


DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE
Log of Fitness Vaule
Generation
10^5
10^0
10^-5
10^-10
10^-15
0
500
1000
1500

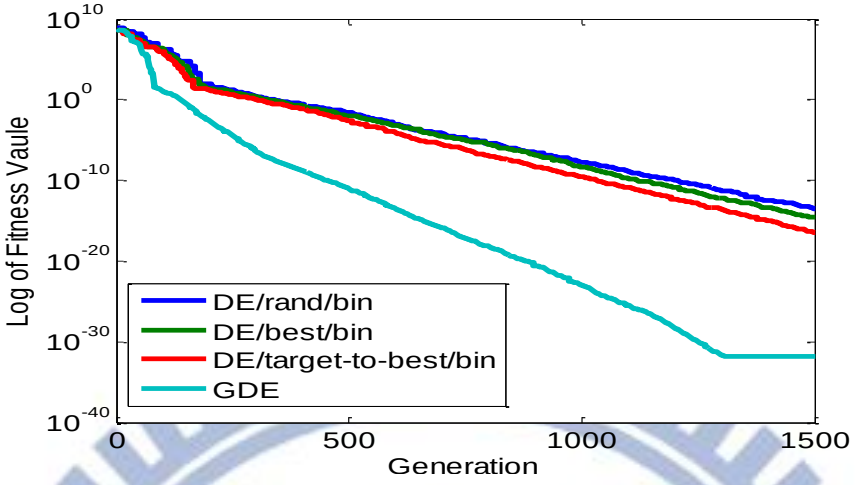
(j)


DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE
Log of Fitness Vaule
Generation
10^5
10^0
10^-5
10^-10
10^-15
10^-20
0
500
1000
1500

(k)


DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE
Log of Fitness Vaule
Generation
10^10
10^0
10^-10
10^-20
10^-30
10^-40
0
500
1000
1500

(l)



(m)

Figure 3.3. The best learning curve of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin on 13 test function for low dimensional (D=30) problems. (a) Function 1: f_1; (b) Function 2: f_2; (c) Function 3: f_3; (d) Function 4: f_4; (e) Function 5: f_5; (f) Function 6: f_6; (g) Function 7: f_7; (h) Function 8: f_8; (i) Function 9: f_9; (j) Function 10: f_{10}; (k) Function 11: f_{11}; (l) Function 12: f_{12}; (m) Function 13: f_{13}.

3.3.3 High-Dimensional Problems

In order to verify the capability of algorithm on high dimensional problems, the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms are applied to 13 benchmark test functions. Table 3.3 and Table 3.4 show the detailed performance of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms, including the mean, best and worst performance over 50 independent runs. Obviously, all algorithms are difficult to find optimal solutions caused by high dimensional problem. In Tables, the GDE algorithm obtains better performance than the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms on 13 benchmark test functions. Notice that the GDE algorithm efficiently searches a global optimal solution at zero on the Function 9 and the Function 11. Among three classical DE algorithms, the DE/target-to-best/bin algorithm obtains obvious difference on the Function 9 and the Function 11.

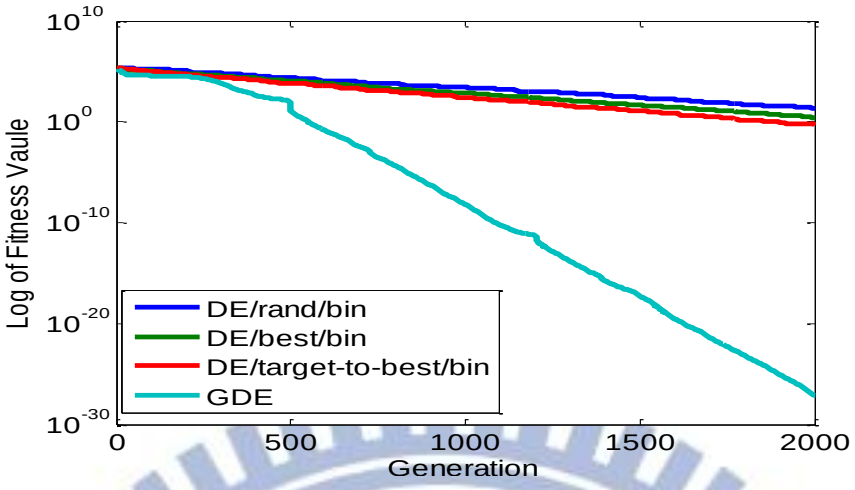
The learning curve of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms on 13 test function for high dimensional (D=100) problems is shown in Figure 3.4. In this Figure, the GDE algorithm also presents speedier convergent curves than other algorithms on high dimensional functions. The stagnation situation is also happened when the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms are performed in Figure 3(b), Figure 3(c), Figure 3(e), Figure 3(g) and Figure 3(i). The GDE algorithm continuously maintains a convergent curve on the Function 2, Function 3, Function 5, Function 7 and Function 9. In this paper, the simulation result show that the proposed GDE algorithm obviously achieves better performance and successfully overcomes the stagnation situation for low dimensional problems and low dimensional problems.

Table 3.3: Experimental results (Function 1 ─ Function 8) of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin for high dimensional problems (D=100), averaged over 50 independent runs.

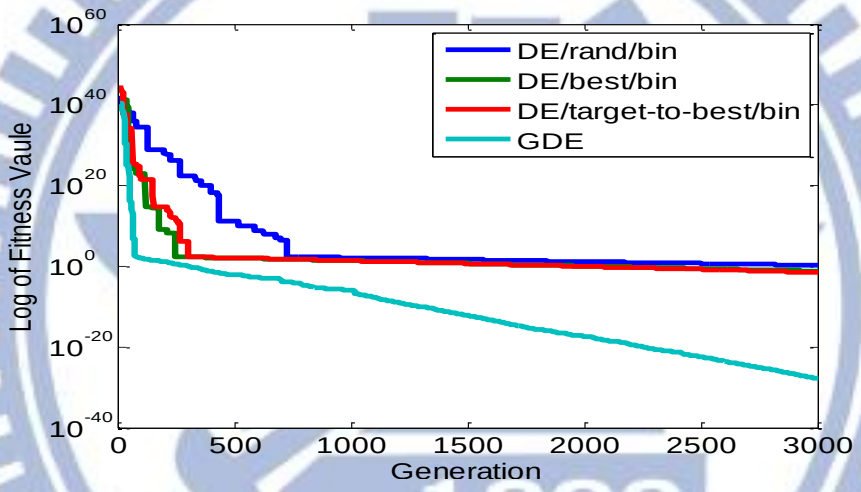
Function	Gen.	GDE	DE/rand/bin	DE/best/bin	DE/target‐to‐best/bin
		Mean (Best, Worst)			
f1	2000	4.95E–21 (8.68E–28, 9.07E–20)	3.71E+01 (2.14E+01, 5.22E+01)	5.25E+00 (2.31E+00, 1.11E+01)	1.13E+00 (5.33E–01, 2.60E+00)
f2	3000	9.81E–23 (1.60E–28, 3.66E–21)	2.46E+00 (1.58E+00, 3.82E+00)	1.41E–01 (7.18E–02, 2.29E–01)	7.27E–02 (2.87E–02, 1.41E–01)
f3	8000	2.74E–10 (7.24E–12, 4.08E–09)	2.23E+05 (1.47E+05, 3.13E+05)	4.91E+04 (2.97E+04, 7.34E+04)	3.04E+04 (1.39E+04, 4.65E+04)
f4	15000	1.23E–02 (1.00E–02, 1.55E–23)	9.19E+01 (5.68E+01, 9.54E+01)	1.08E+01 (5.86E+00, 15.9E+00)	2.40E+00 (1.19E+00, 4.25E+00)
f5	1500	5.27E–22 (1.31E–23, 5.44E–21)	3.70E+02 (2.03E+02, 5.19E+02)	6.93E+01 (4.00E+01, 1.06E+02)	2.08E+01 (1.32E+01, 3.23E+01)
f6	6000	6.15E–03 (4.40E–03, 8.26E–03)	2.98E–02 (2.21E–02, 3.49E–02)	7.27E–02 (4.67E–02, 1.10E–01)	4.24E–02 (2.66E–02, 610E–02)
f7	6000	6.70 E+00 (1.71E–06, 3.72 E+01)	9.11E+01 (9.05E+01, 9.23E+01)	1.52E+02 (8.40E+01, 2.99E+02)	1.06E+02 (7.69E+01, 1.49E+02)
f8	1000	2.66E+03 (9.82E+02, 3.79E+03)	3.14E+04 (2.94E+04, 3.24E+04)	1.36E+04 (1.05E+04, 1.81E+04)	2.86E+04 (2.03E+04, 3.17E+04)

Table 3.4: Experimental results (Function 9 ─ Function 13) of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin for high dimensional problems (D=100), averaged over 50 independent runs.

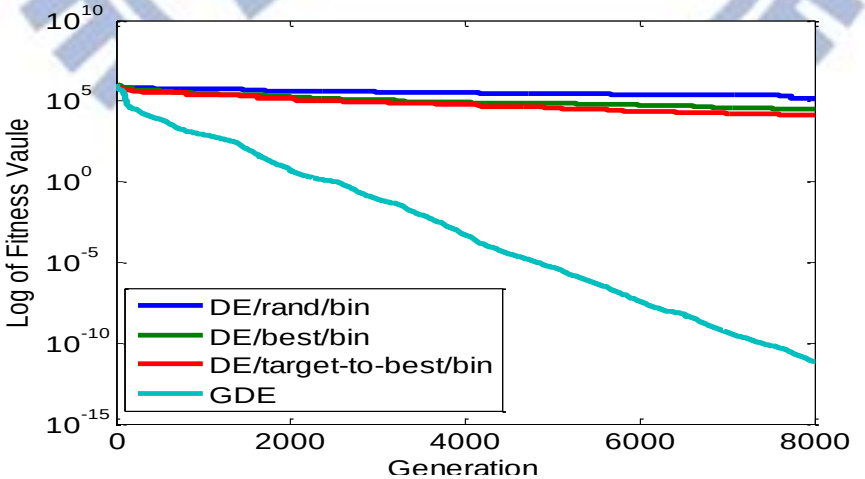
Function	Gen.	GDE	DE/rand/bin	DE/best/bin	DE/target‐to‐best/bin
		Mean (Best, Worst)			
f9	9000	0.0E+00 (0.0E+00, 0.0E+00)	8.08E+02 (7.53E+02, 8.47E+02)	1.74E+02 (1.26E+02, 2.42E+02)	4.20E+02 (7.3E+01, 7.99E+02)
f10	3000	4.41E–13 (3.45E–13, 5.65E–13)	1.54E–01 (9.20E–02, 2.09E–01)	2.68E–01 (2.08E–02, 1.32E+00)	9.45E–03 (5.51E–03, 1.30E–02)
f11	3000	0.0E+00 (0.0E+00, 0.0E+00)	2.48E–01 (1.47E–01, 3.71E–01)	1.72E–02 (7.14E–03, 3.53E–02)	2.32E–03 (5.98E–04, 1.44 E–02)
f12	3000	2.43E–24 (1.80E–28, 2.54E–23)	2.35E+00 (3.11E–01, 1.05E+01)	2.81E+00 (6.42E–01, 6.66E+00)	2.47E–01 (6.86E–04, 1.34E+00)
f13	3000	7.65E–25 (3.38E–28, 4.18E–24)	8.82E+00 (2.32E+00, 2.47E+01)	7.49E+00 (5.72E–01, 3.58E+01)	1.91E–01 (4.17E–03, 3.84E+00)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

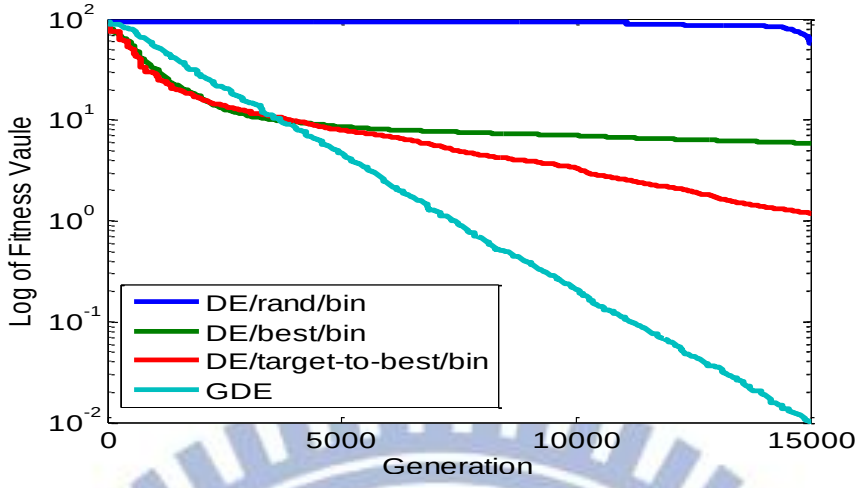
(a)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

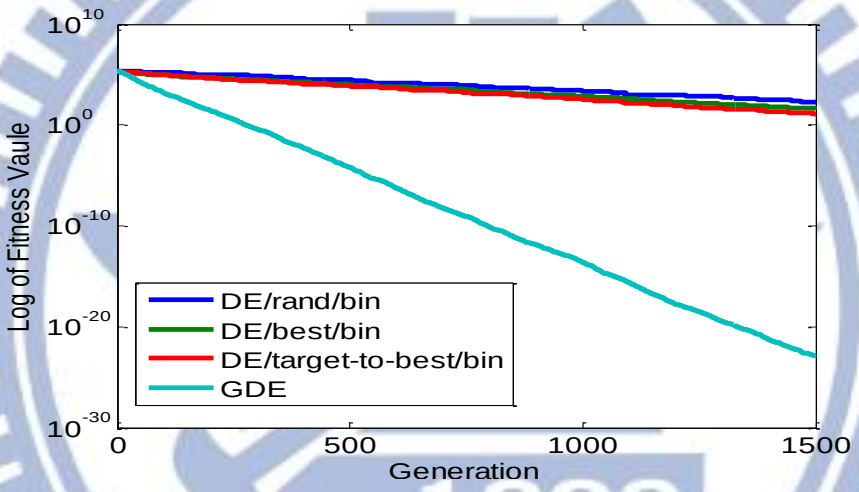
(b)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

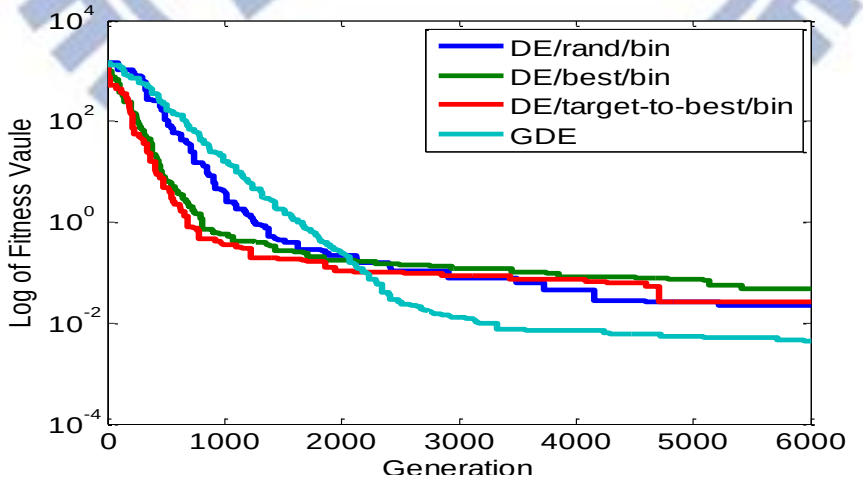
(c)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

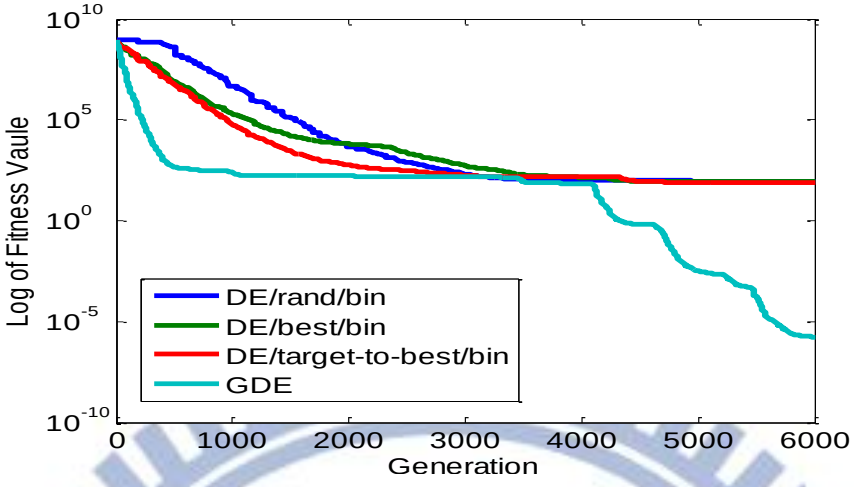
(d)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

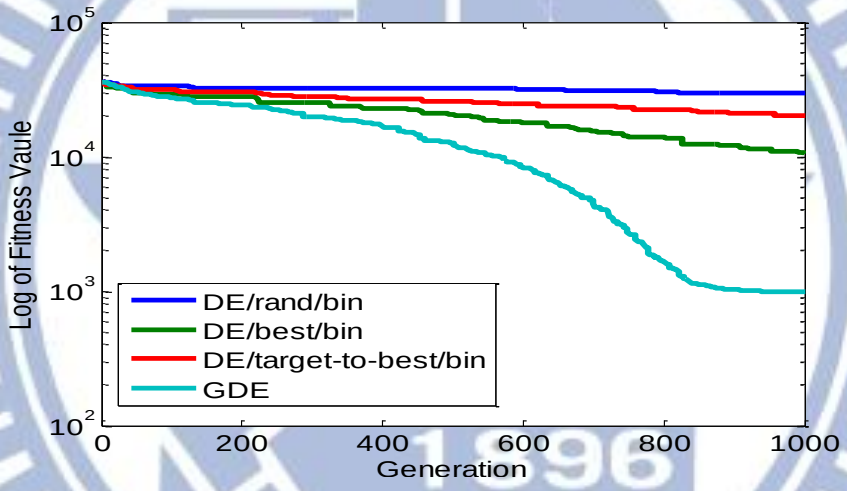
(e)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

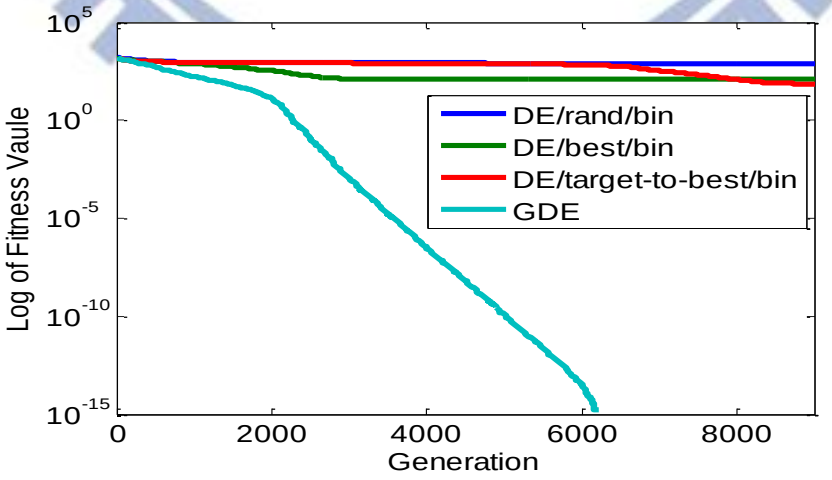
(f)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

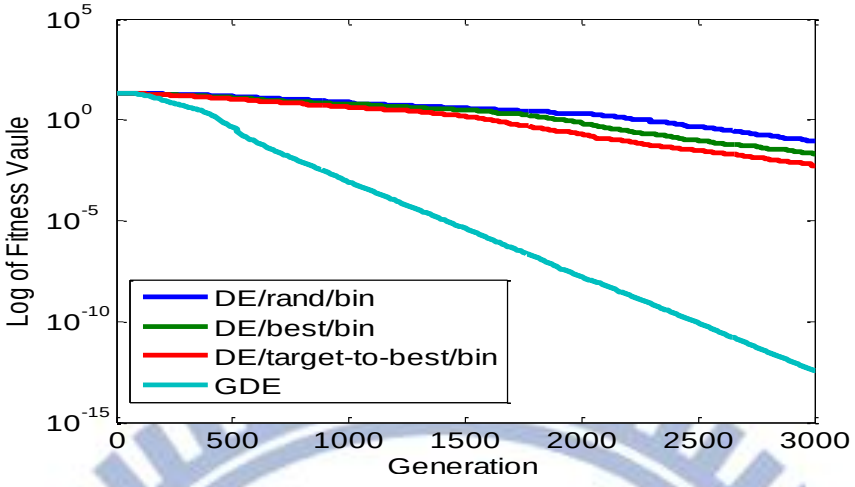
(g)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

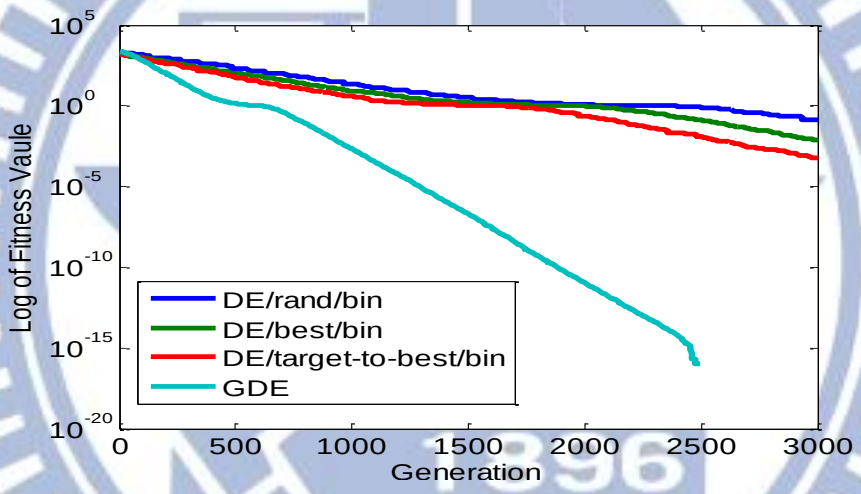
(h)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

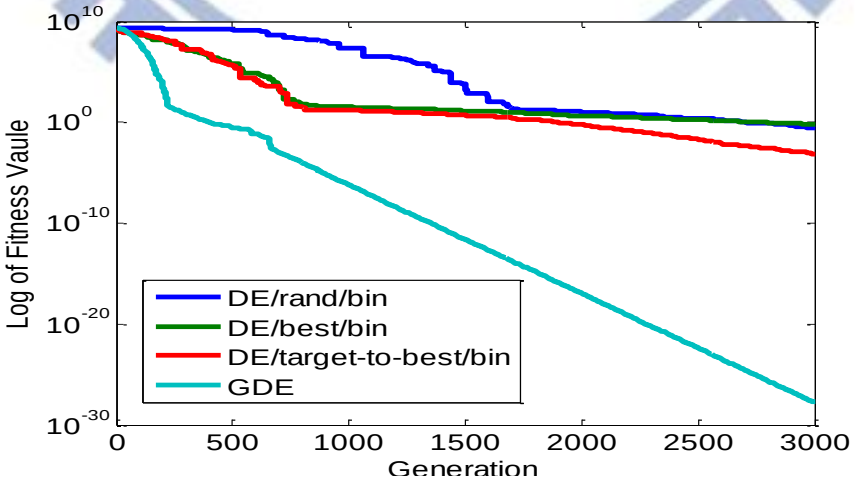
(i)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

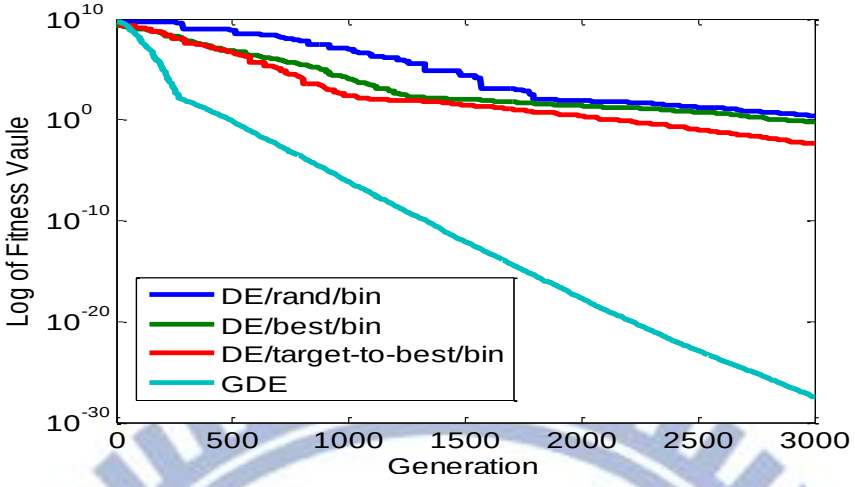
(j)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

(k)


Log of Fitness Vaule
Generation
DE/rand/bin
DE/best/bin
DE/target-to-best/bin
GDE

(l)



(m)

Figure 3.4 : The best learning curve of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin on 13 test function for high dimensional (D=100) problems. (a) Function 1: f_1; (b) Function 2: f_2; (c) Function 3: f_3; (d) Function 4: f_4; (e) Function 5: f_5; (f) Function 6: f_6; (g) Function 7: f_7; (h) Function 8: f_8; (i) Function 9: f_9; (j) Function 10: f_{10}; (k) Function 11: f_{11}; (l) Function 12: f_{12}; (m) Function 13: f_{13};

3.3.4 Statistical Comparison Using Friedman test

In order to understand the significant difference between the GDE and other algorithms over multiple test functions, this paper performed a statistical procedure based on the Friedman test [41, 42] with the corresponding post-hoc tests. We set the GDE algorithm as the control algorithm to compare with other algorithms. The performance of algorithm is significant difference if the corresponding average ranks differ by at least the critical difference (CD)

$$CD = q_{0.05}\sqrt{\frac{j(j+1)}{6T}}, \tag{16}$$

where j is the number of algorithms, T is the number of test functions, and critical values $q_{0.05}$ =2.569 can be found in [42].

A rank relationship of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms is shown in Table 3.5. In this simulation, j=4, T=13 and CD = 0.13. Table 3.6 presents a complete result of Friedman test. Under the 30 dimensional problems, all differences were greater than the critical difference, which means the GDE algorithm is significantly better than the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms in this case. Under the 100 dimensional problems, the difference between the GDE algorithm and the DE/target-to-best/bin algorithm was smaller than the critical difference, which seems to suggest that the GDE algorithm is likely to be different from the DE/target-to-best/bin algorithm. However, in statistics theory, the Friedman test could not prove the significant difference between the GDE algorithm and the DE/target-to-best/bin algorithm. Otherwise, the proposed GDE algorithm was significantly better than the DE/rand/bin algorithm and the DE/best/bin algorithm in 100 dimensional problems. In this paper, an additional statistical test, called Wilcoxon signed-rank test [41], was performed for comparison with the GDE algorithm and the DE/target-to-best/bin algorithm in 100 dimensional problems. Finally, we obtain a *P-value* = 8.53×10^{-19}. This result indicated that the GDE algorithm achieves

significantly better performance than DE/target-to-best/bin algorithms in 100 dimensional problems. The overall result of Friedman test indicates the significant difference between the proposed GDE algorithm and other methods for 100 dimensional problems and 30 dimensional problems.

Table 3.5: The rank table based on experimental results of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin for statistical comparison.

	D=30				D=100			
Function	GDE	DE/rand/bin	DE/best/bin	DE/target–to–best/bin	GDE	DE/rand/bin	DE/best/bin	DE/target–to–best/bin
f1	1	4	3	2	1	4	3	2
f2	1	4	3	2	1	4	3	2
f3	1	4	3	2	1	4	3	2
f4	1	4	3	2	1	4	3	2
f5	1	4	3	2	1	4	3	2
f6	1	4	3	2	1	2	4	3
f7	1	4	3	2	1	4	3	2
f8	1	4	2	3	1	4	2	3
f9	1	4	3	2	1	4	2	3
f10	1	4	3	2	1	3	4	2
f11	1	2	4	3	1	4	3	2
f12	1	2	4	3	1	3	4	2
f13	1	2	4	3	1	4	3	2
Total Rank	13	46	41	30	13	48	40	29
Ave. Rank	1	3.54	3.15	2.31	1	3.69	3.07	2.23

Table 3.6: The result of Friedman test for statistical comparison.

D = 30		
Algorithm	Difference in Rank	Critical Difference(CD)
DE/rand/bin	(3.54－1) ＝ 2.54	1.30
DE/best/bin	(3.15－1) ＝ 2.15	
DE/target–to–best/bin	(2.31－1) ＝ 1.31	
D = 100		
DE/rand/bin	(3.69－1) ＝ 2.69	1.30
DE/best/bin	(3.07－1) ＝ 2.07	
DE/target–to–best/bin	(2.23－1) ＝ 1.23	
D = 30 & D = 100		
DE/rand/bin	(3.61－1) ＝ 2.61	0.91
DE/best/bin	(3.11－1) ＝ 3.11	
DE/target–to–best/bin	(2.26－1) ＝ 1.26	

3.3.5 Comparisons with Other Methods

A further result of the GDE algorithm which compares with other evolutionary algorithms is presented in this section. Table 3.7 shows the comparison of the GDE algorithm and other evolutionary algorithms with 30 dimensional problems. These algorithms include RMEA[45],CEP[30], ALEP[43], BestLevy[43], NSDE[40] and RTEP[44]. The GDE algorithm obtained the best results on six out of eight functions. Table 3.8 shows the comparison of the GDE algorithm and advanced DE algorithms, including jDE[33,47], SaDE[26], ODE[27], SaCDE[16], DEGL[18] and JADE[22,46]. On unimodal

function problems (Function 1 –Function 6); the GDE algorithm obtained the best results on four out of six functions. On multimodal function problems (Function 7 –Function 13), the GDE algorithm obtained the best results on four out of seven functions and has a result near the best solution on f_{10}. The overall results showed that GDE algorithm is a more effective algorithm than other competitive algorithms.

Table 3.7: Comparison with the proposed GDE algorithm and other methods (D=30), including RMEA[45],CEP[30], ALEP[43], BestLevy[43], NSDE[40] and RTEP[44].

Function	GDE	RMEA[45]	CEP[3,44]	ALEP[43]	BestLevy[43]	NSDE[40]	RTEP[44]
	Performance						
f1	**1.83E–42**	1.10E–17	9.10E–04	6.32E–04	6.59E–04	7.10E–17	7.50E–18
f3	**1.13E–25**	2.21E–15	2.10E+02	4.18E–02	3.06E+01	7.90E–16	2.40E–15
f7	3.73E–07	3.10E–04	8.60E+01	4.34E+01	5.77E+01	**5.90E–28**	1.10E+00
f9	5.68E–13	1.74E–08	4.34E+01	5.85E+00	1.30E+01	–	**2.50E–14**
f10	**9.69E–15**	5.08E–06	1.50E+00	1.90E–02	3.10E–02	1.69E–09	2.00E–10
f11	**0.0E+00**	6.41E–20	8.70E-00	2.4E–02	1.80E–02	5.80E–16	2.70E–25
f12	**1.50E–32**	1.72E–08	4.80E–01	6.00E–06	3.00E–05	5.40E–16	3.20E–13
f13	**1.70E–32**	9.29E–05	8.90E–02	9.80E–05	2.60E–04	6.40E–17	7.10E–08

Table 3.8: Comparison with the proposed GDE algorithm and advanced DE algorithms (D=30), including jDE[33,47], SaDE[26], ODE[27], SaCDE[16], DEGL[18] and JADE[22,46].

Function	GDE	jDE[33,47]	SaDE[26]	ODE[27]	SaCDE[16]	DEGL[18]	JADE[22,46]
	Performance						
f1	**1.83E–42**	1.10E–28	4.50E–20	5.61E–24	3.00E–28	8.78E–37	5.50E–28
f2	4.02E–30	1.50E–23	1.90E–14	67.3E–13	1.98E–21	**4.47E–36**	1.03E–26
f3	**1.13E–25**	9.00E–02	9.00E–20	2.95E–08	1.96E–24	3.90E–25	2.40E–18
f4	6.67E–11	1.40E–15	7.40E–11	**2.90E–37**	5.54E–36	4.99E–15	–
f5	**0.0E+00**	**0.00E+00**	**0.00E+00**	–	–	–	–
f6	**2.08E–03**	3.30E–03	4.80E–03	–	–	–	–
f7	3.73E–07	3.10E–15	6.21E–03	2.04E–03	1.66E–03	1.98E–21	7.54E–09
f8	**2.52E+00**	–	–	–	–	–	2.79E+00
f9	5.68E–13	1.50E–15	–	–	–	**1.25E–15**	–
f10	9.69E–15	7.70E–15	3.08E–10	1.90E–13	7.34E–15	1.69E–13	**2.24E–15**
f11	**0.0E+00**	**0.00E+00**	–	**0.00E+00**	–	5.80E–36	–
f12	**1.50E–32**	6.60E–30	4.48E–20	8.14E–25	2.12E–30	–	–
f13	**1.70E–32**	5.00E–29	1.70E–19	5.99E–21	1.37E–28	3.00E–28	–

Chapter 4

A GDE Algorithm for Functional-Link Fuzzy Systems Optimization

4.1 Review of Evolutionary Fuzzy Systems

Fuzzy System (FS) has become a popular research topic and successfully applied to many areas [48-55]. To train the parameters in designing a FS, many papers have employed Backpropagation (BP) algorithm [48, 51, 54-55]. The BP is a powerful training technique that can quickly minimize the error function for NFS. However, the BP algorithm may trap into the local minimum solution and never find the global solution. In order to overcome this disadvantage, many researchers have proposed FS design using evolutionary algorithm (EA) [17, 28, 31, 33, 56-70].

Genetic algorithm (GA) is one of well known evolutionary algorithms. Many researchers had developed GA to implement fuzzy system and neuro-fuzzy system in order to automate the determination of parameters and structures [57-66]. Genetic fuzzy system [60-61] was characterized by using a fuzzy system as an individual in genetic operators. In [65], Karr applied GA to the design of fuzzy controller membership functions, where each fuzzy rule was treated as an individual. Ng and Li [62] applied chromosomes in the GA to optimize sophisticated membership functions for a nonlinear water level control system.

Seng *et al.* [63] proposed a neuro-fuzzy network that is based on the radial basis function neural network all of whose parameters are simultaneously tuned using GA. Juang [66] successfully applied GA to TSK-type recurrent neuro-fuzzy system design for control problem.

Another evolutionary algorithms category for the FS design, called particle swarm optimization (PSO), appears to be efficient and powerful search capability in search space. It is an evolutionary computation technique that was developed by Kennedy and Eberhart in 1995 [13]. The underlying motivation for the development of PSO algorithm is the social behavior of animals, such as bird flocking, fish schooling and swarm theory. PSO has been successfully applied to many optimization problems, such as NFS design [67-78] for control problems, with improved performance over GAs. In [78], the researcher proposed an improved PSO algorithm for a recurrent fuzzy neural network design. The improved PSO algorithm is adopted to adjust the learning rates to improve the online learning capability of the recurrent fuzzy neural network. Juang *et al.*[73] proposed a hierarchical cluster-based multispecies particle-swarm optimization (HCMSPSO) algorithm for fuzzy-system optimization. In their paper, the algorithm combined online cluster-based algorithm and subspecies technique to automatically designs both the structure and the parameters of an FS.

A fast and easy evolutionary algorithm as differential evolution (DE) algorithm, proposed by Storn and Price [15], is an efficient and effective global optimizer in the continuous search domain. In [17, 28], the researcher proposed a modified differential evolution (MODE) for an adaptive neural fuzzy network (ANFN-MODE) design. This MODE provided a cluster-based mutation scheme to prevent the algorithm from being trapped in local optima of the search space. In addition, the MODE algorithm has been applied to locally recurrent neuro-fuzzy system design [31]. An optimization of fuzzy systems using DE algorithm and neighborhood-based mutation operation was proposed by Lin *et al.*[33]. In their paper, they utilized new mutation strategy and adaptive fuzzy c-means

method to find potential individuals in population. Han *et al.*[79] have proposed a new mutation operation based on local information and adaptive parameter tuning method for designing a functional-link-based neural fuzzy network. They successfully applied the proposed model to time series forecasting and achieve a better prediction performance.

Hybrid evolutionary algorithm has been investigated in many studies [76-77, 80-81]. Such a hybrid is often combination of local search in evolutionary algorithm, and referred to as a memetic algorithm [80-81]. In [76], a hybrid of cultural method and cooperative PSO (CPSO) was applied for designing a functional-link-based neural fuzzy network (FLNFN). This method is called FLNFN–CCPSO. In FLNFN–CCPSO, a swarm only optimizes one parameter of an FLNFN. Another hybrid evolutionary algorithm as combination of GA and PSO, which is called HGAPSO [77], was proposed. In HGAPSO, new individuals were created not only by PSO, but also by the crossover and mutation operations of a GA.

4.2 Functional-Link Fuzzy Systems

This section describes the architecture of functional-link fuzzy system. The used system is a novel neural fuzzy network [53, 70, 79, 28]. This system realizes a nonlinear combination of input variables in consequent part. Each fuzzy rule corresponds to an output of functional-link neural network (FLNN). The functional-link fuzzy system realizes a fuzzy IF-THEN rule in the following form:

$$\begin{aligned} \textit{Rule } j \ :\ & \text{IF } x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2j} \text{ and } \ldots x_n \text{ is } A_{nj} \\ & \text{THEN } \ y_j = w_{0j} + w_{1j}\varphi_1 + w_{2j}\varphi_2 + \ldots + w_{lj}\varphi_l \end{aligned} \tag{17}$$

where $x_1,\ldots,x_n$ are input variables, y_j is system output variable, $A_{1j},\ldots,A_{nj}$ are the linguistic term of the precondition part with Gaussian membership function, n is the number of input variables, $w_{0j},\ldots,w_{lj}$ are the functional-link weights, $\varphi_1,\ldots,\varphi_l$ are the basis

trigonometric function of input variables, given by $[x_1\ sin(\pi x_1)\ cos(\pi x_1)\ x_2\ sin(\pi x_2)\ cos(\pi x_2)]$ for two-dimensional input variables.; l is the number of basis function, and $Rule_j$ is the jth fuzzy rule.

In order to present the characteristic of the FLFS, we consider an IF-THEN rule with one input x_1 as follows:

$$\begin{aligned} Rule\ :\ \text{IF } x_1 \text{ is } A_1 \text{ THEN } \quad y &= w_0 + w_1\varphi_1 \\ &= w_0 + w_1 x_1 + w_2 \sin(\pi x_1) + w_3 \cos(\pi x_1) \end{aligned} \tag{18}$$

According to Eq. (18), the FLFS can be degenerated as a TSK-type fuzzy system and Singleton-type fuzzy system when $w_2 = w_3 = 0$ and $w_1 = w_2 = w_3 = 0$. Based on this idea, the TSK-type fuzzy system and Singleton-type fuzzy system are special cases in the proposed FLFS. Therefore, the FLFS presents a diverse combination of input variables to deal with difficult problems more effectively.

The proposed functional-link fuzzy system is five-layered network architecture as shown in Figure 4.1, which is comprised of the input layer, membership function layer, rule layer, functional-link layer and output layer. The operation functions of the nodes in each layer of the FLFS system are now described. In the following description, $O^{(p)}$ denotes the output of a node in the pth layer.

Layer 1—Input layer: No computation is done in this layer. Each node in this layer, which corresponds to one input variable, only transmits input values to the next layer directly. That is

$$O^{(1)} = x_i \qquad i=1,2,...,n \tag{19}$$

where n are the input variables of the functional-link fuzzy system.

Layer 2—Membership function layer: Each node in this layer is a membership function that corresponds one linguistic label of one of the input variables in Layer 1. In other words, the membership value which specifies the degree to which an input value belongs to a fuzzy

set is calculated in Layer 2

$$O^{(2)} = \mu_{ij} = \exp\left(-(x_i - m_{ij})^2 \Big/ \sigma_{ij}^2 \right). \tag{20}$$

Where $j = 1, 2..., M$, M is number of rules in the functional-link fuzzy system, m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of input variable, respectively.

Layer 3—Rule layer: This layer receives 1-D membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described before is adopted to perform the precondition part of the fuzzy rules. As a result, the output function of each inference node is

$$O^{(3)} = R_j = \prod_{i=1}^{n} \mu_{ij} . \tag{21}$$

The output of a layer 3 node represents the firing strength of the corresponding fuzzy rule.

Layer 4—Functional-link layer: The input to a node in layer 4 is the output from layer 3, and the other inputs are calculated from a functional-link neural network that has not used the function $tanh(\cdot)$. For such a node,

$$O^{(4)} = R_j \left(\sum_{k=1}^{l} w_{0j} + w_{kj} \varphi_k \right), \tag{22}$$

where w_{0j} and w_{kj} are the corresponding link weight of functional-link neural network and ϕ_k is the functional expansion of input variables. The functional expansion uses a trigonometric polynomial basis function, given by $[x_1 \; sin(\pi x_1) \; cos(\pi x_1) \; x_2 \; sin(\pi x_2) \; cos(\pi x_2)]$ for two-dimensional input variables. Therefore, l is the number of basis functions, $l = 3 \times n + 1$, where n is the number of input variables. Moreover, the output nodes of functional-link neural network depend on the number of fuzzy rules of the FLFS.

Layer 5—Output layer: Each node in this layer corresponds to one output variable. The node integrates all of the actions recommended by layer 3 and layer 4, which acts as a defuzzifier with

$$O^{(5)} = y = \frac{\sum_{j=1}^{M} R_j (\sum_{k=1}^{l} w_{0j} + w_{kj}\varphi_k)}{\sum_{j=1}^{M} R_j}, \tag{23}$$

where M is the number of fuzzy rules, and y is the output of the FLFS.

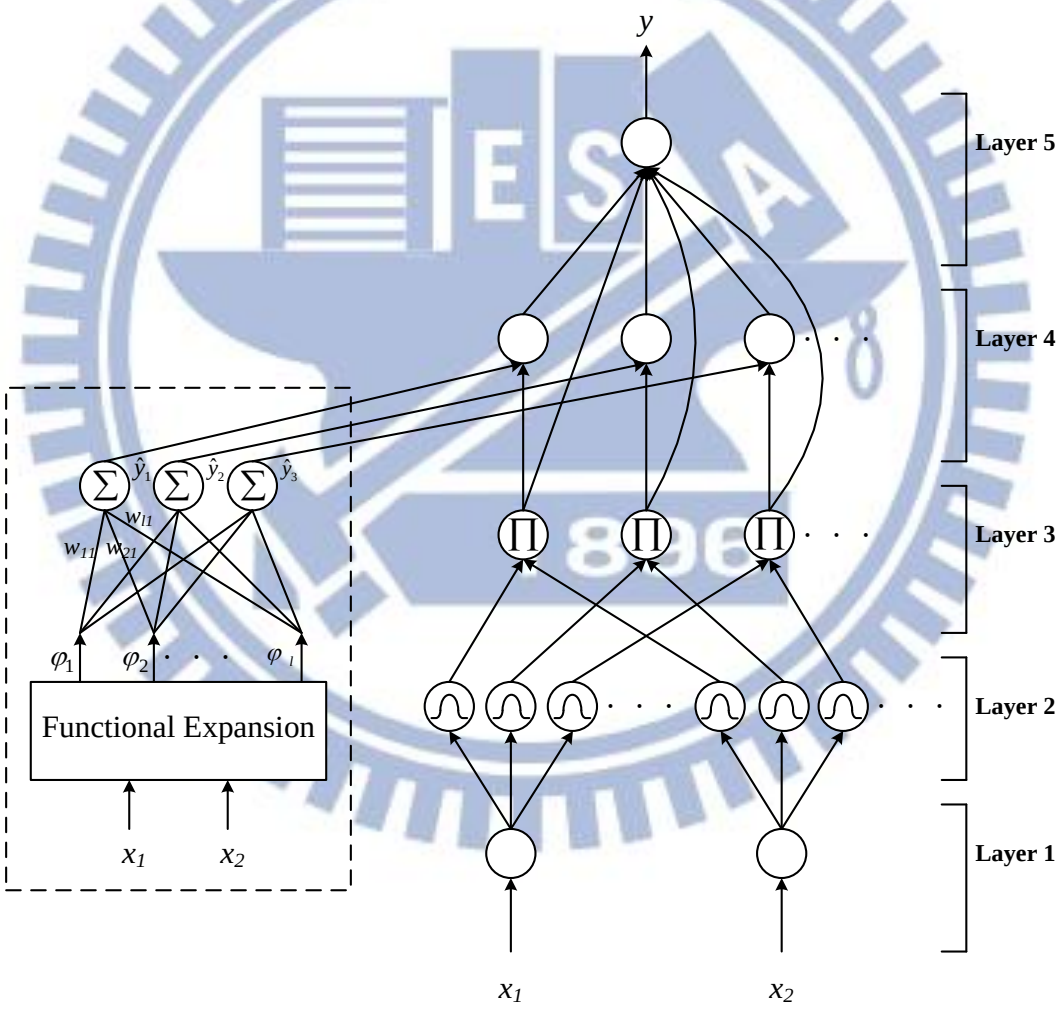


Figure 4.1: The architecture of the functional-link fuzzy system.

4.3 Learning process of Functional-Link Fuzzy Systems

This section describes a learning process based GDE optimization for the functional-link fuzzy system design. Initially, an agglomerative clustering algorithm is to automatically construct a preliminary functional-link fuzzy system and determine optimal number of fuzzy rules. Subsequently, the learning process randomly generates a set of individuals (functional-link fuzzy systems) for the evolution process. All individuals are learned by GDE algorithm for searching an optimal functional-link fuzzy system. The overall learning process is shown in Figure 4.2.

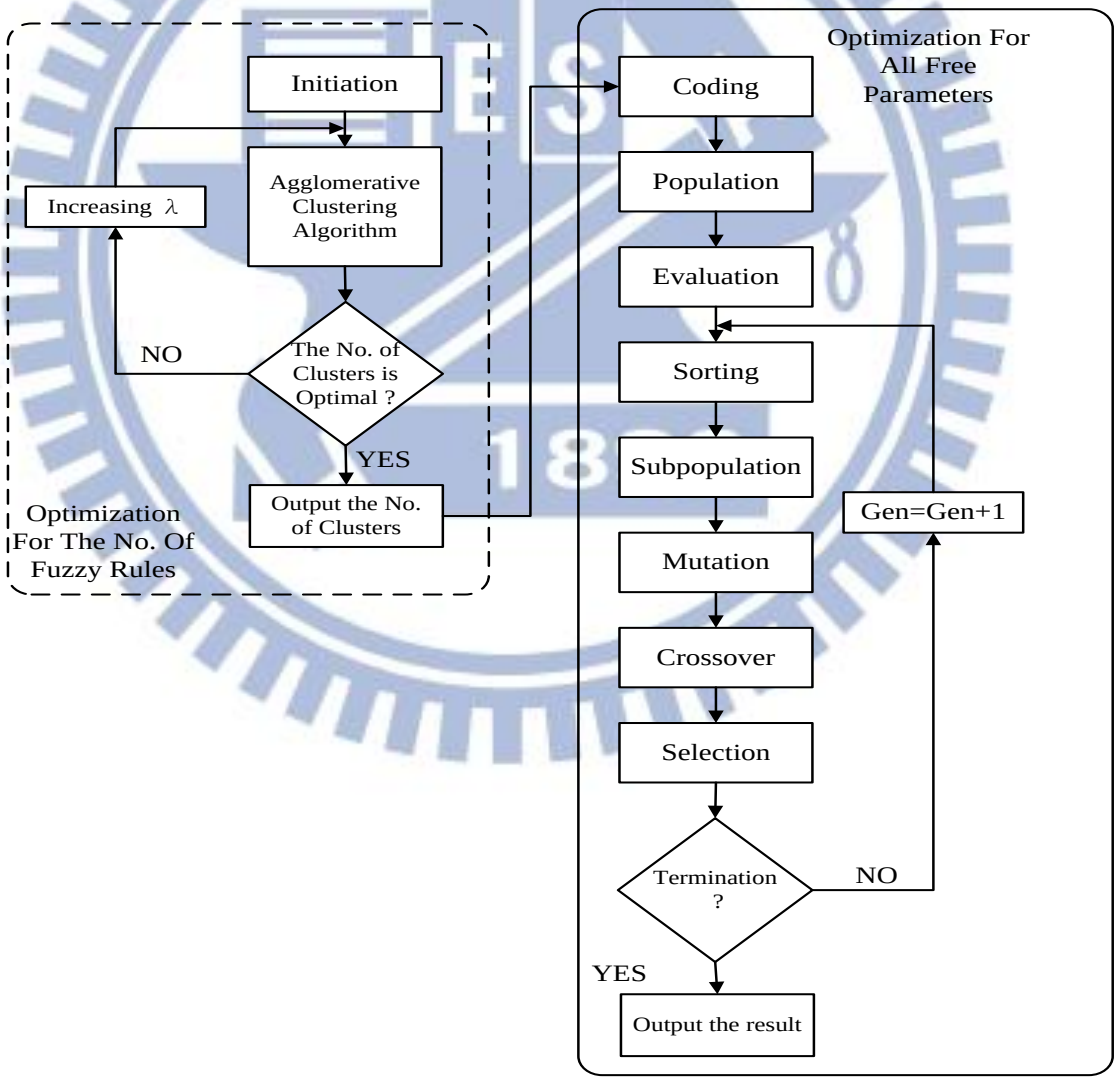


Figure 4.2: The Overall learning process.

4.3.1 An Agglomerative Clustering Algorithm

The first step is to determine whether a new rule should be extracted the training pattern and decide the number of fuzzy sets in the universal 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 σ_{ij} represent the mean and width of that cluster, respectively. Many studies have employed clustering technique for rule generation, such as fuzzy *C*-means, possibilistic *C*-means, and on-line clustering methods [28, 53, 68, 73]. However, such clustering techniques require prior knowledge such as the number of clusters present in a pattern set. To solve this problem, an agglomerative clustering algorithm is proposed for rule generation. The proposed agglomerative clustering algorithm (ACA) [82-83] is an extension to the standard fuzzy C-Means algorithm by a penalty term to the objective function. This algorithm can find the best number of clusters for various problems. Now, we consider a set of samples $X_i = \{x_1, x_2, ..., x_n\}$, $i = 1, 2, ..., N$. To cluster X into M clusters by minimizing the following objective function:

$$Obj(\mathrm{U},\mathrm{C}) = \sum_{j=1}^{M}\sum_{i=1}^{N} u_{ij} \left\| \mathrm{C}_j - \mathrm{X}_i \right\|^2 + \lambda \sum_{j=1}^{M}\sum_{i=1}^{N} u_{ij} \log u_{ij} \tag{24}$$

subject to

$$\sum_{j=1}^{M} u_{ij} = 1, \quad i = 1, 2, ..., N, \tag{25}$$

where $u_{ij} \in [0,1]$ is the membership degree of X_i in the *j*th cluster $\mathrm{C}_j = (\mathrm{c}_{j1}, \mathrm{c}_{j2}, ..., \mathrm{c}_{jn})$, U is the matrix of u_{ij}, λ is a penalty parameter and $\|\cdot\|$ is the Euclidean norm as the dissimilarity measure. λ

Equation (24) and (25) present a class of constrained nonlinear optimization problems. For solving optimization problem, we use Lagrangian multiplier technique and gradient

method to obtain update law

$$c_{jl} = \frac{\sum_{i=1}^{N} u_{ij} x_{il}}{\sum_{i=1}^{N} u_{ij}}, \qquad j = 1, 2, ..., M \text{ and } l = 1, 2, ..., n \tag{26}$$

$$u_{ij} = \frac{\exp\left(\frac{-\left\|\mathrm{C}_j - \mathrm{X}_i\right\|^2}{\lambda}\right)}{\sum_{l=1}^{M} \exp\left(\frac{-\left\|\mathrm{C}_l - \mathrm{X}_i\right\|^2}{\lambda}\right)}, \qquad j = 1, 2, ..., M \text{ and } i = 1, 2, ..., N \tag{27}$$

To obtain the optimal centers, the u_{ij} and c_{ij} are continuously updated by equation (26) and (27). Until the objective function is unchanged, the process of agglomerative clustering algorithm is terminated. The completed flow chart is shown in Figure 4.3.

In the agglomerative clustering algorithm, λ is an important parameter for the minimization process. When λ is large, the minimization process tries to assign each object to more clusters to make the second term more negative. In order to achieve the largest object entropy, the cluster centers move to the same location. On the contrary, agglomerative clustering algorithm is degenerated as standard fuzzy C-Means algorithm when λ is small.

We design an automatic process to discover the best number of clusters. The overall process is shown in Figure 4.3. In the automatic process, there is two input parameters, the number of initial cluster centers $M_{initail}$ and the penalty value $\lambda_{initail}$. In general, the $M_{initail}$ should be larger than the possible number of clusters in the given data set, the $\lambda_{initail}$ should be set a small value. Initially, the agglomerative clustering algorithm with $M_{initail}$ and $\lambda_{initail}$ is performed for our problems. An exactly clustering result $M(t)$ is obtained in the output. We consider that the values $\lambda(t)$ of increase from $\lambda(t) = \lambda(t-1) + \lambda_{initial}$ and perform agglomerative clustering algorithm for every *t*. The whole procedure is stopped when $M(t)$ is equal to 1.

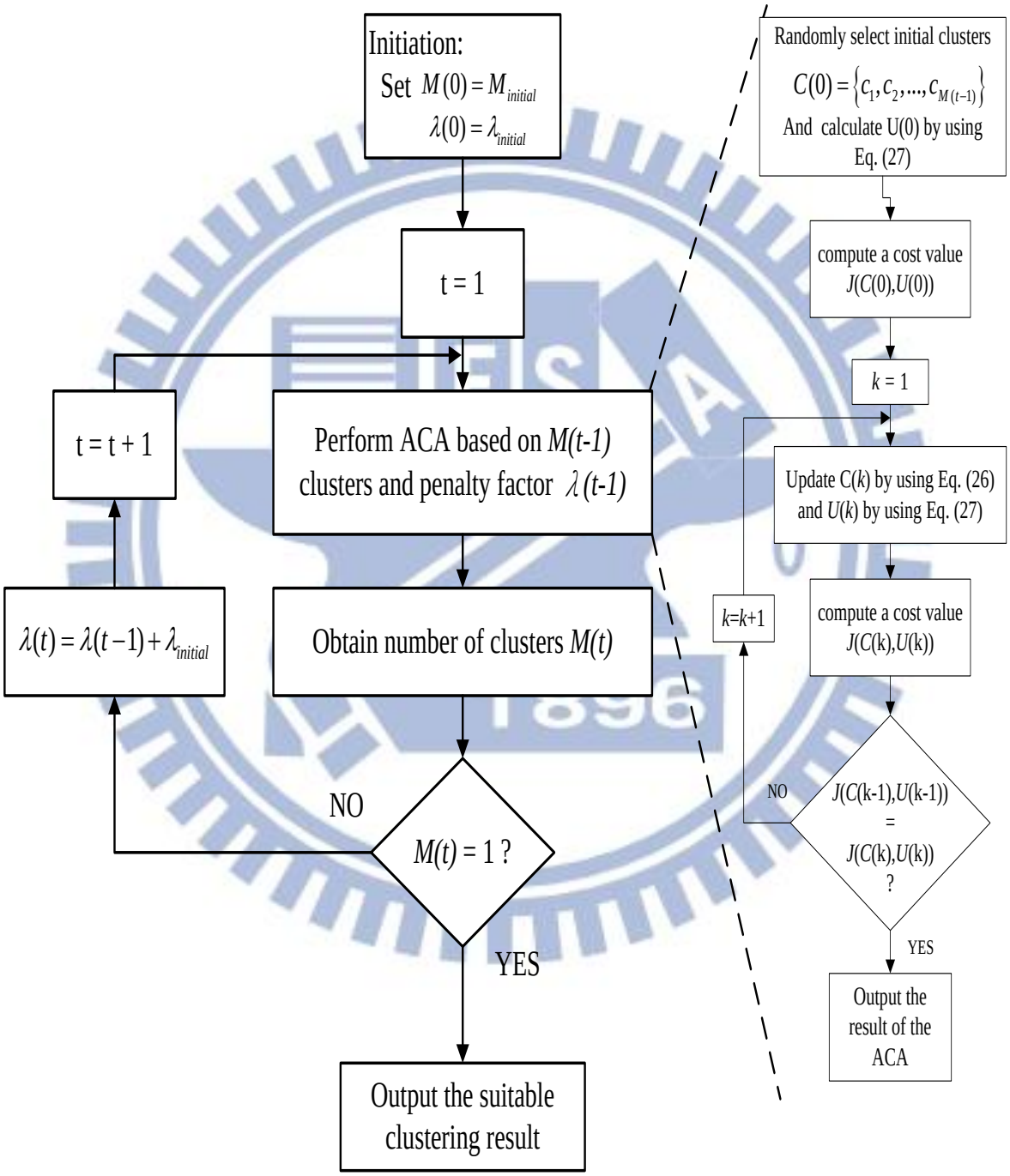


Figure 4.3: A flow chart of the proposed agglomerative clustering algorithm for discovering the optimal number of clusters.

In order to demonstrate the proposed method, we consider a data set of 1,000 points in a two-dimensional (2D) space as shown in Figure 4.5(a). We perform the automatic process to cluster this data set and discover the optimal number of clusters. Initially, λ is a small value, we can see that the number of clusters generated by the algorithm was equal to the number of initial cluster centers. As increased, the number of generated clusters reduced because some initial cluster centers moved to the same locations. In Figure 4.4, the result of clusters = 3 is usually found for this problem. This indicates that the λ setting is right in finding the true clusters by the algorithm. Finally, when λ increased to a certain value, the number of generated clusters became one. Figures 4.5(a)-(d) show the movements of the cluster centers in the iterations k=1, 5, 10 and 21 when λ=1. We can see that the initial cluster centers moved to three locations (1,1), (1,5) and (5,5).

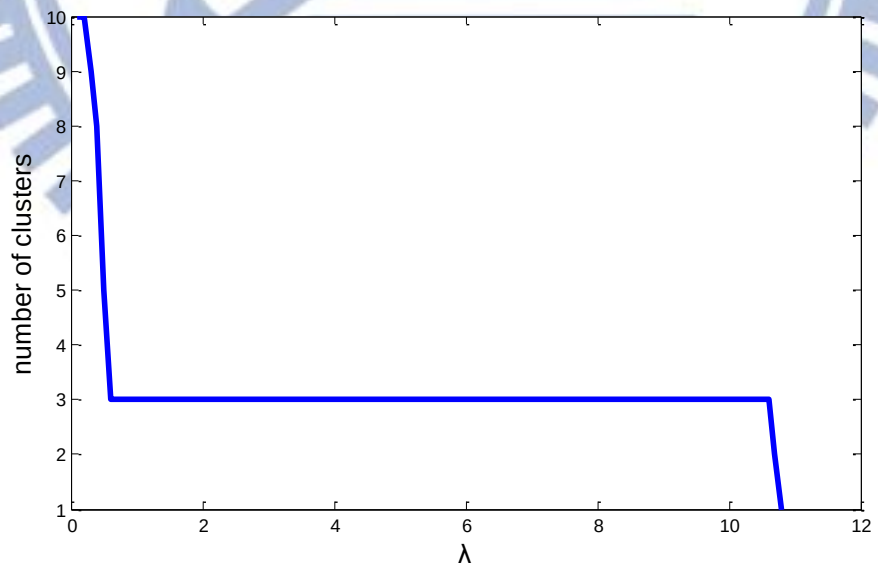
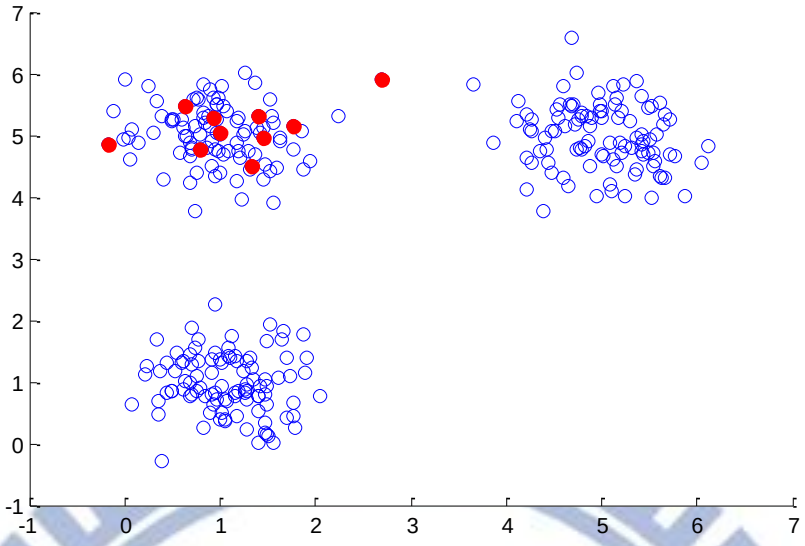
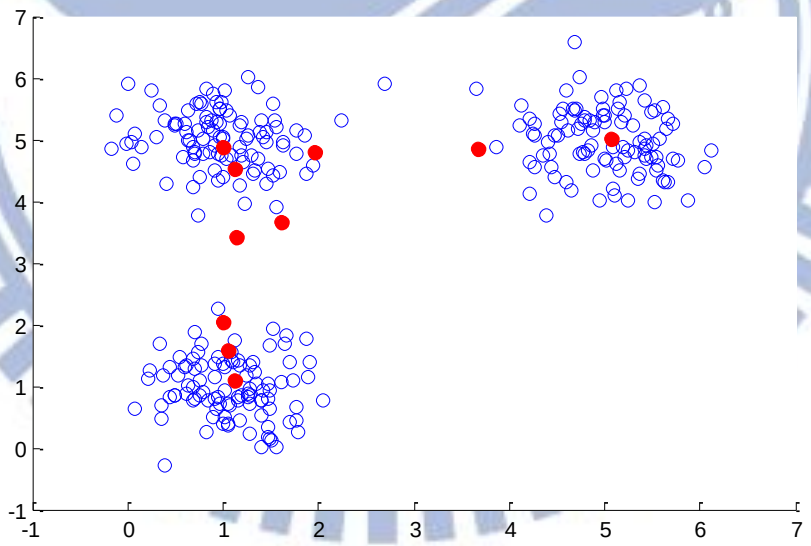


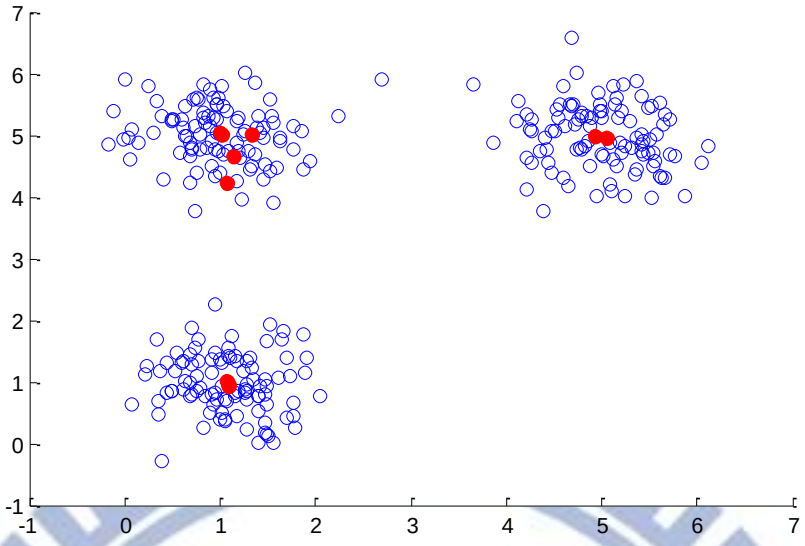
Figure 4.4: The result of proposed agglomerative clustering algorithm with respect to different λ.



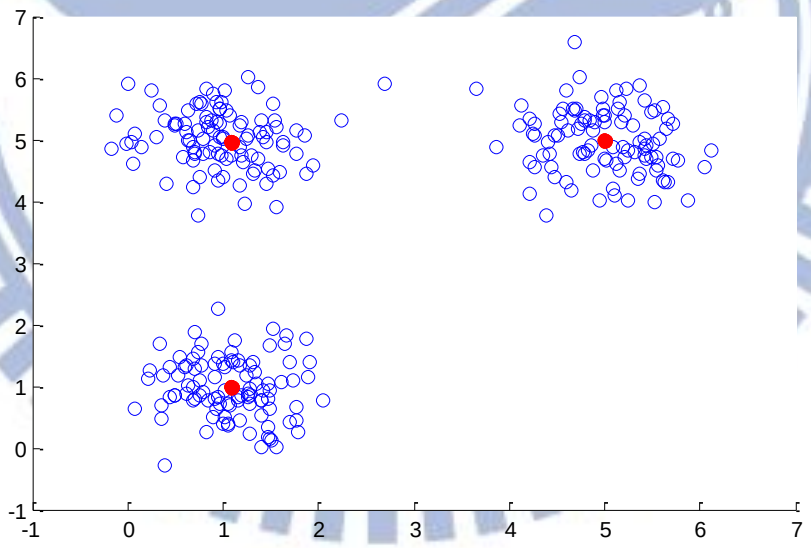
(a)



(b)



(c)



(d)

Figure 4.5: The clustering results by the proposed algorithm with $\lambda = 4$ (a) the result of k =1, (b) the result of k =5, (c) the result of k =10, and (d) the result of k =21.

4.3.2 Evolution Learning Processes

For the effective parameter learning, evolutionary algorithm is usually used. In this paper, we propose the GDE algorithm to tune all free parameters. The proposed GDE algorithm consists of eight major steps：the coding step, population step, evaluation step, sorting step, subpopulation step, mutation step, crossover step, and selection step. The whole learning process is described as follows：

(1) Coding step：The foremost step in the GDE algorithm is the coding of the FLFS system into an individual. Figure 4.6 shows an example of the coding of parameters of the FLFS system into an individual, where *NP* is population size, *M* is number of rules, and *n* is total of input variable. In this study, an individual consists of the mean m_{ij} and width σ_{ij} of a Gaussian membership function, and w_{kj} weight of the consequent part, where *i* and *j* represent the *i*th input variable and the *j*th rule, respectively.

(2) Population Step：Before the proposed GDE algorithm is performed, the individuals that will constitute an initial population must be created. A niching operation [57, 68] is to create good initial population in the input space. The initial population is created according to the range of the mean and variance of the membership function, which were computed by the agglomerative clustering algorithm in section 4.3.1. The following formulations show the generation of the initial population.

$$\begin{aligned} FLFS_q &= [rule_1^q \mid rule_2^q \mid ... \mid rule_M^q] \\ &= [m_{i1}^* + \Delta m_{i1}^q, \sigma_{i1}^* + \Delta\sigma_{i1}^q, w_{k1}^q \mid ... \\ &\quad \mid m_{ij}^* + \Delta m_{ij}^q, \sigma_{ij}^* + \Delta\sigma_{ij}^q, w_{kj}^q \mid ... \\ &\quad \mid m_{iM}^* + \Delta m_{iM}^q, \sigma_{iM}^* + \Delta\sigma_{iM}^q, w_{kM}^q \ ] \end{aligned} \tag{28}$$

where m_{ij}^* and σ_{ij}^* are results of structure learning for the mean and width of the Gaussian membership function of the *j*th rule of the *i*th input variable, Δm_{ij}^q and $\Delta\sigma_{ij}^q$ are small

random deviations that are uniformly generated from the interval [−0.1, 0.1], w_{kj} are randomly and uniformly generated from an interval whose range is identical to the FLFS system output *y* range.

(3) Evaluation Step：In this study, we adopt a fitness function to evaluate the performance of each individual. The fitness function used in this paper is the root mean-squared error (RMSE) between the desired and actual outputs. The fitness function is defined as follows:

$$fitness = \sqrt{\frac{\sum_{k=1}^{n}\left(y_k - \bar{y}_k\right)^2}{n}} \tag{29}$$

where y_k represents the model output of the *k*th pattern, $\bar{y}_k$ represents the desired output of the *k*th pattern, and *n* the number of the training pattern.

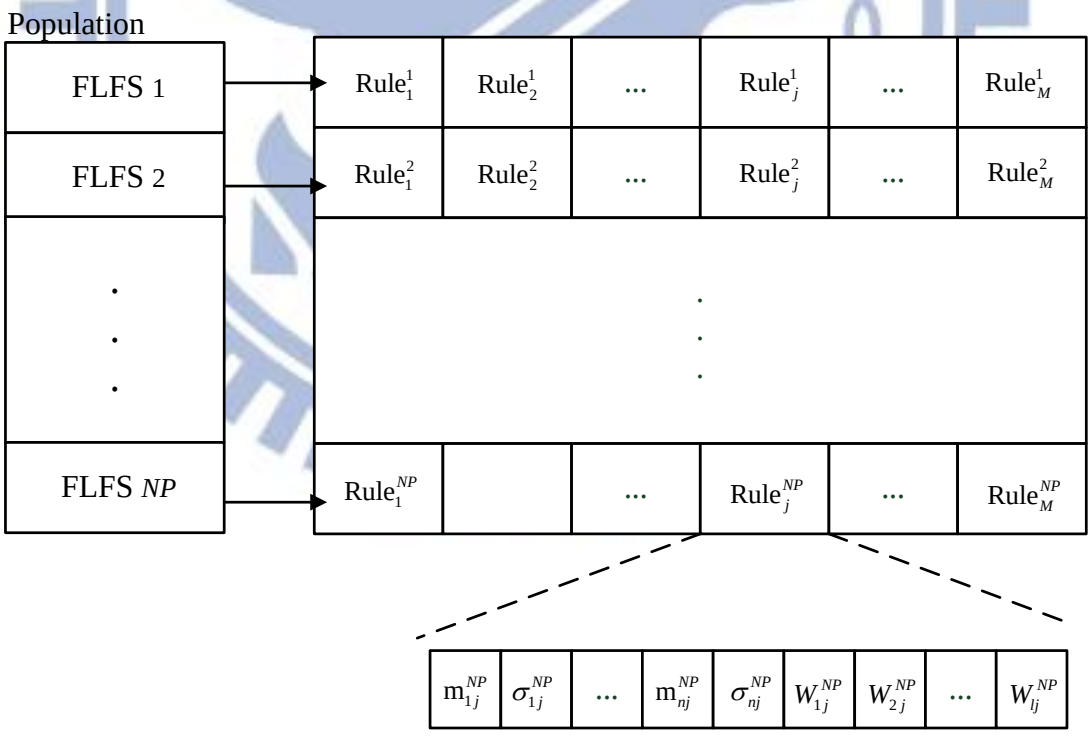


Figure 4.6: Coding FLFSs into individuals.

(4) Sorting Step: A sorting process arranges all FLFSs based on their fitness value as

$$fitness(\text{FLFS}_1) < fitness(\text{FLFS}_2) < fitness(\text{FLFS}_3) < \ldots < fitness(\text{FLFS}_{\text{NP-1}}) < fitness(\text{FLFS}_{\text{NP}})$$

for minimum objective problems. After sorting process, the FLFS_1 is the best system in population for current generation. According to fitness values, all FLFSs are partitioned into an inferior group and a superior group. The two groups perform different evolution strategies.

(5) Subpopulation Step: To enhance performance, we used symbiotic learning method in the proposed algorithm. The basic idea of symbiotic learning method is that an FLFS is combined by fuzzy rules, which are randomly selected from a subpopulation. Every subpopulation is composed of related fuzzy rules, called subindividuals. Every subpopulation performed evolution process to product new subindividuals. This method can increase more possibility to search potential solutions. A completed process of the subpopulation step is shown in Figure 4.7.

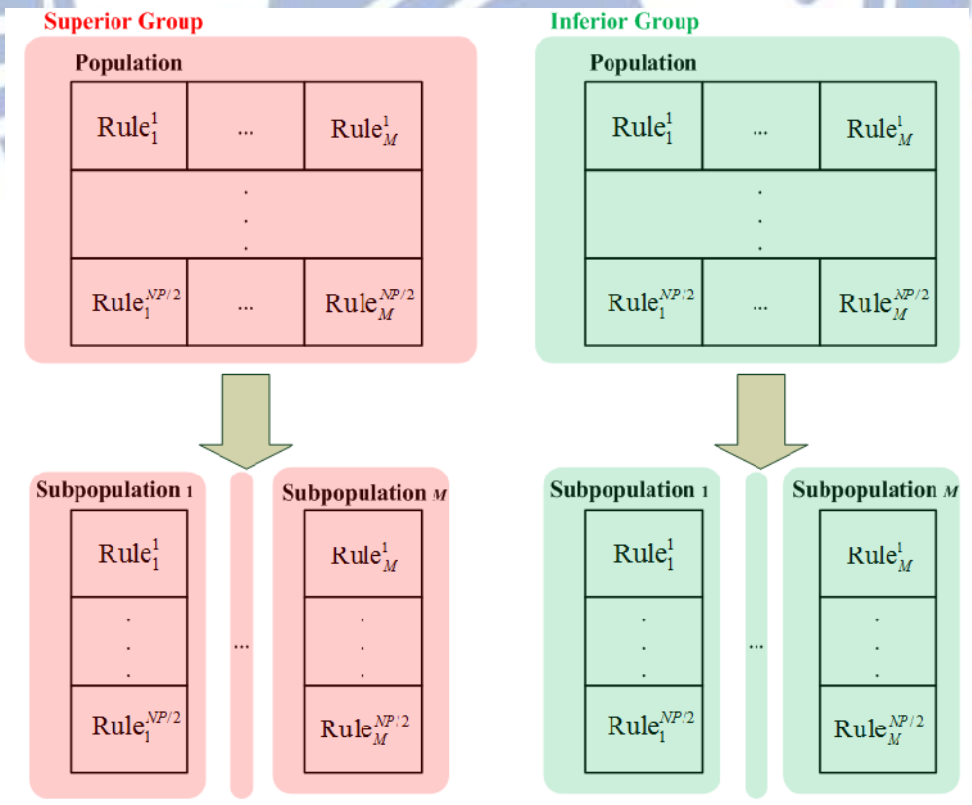


Figure 4.7: A completed process of the subpopulation step.

(6) Mutation Step：Each Sub-individual in the current generation is allowed to breed through mating with other randomly selected sub-individuals from the subpopulation. Specifically, for each subindividual $\boldsymbol{Z}_{i,gen}$, $i=1,2,...,NP$, where g denotes the current generation, NP is subpopulation size, Four other random subindividuals, $\boldsymbol{Z}_{r1,gen}$, $\boldsymbol{Z}_{r2,gen}$, $\boldsymbol{Z}_{r3,gen}$, and $\boldsymbol{Z}_{r4,gen}$ are selected from the subpopulation such that $r1$, $r2$, $r3$,and $r4 \in \{1, 2, \ldots, NP\}$and $i \neq r1 \neq r2 \neq r3 \neq r4$. This way, a parent pool of four subindividuals is formed to produce an offspring. Two mutation operations are applied to generate a mutated subindividual $v_{i,g}$ according to the following equation:

$$\text{Group A: } \boldsymbol{V}_{i,gen} = \boldsymbol{Z}_{i,gen} + F_a(\boldsymbol{Z}_{r1,gen} - \boldsymbol{Z}_{r2,gen}) \tag{30}$$

$$\text{Group B: } \boldsymbol{V}_{i,gen} = \boldsymbol{Z}_{gbest,gen} + F_b(\boldsymbol{Z}_{r3,gen} - \boldsymbol{Z}_{r4,gen}) \tag{31}$$

where F_a and F_b are scaling factors $\in [0,1]$, $\boldsymbol{Z}_{gbest,gen}$ is the best-so-far sub-individual (i.e., $\boldsymbol{Z}_{gbest,gen}$ keeps best fitness value up to now in the subpopulation). Figure 4.8 presents the mutation process of the proposed GDE algorithm.

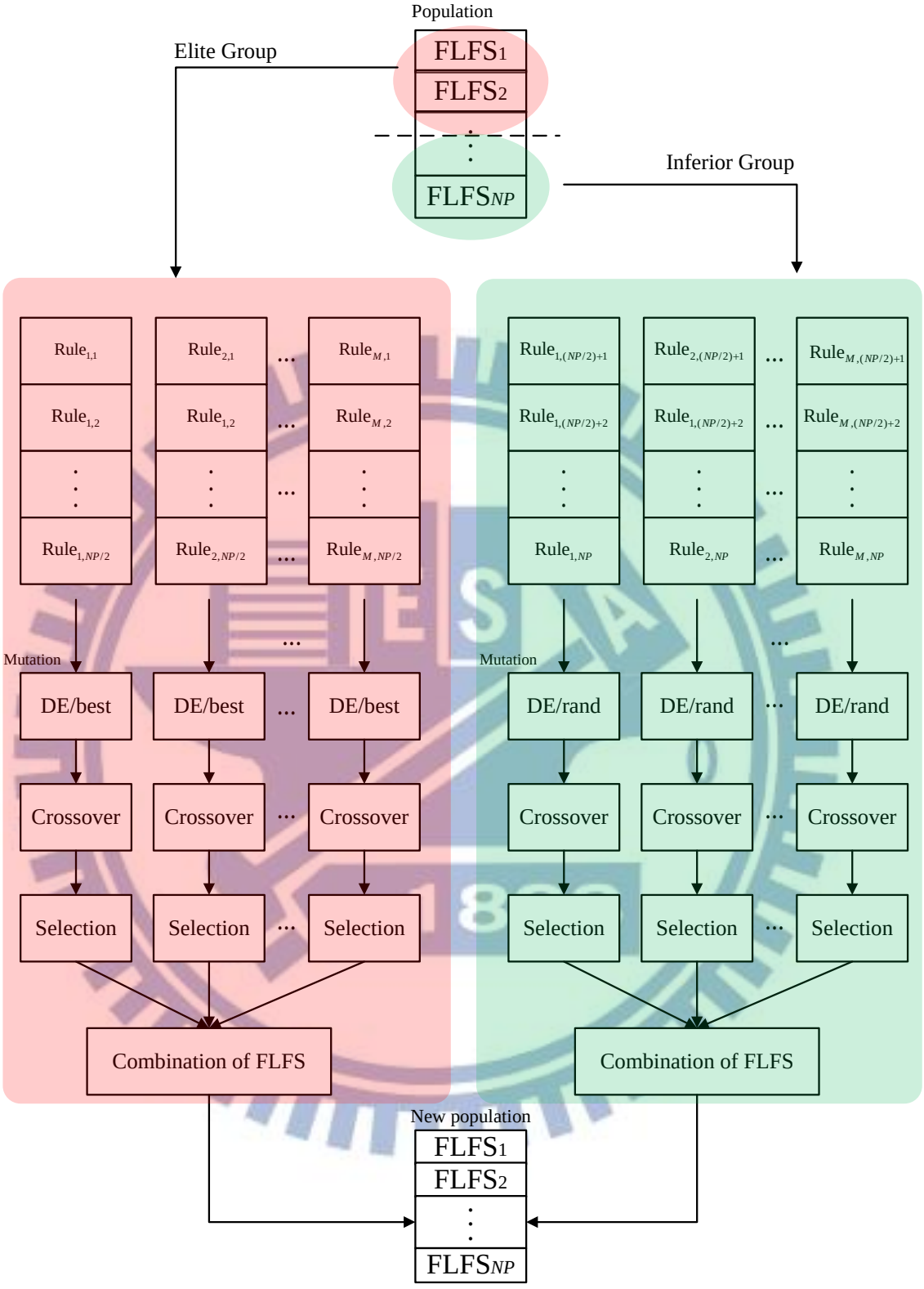


Figure 4.8: A flow chart of the proposed GDE algorithm for FLFSs optimization.

(7) Crossover Step：After mutation operation, the proposed GDE algorithm uses a crossover operation, often referred to as discrete recombination, in which the mutated subindividual $\boldsymbol{V}_{i,gen}$ is mated with $\boldsymbol{Z}_{i,gen}$ and generates the offspring $\boldsymbol{U}_{i,gen}$. The elements of an subindividual $\boldsymbol{U}_{i,gen}$ are inherited from $\boldsymbol{Z}_{i,gen}$ and $\boldsymbol{V}_{i,gen}$, which are determined by a parameter called crossover probability (CR ∈ [0, 1]), as follows:

$$\boldsymbol{U}_{d,i,gen} = \begin{cases} \boldsymbol{V}_{d,i,gen}, & \text{if rand}(d) \le \text{CR} \\ \boldsymbol{Z}_{d,i,gen}, & \text{if rand}(d) > \text{CR} \end{cases} \tag{32}$$

where d = 1, 2, . . . , D denotes the dth element of individual vectors, D is total element of subindividual vector, rand(d) ∈ [0, 1] is the dth evaluation of a random number generator.

(8) Selection Step：The GDE algorithm applies selection operation to determine whether the subindividual survives to the next generation. First, the current composed $\text{FLFS}_{q,gen}$ embeds the current subindividual $Z_{i,g}$ into the $\text{FLFS}_{q,gen-1}$ and the trial composed $\text{FLFS}_{q,gen}$ embeds the trial subindividual $U_{i,gen+1}$ into the $\text{FLFS}_{q,gen-1}$. Second, a knockout competition is played between the current composed $\text{FLFS}_{q,gen}$ and the trial composed $\text{FLFS}_{q,gen}$. Then, the corresponding subindividual of the winner is selected deterministically based on objective function values.

4.4 Simulation

This section discusses five simulations that are considered to evaluate the FLFS model with the GDE algorithm. The five simulations include chaotic time series prediction, Mackey Glass time series prediction, oil price time series prediction, star brightness time series prediction and auto-MPG6 data prediction. Table 4.1 presents the initial parameters prior to training used in each of the five simulations.

For comparison, the evolutionary algorithms, such as DE, jDE and MODE are applied to the same problems for the FLFS optimisation. We used the same population size and number of generations in each of these evolutionary algorithms. The agglomerative clustering algorithm is also used for rule generation.

In the following simulations, the major computation time is evaluating the performance of the FLFS. All evolutionary algorithms are compared using the same population size and number of generations in a single run. Thus, the overall computation time is almost the same for different evolutionary algorithms.

Table 4.1: Initial parameters before training.

Parameter	Value
Population Size	50
CR_a	0.9
CR_b	0.9
F_a	0.5
F_b	0.8
$M_{initail}$	15
$\lambda_{initail}$	0.01
Generation	1000
Coding Type	Real Number

Example 1: Prediction of chaotic time series

In this example, an FLFS model with a GDE learning method (FLFS-GDE) is used to predict a chaotic signal. The classical time series prediction problem is a one-step-ahead prediction, which has been described in [48]. The following equation describes the logistic function.

$$x(k+1) = ax(k)(1-x(k)). \tag{33}$$

The behavior of the time series generated by this equation depends critically on parameter a. If $a<1$, then the system has a single fixed point at the origin, and from a random initial value between [0, 1] the time series collapses to a constant value. For $a>3$, the system generates a periodic attractor. At $a \geq 3.6$, the system becomes chaotic. In this example, a is set to 3.8. The first 60 pairs (from *x(1)* to *x(60)*), with initial value $x(1) = 0.001$, are the training data set, while the remaining 100 pairs (from *x(1)* to *x(100)*), with initial value $x(1) = 0.9$, are the testing data set used to validate the proposed method.

In this example, DE, jDE and MODE are applied to the same problem to show the effectiveness and efficiency of the FLFS model with the GDE learning method. In the DE and jDE, the scale factor F = 0.5, the crossover rate CR=0.9 and the mutation strategy= DE/rand/bin. In the MODE, the scale factor is linearly increased from 0 to 1, the crossover rate CR=0.9 and the mutation strategy= DE/target-to-best/bin. A total of 50 runs are performed for statistical analysis.

After rule generation, the agglomerative clustering algorithm find the optimal number of fuzzy rules = 2 for example 1, as shown in Figure 4.9. The FLFS is learned by DE, jDE, MODE and GDE algorithms. The performance of the FLFS model with DE, jDE, MODE and GDE is shown in Table 4.2, including average and standard deviation (STD) over 50 runs. Figure 4.10 shows the learning curves of the DE, jDE, MODE and GDE algorithms for

example 1. The learning curves of the DE and jDE algorithms present stagnation situations during evolution process. The MODE and GDE algorithms continually keep convergence results. It is clear from these data that the proposed GDE algorithm shows better learning curves than the other methods. The proposed GDE obtains the best performance RMSE= 0.00025. Figure 4.11 plots the results predicted using the proposed GDE algorithm. Figure 4.12 presents the prediction errors of the proposed GDE algorithm.

In addition, we also compare with the performance of the FLFS-GDE model and other papers. Table 4.3 shows that the testing RMSE of FLNFN-PSO [13, 76], FLNFN-CPSO [84, 76] and FLNFN-CCPSO [76] models from other journal papers. These comparative papers use three fuzzy rules for their system. In Table 4.3, our method FLFS-GDE model achieves a better performance than FLNFN-PSO [13, 76], FLNFN-CPSO [84, 76] and FLNFN-CCPSO [76] models.

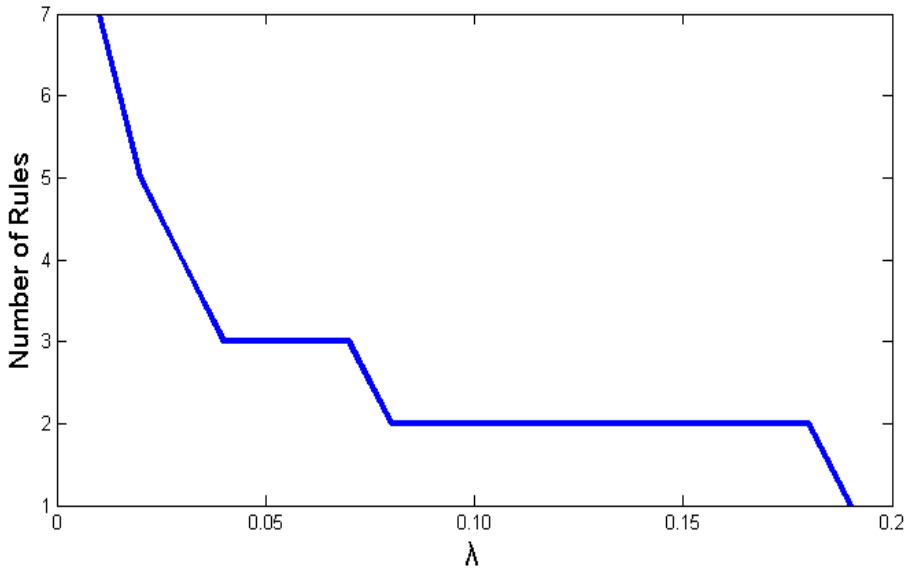


Figure 4.9: The result of the agglomerative clustering algorithm for example 1.

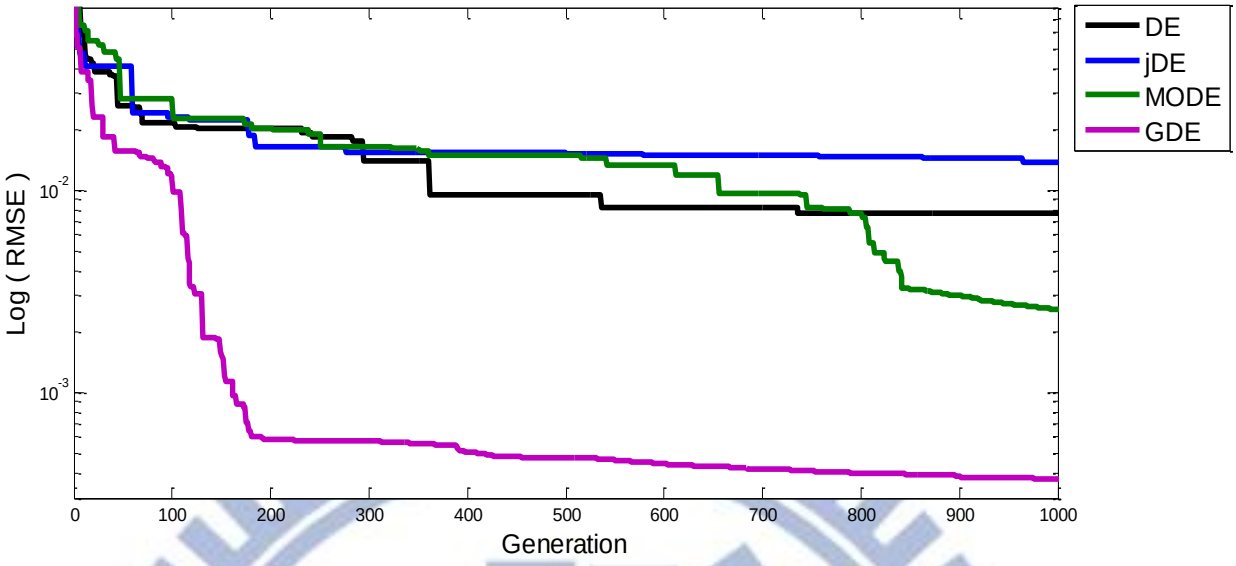


Figure 4.10: Training RMSEs of the DE, jDE, MODE and GDE algorithms at each performance evaluation for example 1.

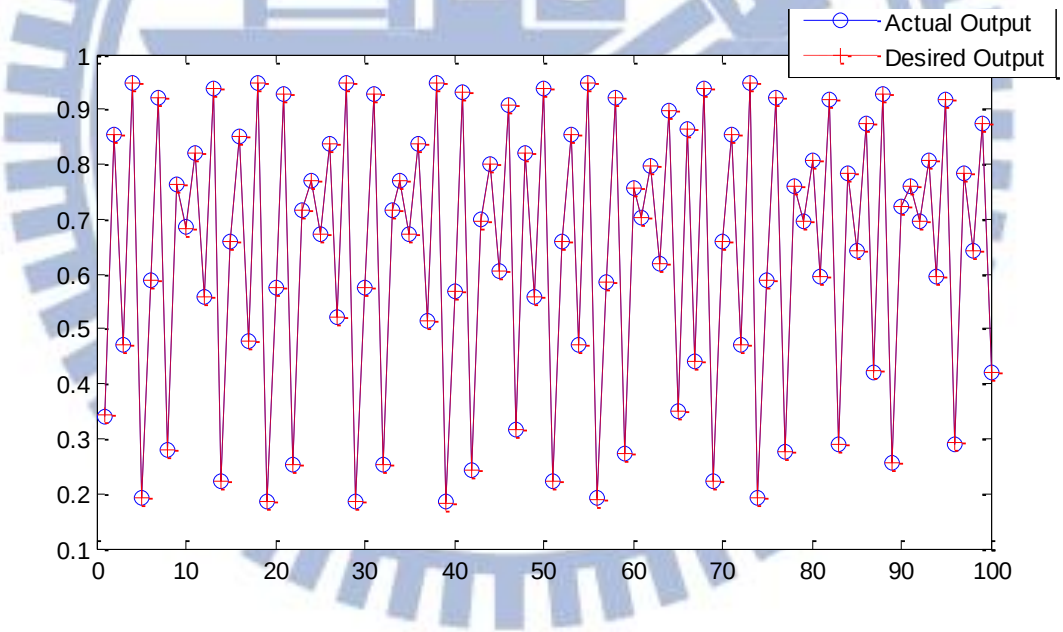


Figure 4.11: Prediction results of the FLFS-GDE model for example 1. Symbol "+" represents the desired results and "O" represents the actual results.

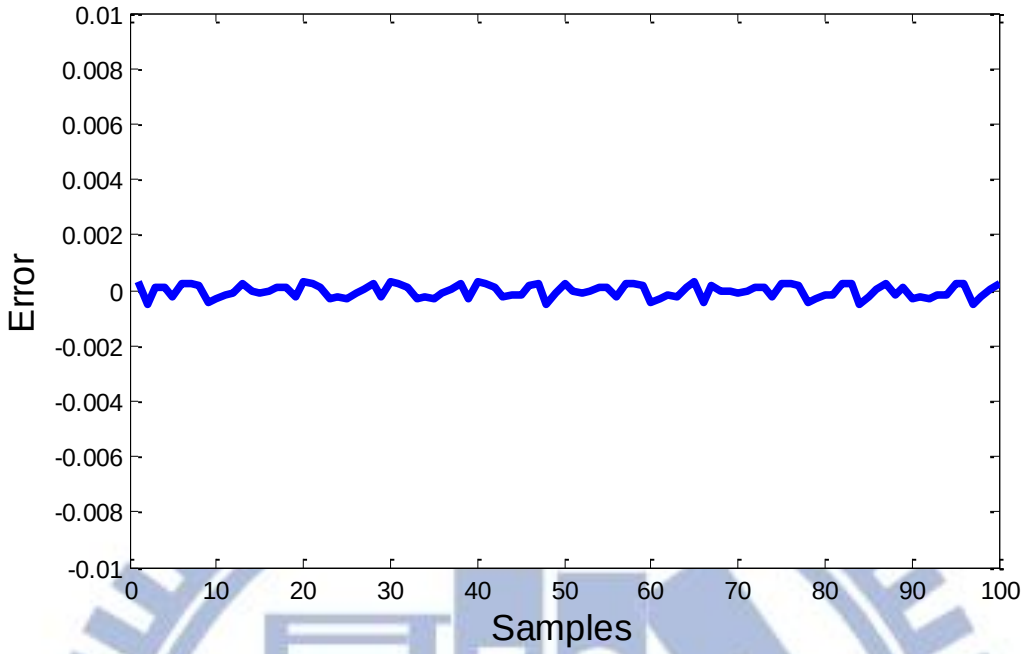


Figure 4.12: Prediction errors of the FLFS-GDE model for example 1.

Table 4.2. Performance of the GDE algorithm and the other algorithms for example 1.

	DE	jDE	MODE	GDE
No. of Rules (Parameters)	2 (12)			
Training RMSE (Mean ± STD)	0.0071 ±0.0018	0.0041 ±0.002	0.0028 ±0.005	**0.0012** ±0.001
Testing RMSE (Mean ± STD)	0.0074 ±0.002	0.0044 ±0.002	0.0023 ±0.006	**0.0015** ±0.002

Table 4.3. The best performance of the FLFS-GDE model and other papers for example 1.

Method	Rules(Parameters)	Testing RMSE
FLNFN-PSO[13,76]	3(18)	0.0055
FLNFN-CPSO[84,76]	3(18)	0.0039
FLNFN-CCPSO[76]	3(18)	0.0027
FLFS-GDE	**2(12)**	**0.00025**

Example 2: Prediction of Mackey–Glass time series

The time-series prediction problem used in this example is the chaotic Mackey–Glass time series, which is generated from the following differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{34}$$

where $\tau > 17$. As in previous studies [76], the parameter $\tau = 30$, and $x(0) = 1.2$ in this simulation. Four past values are used to predict $x(t)$, and the input–output pattern format is given by $[x(t-24), x(t-18), x(t-12), x(t-6) \mid x(t)]$.

A total of 1000 patterns are generated from t = 124 to 1123, where the first 500 patterns [form $x(1)$ to $x(500)$] are used to train, and the last 500 patterns [form $x(501)$ to $x(1000)$] are used to test. A total of 50 runs are performed for statistical analysis. The agglomerative clustering algorithm find the optimal number of fuzzy rules = 3 for Mackey–Glass time series data. The result of agglomerative clustering algorithm is shown in Figure 4.13. Figure 4.14 shows the learning curves of the DE, jDE, MODE and GDE algorithms for example 2. The learning curve of the PSO and DE algorithms present a rapid convergence result over the first 150 generations that became trapped at local minimum solutions at training average

RMSE = 0.066 and 0.069, respectively. The result of the MODE algorithm keep convergence after 500 generations, and this result is better than those of the DE and jDE algorithms. The performance of the GDE algorithm obtained a training average RMSE=0.019, which is better than the other algorithms for example 2. Table 4.4 shows that the average performance of the GDE algorithm compared with those of DE, jDE and MODE over 50 runs. The results show that the GDE algorithm for FLFS optimisation offers a smaller testing RMSE than the other methods. Table 4.5 shows that the best testing RMSE of FLFS-GDE, FLNFN-CCPSO[76], RBF-AFS[86], HyFIS[87], NEFPROX[88], D-FNN[89], GA-FLC [65], SEFC [85], Back-propagation NN, Six-order polynomial, Cascaded-correlation, Auto regressive model and Linear predictive from other journal papers. The proposed FLFS-GDE model achieves a better performance than other evolutionary algorithms. Figure 4.15 shows the prediction results of the FLFS-GDE model for the desired output and the actual output.

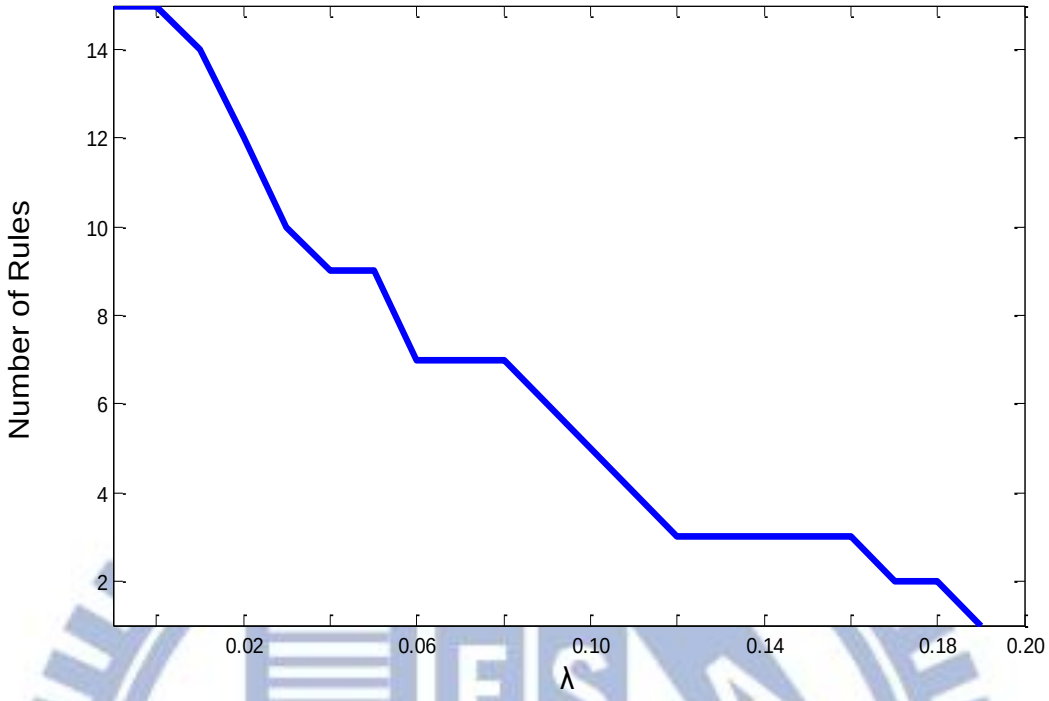


Figure 4.13:The result of the agglomerative clustering algorithm for example 2.

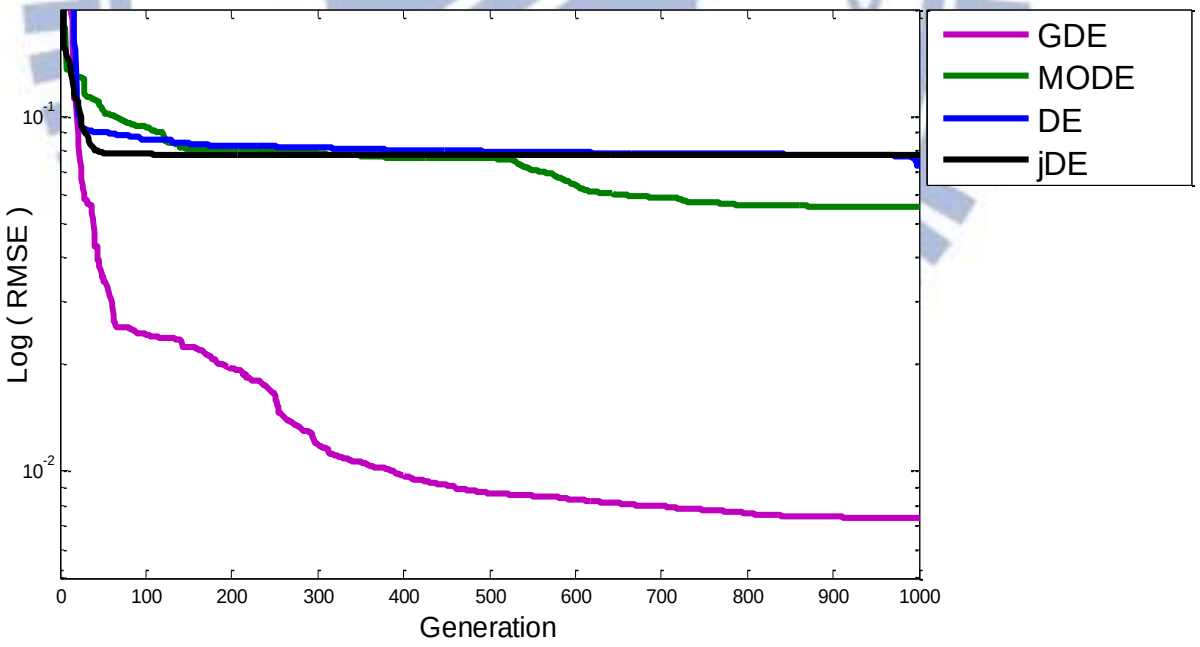


Figure 4.14: Training RMSEs of the DE, jDE, MODE and GDE algorithms at each performance evaluation for example 2.

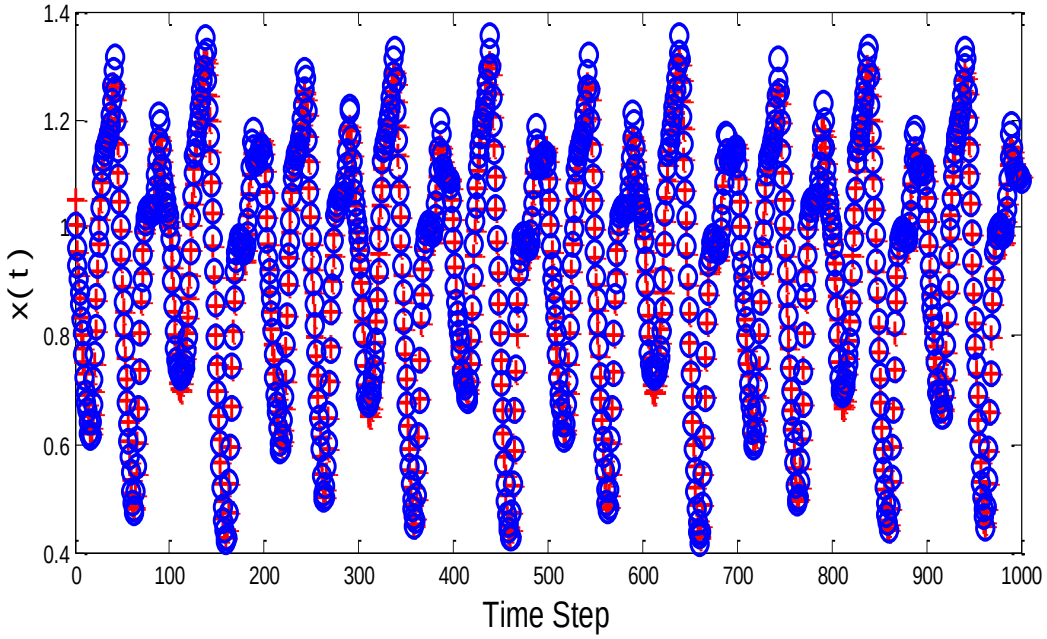


Figure 4.15: Symbol "+" represents the desired results and "O" represents the prediction results of the FLFS-GDE model for example 2.

Table 4.4. Performance of the GDE algorithm and the other algorithms for example 2.

	DE	jDE	MODE	GDE
No. of Rules (Parameters)	3 (63)			
Training RMSE (Mean ± STD)	0.066 ±0.018	0.069 ±0.021	0.048 ±0.015	**0.019** ±0.008
Testing RMSE (Mean ± STD)	0.075 ±0.022	0.072 ±0.020	0.050 ±0.022	**0.023** ±0.014

Table 4.5. The best performance of the FLFS-GDE model and other papers for example 2.

Method	Rules (Parameters)	Testing RMSE
FLFS-GDE	**3 (63)**	**0.0075**
FLNFN-CCPSO[76]	3(63)	0.0082
RBF-AFS[86]	13(130)	0.0131
HyFIS[87]	16(104)	0.0101
NEFPROX[88]	-(105)	0.053
D-FNN[89]	5(100)	0.008
GA-FLC [65]	-	0.26
SEFC [85]	-	0.032
Back-propagation NN	-	0.02
Six-order polynomial	-	0.04
Cascaded-correlation	-	0.06
Auto regressive model	-	0.19
Linear predictive	-	0.55

Example 3: Prediction of Auto-MPG6 data

This is a real-world problem that concerns the prediction of automobile city-cycle fuel consumption, in miles per gallon (MPG). There are five inputs and one output in the prediction model. The real dataset contains 398 examples and can be downloaded from KEEL(http://www.keel.es/)[90]. Evaluation of this model used the five-fold cross-validation datasets in KEEL. The inputs are scaled to the range [0, 1]. For each cross-validation dataset, a learning algorithm is repeated for ten runs. For rule generation, we obtain the best number of fuzzy rules = 4.2 by the agglomerative clustering algorithm. The result of agglomerative clustering algorithm is shown in Figure 4.16. Figure 4.17 shows the learning curves of the DE, jDE, MODE and GDE algorithms for example 3. Table 4.6 shows the performances of the DE, jDE, MODE and GDE algorithms using the same number of rules for the FLFS optimisation. In this table, the result of the GDE algorithm is better than that of the DE, jDE and MODE algorithms for example 1, 2 and 3. We also compare the performance of our method with other papers, and the comparison results are tabulated in Table 4.7. According to these results, the proposed FLFS-GDE model outperforms FS-HGAPSO [77], MOGUL-TSK[92], FS-CPSO[84] and FS-HPSO-TVAC[91]. Figure 4.18 shows the training output of the FLFS-GDE model for the desired output (blue line) and the actual output (red line). Figure 4.19 shows the testing result of the FLFS-GDE model.

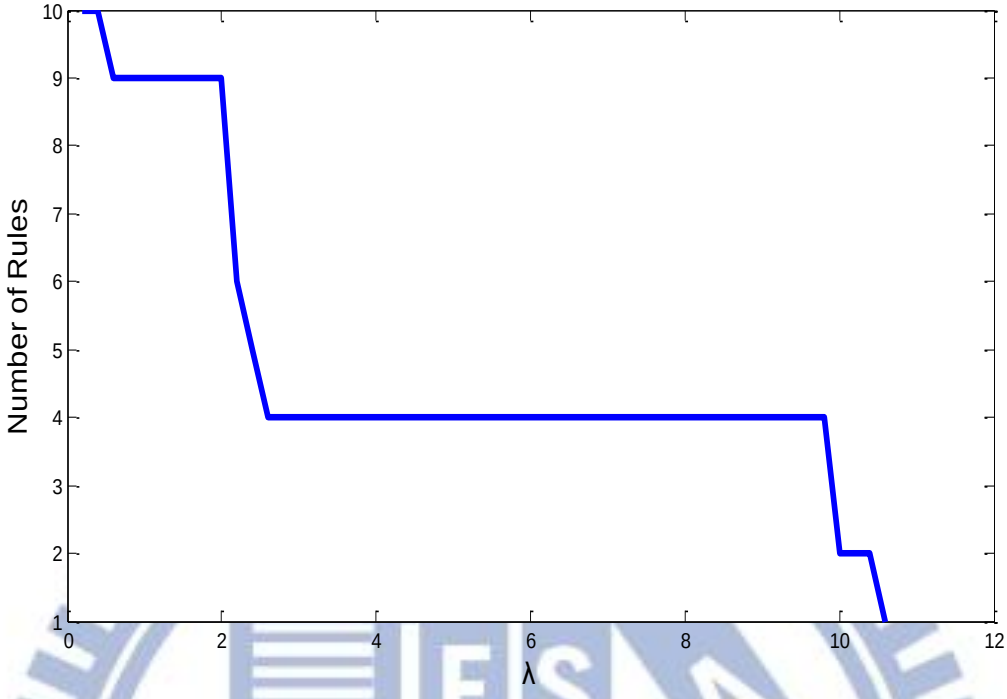


Figure 4.16: The result of the agglomerative clustering algorithm for example 3.

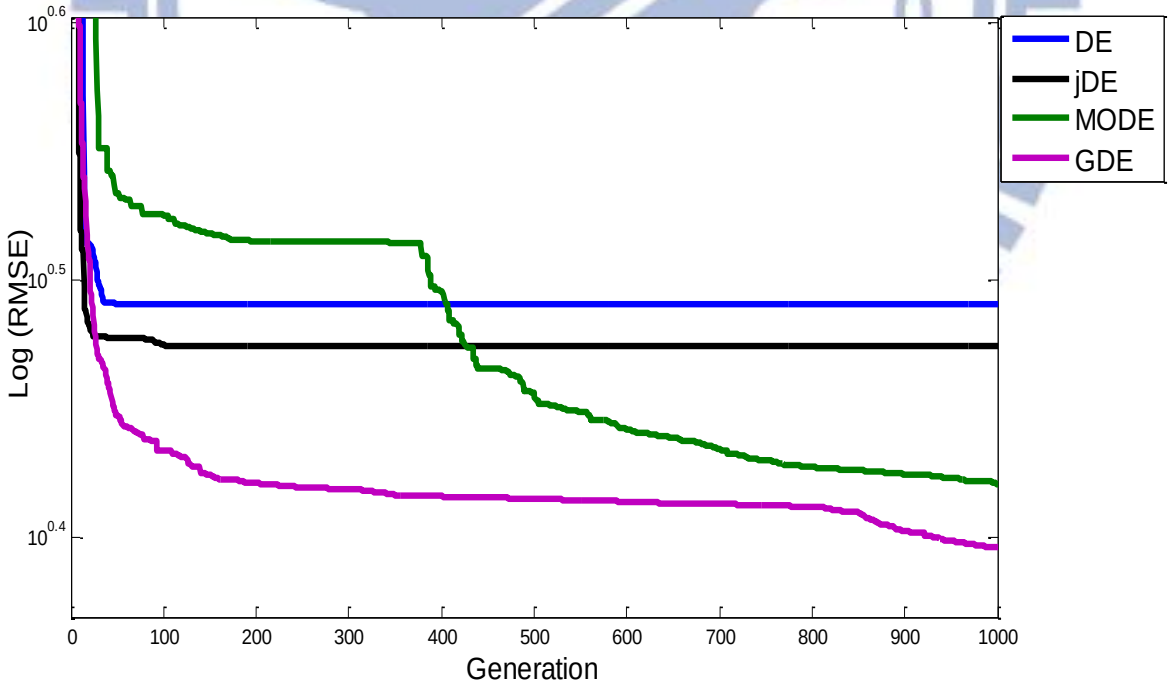


Figure 4.17: Training RMSEs of the DE, jDE, MODE and GDE algorithms at each performance evaluation for example 3.

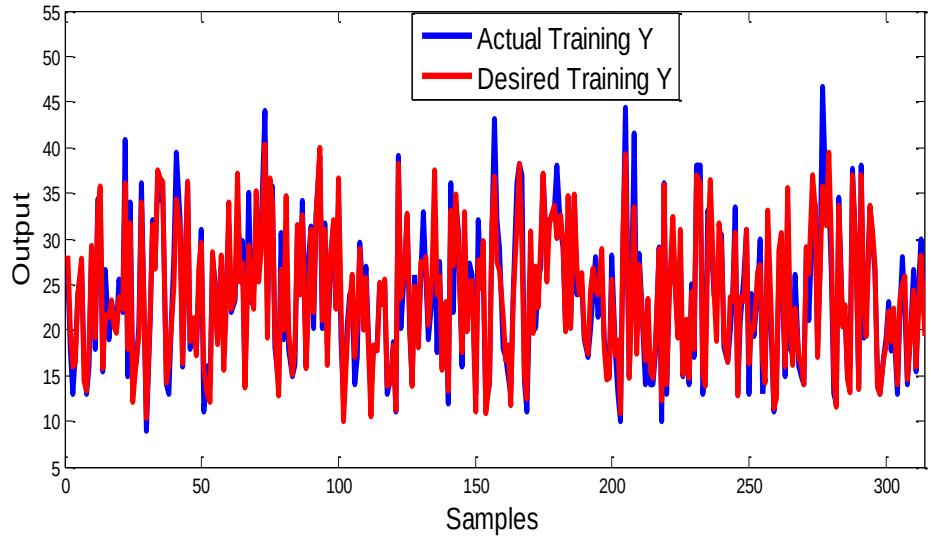


Figure 4.18: The training output of the FLFS-GDE model for example 3.

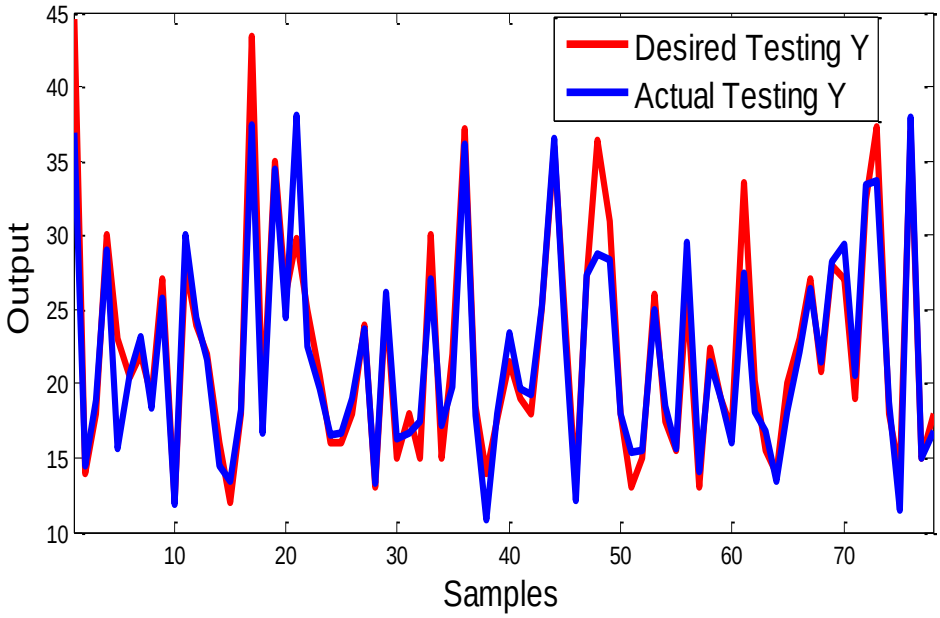


Figure 4.19: The testing output of the FLFS-GDE model for example 3.

Table 4.6: The performance of the GDE algorithm and other algorithms for example 3.

	DE	jDE	MODE	GDE
No. of Rules (Parameters)	4 (104)			
Training RMSE (Mean ± STD)	3.35 ±0.56	3.27 ±0.56	2.51 ±0.22	2.36 ±0.15
Testing RMSE (Mean ± STD)	3.66 ±0.68	3.61 ±0.72	2.89 ±0.34	2.58 ±0.21

Table 4.7: Comparison of the FLFS-GDE model and other papers for example 3.

Method	Testing RMSE
FLFS-GDE	**2.58**
FS-HGAPSO [77]	2.97
MOGUL-TSK[92]	5.16
FS-CPSO[84]	2.66
FS-HPSO-TVAC[91]	2.72

Example 4: Prediction of oil price time series

A practical prediction problem of oil price time series is considered in this paper. This dataset recorded the average annual price of oil time series from 1870 to 1997. The oil price time series dataset can be downloaded from *http://www-personal.buse -co.monash.edu.au/~hyndman/TSDL/*. 128 samples are used, each with two inputs and one output, i.e. $y_t = f(y_{t-1}, y_{t-2})$. The first 64 samples are used for training and the last 64 samples are used for testing. For fair comparison, we perform the same normalized process [5] to scale all samples within the range [-1, 1]. In this simulation, the FLFS-GDE model is repeated for 50 runs and we obtain two fuzzy rules after rule generation. The result of agglomerative clustering algorithm is shown in Figure 4.20. The best prediction performance of FLFS-GDE model is about MSE=0.0132. Figure 4.21 shows the prediction result of FLFS-GDE model for desired output (blue line) and actual output (red line) for example 4. Tables 4.8 shows the performances of TSK-NFIS[93], Autoregressive model[93], Nonlinear autoregressive model[93], Neural network[93], NFS-PSO-RLSE[94], CNFS-PSO-RLSE[95], CNFS-HMSPSO-RLSE[96], FLFS-DE and FLFS-GDE for prediction problems. The proposed FLFS-GDE model which achieves a significant performance is superior to other algorithm for example 4.

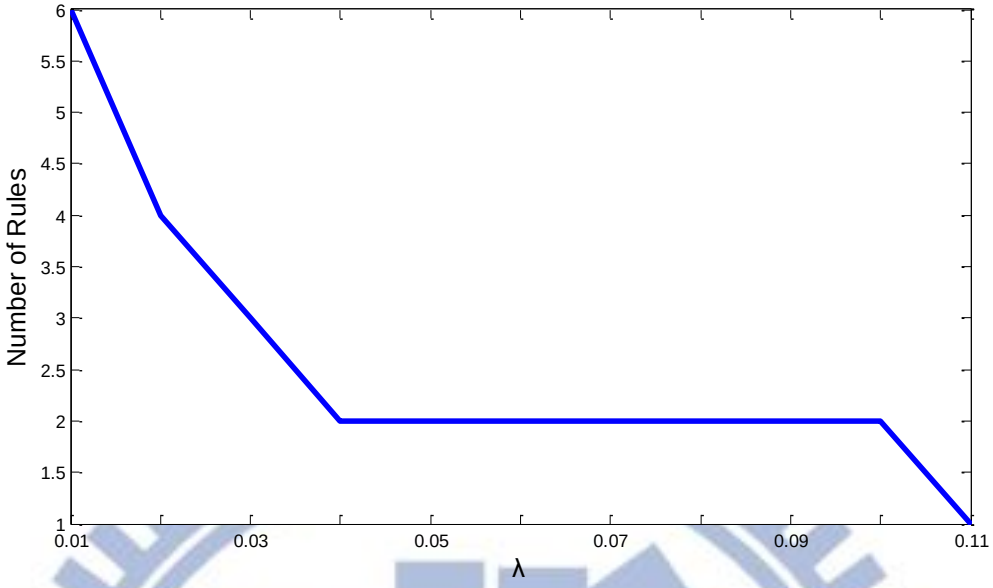


Figure 4.20: The result of the agglomerative clustering algorithm for example 4.

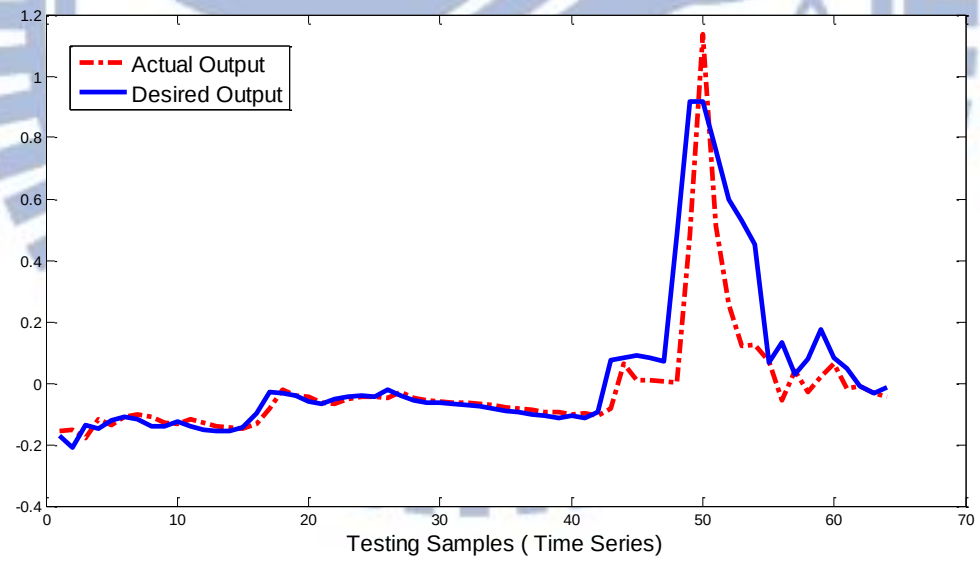


Figure 4.21: Prediction output of FLFS-GDE model for example 4.

Table 4.8: Performance of the FLFS-GDE model and other algorithms for example 4.

Method	Rules (Parameters)	Training MSE	Testing MSE
TSK-NFIS[93]	-	0.00431	0.0237
Autoregressive model[93]	-	0.00545	0.0244
Nonlinear Autoregressive model[93]	-	0.00499	0.0327
Neural network[93]	-	0.00469	0.0254
NFS-PSO-RLSE[94]	4(28)	0.00198	0.0259
CNFS-PSO-RLSE[95]	4(36)	0.00203	0.0163
CNFS-HMSPSO-RLSE[96]	4(36)	0.00221	0.0134
FLFS-DE	2(22)	0.00215	0.0244
FLFS-GDE	2(22)	0.00258	**0.0132**

Example 5 : Prediction of star brightness time series

In this example, an FLFS-GDE model is used to predict the star brightness time series. This real data measures the brightness of a star in 600 successive midnights. The dataset is obtained from *http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/*. 600 samples are used, each with three inputs and one output, i.e. $y_t = f(y_{t-1}, y_{t-2}, y_{t-3})$. The first 300 samples are used for training the FLFS-GDE model and the remaining 300 samples are used for testing phase.

In this simulation, we perform the same normalized process [5] to scale all samples within the range [-1, 1]. The FLFS-GDE model and the FLFS-DE model are repeated for 50 runs and the standard deviation of performance error is a small value. After agglomerative clustering algorithm, three fuzzy rules are generated for predicting star brightness time series.

The result of agglomerative clustering algorithm is shown in Figure 4.22. Figure 4.23 plots the prediction outputs of FLFS-GDE model for predicting star brightness time series. Table 4.9 shows the performances of the proposed FLFS-GDE model and other journal papers. The proposed FLFS-GDE model obtains the best performance MSE=0.000249 which is better than TSK-NFIS[93], Autoregressive model[93], Nonlinear autoregressive model[93], Neural network[93], NFS-PSO-RLSE[94], NFS-ARIMA[97], CNFS-PSO-RLSE[95] and CNFS-HMSPSO-RLSE[96].

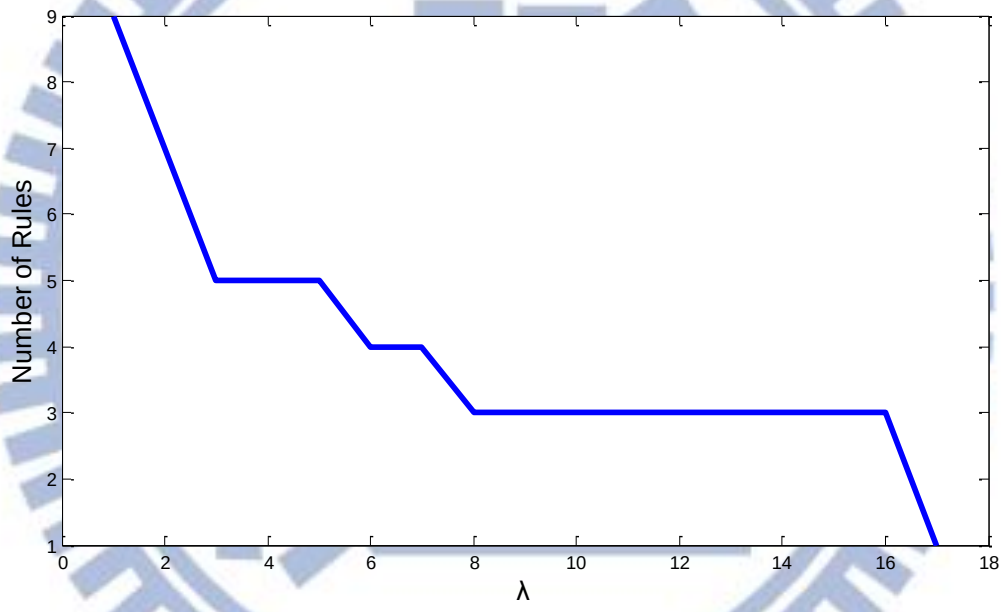


Figure 4.22: The result of the agglomerative clustering algorithm for example 5.

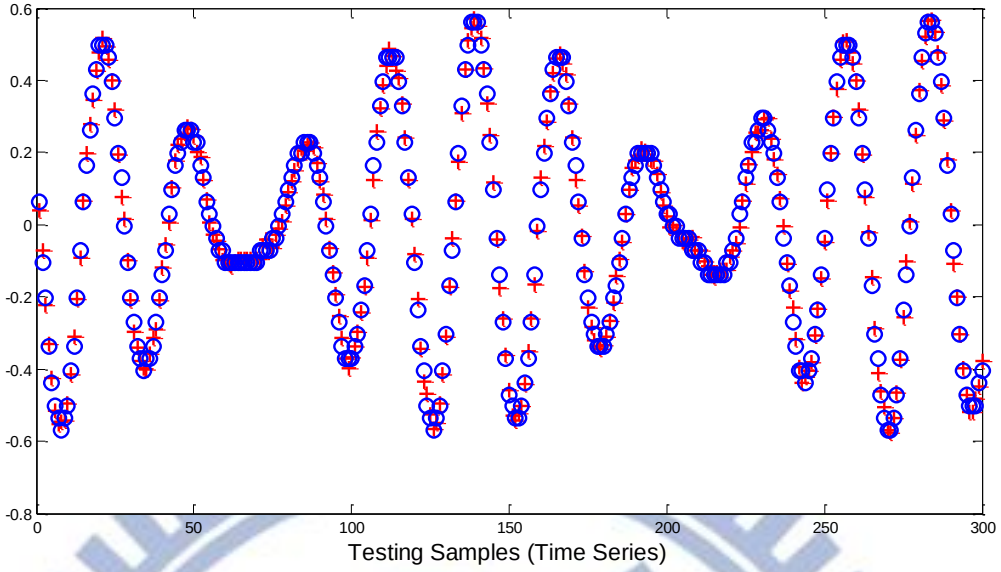


Figure 4.23: Symbol "+" represents desired and "O" represents prediction results of the FLFS-GDE model for example 5.

Table 4.9: Performance of the FLFS-GDE model and other algorithms for example 5.

Method	Rules (Parameters)	Training MSE	Testing MSE
TSK-NFIS[93]	-	0.000313	0.000331
Autoregressive model[93]	-	0.000304	0.000322
Nonlinear Autoregressive model[93]	-	0.000320	0.000312
Neural network[93]	-	0.000301	0.000311
NFS-PSO-RLSE[94]	8(84)	0.000199	0.000324
CNFS-PSO-RLSE [95]	8(108)	0.000198	0.000280
CNFS-HMSPSO-RLSE[96]	8(108)	0.000198	0.000272
NFS-ARIMA[97]	8(84)	0.000209	0.000264
FLFS-DE	3(48)	0.000255	0.000282
FLFS-GDE	3(48)	0.000246	**0.000249**

Chapter 5

Conclusions

This dissertation proposes a group-based differential evolution algorithm (GDE) for global optimization problems. The GDE algorithm combines two classical mutation strategies instead of a single mutation model for solving the stagnation problem. An adaptive strategy is also proposed in this dissertation. This strategy uses successful information to automatically tune factor *F* and crossover rate *CR*. The advantages of the proposed GDE algorithm are summarized below.

(1) The proposed GDE algorithm employs the inherent properties of the DE algorithm to solve the stagnation problem. The GDE algorithm combines the two mutation operations to tradeoff between the exploration ability and the exploitation ability.

(2) An adaptive strategy automatically tunes parameters without the user's prior knowledge. This strategy collects successful factor *F* and crossover rate *CR* to generate potential parameters for the next generation.

(3) Thirteen well-known numerical benchmark functions are tested to validate the performance of the proposed GDE algorithm. The GDE algorithm shows significantly better performance than other EAs in statistical tests.

Furthermore, we also propose a learning algorithm for function-link fuzzy system (FLFS) optimization. The proposed learning algorithm includes agglomerative clustering

algorithm and evolution process. The agglomerative clustering algorithm constructs the optimal structure. The evolution process comprises multi-subpopulation that uses each individual represents a single fuzzy rule and each individual in each subpopulation evolves separately using a GDE algorithm. The advantages of the FLFS model with GDE algorithm (FLFS-GDE) are summarized as follows:

(1) The consequent of the FLFS model is a nonlinear combination of input variables. This study uses the functional-link neural network to the consequent part of the fuzzy rules. The local properties of the consequent part in the FLNFN model enable a nonlinear combination of input variables to be approximated more effectively.

(2) An automatic process based on agglomerative clustering algorithm can construct the optimal number of fuzzy rules for the structure of the FLFS model. In this algorithm, we just easily assign two parameter values instead of the trial and error process.

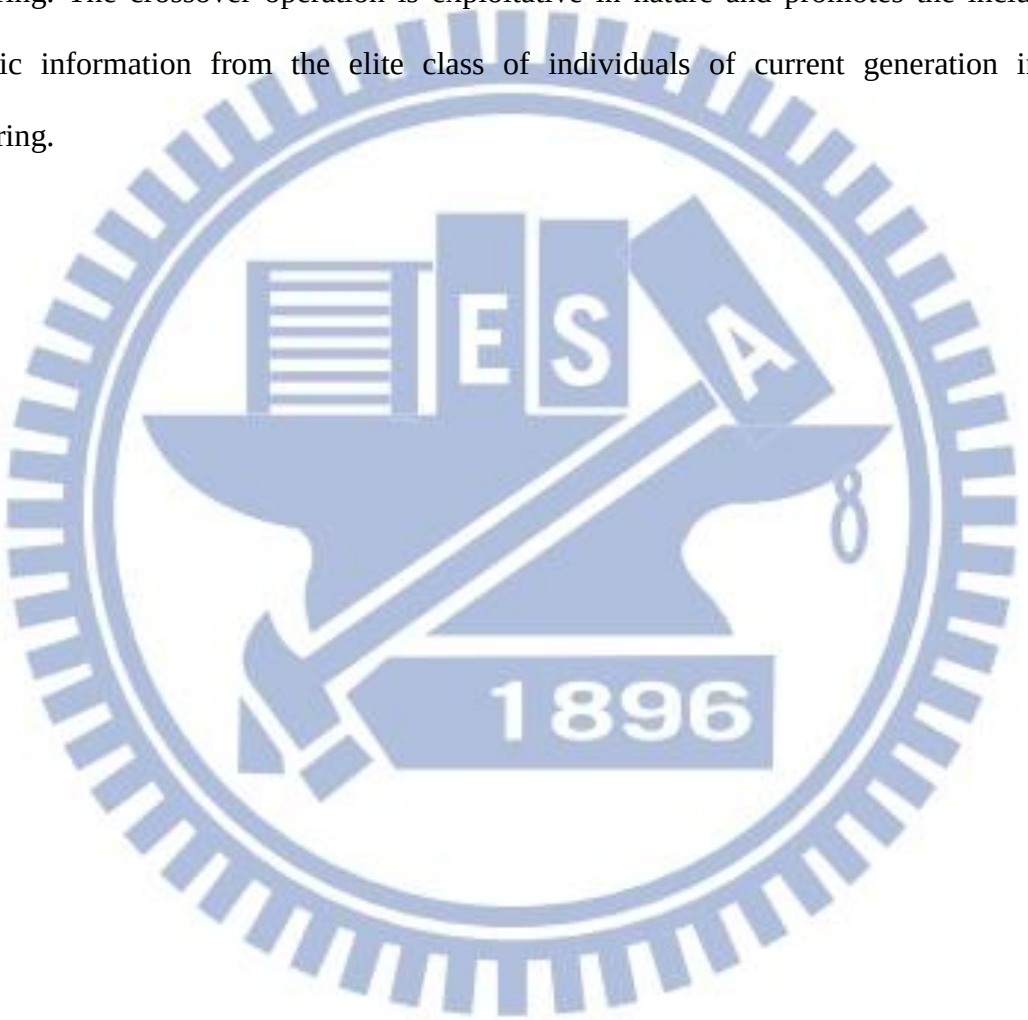
(3) The evolution process adopts a subpopulation symbiotic method which uses the rule-based subpopulation to evolve separately.

(4) The evolution process adopts a GDE algorithm to effectively search potential individuals.

(5) As demonstrated in section 4.4, the proposed FLFS-GDE model is a more adaptive and effective predictor than the other models.

Two advanced topics on the proposed model should be addressed in future research. First, the proposed GDE algorithm will tend to apply large-scale problems or overly complex problems. In this dissertation, the proposed GDE algorithm is limited to some small-scale problems (less than 100 dimensions). The scalability performance of the GDE algorithm is unclear. Second, the crossover operation in the GDE algorithm is also an important evolution operation which influences the performance of the proposed algorithm. We may modify the

crossover operation to improve the performance. For example, Islam *et al.* [46] presented a novel crossover operation for the DE algorithm. The novel crossover operation incorporates a greedy parent selection strategy with the conventional binomial crossover scheme of the DE algorithm. In the crossover operation, a binomial crossover is performed between the current donor vector and any other individual from p top-ranked individuals for the new offspring. The crossover operation is exploitative in nature and promotes the inclusion of genetic information from the elite class of individuals of current generation into the offspring.



References

[1] C. C. Carlos A, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 1245-1287, 2002.

[2] P. Fei, T. Ke, C. Guoliang, and Y. Xin, "Population-based algorithm portfolios for numerical optimization," *IEEE Trans. Evolutionary Computation*, vol. 14, pp. 782-800, 2010.

[3] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. New York: IEEE Press, 1995.

[4] M. J. Gacto, R. Alcalá and F. Herrera, "A multi-objective evolutionary algorithm for an effective tuning of fuzzy logic controllers in heating, ventilating and air conditioning systems," *Applied Intelligence*, vol. 36, No. 2, pp. 330-347, 2012.

[5] K. S. Shin, Y.-S. Jeong and M. K. Jeong, "A two-leveled symbiotic evolutionary algorithm for clustering problems," *Applied Intelligence*, vol. 36, No. 4, pp. 788-799, 2012.

[6] D. Ayvaz, H. R. Topcuoglu and F. Gurgen," Performance evaluation of evolutionary heuristics in dynamic environments," *Applied Intelligence*, vol. 37, No. 1, pp. 130-144, 2012.

[7] E. E. Korkmaz, "Multi-objective Genetic Algorithms for grouping problems," *Applied Intelligence*, vol. 33, No. 2, pp. 179-192, 2010.

[8] M. Mitchell, *An Introduction to Genetic Algorithms* (Complex Adaptive Systems). Cambridge, MA: MIT Press, 1998.

[9] L. J. Fogel, *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. New York: Wiley, 1999.

[10] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, vol. 3, pp. 82 - 102, 1999.

[11] H. G. Beyer and H. P. Schwefel, "Evolution strategies: A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[12] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. San Francisco,CA: Morgan Kaufmann Publishers, 2001.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. *IEEE Int. Neural Netw*, 1995.

[14] K. Price, R. Storn and J. Lampinen, *DifferentialEvolution: A Practical Approach to GlobalOptimization*. Berlin: Springer-Verlag,2005.

[15] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization,*, vol.

11, pp. 341-359, 1997.

[16] Z. Cai, W. Gong, C. X. Ling and H. Zhang, "A clustering-based differential evolution for global optimization," *Applied Soft Computing,* vol. 11, pp. 1363-1379, 2011.

[17] C.-H. Chen, C.-J. Lin and C.-T. Lin, "Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution," *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews,* , vol. 39, pp. 459-473, 2009.

[18] S. Das, A. Abraham, U. K. Chakraborty and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evolutionary Computation,* vol. 13, pp. 526-553, 2009.

[19] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Trans. Evolutionary Computation,* vol. 15, pp. 4-31, 2011.

[20] H. R. Cheshmehgaz, M. I. Desa and A. Wibowo, "Effective local evolutionary searches distributed on an island model solving bi-objective optimization problems," *Applied Intelligence,* 2012. (In Press)

[21] R. Vafashoar, M. R. Meybodi and A. H. Momeni Azandaryani, "CLA-DE: a hybrid model based on cellular learning automata for numerical optimization," *Applied Intelligence*, vol. 36, No. 3. pp. 735-748, 2012.

[22] Z. Jingqiao and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Trans. Evolutionary Computation,* vol. 13, pp. 945-958, 2009.

[23] E. Mezura-Montes, M. E. Miranda-Varela and R. del Carmen Gmez-Ramn, "Differential evolution in constrained numerical optimization: An empirical study," *Information Sciences,* vol. 180, pp. 4223-4262, 2010.

[24] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evolutionary Computation,* vol. 12, pp. 107-125, 2008.

[25] A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Trans.Evolutionary Computation,* , vol. 13, pp. 398-417, 2009.

[26] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," Congress on IEEE Evolutionary Computation, pp. 1785-1791 Vol. 2, 2005.

[27] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evolutionary Computation,* vol. 12, pp. 64-79, 2008.

[28] 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, pp. 4847-4858, 2011.

[29] G. Wenyin, C. Zhihua, C. X. Ling and L. Hui, "Enhanced differential evolution with

adaptive strategies for numerical optimization," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 41, pp. 397-413, 2011.

[30] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," Congress on Evolutionary Computation, 2004, pp. 1980-1987 Vol.2, 2004.

[31] C.-T. Lin, M.-F. Han, Y.-Y. Lin, J.-Y. Chang and L.-W. Ko, "Differential Evolution based Optimization of Locally Recurrent Neuro-Fuzzy System for Dynamic System Identification," presented at the The 17th National Conference on Fuzzy Theory and its Applications, 2010.

[32] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, "Self-Adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Trans. Evolutionary Computation,* vol. 10, pp. 646-657, 2006.

[33] C.-T. L., M.-F. H., Y.-Y. L., S.-H. L. and J.-Y. C., "Neuro-fuzzy system design using differential evolution with local information," in Fuzzy Systems (FUZZ), 2011 IEEE International Conference on, 2011, pp. 1003-1006.

[34] T. Josef, "Adaptation in differential evolution: A numerical comparison," *Applied Soft Computing,* vol. 9, pp. 1149-1155, 2009.

[35] L. Junhong and L. Jouni, "A fuzzy adaptive differential evolution algorithm," in *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, pp. 606-611 vol.1, 2002.

[36] M. Ali and M. Pant, "Improving the performance of differential evolution algorithm using Cauchy mutation", *Soft Computing,* vol. 15, pp. 991-1007, 2011.

[37] T. Niknam, HD. Mojarrad, and M. Nayeripour, "A New Hybrid Fuzzy Adaptive Particle Swarm Optimization for Non-Convex Economic Dispatch," *International Journal of Innovative Computing Information and Control*, vol. 7, pp. 189-202, JAN. 2011.

[38] Y.-W. Shang and Y.-H. Qiu, "A note on the extended rosenbrock function," *Evolutionary Computation,* vol. 14, pp. 119–126, 2006.

[39] X. Yao, Y. Liu, K.-H. Liang and G. Lin, "Fast evolutionary algorithms," presented at the Advances Evol. Computing: Theory Applicat., New York, 2003.

[40] Z. Yang, J. He and X. Yao, "Making a difference to differential evolution," in *Advances Metaheuristics Hard Optimization*, pp. 397–414, 2007.

[41] J. Demˇsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research,* pp. 1-30, 2006.

[42] S. Garc´ıa and F. Herrera, "An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons," *Journal of Machine Learning Research* pp. 2677-2694, 2008.

[43] C. Lee and X. Yao, "Evolutionary programming using mutations based on the Lévy probability distribution," *IEEE Trans. Evolutionary Computation*, vol. 8, pp. 1–13, 2004.

[44] M. S. Alam, M. M. Islam, F. Xin Yao and K. Murase, "Recurring two-stage evolutionary programming: a novel approach for numeric optimization," *IEEE Trans.n Systems, Man, And Cybernetics,Part B: Cybernetics*, vol. 41, pp. 1352-1365, 2011.

[45] M. M. Islam, M. S. Alam and K. Murase, "A new recurring multistage evolutionary algorithm for solving problems efficiently," *Lecture Notes in Computer Science*, vol. 4881, pp. 97–106, 2007.

[46] Sk. M. Islam, S. Das, S. Ghosh, S. Roy and P.N.Suganthan, " An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics, ,* vol. 42, No. 2, pp. 482-500,2012.

[47] A. Ghosh, S. Das, A. Chowdhury and R. Giri, " An improved differential evolution algorithm with fitness-based adaptation of the control parameters," *Information Sciences*, vol. 181, No. 18, pp. 3749–3765, 2011.

[48] C. T. Lin and C. S. G. Lee, Neural Fuzzy Systems: *A Neuro-Fuzzy Synergism to Intelligent System*. Englewood Cliffs, NJ: Prentice-Hall, 1996

[49] Y. H. Chien, W. Y. Wang, Y. G. Leu and T.T. Lee, "Robust adaptive controller design for a class of uncertain nonlinear systems using online T–S fuzzy-neural modeling approach," *IEEE Trans. Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 41, no. 2, pp. 542–552, Apr. 2011.

[50] R. Prakash and R. Anita, "Modeling and simulation of fuzzy logic controller-based model reference adaptive controller", *International Journal of Innovative Computing Information and Control*, vol. 8, no.4, pp. 2533-2550, Apr. 2012.

[51] C.-J. Lin and C.-C. Peng," Self-Adaptive quantum radial basis function network for classification applications ", *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 8, Aug. 2011.

[52] T. Chen and Y.-C. Lin, "A collaborative fuzzy-neural approach for internal due date assignment in a wafer fabrication plant," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 9, pp. 5193–5210, Sep. 2011.

[53] C.-H. Chen, C.-J. Lin and C.-T. Lin, "A functional-link-based neurofuzzy network for nonlinear system control," *IEEE Trans. Fuzzy System*, vol. 16, no. 5, pp. 1362–1378, Oct. 2008.

[54] C. F. Juang, R. B. Huang and Y. Y. Lin, "A recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing," *IEEE Trans. Fuzzy System*, vol.17, no.5, pp.1092-1105, Oct. 2009.

[55] M.-F. Han, C.-T. Lin and J.-Y. Chang, "A compensatory neurofuzzy system with online constructing and parameter learning," Proc. of 2010 IEEE International Conference on Systems, Man, and Cybernetics., pp. 552–556, Oct. 2010.

[56] C.-F. Juang and P.-H. Chang, "Designing fuzzy rule-based systems using continuous ant colony optimization," *IEEE Trans. Fuzzy System*, vol.18, no.1, pp.138-149, Feb. 2010.

[57] C.-F. Juang, "Combination of on-line clustering and Q-value based GA for reinforcement fuzzy system design," *IEEE Trans. Fuzzy System*, vol. 13, no. 3, pp. 289–302, Jun. 2005.

[58] F. Hoffmann, D. Schauten and S. Holemann, "Incremental evolutionary design of TSK fuzzy controllers," *IEEE Trans. Fuzzy Syst*em, vol. 15, no. 4, pp. 563–577, Aug. 2007.

[59] E. Sanchez, T. Shibata and L. A. Zadeh, *Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives.* Singapore: World Scientific, 1997.

[60] O. Cordoon, F. Herrera, F. Hoffmann and L. Magdalena, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, Advances in Fuzzy Systems—Applications and Theory.*, Singapore: World Scientific, 2001.

[61] M. Russo, "Genetic fuzzy learning," *IEEE Trans. Evolutionary Computation*, vol. 4, no. 3, pp. 259–273, Sep. 2000.

[62] K. C. Ng and T. Li, "Design of sophisticated fuzzy logic controllers using genetic algorithms," in Proc. 3rd IEEE Int. Conf. Fuzzy Systems, pp. 1708-1711, 1994.

[63] T. L. Seng, M. B. Khalid and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithm," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, pp. 226-236, Apr. 1999.

[64] C.-H. Chou, "Genetic algorithm-based optimal fuzzy controller design in the linguistic space," *IEEE Trans. Fuzzy System*, vol. 14, no. 3, pp. 372–385, Jun. 2006.

[65] C. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," Proceeding of 4th International Conference on Genetic Algorithms, pp. 450-457, 1991.

[66] C.-F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," *IEEE Trans. Fuzzy System*, vol. 10, no. 2, pp. 155–170, Apr. 2002.

[67] K. D. Sharma, A. Chatterjee and A. Rakshit, "A hybrid approach for design of stable adaptive fuzzy controllers employing Lyapunov theory and particle swarm optimization," *IEEE Trans. Fuzzy System*, vol. 17, no. 2, pp. 329–342, Apr. 2009.

[68] C.-F. Juang, C. M. Hsiao and C. H. Hsu,"Hierarchical cluster-based multispecies particle-swarm optimization for fuzzy-system optimization," *IEEE Trans. Fuzzy System*, vol. 18, no. 1, pp. 14–26, Feb. 2010.

[69] H. Lu, E. Pi, Q. Peng, L. Wang and C. Zhang, "A particle swarm optimization-aided fuzzy cloud classifier applied for plant numerical taxonomy based on attribute similarity," *Expert Systems with Applications*, vol. 36, no. 5, pp. 9388–9397, Jul.

2009.

[70] C.-J. Lin, C. C. Weng, C. Y. Lee and C. L. Lee, 2009, "Using an efficient hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy network design," 2009 International Conference on Machine Learning and Cybernetics, pp. 3076-3082, July, 2009.

[71] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[72] K. D. Sharma, A. Chatterjee and A. Rakshit, "A hybrid approach for design of stable adaptive fuzzy controllers employing Lyapunov theory and particle swarm optimization," *IEEE Trans. Fuzzy System*, vol. 17, no. 2, pp. 329–342, Apr. 2009.

[73] C.-J. Lin and C.-L. Lee ," A self-organizing neural network using hierarchical particle swarm optimization," International Joint Conference on Neural Networks (IJCNN 2011), 2011.

[74] J.-Y. Chang, M.-F. Han and C.-T. Lin, "Optimization of Fuzzy Systems Using Group-Based Evolutionary Algorithm," *Lecture Notes in Computer Science*, Vol. 7665, pp 291-298, 2012.

[75] M.-F. Han, C.-T. Lin, J.-Y. Chang and D.-L. Li, "Group-Based Differential Evolution for Numerical Optimization Problems," *International Journal of Innovative Computing, Information and Control*. Vol. 9, No. 3, pp. 1357-1372, Mar. 2013.

[76] C.-J. Lin, C.-H. Chen and C.-T. Lin, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy network and its prediction applications," *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39, no. 1, pp. 55–68, Jan. 2009.

[77] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[78] F. J. Lin, S. Y. Chen, L. T. Teng and H. Chu, "A recurrent FL-based fuzzy neural network controller with improved particle swarm optimization for linear synchronous motor drive," *IEEE Trans. Magnetics*, vol. 45, no. 8, pp.3151-3165, 2009.

[79] M.-F. Han, C.-T. Lin and J.-Y. Chang, "Efficient differential evolution algorithm based optimization of fuzzy prediction model for time series forecasting," *International Journal of Intelligent Information and Database Systems*. (Accepted)

[80] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in Proc. Genetic and Evolutionary Computation Conf., Las Vegas, NV, pp. 987–994, July 2000.

[81] H. Ishibuchi, T. Yoshida and T. Murata, "Balance between genetic algorithm and local search in memetic algorithms for multiobjective permutation flowshop scheduling,"

IEEE Trans. Evolutionary Computation, vol. 7, pp.204–223, Apr. 2003.

[82] M. J. Li, M. K. Ng, Y.-M. Cheung and Jo. Z. Huang, "Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters," *IEEE Trans. Knowledge and Data Engineering*, vol. 20, No. 11, pp.1519–1534, Nov. 2008.

[83] S. Miyamoto and M. Mukaidono, "Fuzzy c-means as a regularization and maximum entropy approach," Proc. Seventh Int'l Fuzzy Systems Assoc. World Congress (IFSA '97), vol. 2, pp. 86-92, 1997.

[84] F. Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, Jun. 2004.

[85] C. F. Juang, J. Y. Lin and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 2, pp. 290-302, Apr. 2000.

[86] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification," *Fuzzy Sets System*, vol. 83, pp. 325–339, 1996.

[87] J. Kim and N. K. Kasabov, "HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamic systems," *Neural Network*, vol. 12, pp. 1301–1319, 1999.

[88] D. Nauk and R. Kruse, "Neuro-fuzzy systems for function approximation," *Fuzzy Sets Syst.*, vol. 101, no. 2, pp. 261–271, 1999.

[89] S. Wu and M. J. Er, "Dynamic fuzzy neural networks—A novel approach to function approximation," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 2, pp. 358–364, Apr. 2000.

[90] J. Alcal´a-Fdez, L. S´anchez, S. Garc´ıa, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fern´andez and F. Herrera, "KEEL: A software tool to assess evolutionary algorithms to data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[91] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, Jun. 2004.

[92] R. Alcal´a, J. Alcal´a-Fdez, J. Casillas, O. Cord´on and F. Herrera, "Local identification of prototypes for genetic learning of accurate TSK fuzzy rule-based systems," *International Journal of Intelligent Systems*, vol. 22, pp. 909–941, 2007.

[93] D. Graves and W. Pedrycz," Fuzzy prediction architecture using recurrent neural networks", *Neurocomputing*, no. 72, pp. 1668– 1679, 2009.

[94] C. Li and T.-W. Chiang, "Function approximation with complex neuro-fuzzy system using complex fuzzy sets - a new approach," *New Generation Computing*, vol. 29, no. 3,

pp. 261-276, Jul. 2011.

[95] C. Li and T.-W. Chiang, "Complex fuzzy computing to time series prediction - a multi-swarm PSO learning approach," *Lecture Notes in Artificial Intelligence*, vol. 6592, pp.242-251, Apr. 2011.

[96] C. Li and T.-W. Chiang, " Complex fuzzy model with PSO-RLSE hybrid learning approach to function approximation", *International Journal of Intelligent Information and Database Systems*, vol. 5, no. 4, pp. 409 - 430, July 2011.

[97] C. Li and J.-W.Hu, "A new ARIMA-based neuro-fuzzy approach and swarm intelligence for time series forecasting," *Engineering Applications of Artificial Intelligence*. vol. 25, no. 2, pp. 295-308, March 2012.