Efficient Immune-Based Particle Swarm Optimization Learning for Neuro-Fuzzy Networks Design

CHENG-JIAN LIN[1], CHENG-HUNG CHEN[2] AND CHI-YUNG LEE[3]
[1]*Department of Computer Science and Information Engineering*
National Chin-Yi University of Technology
Taichung, 411 Taiwan
E-mail: cjlin@cyut.edu.tw
[2]*Department of Electrical and Control Engineering*
National Chiao Tung University
Hsinchu, 300 Taiwan
[3]*Department of Computer Science and Information Engineering*
Nan Kai Institute of Technology
Nantou, 542 Taiwan

In order to enhance the immune algorithm (IA) performance and find the optimal solution when dealing with difficult problems, we propose an efficient immune-based particle swarm optimization (IPSO) for use in TSK-type neuro-fuzzy networks for solving the identification and prediction problems. The proposed IPSO combines the immune algorithm (IA) and particle swarm optimization (PSO) to perform parameter learning. The IA uses the clonal selection principle, such that antibodies between others of high similar degree are affected, and these antibodies, after the process, will have higher quality, accelerating the search and increasing the global search capacity. The PSO algorithm has proved to be very effective for solving global optimization. It is not only a recently invented high-performance optimizer that is easy to understand and implement, but it also requires little computational bookkeeping and generally only a few lines of code. Hence, we employed the advantages of PSO to improve the mutation mechanism of immune algorithm. Experiments with synthetic and real data sets have performed in order to show the applicability of the proposed approach and also to compare with other methods in the literature.

Keywords: neuro-fuzzy network, immune system algorithm, particle swarm optimization, backpropagation, identification, prediction

1. INTRODUCTION

Neuro-fuzzy networks [1-4] are gaining research interest. They not only have attracted considerable attention in recent years due to their diverse applications in fields such as pattern recognition, image processing, and control, but they can also handle imprecise information through linguistic expressions. However, these models often have a serious drawback; that is, they use the backpropagation (BP) learning algorithm [2, 3]. Using the steepest descent optimization technique in BP training could minimize the error function, allowing the algorithm to reach the local minima very fast, but never finds a global solution. In addition, BP training performance depends on the initial system parameter values. For different network topologies one must derive new mathematical expressions for each network layer.

Received October 26, 2006; revised April 24, 2007; accepted June 1, 2007.
Communicated by Pau-Choo Chung.

Considering the aforementioned disadvantages, suboptimal performance occurs, even for a suitable neuro-fuzzy network topology. Hence, technologies capable of training the system parameters and finding the global solution while optimizing the overall structure are needed.

Recently, there has been a great deal of interest in the use of immune systems and algorithms in computer science and engineering [5-9]. Adem Kalinli [5] proposed an artificial immune algorithm for an infinite impulse response (IIR) filter design. Liao [6] embedded chaos search capability in immune genetic algorithms to solve short-term thermal generating unit commitment problems. Liao's method uses fuzzy systems to determine the rate of crossover and mutation mechanism. Wen [7] proposed an immune evolutionary algorithm for sphericity error evaluation. A self-adaptive mutation operator was constructed to divide the mutation step size of every antibody according to its environment. Zhou's proposal [8] was based on the immune recognition principle to predict the performances of hot-rolled steel bars. Chun [9] employed an immune algorithm (IA) as the search method for the shape optimization of an electromagnetic device. Chun's search method improved the global search performance of the genetic algorithm by using the immune network theory. However, these approaches in the fundamental methodologies are not dramatic in terms of overall performance. Recently, some researchers [10-12] have developed several hybrid methods that combine particle swarm optimization and immune algorithm. Ge and Liang [10] proposed an immune particle swarm optimization that based on the receptor editing in immune systems. The inactive particles are recognized and 25% of them are edited after each generation. Wang *et al.* [11] embedded the idea of the particle swarm optimization into the clonal selection algorithm. The antibodies can be improved using the particle swarm optimization before clonal selection. The principles of information diffusion and clonal selection are incorporated into the particle swarm optimization to achieve a better diversity and break away from local optimal solutions using InformPSO by Lv *et al.* [12]. In this study, we also proposed an efficient hybrid learning algorithm to avoid falling in a local optimal solution. Our method is different from [10-12]. We employed the advantages of PSO to improve the mutation scheme of immune algorithm.

This study presents the efficient immune-based particle swarm optimization (IPSO) for use in TSK-type neuro-fuzzy network to solve the identification and prediction problems. The proposed IPSO combines the immune algorithm (IA) and particle swarm optimization (PSO) to perform parameter learning. The IA uses the clonal selection principle to affect antibodies between others of high similar degree, and these antibodies, after the process, will be of higher quality, accelerating the search, and increasing the global search capacity. The PSO algorithm, proposed by Kennedy and Eberhart [13], has proved to be very effective for solving global optimization. It is not only a recently invented high-performance optimizer that is easy to understand and implement, but it also requires little computational bookkeeping and generally only a few lines of code [14]. In order to avoid trapping in a local optimal solution and to ensure the search capability of a near global optimal solution, mutation plays an important role in IPSO. Therefore, we employ the advantages of PSO to improve mutation mechanism of immune algorithm. The proposed method can improve the searching ability and greatly increase the converging speed that we can observe in the simulations.

2. STRUCTURE OF THE TSK-TYPE NEURO-FUZZY NETWORK

In this paper, we adopt a TSK-type neuro-fuzzy network with IPSO to solve the identification and prediction problems. The structure of a TSK-type neuro-fuzzy network is shown in Fig. 1, where N and R are, respectively, the number of input dimensions and the number of rules. It is a five-layer network structure. The representation of input x_i and output y is

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} \left\{ \prod_{i=1}^{N} \exp\left[-\frac{(u_i^{(1)} - m_{ij})^2}{\sigma_{ij}^2} \right] \right\} \cdot \left\{ w_{0j} + \sum_{i=1}^{N} w_{ij} x_i \right\}}{\sum_{j=1}^{R} \prod_{i=1}^{N} \exp\left[-\frac{(u_i^{(1)} - m_{ij})^2}{\sigma_{ij}^2} \right]} \tag{1}$$

where $u^{(l)}$ denotes the output of a node in the lth layer; $u_i^{(1)} = x_i$; m_{ij} and σ_{ij} are, respectively, the mean and the deviation of a Gaussian membership function of the jth term of the ith input variable x_i; and w_{ij} are the corresponding parameters of the consequent part. The detailed TSK-type neuro-fuzzy network is described in our previous research [15].

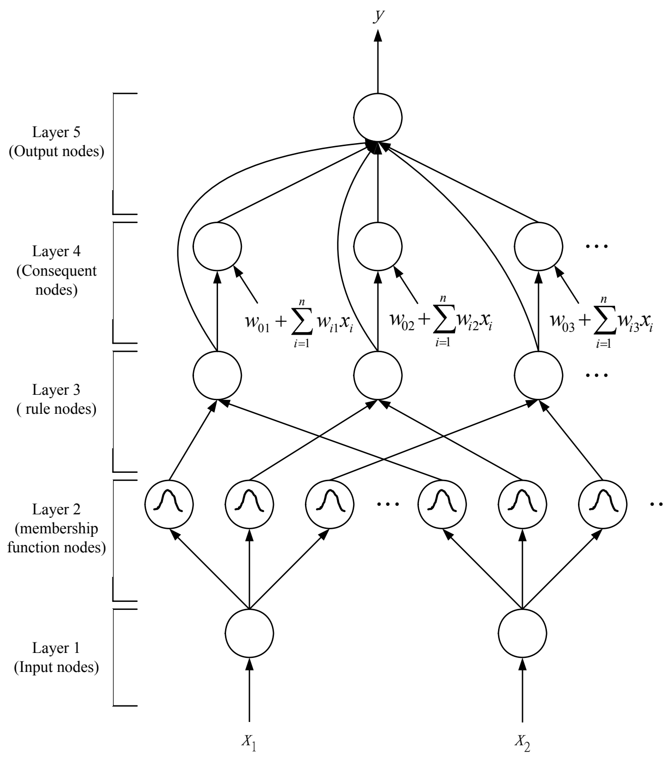

y
Layer 5 (Output nodes)
Layer 4 (Consequent nodes)
Layer 3 (rule nodes)
Layer 2 (membership function nodes)
Layer 1 (Input nodes)
$w_{01} + \sum_{i=1}^{n} w_{i1} x_i$
$w_{02} + \sum_{i=1}^{n} w_{i2} x_i$
$w_{03} + \sum_{i=1}^{n} w_{i3} x_i$
x_1
x_2

Fig. 1. The structure of the TSK-type neuro-fuzzy network.

3. THE EFFICIENT IMMUNE-BASED PARTICLE SWARM OPTIMIZATION (IPSO)

In this section, we describe and the immune-based particle swarm optimization (IPSO). The IA uses the clonal selection principle to accelerate the search and increase global search capacity. The PSO algorithm has proved to be very effective for solving global optimization. Therefore, an efficient immune-based particle swarm optimization (IPSO), which combines the immune algorithm (IA) and particle swarm optimization (PSO), is proposed for the TSK-type neuro-fuzzy network in this study.

Analogous to the biological immune system, the proposed algorithm has the capability of seeking feasible solutions while maintaining diversity. The proposed IPSO combines the immune algorithm (IA) and particle swarm optimization (PSO) to perform parameter learning. The IA uses the clonal selection principle to accelerate the search and increase global search capacity. The PSO algorithm has proved to be very effective for solving global optimization. It is not only a recently invented high-performance optimizer that is very easy to understand and implement, but it also requires little computational bookkeeping and generally only a few lines of code. In order to avoid trapping in a local optimal solution and to ensure the search capability of a near global optimal solution, mutation plays an important role in IPSO. Moreover, the PSO adopted in evolutionary algorithm yields high diversity to increase the global search capacity, as well as the mutation scheme. Therefore, we employed the advantages of PSO to improve the mutation mechanism of immune algorithm. A detailed IPSO of the TSK-type neuro-fuzzy network is presented in Fig. 2. The whole learning process is described step-by-step below.

(A) Coding step

The coding step is concerned with the membership functions and the corresponding parameters of the consequent part of a fuzzy rule that represent antibodies suitable for IPSO. We will discuss this first. This step codes a rule of a TSK-type neuro-fuzzy network into an antibody. Fig. 3 shows an example of a TSK-type neuro-fuzzy network coded into an antibody (*i.e.* an antibody represents a rule set), where i and j represent the ith dimension and the jth rule, respectively. In this paper, a Gaussian membership function is used with variables representing the mean and deviation of the membership function. Each fuzzy rule has the form in Fig. 1, where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation of the ith dimension and jth rule node and w_{ij} represents the corresponding parameters of consequent part.

(B) Determine the initial parameters by self-clustering algorithm

The initial parameters of a TSK-type neuro-fuzzy network were computed by the self-clustering algorithm (SCA) method [16]. That is, we used SCA method to determine the initial mean and deviation of the antecedent part. Before the IPSO method is designed, the initial antibodies in the populations are generated according to the initial parameters of the antecedent part and the consequent part.

It is a distance-based connectionist clustering algorithm. In any cluster, the maximum distance between an example point and the cluster center is less than a threshold value. This clustering algorithm sets clustering parameters and affects the number of

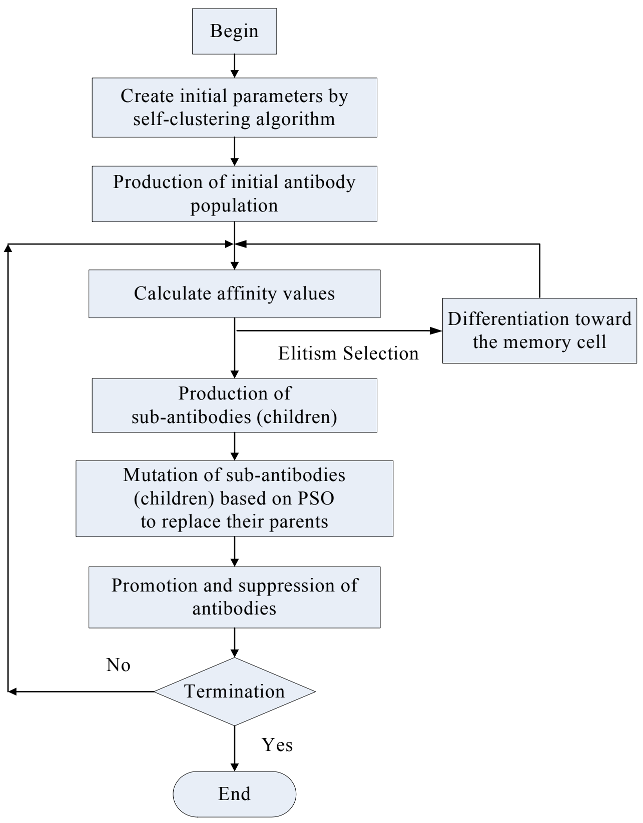

Begin
Create initial parameters by self-clustering algorithm
Production of initial antibody population
Calculate affinity values
Differentiation toward the memory cell
Elitism Selection
Production of sub-antibodies (children)
Mutation of sub-antibodies (children) based on PSO to replace their parents
Promotion and suppression of antibodies
No
Termination
Yes
End

Fig. 2. Flowchart of the proposed IPSO.

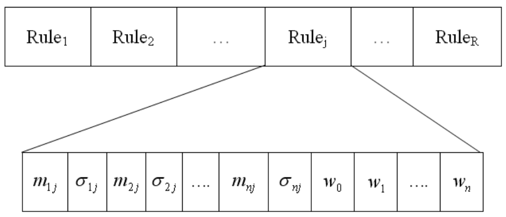


Fig. 3. Coding a TSK-type neuro-fuzzy network into an antibody in the IPSO method.

clusters to be estimated. In the clustering process, the data examples come from a data stream. The clustering process starts with an empty set of clusters. The clusters will be updated and changed depending on the position of the current example in the input space.

(C) Produce initial population

In the immune system, the antibodies are produced in order to cope with the antigens. In other words, the antigens are recognized by a few of high affinity antibodies (*i.e.* the antigens are optimal solutions). The first initial antibody utilizing a real variable string is generated by self-clustering algorithm, and the other antibodies of population are generated to add random value by first antibody.

(D) Calculate affinity values

For the large number of various antigens, the immune system has to recognize them for their posterior influence. In biological immune system, *affinity* refers to the binding strength between a single antigenic determinants and an individual antibody-combining site. The process of recognizing antigens is to search for antibodies with the maximum affinity with antigens. The affinity value is a performance measure of an antibody and its value is obtained according to the error function. In this paper, the affinity value is designed according to the follow formulation:

$$\mathit{Affinity\ value} = \frac{1}{\sqrt{\frac{1}{N_t}\sum_{k=1}^{N_t}(y_k - y_k^d)^2}} \tag{2}$$

where y_k represents the kth model output, y_k^d represents the desired output, and N_t represents the number of the training data. In the problems, the higher affinity refers to the better antibody.

(E) Production of sub-antibodies

In this step, we will generate several neighborhoods to maintain solution variation. This strategy can prevent the search process from becoming premature. We can generate several clones for each antibody on feasible space by Eqs. (3) and (4). Each antibody regarded as parent while the clones regarded as children (sub-antibodies). In other words, children regarded as several neighborhoods of near parent.

$$\text{mean and deviation}: \mathit{clons}[\mathit{children}_{i\_c}] = \mathit{antibody}[\mathit{parent}_i] + \alpha \tag{3}$$

$$\text{weight}: \mathit{clons}[\mathit{children}_{i\_c}] = \mathit{antibody}[\mathit{parent}_i] + \beta \tag{4}$$

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

(F) Mutation of sub-antibodies based on particle swarm optimization

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

PSO is not only a recently invented high-performance optimizer that is very easy to understand and implement, but it also requires little computational bookkeeping and, generally, only a few lines of code. Each particle has a velocity vector $\vec{v}_i$ and a position vector $\vec{x}_i$ to represent a possible solution. The velocity for each particle is updated by

$$\vec{v}_i(k+1) = \omega * \vec{v}_i(k) + \phi_1 * rand() * (Lbest - \vec{x}_i(k)) + \phi_2 * rand() * (Gbest - \vec{x}_i(k)) \quad (5)$$

where ω is the coefficient of inertia, ϕ_1 is the cognitive study, and ϕ_2 is the group study. The *rand*() is uniformly distributed random numbers in [0, 1]. The term $\vec{v}_i$ is limited to the range $\pm\,\vec{v}_{\max}$. If the velocity violates this limit, it will be set at its proper limit. Changing velocity enables every particle to search around its individual best position and global best position. Based on the updated velocities, each particle changes its position according to the following:

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1). \quad (6)$$

When every particle is updated, the affinity value of each particle is calculated again. If the affinity value of the new particle is higher than those of local best, then the local best will be replaced with the new particle. Moreover, in the mutation step, each antibody in the population must be mutated only one time by PSO in each generation. The mutation step flowchart is presented in Fig. 4.

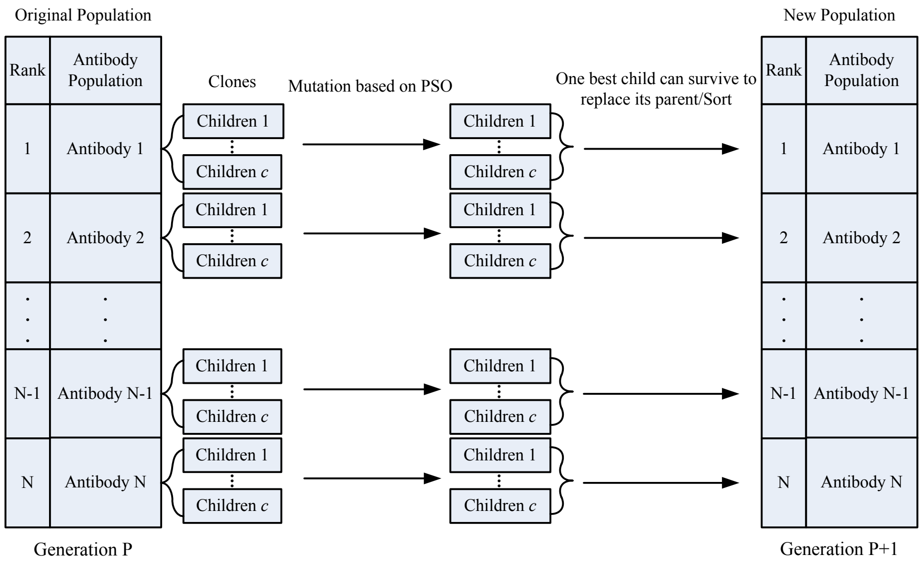


Fig. 4. The flowchart of the mutation step.

(G) Promotion and suppression of antibodies

In order to affect antigens and keep diversity to a certain degree, we use information entropy theory to measure the diversity of antibodies. If the affinity between two antibodies is greater than the suppression threshold Th_{aff}, these two antibodies are similar,

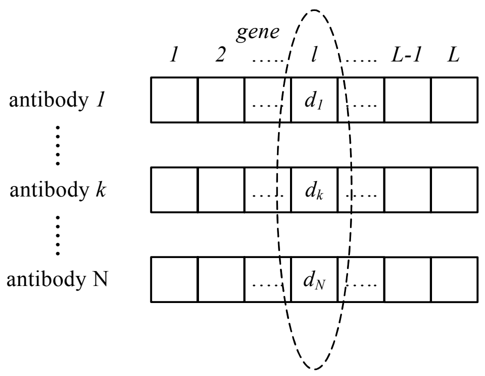


Fig. 5. The coding of antibody population.

and the antibody of lower affinity value is reduced a small amount of value λ. Fig. 5 shows the immune algorithm composed of N antibodies having L genes.

From information entropy theory, we get

$$IE_l(N) = \sum_{i=1}^{N} -P_{il} \log P_{il} \tag{7}$$

where P_{il} is the probability that the ith allele comes out at the lth gene. The diversity of the genes is calculated using Eq. (7). The average entropy value $IE(N)$ of diversity can be also computed as follows:

$$IE(N) = \frac{1}{L} \sum_{l=1}^{L} IE_l(N) \tag{8}$$

where L is the size of the gene in a antibody. Eq. (8) yields the diversity of the antibody pool in terms of the entropy. There are two kinds of affinities in IPSO. One explains the relationship between an antibody and an antigen using Eq. (2). The other accounts for the degree of association between the jth antibody and the kth antibody and measures how similar these two antibodies are. It can be calculated by using

$$Affinity\_Ab_{jk} = \frac{1}{1 + IE(2)} \tag{9}$$

where $Affinity\_Ab_{jk}$ is the affinity between two antibodies j and k, and $IE(2)$ is the entropy of only the antibodies j and k. This affinity is constrained from zero to one. When $IE(2)$ is zero, the genes of the ith antibody and the kth antibody are the same.

(H) Elitism selection

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

4. ILLUSTRATIVE EXAMPLES

In order to demonstrate the performance of the proposed IPSO approach, several experiments were conducted on the identification and prediction problems. The initial parameters before training are given in Table 1. All the programs were developed using Visual C++ 6.0 on a Pentium IV 3.2GHz desktop computer.

Table 1. The initial parameters before training.

Parameters	Value
Antibody Population Size	50
Coding Type	Real Number
Clones number *c*	5

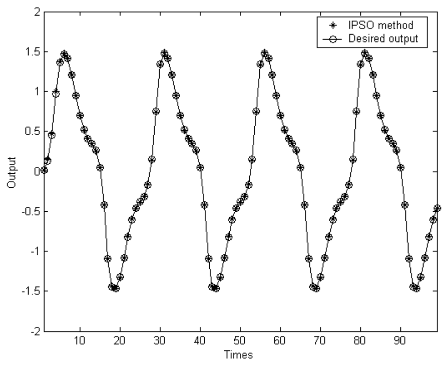
Example 1: Identification of Nonlinear Dynamic System.

The first example used for identification is described by the difference equation

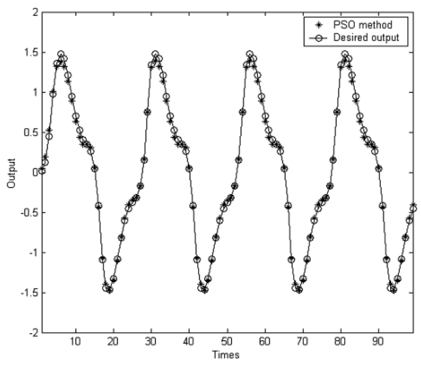
$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k). \tag{10}$$

The output of this equation depends nonlinearly on both its past value and the input, but the effects of the input and output values are additive. The training input patterns are randomly generated in the interval [– 2, 2] for training data, λ is 0.01, α is in the interval [– 0.009, 0.001], and β is in the interval [– 0.001, 0.049]. Evolution progressed for 1000 generations. After using the self-clustering algorithm (SCA) [16] for performing input space partition, we obtain seven clusters (fuzzy rules). To show the effectiveness of the proposed IPSO method, an immune algorithm (IA) [9] and the particle swarm optimization (PSO) [13] are applied to the same problem. Figs. 6 (a)-(c) show the outputs of the three methods for the input $u(k) = \sin(2\pi k/25)$. According to these results, the identification ability of the IPSO method was better than those of the IA and PSO methods.

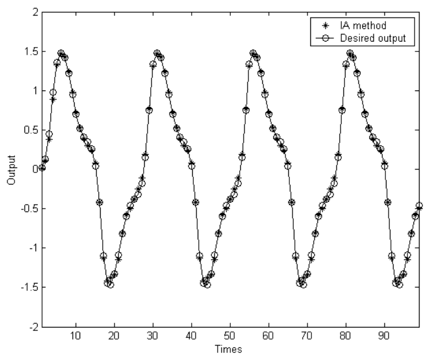
Fig. 7 shows the best situation learning curves of the three methods. In this figure, we find that the proposed IPSO method obtains a lower RMS (root mean square) error than the others. We also compare the performance of our model with some existing models [17, 18]. The performance indices considered include rms error, number of parameters, and training steps. The comparison results are tabulated in Table 2. Lin [18] proposed a hybrid system that incorporates a priori knowledge into the selection of initial parameter values. The fuzzy system contains five rules. After the parameter learning, the RMS error of hybrid system approximates 0.04. The results are shown in the fourth column of Table 2. Narendra and Paethasarathy [17] using neural networks with two hidden layers, one with twenty units and other with ten units, and carried out the identification process for 100000 training steps. After the parameter learning, the RMS error of neural



(a) The proposed IPSO method.



(b) The PSO method [13].



(c) The IA method [9].

Fig. 6. Results of the desired output.

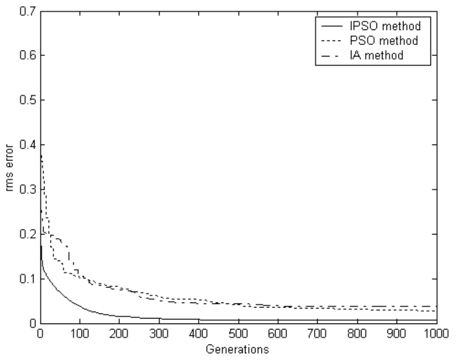


Fig. 7. The learning curves of the proposed IPSO method, the PSO [13] and the IA [9].

Table 2. Performance comparison of various existing models.

	IPSO	PSO [13]	IA [9]	Hybrid System [18]	Neural Networks [17]
RMS error	**0.017**	0.097	0.1	0.04	0.07
Number of parameters	**63**	63	63	80	270
Training steps	**1000**	1000	1000	30000	100000

	m_{11}	σ_{11}	m_{21}	σ_{21}	w_{01}	w_{11}	w_{21}	m_{12}	σ_{12}	m_{22}	…	m_{27}	σ_{27}	w_{07}	w_{17}	w_{27}	
Antibody 1	0.51	0.35	-0.87	0.12	-0.03	-0.10	-0.21	-0.65	-0.92	0.54	…	0.18	-0.33	0.51	0.15	0.32	affinity = 5.22
Antibody 2	0.81	1.05	-1.22	0.07	-0.29	0.36	-0.70	-0.82	-1.26	0.38	…	0.03	-0.32	0.51	0.42	0.32	affinity = 5.20
Antibody 3	0.59	-0.14	-1.21	-0.02	0.76	0.16	-0.38	-0.53	-1.11	0.59	…	0.13	-0.22	0.44	0.42	0.42	affinity = 5.19
Antibody 4	0.90	0.35	-1.22	0.00	0.60	0.05	-0.26	-0.65	-1.16	0.51	…	0.11	-0.32	0.45	0.42	0.32	affinity = 5.06
Antibody 5	0.90	0.35	-1.22	0.00	0.60	0.05	-0.25	-0.65	-1.16	0.51	…	0.12	-0.32	0.47	0.42	0.34	affinity = 5.06
Antibody 6	0.21	0.18	-1.09	-0.01	0.88	-0.09	-0.64	-0.66	-1.04	0.68	…	0.07	-0.30	0.51	0.44	0.31	affinity = 5.02
Antibody 7	0.95	0.40	-1.25	0.03	0.60	0.07	-0.20	-0.59	-1.19	0.52	…	0.09	-0.33	0.47	0.41	0.31	affinity = 4.99
Antibody 8	0.24	0.33	-1.14	-0.01	-0.04	-0.04	-0.07	-0.68	-0.93	0.80	…	0.08	-0.20	0.16	0.47	0.29	affinity = 4.95
Antibody 9	0.87	0.05	-1.18	0.25	0.62	0.06	-0.28	-0.23	-1.13	0.55	…	0.18	-0.33	0.35	0.42	0.28	affinity = 4.95
Antibody 10	-0.20	-0.62	-1.21	-0.03	0.79	0.26	-0.49	-0.53	-1.07	0.59	…	0.16	-0.18	0.15	0.40	0.41	affinity = 4.91
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Antibody 46	0.09	0.44	-0.97	-0.09	0.66	-0.12	-0.66	-0.63	-0.71	0.67	…	0.36	-0.27	0.04	0.42	0.36	affinity = 4.11
Antibody 47	0.33	0.95	-1.30	0.09	0.53	-0.07	-0.33	-0.66	-1.03	0.82	…	0.23	-0.38	0.10	0.49	0.37	affinity = 4.10
Antibody 48	0.82	0.29	-1.18	-0.12	0.58	0.13	-0.35	-0.66	-1.11	0.65	…	0.12	-0.23	0.36	0.30	0.28	affinity = 3.98
Antibody 49	0.68	0.10	-1.23	0.12	0.50	0.05	-0.41	-0.62	-1.15	0.49	…	0.14	-0.36	0.51	0.41	0.34	affinity = 3.98
Antibody 50	0.13	-0.17	-1.17	0.07	0.05	0.27	-0.27	-0.60	-0.94	0.81	…	0.08	-0.18	0.14	0.39	0.36	affinity = 3.94

(a) The initial populations using SCA learning method.

	m_{11}	σ_{11}	m_{21}	σ_{21}	w_{01}	w_{11}	w_{21}	m_{12}	σ_{12}	m_{22}	…	m_{27}	σ_{27}	w_{07}	w_{17}	w_{27}	
Memory cell 1	0.51	0.35	-0.87	0.12	-0.03	-0.10	-0.21	-0.65	-0.92	0.54	…	0.18	-0.33	0.51	0.15	0.32	affinity = 5.22
Memory cell 2	0.59	-0.14	-1.21	-0.02	0.76	0.16	-0.38	-0.53	-1.11	0.59	…	0.13	-0.22	0.44	0.42	0.42	affinity = 5.21
Memory cell 3	0.81	1.05	-1.22	0.07	-0.29	0.36	-0.70	-0.82	-1.26	0.38	…	0.03	-0.32	0.51	0.42	0.32	affinity = 5.21
Memory cell 4	0.90	0.35	-1.22	0.00	0.60	0.05	-0.25	-0.65	-1.16	0.51	…	0.12	-0.32	0.47	0.42	0.34	affinity = 5.09
Memory cell 5	0.21	0.18	-1.09	-0.01	0.88	-0.09	-0.64	-0.66	-1.04	0.68	…	0.07	-0.30	0.51	0.44	0.31	affinity = 5.06
Memory cell 6	0.90	0.35	-1.22	0.00	0.60	0.05	-0.26	-0.65	-1.16	0.51	…	0.11	-0.32	0.45	0.42	0.32	affinity = 5.06
Memory cell 7	0.95	0.40	-1.25	0.03	0.60	0.07	-0.20	-0.59	-1.19	0.52	…	0.09	-0.33	0.47	0.41	0.31	affinity = 5.04
Memory cell 8	0.87	0.05	-1.18	0.25	0.62	0.06	-0.28	-0.23	-1.13	0.55	…	0.18	-0.33	0.35	0.42	0.28	affinity = 5.01
Memory cell 9	0.43	0.01	-1.22	-0.07	0.63	0.01	-0.50	-0.65	-1.15	0.50	…	0.10	-0.30	0.34	0.41	0.33	affinity = 4.97
Memory cell 10	0.43	-0.06	-1.21	-0.07	0.50	0.00	-0.42	-0.64	-1.14	0.47	…	0.13	-0.28	0.21	0.48	0.32	affinity = 4.96
Memory cell 11	0.24	0.33	-1.14	-0.01	-0.04	-0.04	-0.07	-0.68	-0.93	0.80	…	0.08	-0.20	0.16	0.47	0.29	affinity = 4.95
Memory cell 12	0.24	0.33	-1.14	-0.01	-0.04	-0.04	-0.07	-0.68	-0.93	0.80	…	0.08	-0.20	0.16	0.47	0.29	affinity = 4.95
Memory cell 13	-0.20	-0.62	-1.21	-0.03	0.79	0.26	-0.49	-0.53	-1.07	0.59	…	0.16	-0.18	0.15	0.40	0.41	affinity = 4.91
Memory cell 14	0.87	-0.18	-1.22	0.02	0.51	0.06	-0.24	-0.44	-1.11	0.37	…	0.08	-0.34	0.36	0.51	0.33	affinity = 4.90
Memory cell 15	0.80	0.18	-0.62	0.25	0.13	0.07	-0.21	-0.52	-0.92	0.74	…	-0.02	-0.32	0.26	0.38	0.33	affinity = 4.88

(b) The antibodies into long-lived B memory cells by elitism selection.

	m_{11}	σ_{11}	m_{21}	σ_{21}	w_{01}	w_{11}	w_{21}	m_{12}	σ_{12}	m_{22}	…	m_{27}	σ_{27}	w_{07}	w_{17}	w_{27}	
Antibody 1	**0.51**	**0.35**	**-0.87**	**0.12**	**-0.03**	**-0.10**	**-0.21**	**-0.65**	**-0.92**	**0.54**	…	**0.18**	**-0.33**	**0.51**	**0.15**	**0.32**	**affinity = 5.22**
Children 1	0.51	0.35	-0.87	0.12	-0.04	-0.10	-0.21	-0.65	-0.92	0.54	…	0.18	-0.33	0.51	0.16	0.33	affinity = 5.23
Children 2	0.51	0.35	-0.87	0.12	-0.03	-0.11	-0.21	-0.65	-0.92	0.54	…	0.17	-0.33	0.51	0.15	0.34	affinity = 5.24
Children 3	0.51	0.35	-0.87	0.12	-0.03	-0.10	-0.21	-0.65	-0.92	0.54	…	0.17	-0.33	0.52	0.15	0.30	affinity = 5.28
Children 4	0.51	0.35	-0.87	0.12	-0.03	-0.10	-0.21	-0.65	-0.92	0.54	…	0.18	-0.33	0.50	0.15	0.34	affinity = 5.25
Children 5	0.51	0.35	-0.87	0.12	-0.04	-0.10	-0.21	-0.65	-0.92	0.54	…	0.19	-0.32	0.48	0.15	0.33	affinity = 5.22
Antibody 2	**0.81**	**1.05**	**-1.22**	**0.07**	**-0.29**	**0.36**	**-0.70**	**-0.82**	**-1.26**	**0.38**	…	**0.03**	**-0.32**	**0.51**	**0.42**	**0.32**	**affinity = 5.21**
Children 1	0.55	-0.02	-0.99	0.04	0.07	0.14	-0.23	-0.58	-1.05	0.58	…	0.14	-0.31	0.44	0.34	0.36	affinity = 4.69
Children 2	0.56	0.08	-0.96	0.10	0.52	-0.03	-0.25	-0.64	-1.07	0.57	…	0.17	-0.21	0.46	0.25	0.44	affinity = 5.11
Children 3	0.52	0.27	-0.96	0.04	0.13	0.02	-0.31	-0.57	-1.09	0.57	…	0.15	-0.33	0.46	0.28	0.38	affinity = 5.16
Children 4	0.56	0.21	-0.87	-0.02	0.27	-0.06	-0.24	-0.61	-0.96	0.54	…	0.15	-0.32	0.45	0.27	0.33	affinity = 4.70
Children 5	0.54	0.24	-1.16	0.06	0.14	-0.05	-0.37	-0.60	-0.93	0.56	…	0.16	-0.27	0.49	0.15	0.35	affinity = 4.96
Antibody 3	**0.59**	**-0.14**	**-1.21**	**-0.02**	**0.76**	**0.16**	**-0.38**	**-0.53**	**-1.11**	**0.59**	…	**0.13**	**-0.22**	**0.44**	**0.42**	**0.42**	**affinity = 5.21**
Children 1	0.78	0.50	-1.15	0.07	-0.05	0.35	-0.40	-0.81	-1.07	0.50	…	0.08	-0.32	0.52	0.17	0.32	affinity = 5.01
Children 2	0.80	0.86	-1.15	0.12	-0.28	0.24	-0.68	-0.68	-1.06	0.44	…	0.07	-0.33	0.51	0.18	0.32	affinity = 4.39
Children 3	0.61	0.62	-1.15	0.10	-0.24	0.33	-0.42	-0.73	-1.21	0.52	…	0.15	-0.32	0.51	0.44	0.31	affinity = 5.04
Children 4	0.74	0.86	-0.98	0.11	-0.21	0.02	-0.56	-0.68	-1.23	0.39	…	0.06	-0.34	0.51	0.41	0.31	affinity = 3.63
Children 5	0.72	0.53	-1.04	0.12	-0.24	0.13	-0.67	-0.72	-1.23	0.53	…	0.10	-0.32	0.51	0.27	0.32	affinity = 4.83
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Antibody 50	**0.13**	**-0.17**	**-1.17**	**0.07**	**0.05**	**0.27**	**-0.27**	**-0.60**	**-0.94**	**0.81**	…	**0.08**	**-0.18**	**0.14**	**0.39**	**0.36**	**affinity = 3.94**
Children 1	0.33	0.90	-1.25	0.11	0.43	-0.09	-0.30	-0.65	-1.03	0.78	…	0.23	-0.34	0.13	0.24	0.34	affinity = 4.41
Children 2	0.49	0.76	-0.93	0.11	0.17	-0.08	-0.33	-0.66	-0.98	0.67	…	0.23	-0.37	0.20	0.21	0.36	affinity = 3.15
Children 3	0.37	0.74	-1.08	0.10	0.37	-0.08	-0.29	-0.66	-1.03	0.80	…	0.20	-0.34	0.41	0.38	0.38	affinity = 4.41
Children 4	0.48	0.60	-0.97	0.11	0.03	-0.09	-0.22	-0.66	-0.96	0.77	…	0.21	-0.36	0.45	0.17	0.35	affinity = 4.42
Children 5	0.38	0.72	-1.06	0.11	0.18	-0.08	-0.24	-0.66	-0.93	0.56	…	0.18	-0.35	0.33	0.25	0.34	affinity = 4.25

(c) The generated sub-antibodies (children).

Fig. 8. The evolutionary process of population for the proposed IPSO.

	m_{11}	σ_{11}	m_{21}	σ_{21}	w_{01}	w_{11}	w_{21}	m_{12}	σ_{12}	m_{22}	…	m_{27}	σ_{27}	w_{07}	w_{17}	w_{27}	
Antibody 1	0.51	0.35	-0.87	0.12	-0.03	-0.10	-0.21	-0.65	-0.92	0.54	…	0.17	-0.33	0.52	0.15	0.30	affinity = 5.28
Antibody 2	0.59	-0.14	-1.21	-0.02	0.76	0.16	-0.38	-0.53	-1.11	0.59	…	0.13	-0.22	0.44	0.42	0.42	affinity = 5.21
Antibody 3	0.81	1.05	-1.22	0.07	-0.29	0.36	-0.70	-0.82	-1.26	0.38	…	0.03	-0.32	0.51	0.42	0.32	affinity = 5.21
Antibody 4	0.90	0.35	-1.22	0.00	0.60	0.05	-0.25	-0.65	-1.16	0.51	…	0.12	-0.32	0.47	0.42	0.34	affinity = 5.09
Antibody 5	0.30	0.25	-1.07	0.10	0.62	-0.10	-0.36	-0.65	-0.98	0.55	…	0.12	-0.32	0.51	0.34	0.32	affinity = 5.15
Antibody 6	0.90	0.35	-1.22	0.00	0.60	0.05	-0.26	-0.65	-1.16	0.51	…	0.11	-0.32	0.45	0.42	0.32	affinity = 5.06
Antibody 7	0.74	0.39	-1.12	0.10	0.21	0.00	-0.20	-0.64	-1.12	0.52	…	0.13	-0.32	0.51	0.36	0.33	affinity = 5.41
Antibody 8	0.87	0.05	-1.18	0.25	0.62	0.06	-0.28	-0.23	-1.13	0.55	…	0.18	-0.33	0.35	0.42	0.28	affinity = 5.01
Antibody 9	0.43	0.01	-1.22	-0.07	0.63	0.01	-0.50	-0.65	-1.15	0.50	…	0.10	-0.30	0.34	0.41	0.33	affinity = 4.97
Antibody 10	0.46	0.07	-1.15	-0.01	0.17	-0.09	-0.25	-0.65	-1.05	0.50	…	0.16	-0.31	0.38	0.42	0.31	affinity = 5.02
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Antibody 46	0.48	0.26	-0.98	0.02	0.04	-0.03	-0.35	-0.67	-1.02	0.54	…	0.16	-0.27	0.31	0.21	0.37	affinity = 5.27
Antibody 47	0.54	0.43	-1.21	0.11	-0.05	0.00	-0.15	-0.43	-1.19	0.73	…	0.14	-0.34	0.50	0.38	0.15	affinity = 4.85
Antibody 48	0.39	0.23	-0.74	0.02	0.15	-0.14	-0.43	-0.67	-0.94	0.70	…	0.13	-0.29	0.49	0.21	0.32	affinity = 4.71
Antibody 49	0.26	0.44	-0.97	-0.02	0.40	-0.12	-0.49	-0.64	-0.74	0.65	…	0.25	-0.32	0.40	0.41	0.37	affinity = 4.96

(d) The populations through the mutation step based on PSO.

	m_{11}	σ_{11}	m_{21}	σ_{21}	w_{01}	w_{11}	w_{21}	m_{12}	σ_{12}	m_{22}	…	m_{27}	σ_{27}	w_{07}	w_{17}	w_{27}	
Antibody 1	0.26	0.29	-0.96	0.08	-0.02	-0.08	-0.23	-0.65	-0.93	0.58	…	0.14	-0.30	0.24	0.21	0.36	affinity = 5.61
Antibody 2	0.56	0.12	-1.03	0.08	0.08	-0.02	-0.22	-0.65	-1.04	0.53	…	0.14	-0.32	0.49	0.37	0.36	affinity = 5.59
Antibody 3	0.34	0.76	-1.19	0.10	-0.03	-0.03	-0.15	-0.64	-0.96	0.60	…	0.14	-0.32	0.41	0.39	0.30	affinity = 5.53
Antibody 4	0.51	0.17	-1.07	0.12	0.18	0.01	-0.27	-0.63	-1.10	0.53	…	0.14	-0.36	0.49	0.32	0.36	affinity = 5.46
Antibody 5	0.49	0.30	-0.92	0.07	0.01	-0.10	-0.34	-0.66	-0.94	0.52	…	0.06	-0.34	0.33	0.29	0.31	affinity = 5.46
Antibody 6	0.70	-0.07	-0.91	0.12	0.10	-0.06	-0.21	-0.56	-1.02	0.47	…	0.17	-0.35	0.47	0.37	0.31	affinity = 5.42
Antibody 7	0.74	0.39	-1.12	0.10	0.21	0.00	-0.20	-0.64	-1.12	0.52	…	0.13	-0.32	0.51	0.36	0.33	affinity = 5.34
Antibody 8	0.82	0.33	-1.21	-0.02	0.14	0.02	-0.21	-0.66	-1.11	0.53	…	0.12	-0.30	0.44	0.23	0.31	affinity = 5.33
Antibody 9	0.55	0.26	-1.02	0.07	-0.12	-0.03	-0.09	-0.65	-0.90	0.63	…	0.21	-0.32	0.44	0.28	0.34	affinity = 5.26
Antibody 10	0.51	0.35	-0.87	0.12	-0.03	-0.10	-0.21	-0.65	-0.92	0.54	…	0.18	-0.33	0.51	0.15	0.32	affinity = 5.22
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Antibody 46	0.26	0.44	-0.97	-0.02	0.40	-0.12	-0.49	-0.64	-0.74	0.65	…	0.25	-0.32	0.40	0.41	0.37	affinity = 4.70
Antibody 47	0.61	0.35	-0.87	0.08	-0.04	-0.09	-0.15	-0.64	-0.91	0.60	…	0.17	-0.33	0.51	0.14	0.36	affinity = 4.69
Antibody 48	0.26	0.04	-0.99	0.09	0.04	0.27	-0.21	-0.62	-0.94	0.63	…	0.10	-0.30	0.30	0.22	0.32	affinity = 4.69
Antibody 49	0.51	0.34	-1.06	0.09	0.09	0.30	-0.30	-0.69	-1.01	0.44	…	0.16	-0.30	0.42	0.15	0.32	affinity = 4.67

(e) The evolved populations through promotion and suppression.

Fig. 8. (Cont'd) The evolutionary process of population for the proposed IPSO.

networks approximates 0.07. The results are shown in the fifth column of Table 2. In Table 2, we obtain a smaller RMS error (0.017) and require fewer adjustable parameters than some existing models. Finally, Fig. 8 shows the evolutionary process of populations using the proposed IPSO. The initial populations using SCA learning method are shown in Fig. 8 (a). Fig. 8 (b) presents the antibodies into long-lived B memory cells by elitism selection. The generated sub-antibodies using Eqs. (3) and (4) are shown in Fig. 8 (c). Through the mutation step based on PSO, the evolved population is shown in Fig. 8 (d). Fig. 8 (e) shows the new population through promotion and suppression.

Example 2: Forecast of the Sunspot Number.

The sunspot numbers exhibit nonlinear from 1700 to 2004, non-stationary, and non-Gaussian cycles that is difficult to predict [19]. The inputs x_i are defined as $x_1(t) = y_1^d(t - 1)$, $x_2(t) = y_1^d(t - 2)$, and, $x_3(t) = y_1^d(t - 3)$, where t represents the year and $y_1^d(t)$ is the sunspot numbers at the t year.

In this example, the first 180 years (from 1705 to 1884) of the sunspot numbers were used to train the proposed IPSO model while the remaining 121 years (from 1885 to 2004) of the sunspot numbers were the used to test. We set the threshold value in the self-clustering algorithm (SCA) [16] to 80. After self-clustering algorithm process, the nine clusters (fuzzy rules) were obtained, the λ is 0.008, α is in the interval [− 3, 3], and β is in the interval [− 0.2, 0.2]. After 500 generations of training, the final average RMS error of the prediction output approximates 11.36. We also compared the performance with the IA [9] and PSO [13] method. The evolution learning also was 500 generations. The best situation learning curves of the three methods are shown in Fig. 9 Table 3 tabulated the RMS error, the training error (governed by $\sum_{t=1705}^{1884} \frac{|y_1^d(t) - y_1(t)|}{180}$), and the forecasting error (governed by $\sum_{t=1885}^{2004} \frac{|y_1^d(t) - y_1(t)|}{121}$). As shown in Table 3, the proposed IPSO method performs a better performance than other models.

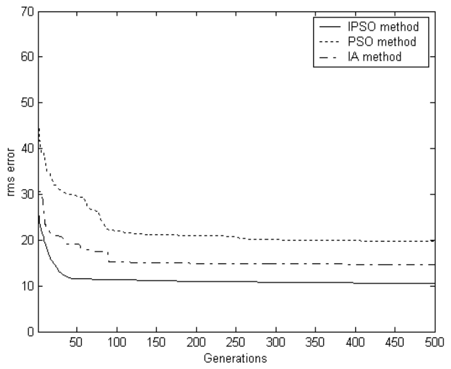


Fig. 9. The learning curves of the proposed method, the IA [9], and the PSO [13].

Table 3. Performance comparison of various existing models.

Methods	Training case	RMS error	Training error	Forecast error
IPSO	500	**11.36**	**9.47**	**14.51**
IA [9]	500	17.10	16.70	18.78
PSO [13]	500	22.49	16.73	19.17

5. CONCLUSION

In this paper, the efficient immune-based particle swarm optimization (IPSO) is proposed to improve the searching ability and the converge speed. We proposed the IPSO for use in TSK-type neuro-fuzzy networks. The advantages of the proposed IPSO method are summarized as follows: (1) We employed the advantages of PSO to improve the mutation mechanism; (2) The complicated problems can be better solved than IA and PSO; (3) There is more of a likelihood to get a global optimum compared to heuristic methods; (4) The experimental results have shown that our method obtains better results than other existing methods in accuracy rate and convergence speed.

In this study, the size of each coding of antibodies is fixed for overall population. If the linguistic terms are the same or the similar in different rules, then the similar parameters will be generated constantly. In the future work, we will make the coding of antibodies more flexible according to the fuzzy rules similarity measure [20].

REFERENCES

1. C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*, Prentice-Hall, NJ, 1996.
2. L. Chen, D. H. Cooley, and J. Zhang, "Possibility-based fuzzy neural networks and their application to image processing," *IEEE Transactions on Systems, Man and Cybernetics − Part B: Cybernetics*, Vol. 29, 1999, pp. 119-126.
3. Q. Jishuang, W. Chao, and W. Zhengzhi, "Structure-context based fuzzy neural network approach for automatic target detection," in *Proceedings of IEEE International Geoscience and Remote Sensing Symposium*, Vol. 2, 2003, pp. 767-769.
4. B. Kusumoputo, P. Irwanto, and W. Jatmiko, "Optimization of fuzzy-neural structure through genetic algorithm and its application in artificial odor recognitions," in *Proceedings of Asia-Pacific Conference on Circuits and Systems*, Vol. 2, 2002, pp. 47-51.
5. A. Kalinli and N. Karabogab, "Artificial immune algorithm for IIR filter design," *Engineering Applications of Artificial Intelligence*, Vol. 18, 2005, pp. 919-929.
6. G. C. Liao and T. P. Tsao, "Application embedded chaos search immune genetic algorithm for short-term unit commitment," *Electric Power Systems Research*, Vol. 71, 2004, pp. 135-144.
7. X. Wen and A. Song, "An immune evolutionary algorithm for sphericity error evaluation," *International Journal of Machine Tools & Manufacture*, Vol. 44, 2004, pp. 1077-0184.
8. Y. Zhou, D. L. Zheng, Z. L. Qiu, and G. Y. Dong, "The application of RBF networks based on artificial immune algorithm in the performance prediction of steel bars," in *Proceedings of the 3rd International Conference on Machine Learning and Cybernetics*, 2004, pp. 26-29.
9. J. S. Chun, M. K. Kim, and H. K. Jung, "Shape optimization of electromagnetic devices using immune algorithm," *IEEE Transactions on Magentics*, Vol. 33, 1997, pp. 1876-1879.
10. H. W. Ge and Y. C. Liang, "A hidden markov model and immune particle swarm optimization-based algorithm for multiple sequence alignment," in *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence*, 2005, pp. 756-765.
11. Q. Wang, C. Wang, and X. Z. Gao, "A hybrid optimization algorithm based on clonal selection principle and particle swarm intelligence," in *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications*, Vol. 2, 2006, pp. 975-9792.
12. Y. Lv, S. Li, S. Chen, Q. Jiang, and W. Guo, "Particle swarm optimization based on information diffusion and clonal selection," in *Proceedings of the 6th International Conference on Simulated Evolution and Learning*, 2006, pp. 521-528.
13. J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.
14. D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Transactions on Antennas and Propagation*, Vol. 52, 2004, pp. 771-779.
15. C. J. Lin and Y. J. Xu, "The design of TSK-type fuzzy controllers using a new hybrid learning approach," *International Journal of Adaptive Control and Processing*,

Vol. 20, 2006, pp. 1-25.
16. C. J. Lin and Y. J. Xu, "A self-constructing neural fuzzy network with dynamic from symbiotic evolution," *Intelligent Automation and Soft Computing*, Vol. 11, 2005, pp. 1-15.
17. K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, Vol. 1, 1990, pp. 4-27.
18. C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Transactions on Fuzzy Systems*, Vol. 6, 1998, pp. 12-31.
19. S. H. Ling, F. H. F. Leung, H. K. Lam, Y. S. Lee, and P. K. S. Tam, "A novel genetic-algorithm-based neural network for short-term load forecasting," *IEEE Transactions on Industrial Electronics*, Vol. 50, 2003, pp. 793-799.
20. C. J. Lin and W. H. Ho, "An asymmetry-similarity-measure-based neural fuzzy inference system," *Fuzzy Sets and Systems*, Vol. 152, 2005, pp. 535-551.

Cheng-Jian Lin (林正堅) received the B.S. degree in Electrical Engineering from Tatung University, Taiwan, R.O.C., in 1986 and the M.S. and Ph.D. degrees in Electrical and Control Engineering from the National Chiao Tung University, Taiwan, R.O.C., in 1991 and 1996. From April 1996 to July 1999, he was an Associate Professor in the Department of Electronic Engineering, Nan Kai College, Nantou, Taiwan, R.O.C. From August 1999 to January 2005, he was an Associate Professor in the Department of Computer Science and Information Engineering, Chaoyang University of Technology. From February 2005 to July 2007, he was a full Professor in the Department of Computer Science and Information Engineering, Chaoyang University of Technology. From August 2007 to July 2008, he was a full Professor in the Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung, Taiwan, R.O.C. Currently, he is a full Professor of Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan, R.O.C. He served as the chairman of Computer Science and Information Engineering Department, Chaoyang University of Technology from 2001 to 2005. He served as the library director of Poding Memorial Library, Chaoyang University of Technology from 2005 to 2007. Dr. Lin served as the Associate Editor of International Journal of Applied Science and Engineering from 2002 to 2005. His current research interests are soft computing, pattern recognition, intelligent control, image processing, bioinformatics, and FPGA design. He has published more than 150 papers in the referred journals and conference proceedings. Dr. Lin is a member of the Phi Tau Phi. He is also a member of the Chinese Fuzzy Systems Association (CFSA), the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence (TAAI), the IEICE (The Institute of Electronics, Information and Communication Engineers), and the IEEE Computational Intelligence Society. He is an executive committee member of the Taiwanese Association for Artificial Intelligence (TAAI).

Cheng-Hung Chen (陳政宏) was born in Kaohsiung, Taiwan, R.O.C. in 1979. He received the B.S. and M.S. degree in Computer Science and Information Engineering from the Chaoyang University of Technology, Taiwan, R.O.C., in 2002 and 2004. He is currently working toward the Ph.D. degree in Electrical and Control Engineering at National Chiao Tung University, Taiwan, R.O.C. His current research interests are neural networks, fuzzy systems, evolutionary algorithms, intelligent control, and pattern recognition.

Chi-Yung Lee (李繼永) received the B.S. degree in Electronic Engineering from the National Taiwan University of Science and Technology, Taiwan, R.O.C., in 1981 and the M.S. degrees in Industrial Education from the National Taiwan Normal University, Taiwan, R.O.C., in 1989. From August 1989 to July 2004, he was a lecturer in the Department of Electronic Engineering, Nan Kai Institute of Technology, Nantou, Taiwan, R.O.C. Since August 2004, he has been with the Department of Computer Science and Information Engineering, Nan Kai Institute of Technology. Currently, he is an Associate professor of Computer Science and Information Engineering Department, Nan Kai Institute of Technology, Nantou, Taiwan, R.O.C. His current research interests are neural networks, fuzzy systems, intelligence control, and FPGA design.