Adaptive Group Organization Cooperative Evolutionary Algorithm for TSK-type Neural Fuzzy Networks Design

Sheng-Fuu Lin* and Jyun-Wei Chang
Department of Electrical Engineering
National Chiao Tung University
Hsinchu, Taiwan

***Abstract*—This paper proposes a novel adaptive group organization cooperative evolutionary algorithm (AGOCEA) for TSK-type neural fuzzy networks design. The proposed AGOCEA uses group-based cooperative evolutionary algorithm and self-organizing technique to automatically design neural fuzzy networks. The group-based evolutionary divided populations to several groups and each group can evolve itself. In the proposed self-organizing technique, it can automatically determine the parameters of the neural fuzzy networks, and therefore some critical parameters have no need to be assigned in advance. The simulation results are shown the better performance of the proposed algorithm than the other learning algorithms.**

Keywords—TSK-type Neural Fuzzy Networks; Evolutionary Algorithm; Group based Symbiotic; Self Organization; System Identification

I. INTRODUCTION

In the field of artificial intelligence, neural fuzzy network [1-3] refers to combinations of neural networks and fuzzy logic. Because the advantages of neural fuzzy networks are powerful computation ability and human-like reasoning ability from neural networks and fuzzy systems, respectively, it has good performance for solving complex nonlinear problems. The neural fuzzy networks can perform the nonlinear mapping once the system parameters are trained based on a sequence of input and desired response pairs, and it does not require mathematical descriptions of system. Therefore, the determination of parameters is a critical issue for neural fuzzy networks.

The backpropagation (BP) [4, 5] is a common method and widely used for training neural fuzzy networks. It is well known that BP is an approximate steepest descent algorithm. The steepest descent algorithm is the simplest, and the minimization method. The advantage of steepest decent algorithm is very simple, requiring calculation only of the gradient, the disadvantage of steepest descent algorithm is that training time is generally longer than other algorithms; based on initial weight values, it is often to find the local optimal solution but not global optimal solution. Besides, BP training performance depends on the initial values of the system parameters, and for different network topologies one has to derive new mathematical expressions for each network layer.

Considering the aforementioned disadvantages one may be faced with suboptimal performance even for a suitable neural fuzzy network topology. Hence, the capability of training parameters and finding the global solution while optimizing the overall structure are important. The evolutionary methods using for training the fuzzy model has become a popular research field [6-20] because evolutionary methods simultaneously evaluate many points in the search space and are more likely to converge toward the global solution. The evolutionary fuzzy model is a learning process to generate a fuzzy model automatically by incorporating evolutionary learning procedures.

Recently, several improved evolutionary algorithms have been proposed [16-22].In [16], Bandyopadhyay et al. used the variable-length genetic algorithm (VGA) that allows the differentia of lengths of chromosomes in a population. Carse et. al. [17] used the genetic algorithm to evolve fuzzy rule based controllers. In [18], Chen proposed an efficient immune symbiotic evolution learning algorithm for compensatory neuro-fuzzy controller. In [19], Lin presented a novel self-constructing evolutionary algorithm to design a TSK-type fuzzy model. In [20], the group-based symbiotic evolution (GSE) was proposed to solve the issue of the traditional genetic algorithm that all the fuzzy rules were encoded into one chromosome. In [21], Lin proposed a hybrid evolutionary learning algorithm to combine the compact genetic algorithm and the modified variable-length genetic algorithm to perform structure/parameter learning to construct a network dynamically. Hsu [22] proposed a multi-groups cooperation-based symbiotic evolution (MGCSE) to train a TSK-type neural fuzzy network (TNFN). Their results showed that MGCSE can obtain better performance and convergence than symbiotic evolution. Although MGCSE being a good approach for training a TNFN, there is no systematic way to determine suitable groups for selecting chromosomes.

Although the above evolution learning algorithms [16-22] can improve the traditional genetic algorithms, these algorithms may encounter one or more of the following issues: 1) all fuzzy rules represent one chromosome; 2) the random group selection of fuzzy rules; 3) the numbers of fuzzy rules and group numbers have to be assigned in advance.

Recently, the data mining approach has been widespread used in several fields [23-30]. The data mining can be regarded as a new way of data analysis. One goal of data mining is to find association rules among frequent item sets in transactions. In [23], the authors proposed a mining method of ascertain

*Corresponding author: Sheng-Fuu Lin,
Tel.: +883-3-5712121 ext. 54365.

large item sets to find association rules in transactions. Han et al. [24] proposed the frequent pattern growth (FP-Growth) to mine frequent patterns without candidate generation. Wu et al. [30] proposed a data mining method based on the genetic algorithm that efficiently improve the Traditional genetic algorithm by using analysis support and confidence parameters.

In order to solve aforementioned problems, this paper proposes an adaptive group organization cooperative evolutionary algorithm (AGOCEA) for designing a TSK-type neural fuzzy network. The AGOCEA adopts the group symbiotic evolution (GSE). Each population in the GSE is divided to several groups and each group represents a set of the chromosome that belongs to one single fuzzy rule. To solve the problem of random group selection, a data mining based group selection method was used to select the better groups. The adaptive group organization was used to solve the some parameters have to be assigned in advance.

This paper is organized as follows. In the Section II, a brief description of TSK-type neural fuzzy network is introduced. The proposed AGOCEA is described in the Section III. In the Section IV, the simulation results are presented. The conclusions are summarized in the Section V.

II. The Concept of the TSK-type Neural Fuzzy Networks

A Takagi-Sugeno-Kang (TSK) type neural fuzzy network (TNFN) [1] employs different implication and aggregation methods from the standard Mamdani fuzzy model[3]. Instead of using fuzzy sets the conclusion part of a rule, is a linear combination of the crisp inputs. The fuzzy rule of TSK-type neural fuzzy network is shown as following equation:

IF x_1 is $A_{1j}(m_{1j}, \sigma_{1j})$ and x_2 is $A_{2j}(m_{2j}, \sigma_{2j})$ and … and x_n is $A_{nj}(m_{nj}, \sigma_{nj})$

THEN $y=w_{0j}+w_{1j}x_1+\ldots+w_{nj}x_n$. (1)

where n is the number of the input dimensions and j is the number of the fuzzy rules.

The structure of the TSK-type neural fuzzy network is shown in Fig. 1. It is a five-layer network structure. The functions of the nodes in each layer are described as follows:

Layer 1 (Input Node): No function is performed in this layer. The node only transmits input values to layer 2. That is

$$u_i^{(1)} = x_i . \quad (2)$$

Layer 2 (Membership Function Node): Nodes in this layer correspond to one linguistic label of the input variables in layer 1; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set is calculated in this layer. For an external input x_i, the following Gaussian membership function is used:

$$u_{ij}^{(2)} = \exp\left(-\frac{\left[u_i^{(1)} - m_{ij} \right]^2}{\sigma_{ij}^2} \right), \quad (3)$$

where m_{ij} and σ_{ij} are, respectively, the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i.

Layer 3 (Rule Node): The output of each node in this layer is determined by the fuzzy AND operation. Here, the product operation is utilized to determine the firing strength of each rule.

The function of each rule is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} = \prod_i \exp\left(-\frac{\left[u_i^{(1)} - m_{ij} \right]^2}{\sigma_{ij}^2} \right). \quad (4)$$

Layer 4 (Consequent Node): Nodes in this layer are called consequent nodes.

The input to a node in layer 4 is the output delivered from layer 3, and the other inputs are the input variables from layer 1 as depicted in Fig. 1.

$$u_j^{(4)} = u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i) , \quad (5)$$

where the summation is over all the inputs and where w_{ij} are the corresponding parameters of the consequent part.

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

$$y = u^{(5)} = \frac{\sum_{j=1}^{M_k} u_j^{(4)}}{\sum_{j=1}^{M_k} u_j^{(3)}} = \frac{\sum_{j=1}^{M_k} u_j^{(3)}(w_{0j} + \sum_{i=1}^{M_k} w_{ij} x_i)}{\sum_{j=1}^{M_k} u_j^{(3)}} , \quad (6)$$

where M_k is the number of fuzzy rule.

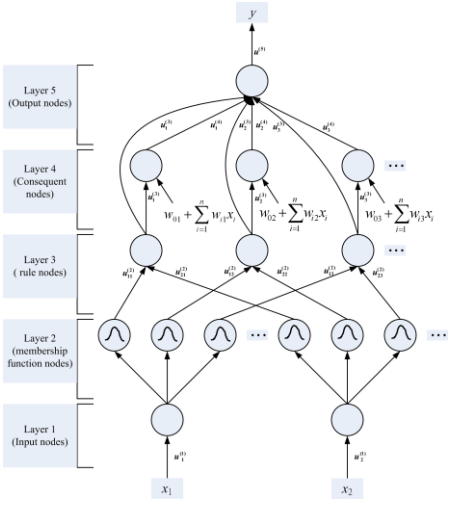

Fig. 1. Structure of TSK-type neural fuzzy network.

III. Adaptive Group Organization Cooperative Evolutionary Algorithm

The flowchart of the proposed adaptive group organization cooperative evolutionary algorithm (AGOCEA) is shown in Fig. 2. The structure of chromosomes to construct a TNFN is shown in Fig. 3. The coding structure of chromosomes is shown in Fig. 4.

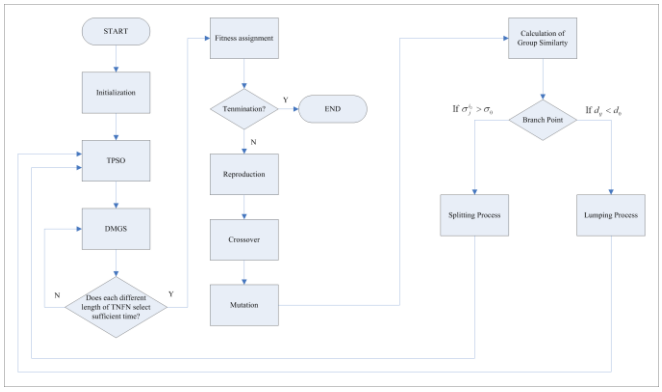

Fig. 2. The flowchart of the AGOCEA.

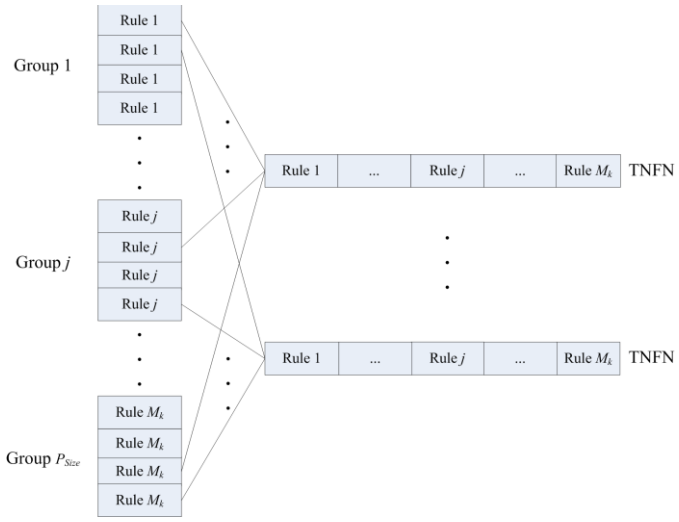

Fig. 3. The structure of the chromosome in the AGOCEA.

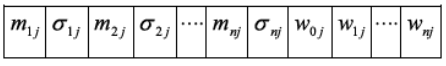

Fig. 4. The structure of the chromosome in the AGOCEA.

The learning process of the AGOCEA involves ten operators: initialization, two phase self organization, data mining based group selection, fitness assignment, reproduction, crossover, mutation, calculation of group similarity, splitting process, and lumping process. The whole learning process is described step-by-step as follows:

A. Initialization

Before the AGOCEA is designed, individuals forming several initial groups should be generated. The initial groups of the AGOCEA are generated randomly within a fixed range. The following formulations show how to generate the initial chromosomes in each group:

$$\text{Deviation: } Chr_{g,c}[p]=random[\sigma_{\min}, \sigma_{\max}] \quad (7)$$

where p=2, 4,⋯, 2n; g=1, 2,⋯, M_k; c=1, 2,⋯, N_C,

$$\text{Mean: } Chr_{g,c}[p]= random[m_{\min}, m_{\max}] \quad (8)$$

where p=1, 3,⋯, 2n-1;

$$\text{Weight: } Chr_{g,c}[p]= random[w_{\min}, w_{\max}] \quad (9)$$

where p=2n+1, 2n+2,⋯, 2n+(n+1),

where $Chr_{g,c}$ represents cth chromosome in gth group; M_k represents k rules that used to form a TNFN and N_C is the total number of chromosomes in each group; p represents the pth gene in a $Chr_{g,c}$; and $[\sigma_{\min},\sigma_{\max}]$, $[m_{\min},m_{\max}]$, and $[w_{\min},w_{\max}]$ represent the range that are predefined to generate the chromosomes.

B. Two phase self organization (TPSO)

After every group is initialized, the AGOCEA adopts pervious research which was TPSO [29, 31] to decide the suitable selection times of each number of rules (in this paper the number of rules lie between [$M_{\max}$, $M_{\min}$]); that is, it determines the selection times of M_k groups which form a TNFN with k rules.

After the TPSO, the selection times of the suitable number of rules in a TNFN will increase, and the selection times of the unsuitable number of rules will decrease. The details of the TPSO are listed as follows:

Step 0. Initialize the probability vectors:

$$V_{M_k}=0.5, \quad \text{where} \quad M_k=M_{\min},\ M_{\min+1},\cdots,\ M_{\max} \quad (11)$$

$$Accumulator = 0 \quad (12)$$

Step 1. Update the probability vectors according to the following equations:

$$\begin{cases} V_{M_k}=V_{M_k}+(Upt\_value_{M_k}*\lambda), & \text{if} \quad Avg \le fit_{M_k} \\ V_{M_k}=V_{M_k}-(Upt\_value_{M_k}*\lambda), & \text{otherwise} \end{cases} \quad (13)$$

$$Avg=\sum_{M_k=M_{\min}}^{M_{\max}} fit_{M_k}/(M_{\max}-M_{\min}+1) \quad (14)$$

$$Upt\_value_{M_k}=fit_{M_k}\Big/\sum_{M_k=M_{\min}}^{M_{\max}} fit_{M_k} \quad (15)$$

$$\text{if} \quad Fitness_{M_k} \ge (Best\_Fitness_{M_k}-ThresholdFitnessvalue) \quad (16)$$

$$\text{then} \quad fit_{M_k}=fit_{M_k}+Fitness_{M_k}$$

where V_{M_k} is the probability vector, λ is a predefine threshold value, *Avg* represents the average fitness value in the whole population, $Best\_Fitness_{M_k}$ represents the best fitness value with M_k rules, fit_{M_k} is the sum of fitness value of TNFN with M_k rules, $Fitness_{M_k}$ is the fitness value with M_k rules, *ThresholdFitnessvalue* is a predefined threshold value.

Step 2. Determine the selection times according to the probability vectors as follows:

$$R_{p_{M_k}} = (Selection\_Times) * (V_{M_k} / Total\_Velocity) \ , \tag{17}$$

$$Total\_Velocity = \sum_{M_k = V_{\min}}^{V_{\max}} V_{M_k} \ , \tag{18}$$

where *Selection_Times* represents total selection times in each generation, *Total_Velocy* is summation of the probability vectors V_{M_k}, $R_{p_{M_k}}$ is the selection times of M_k groups that use to select *k* chromosomes for constructing a TNFN.

Step 3. After step 2, the selection times of every numbers of rules in a TNFN are obtained. Then the $R_{p_{M_k}}$ times are used to select *k* chromosomes form M_k groups to construct a TNFN.

Step 4. In the proposed TPSO, for avoiding suitable M_k groups may fall in the local optima solution, the TPSO proposed two different actions to update the V_{M_k}. Decide the deferent action according to the following equations:

If $Accumulator \le TPSOTimes$ (19)
Then do Step1, Step2, and Step 3,

If $Best\_Fitness_g = Best\_Fitness$ (20)
Then $Accumulator = Accumulator + 1$,

If $Accumulator > TPSOTimes$ (21)
Then do Step 0 and $Accumulator = 0$,

where *TPSOTimes* is a predefined value; $Best\_Fitness_g$ represents the best fitness value of the best combination of chromosomes in *g*th generation; *Best_Fitness* represents the best fitness value of the best combination of chromosomes in current generations. Eqs. (19)-(21) represents that if the fitness is not changed for a sufficient number of generations, the suitable M_k groups may fall in the local optima solution.

C. Data mining based group selection (DMGS)

After the TPSO step, the selection times of each rule number in a TNFN is decided. The AGOCEA then performs selection step. The selection step in the AGOCEA can be divided by selection of groups and chromosomes. In the selection of groups, this paper uses the DMGS to improve the random selection. In the DMGS, the groups are selected according to the frequent patterns found by FP-Growth. In the proposed DMGS, the FP-Growth finds the frequent groups from a transaction (in this paper a transaction means a set of the M_k group indexes that perform well). After the frequent group indexes have been found, the DMGS selects the M_k groups indexes according to the frequent group indexes. To avoid the frequently-occurring groups from falling in the local optimal solution, the DMGS uses three actions to select M_k groups. The three actions defined in this paper are normal, search, and explore. The detail of the DMGS is shown as follows:

Step 0. The transactions are building as follow equation:

$$\begin{aligned} &\text{if} \quad Fitness_{M_k} \ge Best\_Fitness_{M_k} - ThresholdFitnessvalue \\ &\text{then} \quad Transaction_j[i] = TNFNRuleSet_{M_k}[i] \\ &\text{where} \quad i = 1,\ 2, \cdots,\ M_k \\ &\qquad\quad j = 1,\ 2, \cdots,\ TransactionNum \end{aligned} \tag{22}$$

where the $Fitness_{M_k}$ represents the fitness value of TNFN with M_k rules, *TransactionNum* is the total number of transactions $Transaction_j[i]$ represents the *i*th item in the *j*th transaction, $TNFNRuleSet_{M_k}[i]$ represents *i*th group index in M_k group indexes that are selected to form a TNFN with M_k rules. For example, as shown in Table I, the first transaction of the transaction set means the 3 rules TNFN that select from 1st group, 4th group, and 8th group has a well performance.

TABLE I. TRANSACTIONS IN A FP-GROWTH.

Transaction index	Group indexes
1	1, 4, 8
2	2, 4, 7, 10
⋮	⋮
TransactionNum	1, 3, 4, 6, 8, 9

Step 1. Normal action:

After building up the transactions, the DMGS selects group according to different action types. If the action type is normal action, the DMGS selects the group as following equation:

$$\begin{aligned} &\text{if} \quad Accumulator \le NormalTimes \\ &\text{then} \quad GroupIndex[i] = Random[1,\ P_{size}] \\ &\text{where} \quad i = 1,\ 2,\ \cdots,\ M_k; M_k = M_{\min},\ M_{\min+1}, \cdots,\ M_{\max}, \end{aligned} \tag{23}$$

where *Accumulator* is defined in Eq. (20); *GroupIndex*[*i*] represents selected *i*th group index of the M_k group indexes and P_{size} represents there are P_{size} groups in a population in the AGOCEA.

Step 2. Find the frequent groups:

If the action is searching or exploring action, the DMGS uses the FP-Growth [24] to find frequent group indexes in transactions. The frequent group indexes are found according to the predefined *Minimum_Support*. The *Minimum_Support* means the minimum fraction of transactions that contain an item set. The FP-Growth algorithm can be viewed as two parts: construction of the FP-tree and FP-growth. The sample transactions shown in Table II are taken as examples. *Minimum_Support* = 3 is considered in this example. Frequent group indexes generated by FP-growth shown in Table III are then thrown into the pool that's named *FrequentPool*.

TABLE II. SAMPLE TRANSACTIONS.

Transaction index	Group indexes
1	{b, c, e, f, g, h, p}
2	{a, b, c, f, i, m, o}
3	{c, f, i, m, o}
4	{b, c, e, s, p}
5	{a, b, c, d, f, m, o}

TABLE III. FREQUENT GROUP INDEXES GENERATED BY FP-GROWTH WITH *MINIMUM_SUPPORT* = 3.

Suffix group	Cond. group base	Cond. FP-tree	Frequent group indexes
B	c:4	c:4	cb:4
F	cb:3, c:1	c:4, cb:3	cf:4, bf:3, cbf:3
M	cbf:2, cf:1	cf:3	cm:3, fm:3, cfm:3
O	cbfm:2, cfm:1	cfm:3	co:3, fo:3, mo:3, cfo:3, cmo:3, fmo:3, cfmo:3

Step 3. Select the group indexes according to different actions:

After obtaining the frequent item sets, the DMGS selected group indexes according to different actions that describe as follows:

In the searching action, the group indexes are selected from the frequent item as follow equations:

$$\begin{aligned}&\text{if} \quad NormalTimes < Accumulator \le SearchingTimes\\ &\text{then} \quad GroupIndex[i] = w,\\ &\text{where}\\ &\quad w = Random[1,\ P_{size}] \ \text{ and } \ w \in FrequentItemSet[q];\\ &\quad FrequentItemSet[q] = Random[FrequentPool];\\ &\quad q = 1,\ 2, \cdots,\ FrequentPoolNum;\\ &\quad i = 1,\ 2, \cdots,\ M_k; M_k = M_{\min},\ M_{\min+1}, \cdots,\ M_{\max},\end{aligned} \tag{24}$$

where *SearchingTimes* is a predefined value that judge to perform searching action; *FrequentPool* represents the sets of frequent item set that obtain from FP-Growth; *FrequentPoolNum* presents the total number of sets in *FrequentPool* and *FrequentItemSet*[*i*] presents a frequent item set that select from *FrequentPool* randomly. In Eq. (24), if M_k greater than the size of *FrequentItemSet*[*i*], the remaining groups are selected by Eq. (23).

In the exploring action, the group indexes are selected according to the frequent item as follow equations:

$$\begin{aligned}&\text{if} \quad SearchingTimes < Accumulator \le ExploringTimes\\ &\text{then} \quad GroupIndex[i] = w,\\ &\text{where}\\ &\quad w = Random[1, P_{size}] \ \text{ and } \ w \notin FrequentItemSet[i];\\ &\quad FrequentItemSet[i] = Random[FrequentPool];\\ &\quad i = 1,\ 2, \cdots,\ M_k; M_k = M_{\min},\ M_{\min+1}, \cdots,\ M_{\max},\end{aligned} \tag{25}$$

where *ExploringTimes* is a predefined value that judge to perform exploring action.

Step 4. After selecting M_k group indexes, the *k* chromosomes are selected from M_k group as follows:

$$\begin{aligned}&ChromosomeIndex[i] = q,\\ &\text{where} \quad q = Random[1,\ N_c]\\ &\quad i = 1,\ 2, \cdots,\ k\end{aligned} \tag{26}$$

where N_c represents the number of chromosomes in each group; *ChromosomeIndex*[*i*] represents the index of a chromosome that select from *i*th group.

The illustration of the DMGS is shown in Fig. 5 with simple descriptions as follows: suppose the TPSO determines that 4 rules are expected, and 3 out of 7 groups, group 2, 3 and 6, are deemed as frequent groups. If the current action type of the DMGS is normal action, then 4 random groups will be selected to form a TFS. If the search action is taken, then frequent group 2, 3 and 6 will be selected. The remaining one group will be draw randomly from group 1, 4, 5 and 7. If the explore action is taken, then the 4 non-frequent group 1, 4, 5 and 7 will be selected in case of the problem of local optimum.

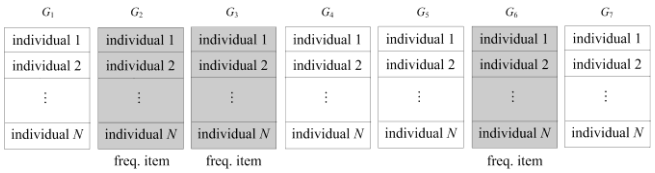

Fig. 5. The example of the DMGS.

D. Fitness assignment

The fitness value of a rule (an individual) is calculated by concatenating this individual with elites of other groups selected by DMGS. The details for assigning the fitness value are described as follows:

Denote G_1, G_2, …, G_{M_k}, the M_k groups selected by the DMGS; $G_j \cdot p_i$ denotes the *i*th individual of the *j*th group; y_j refers to the elite individual of the *j*th group. Then the fitness of the individual $G_j \cdot p_i$ can be computed as follows:

$$fitness(G_j \cdot p_i) = fitness(G_1 \cdot y_1, \cdots, G_j \cdot p_i, G_{j+1} \cdot y_{j+1}, \cdots, G_{M_k} \cdot y_{M_k}) \tag{27}$$

E. Reproduction

To perform reproduction, elite-based reproduction strategy (ERS) [22] is adopted in this study. In ERS, every chromosome in the best combination of Mk groups must be kept by performing reproduction step. In the remaining chromosomes in each group, the roulette-wheel selection method [32] is adopted for proceeding with the reproduction process.

Then the well-performed chromosomes in the top half of each group [14] proceed to the next generation. The other half is generated by performing crossover and mutation operations on chromosomes in the top half of the parent individuals.

F. Crossover

In this step, a two-point crossover strategy [32] is adopted. Once the crossover points are selected, exchanging the site's values between the selected sites of individual parents can create new individuals. These individuals are offspring which inherent the parents' merits.

G. Mutation

The utility of the mutation step can provide some new information to every group at the site of an individual by randomly altering the allele of a gene.

Thus mutation can lead to search new space which can avoid falling into the local minimal solution. In the mutation step, uniform mutation [33] is adopted, and the mutated gene is drawn randomly from the domain of the corresponding variable.

H. Calculation of group similarty

In order to achieve self adaptive group organization, it must determine the group similarity first. This paper involves the three measurements to determine the group similarity: 1. group centers, 2. group distance, and 3. group standard deviation.

Group centers: $$\mathbf{y}_j = \frac{1}{m_j}\sum_{i=1}^{m_j}\mathbf{x}_{l_{ij}},\ j = 1,2,\cdots,N_C \tag{28}$$

Group distance: $$d_{ij} = \left\|\mathbf{y}_i - \mathbf{y}_j\right\| = \sqrt{\sum_{i=1,\,j=1}^{P_{size}}\left(\mathbf{y}_i - \mathbf{y}_j\right)^2} \tag{29}$$

Group standard deviation: $$\sigma_j^{(i)} = \sqrt{\frac{1}{m_j}\sum_{p=1}^{m_j}\left(x_{l_{pj}}^{(i)} - y_j^{(i)}\right)^2} \tag{30}$$

$$\sigma_j^{(i_0)} = \max_{i=1,\cdots,n}\sigma_j^{(i)}$$

where $\mathbf{y}_j$ is the center of the *j*th group, m_j is the total number of *j*th group, $\mathbf{x}_{l_{ij}}$ is *i*th chromosome in the *j*th group, d_{ij} is the Euclidean distance between the *i*th group and *j*th group, $\sigma_j^{(i)}$ is the *i*th bit standard deviation in the *j*th group, $\sigma_j^{(i_0)}$ is the largest standard deviation of *j*th group.

After calculating above three measurements, it will be used to adjust the group organization of the neural fuzzy network by following two processes: splitting process and lumping process.

I. Splitting process

If $\sigma_j^{(i_0)} > \sigma_0$, where σ_0 is standard deviation splitting threshold, it means the chromosomes in the jth group are very dissimilar, so the AGOCEA will call Splitting process.

The Splitting process will divide jth group into two groups, which are j+ group and j- group, by following step: the top 50% (fitness value) chromosomes in the *j*th group will put into *j*+ group, the other 50% chromosomes in the *j*th group will put into *j*- group. The other 50% in the *j*+ group and *j*- group will generate randomly. After Splitting process, the dissimilar group will be separated into different groups, and total number of group will increase.

3.10 Lumping process

If $d_{ij} < d_0$, where d_0 is lumping threshold, it means the chromosomes are very similar between *i*th group and *j*th group, so the AGOCEA will call Lumping process. The Lumping process will merge *i*th group and *j*th group into a new group. The new group consists of the top 50% chromosomes from *i*th group and the top 50% chromosomes from *j*th group. After Lumping process, the similar groups will be merged into a new group, and total number of group will decrease.

IV. Simulation Results

The example used for identification of nonlinear dynamic system given by Narendra and Parthasarathy [34] is described as following difference equation:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{31}$$

The output of above equation depends nonlinearly on both its past value and the input, but the effects of the input and output values are not additive. The training input patterns are random value in the interval [-2, 2]. To determine the performance of the algorithms, this example adopts the root mean square error (RMSE). The definition of the RMSE is:

$$\text{RMSE} = \sqrt{\frac{\sum_{k=1}^{N}\left(\hat{y}(k) - y(k)\right)^2}{N}} \tag{32}$$

where $\hat{y}(k)$ is desired output, $y(k)$ is model output, and N is number of data.

In order to determine performance of the difference learning algorithm, this example is compared AGOCEA with HESP [35], ESP [36], MCGSE [22], SANE [37], and GA [6]. All algorithms were learned for 500 generations and repeated for 50 trails. The initial parameters of the AGOCEA are given in Table IV. Figure 6-11 show the output of all algorithms for the input signal $u(k) = \sin\left(\frac{2\pi k}{25}\right)$.

In these figures, the symbol "o" represents the desired output of the nonlinear dynamic system, and the symbol "*" represents the output of all algorithms. It can be seen from the Fig. 5-10 that the model output of AGOCEA has more accuracy than the other comparing learning algorithms.

TABLE IV. INITIAL PARAMETERS OF THE AGOCEA.

Parameters	Value
P_{size}	30
N_c	20
Selection_Times	40
NormalTimes	10
Searching Times	20
ExploringTimes	30
Crossover Rate	0.6
Mutation Rate	0.3
$[M_{min}, M_{max}]$	[5, 15]
$[m_{min}, m_{max}]$	[-10, 10]
$[\sigma_{min}, \sigma_{max}]$	[1,15]
$[w_{min}, w_{max}]$	[-10, 10]
σ_0	6
d_0	12

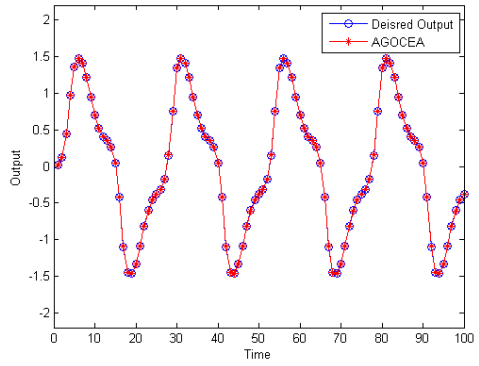

Fig. 6. Identification Results Of The Desired Output And The AGOCEA.

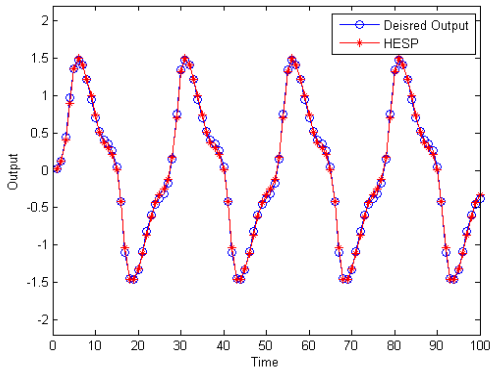

Fig. 7. Figure 4.2 Identification Results Of The Desired Output And The HESP.

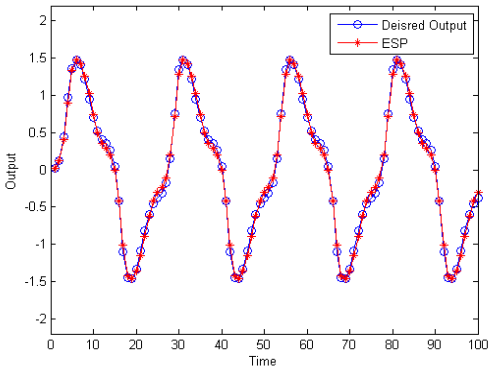

Fig. 8. Identification Results Of The Desired Output And The ESP.

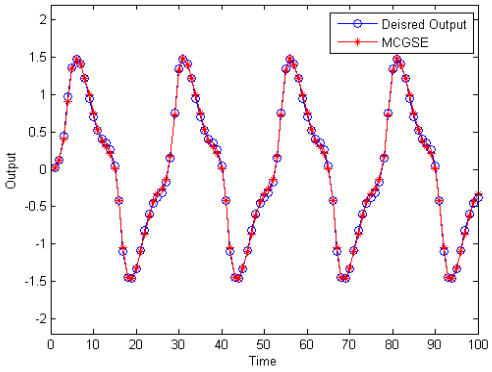

Fig. 9. Identification Results Of The Desired Output And The MCGSE.

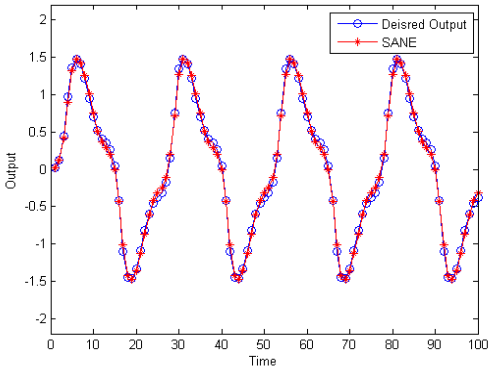

Fig. 10. Identification Results Of The Desired Output And The SANE.

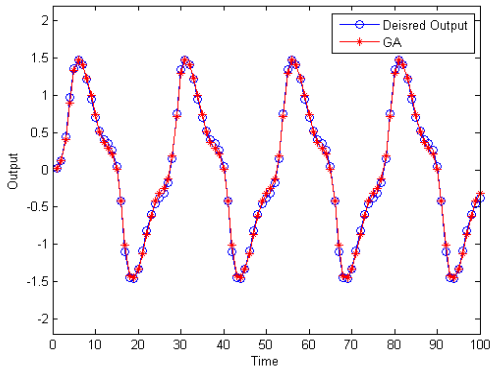

Fig. 11. dentification Results Of The Desired Output And The GA.

Figure 12 (a)-(f) show the identification error between the desired output and all algorithms' output. As shown in Fig. 12 (a)-(f), the AGOCEA illustrated the smaller error than other algorithms. Figure 13 provides the learning curve of the various learning algorithms, it can be seen from the learning curve that the AGOCEA converge faster and better than the other learning algorithms.

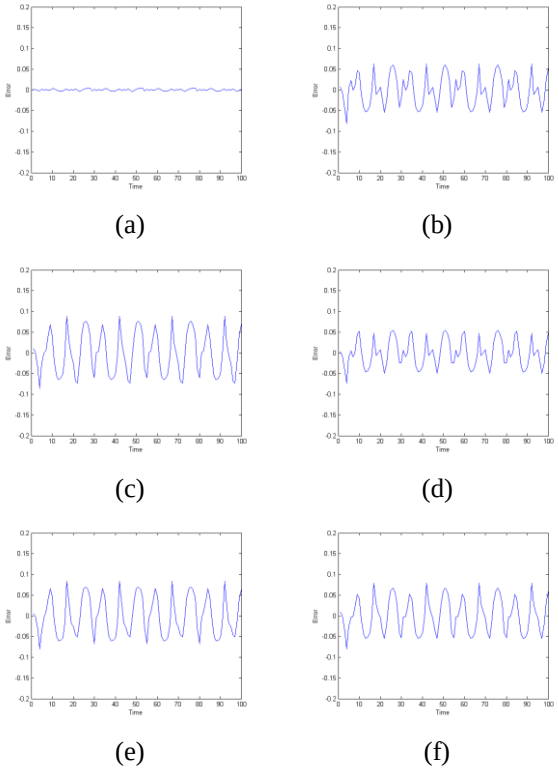

Fig. 12. Identification Errors Of The (A) AGOCEA, (B) HESP, (C) ESP (D) MCGSE, (E) SANE, And (F) GA.

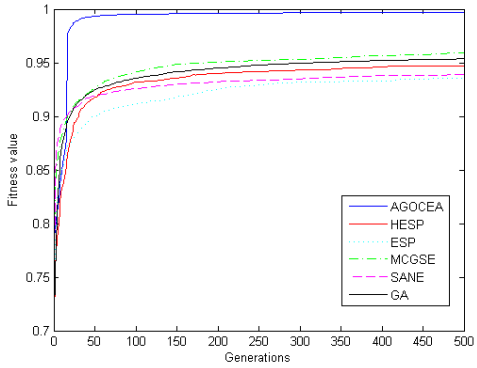

Fig. 13. The Learning Curve Of AGOCEA, HESP, ESP, MCGSE, SANE, And GA.

Table V shows the results obtained from a RMSE analysis of the various learning algorithms. There was a significant difference between the proposed AGOCEA and the other learning algorithms. No matter which performance index is, the proposed AGOCEA has the better performance than the other learning algorithms.

TABLE V. RMSE COMPARISON OF VARIOUS LEARNING ALGORITHMS.

Algorithm	RMSE			
	Mean	Best	Worst	STD
AGOCEA	**0.0023**	**0.0011**	**0.0034**	**0.0006**
HESP	0.0417	0.0285	0.0584	0.0071
ESP	0.0527	0.0342	0.0744	0.0112
MCGSE	0.0326	0.0228	0.0625	0.0062
SANE	0.0501	0.0274	0.0765	0.0117
GA	0.0470	0.0257	0.1117	0.0212

V. CONCLUSIONS

In this paper, the AGOCEA is proposed for designing TSK-type neural fuzzy network. The proposed AGOCEA not only determine the suitable number of fuzzy rules and group number but also efficiently tune the free parameters in the TNFN. The AGOCEA adopts the GSE that each population is divided to several groups and each group represents only one fuzzy rule. In order to solve the problem of random group selection, a data mining based group selection method was used to select the better groups. Furthermore, the adaptive group organization was proposed to solve the some parameters have to be assigned in advance. The simulation results show that the AGOCEA trained TNFN is superior to other methods. Although the proposed AGOCEA can obtain better results in comparison with the other learning algorithms, it still has a limitation. The important limitation lies in the fact that the proposed AGOCEA emphasize the network parameter learning and the group structure organization, it is a two level learning structure. While the problems become more complex, there is possible that increase the levels of learning structure.

Further research might explore how to determine the suitable levels of the learning structure for dealing with more complex problems.

REFERENCES

[1] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 15, no. 1, pp. 116-132, 1985.

[2] J. SR Jang, C. T. Sun, and E. Mizutani, *Neuro-fuzzy and soft computing : A computational approach to learning and machine intelligence*, Prentice Hall, Upper Saddle River, NJ, 1997.

[3] C. T. Lin and C. S. G. Lee, *Neural fuzzy systems: A neural-fuzzy synergism to intelligent systems*, Prentice Hall PTR, Upper Saddle River, NJ, 1996.

[4] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. on Fuzzy Systems*, vol. 5, no. 4, pp. 477-496, 1997.

[5] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. on Fuzzy Systems*, vol. 6, no. 1, pp.12-31, 1998.

[6] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, Reading, 1989.

[7] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, MIT Press, Cambridge, 1992.

[8] L. J. Fogel, *Evolutionary programming in perspective: The top-down view*, In: Zurada JM,Marks JM, Goldberg C (eds) Computational intelligence: imitating life, IEEE Press, New York, 1994.

[9] I. Rechenberg, *Evolution strategy*, In: Zurada JM, Marks JM, Goldberg C (eds) Computational intelligence: imitating life. IEEE Press, New York, 1994.

[10] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," in *Proc. of the 4th International Conference on Genetic Algorithms*, pp. 450-457, 1991.

[11] A. Homaifar and E. Mccormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 2, pp. 129-139, 1995.

[12] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proc. of the IEEE International Conference on Fuzzy Systems*, pp. 612-617, 1993.

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

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

[15] P. Kumar, V. K. Chandna, and M. S. Thomas, "Fuzzy-genetic algorithm for pre-processing data at the RTU," *IEEE Trans. on Power Systems*, vol. 19, no. 2, pp. 718-723, 2004.

[16] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "VGA-classifier: Design and applications," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 30, no. 6, pp. 890-895, 2000.

[17] B. Carse, T. C. Fogarty, and A. Munro, "Evolving fuzzy rule based controllers using genetic algorithms," *Fuzzy Sets and Systems*, vol. 80, no. 3, pp. 273-293, 1996.

[18] C. H. Chen, C. J. Lin, and C. T. Lin, "Using an efficient immune symbiotic evolution learning for compensatory neuro-fuzzy controller," *IEEE Trans. on Fuzzy Systems*, vol. 17, no. 3, pp. 668-682, 2009.

[19] C. J. Lin, C. H. Chen, C. T. Lin, "An efficient evolutionary algorithm for fuzzy inference systems," *Evolving Systems*, vol. 2, no. 2, pp. 83-99, 2011.

[20] C. H. Lin and Y. J. Xu, A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications, *Fuzzy Sets and Systems*, vol 157, no. 8, pp. 1036-1056, 2006.

[21] C. J. Lin and Y. C. Hsu, "Reinforcement hybrid evolutionary learning for recurrent wavelet-based neuro-fuzzy systems," *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 4, pp. 729-745, 2007.

[22] Y. C. Hsu, S. F. Lin, and Y. C. Cheng "Multi groups cooperation based symbiotic evolution for TSK-type neuro-fuzzy systems design," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5320-5330, 2010.

[23] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. of the 20th International Conference on Very Large Data Bases*, pp 487-499, 1994.

[24] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 1-12, 2000.

[25] D. T. Larose, *Discovering knowledge in data: an introduction to data mining*, Wiley-Interscience, Hoboken, 2005.

[26] U. Fayyad, "Data mining and knowledge discovery in databases: implications for scientific database," in *Proc. of International Conference on Scientific and Statistical Database Management*, pp 2-11, 1997.

[27] J. T. Lee, H. W. Wu, T. Y. Lee, Y. H. Liu, and K. T. Chen, "Mining closed patterns in multi-sequence time-series database," *Data and Knowledge Engineering*, vol. 68, no. 10, pp. 1071-1090, 2009.

[28] S. K. Tanbeer, C. F. Ahmed, and B. S. Jeong, "Parallel and distributed algorithm for frequent pattern mining in large database," *IETE Technical Review*, vol. 26, no. 1, pp. 55-65, 2009.

[29] S. F. Lin, J. W. Chang, and Y. C. Hsu, "A self-organization mining based hybrid evolution learning for TSK-type fuzzy model design," *Applied Intelligence*, vol. 36, no. 2, pp. 454-471, 2012.

[30] Y. T. Wu, Y. J. An, J. Geller, and Y. T. Wu, "A data mining based genetic algorithm," in *Proc. of the Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the second International Workshop on Collaborative Computing, Integration, and Assurance*, pp 52-62, 2006.

[31] S. F. Lin and Y. C. Cheng, "Two-strategy reinforcement evolutionary algorithm using data-mining based crossover strategy with TSK-type fuzzy controllers," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 9, pp.3863-3885, 2010.

[32] O. Cordon, 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*, vol 19. World Scientific Publishing, NJ, USA, 2001.

[33] E. Cox, *Fuzzy modeling and genetic algorithms for data mining and exploration*, 1st ed. Morgan Kaufman Publications, San Francisco, USA, 2005.

[34] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.

[35] F. Gomez F and J. Schmidhuber (2005) "Co-evolving recurrent neurons learn deep memory POMDPs," in *Proc. of Conference on Genetic and Evolutionary Computation*, pp 491-498

[36] F. J. Gomez, Robust non-linear control through neuroevolution, Ph. D. Disseration, The University of Texas at Austin, 2003.

[37] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement leraning through symbiotic evolution," *Meachin Learning*, vol. 22, pp. 11-32, 1996.