Neural Comput & Applic (2013) 23:485–498
DOI 10.1007/s00521-012-0943-0

ORIGINAL ARTICLE

Data mining–based hierarchical cooperative coevolutionary algorithm for TSK-type neuro-fuzzy networks design

Chi-Yao Hsu · Sheng-Fuu Lin · Jyun-Wei Chang

Received: 2 January 2012 / Accepted: 12 April 2012 / Published online: 25 April 2012

Abstract This study proposes a data mining–based hierarchical cooperative coevolutionary algorithm (DMHCCA) for TSK-type neuro-fuzzy networks design. The proposed DMHCCA consists of two-level evolutions: the neuro-level evolution (NULE) and the network-level evolution (NWLE). In NULE, a data mining–based evolutionary learning algorithm is utilized to evolve neurons. The good combinations of neurons evolved in NULE are reserved for being the initial populations of NWLE. In NWLE, the initial population are mated and mutated to produce new structure of networks. Similar to NULE, the good neurons of evolved network in NWLE are inserted into the NULE. Thus, by interactive two-level evolutions, the neurons and structure of network can be evolved locally and globally, respectively. Simulation results using DMHCCA are reported and compared with other existing models. Application of DMHCCA to a three-dimensional (3D) surface alignment task is also described, and experimental results are presented better performance than other alignment systems.

Keywords Hierarchical cooperative coevolutionary algorithm · Neuro-level evolution · Network-level evolution · Data mining–based evolutionary learning algorithm · Three-dimensional surface alignment

1 Introduction

In recent years, a fuzzy system used for several problems has become a popular research topic [1–10], especially for solving nonlinear and complex problems [11–14]. The reason is that fuzzy systems use fuzzy sets, instead of a mathematical model, for designing controllers. Therefore, fuzzy systems can solve the problem that inaccurate mathematical modeling degrades the performance of the controllers.

The fuzzy system consists of a set of fuzzy if–then rules that are selected according to a substantial amount of heuristic observations to express the knowledge of proper strategies frequently. Thus, it is difficult for human experts to examine all the input–output data from a complex system to find proper rules for a fuzzy system. To face with this challenge, there are several approaches proposed for generating if–then rules from numerical data [2, 3, 6]. These methods are all developed for supervised learning; that is, the correct "target" output value is given for each input pattern to guide the network's learning. Among them, the most well-known supervised learning algorithm is back-propagation (BP) [3, 6], which is a powerful training technique when tuning the parameters of networks. In addition, M. Riemiller and H. Braun [7] proposed a direct adaptive method for faster back-propagation learning: the RPROP algorithm. In their results, RPROP has shown a better performance in comparison with the gradient-descent method. Since the BP and RPROP algorithms are widespread to minimize the error function when training the networks, it may reach the local minima but never find the global solution. In addition, the performance of BP training depends on the initial values of the system parameters. Moreover, for different network topologies, one has to derive new mathematical expressions for each network layer.

Considering the above disadvantages, one may face with suboptimal performances, even for a suitable neural fuzzy network topology. Hence, the techniques capable of

C.-Y. Hsu · S.-F. Lin (✉) · J.-W. Chang
Department of Electrical Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC
e-mail: sflin@mail.nctu.edu.tw

training the parameters and finding a global solution while optimizing the overall structure are needed. To this end, evolutionary algorithms appear to be better candidates than the BP algorithm. Recently, an evolutionary fuzzy model has become a popular research field [15–28]. The evolutionary fuzzy model is a learning process to generate a fuzzy system automatically by incorporating evolutionary learning procedures. Among these evolutionary fuzzy models, the well-known algorithms are the genetic fuzzy models, that is, fuzzy models that are augmented by a learning process based on genetic algorithms (GAs). In spite of the genetic fuzzy models being used to seek the optima solutions, they may have some limitations such as same lengths of chromosomes, predefined parameters, and so on. Hence, several improved evolutionary algorithms have been proposed [22–28] to take into account these limitations. In [22], Bandyopadhyay et al. used the variable-length genetic algorithm to let chromosomes with different lengths in a same population. Carse et al. [23] used the genetic algorithm to evolve fuzzy rule-based controllers. In [24], authors presented an efficient immune symbiotic evolution learning algorithm to compensate the neuro-fuzzy controller. The experimental results showed that their approach has adopted to solve several nonlinear control problems. Lin et al. [25] proposed a novel self-constructing evolutionary algorithm for designing a TSK-type fuzzy model. Their algorithm exhibited good results on the water bath temperature control problem. Gomez and Schmidhuber proposed lots of work to consider these limitations [26, 27]. In their work, enforced subpopulations (ESP) are proposed to use subpopulations of neurons for the fitness evaluation and overall control. As shown in [26], the subpopulations that use to evaluate the solution locally can obtain better performance compared to systems with only one population used to evaluate the solution. Nevertheless, ESP do not reserve the good combinations of subpopulations whose fitness is high. It indicates that information about potential combinations of subpopulations is lost. In [28], Lin and Hsu 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. More recently, Hsu and Lin [29] proposed a multi-groups cooperation-based symbiotic evolution (MGCSE) to train a TSK-type neuro-fuzzy network (TNFN). Their results showed that MGCSE can obtain better performance and convergence than symbiotic evolution. In spite of MGCSE being a good approach for training a TNFN, it would not be suitable for complex problems. The reason is that complex problems lead to large amount of parameters must be trained. Thus, it could result in slow rate of convergence. In addition, MGCSE performed random group combination to construct a network. In spite of the fact that such action can sustain diversity, there is no systematic way to identify suitable groups for selecting chromosomes.

Although the above evolutionary learning algorithms [22–29] can improve traditional genetic algorithms, these algorithms may conduct one or more of the following problems: (1) the random group selection of fuzzy rules, (2) low convergence rate as the problem becomes complex, and (3) potential fuzzy rules combinations is lost.

To this end, this study proposes data mining–based hierarchical cooperative coevolutionary algorithm (DMH CCA) for improving the problems of evolutionary learning algorithms that were mentioned above. The notion of DMHCCA is to utilize two-level evolutions: the neuro level and the network level. At the neuro level, to solve the problem of the random group selection, this paper utilizes a data mining–based evolutionary learning algorithm (DELA) to evolve neurons. The reason why we adopt the data mining approach is that data mining has been widespread used in several fields [30, 31]. Data mining is a method of mining information from a database called "transactions." Data mining can be regarded as a new way for performing data analysis. One aim of data mining is to find association rules among sets of items that occur frequently in transactions. To achieve this aim, several methods have been proposed [32–34], and a comprehensive survey of discovering frequent item sets and association rules have been presented in [35]. In [32], the authors proposed a mining method, which ascertains large sets of items to find out the association rules in transactions. Hang et al. [33] proposed frequent pattern growth (FP-growth) to mine frequent patterns without candidate generation. In Hang's work, items that occur more frequently will have better chances of sharing information than items that occur less frequently. Wu et al. [34] proposed a data mining method based on the GA algorithm that efficiently improves the traditional GA by using analysis and confidence parameters. Thus, the DELA method adopts a data mining method to systematically select the group of fuzzy rules that can solve the problem of the random group selection. Besides, the regularized least square (RLS) is proposed to increase the convergence rate. At the network level, the good combinations of neurons (fuzzy rules) are reserved and evolved into new ones. Moreover, DMHCCA proposed variable antecedent-part crossover (VAC) and variable antecedent-part mutation (VAM) at network level such that the variable length of chromosomes can be mated and mutated. Therefore, DMHCCA tries to improve hierarchical enforced subpopulations (HESP) [27] that only fixed length of networks can be evaluated in one generation. In the first example of our experimental section, DMHCCA is compared with H-ESP, and the results show that the proposed DMHCCA is proven superior to H-ESP. In the second example, a three-dimensional (3D) surface alignment task is adopted to examine the learning performance of DMHCCA. The experimental results show that the proposed DMHCCA trained TNFN-based alignment approach is better than other alignment systems.

This paper is organized as follows. In Sect. 2, a TSK-type neuro-fuzzy network is introduced. The proposed DMHCCA is described in Sect. 3. In Sect. 4, the illustration examples are presented. The conclusions are given in the last section.

2 Structure of TSK-type neuro-fuzzy network (TNFN)

A TSK-type neuro-fuzzy network (TNFN) [5] employs different implication and aggregation methods from a standard Mamdani fuzzy system. According to [6, 36], authors have shown that a TSK-type NFN can offer better network size and learning accuracy than a Mamdani-type NFN. Thus, instead of using fuzzy sets, the conclusion part of a fuzzy rule is a linear combination of the crisp inputs. The fuzzy rule of TNFN is shown in Eq. (1), where n and j represent the number of the input dimensions and the serial number of the fuzzy rules, respectively.

$$\begin{aligned}&\text{IF } x_1 \text{ is } A_{1j}(m_{1j}, \sigma_{1j}) \text{ and } x_2 \text{ is } A_{2j}(m_{2j}, \sigma_{2j})\\ &\quad \ldots \text{and } x_n \text{ is } A_{nj}(m_{nj}, \sigma_{nj})\\ &\quad \text{THEN } y' = w_{0j} + w_{1j}x_1 + \cdots + w_{nj}x_n \end{aligned} \tag{1}$$

The structure of a TNFN is shown in Fig. 1, where n represents the number of input dimensions. It is a five-layer network structure. In the TNFN, the firing strength of a fuzzy rule is calculated by performing the following "AND" operation on the truth values of each variable to its corresponding fuzzy sets by:

$$u_{ij}^{(3)} = \prod_{i=1}^{n} \exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2}\right) \tag{2}$$

where $u_i^{(1)} = x_i$ and $u_{ij}^{(3)}$ are the outputs of 1st and 3rd layers; m_{ij} and σ_{ij} are the center and the width of the

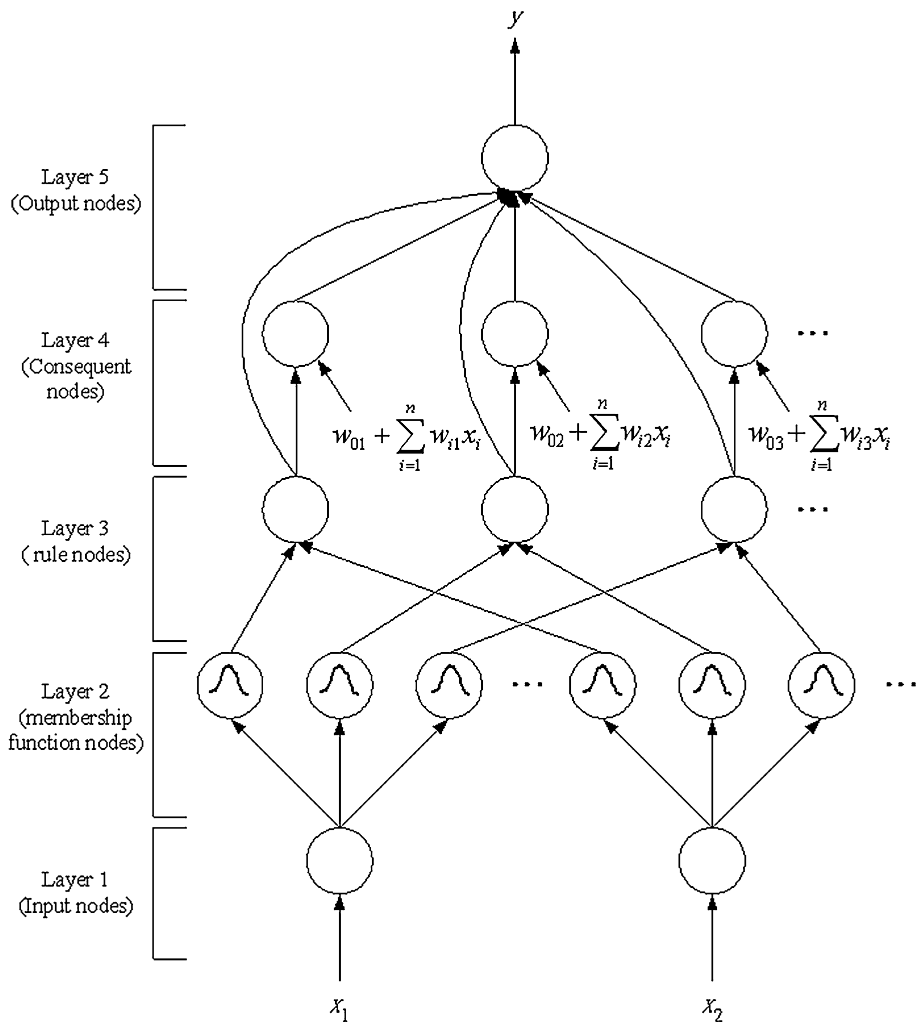
y
Layer 5
(Output nodes)
Layer 4
(Consequent nodes)
$w_{01} + \sum_{i=1}^{n} w_{i1}x_i$
$w_{02} + \sum_{i=1}^{n} w_{i2}x_i$
$w_{03} + \sum_{i=1}^{n} w_{i3}x_i$
Layer 3
(rule nodes)
Layer 2
(membership function nodes)
Layer 1
(Input nodes)
x_1
x_2

Fig. 1 Structure of the TSK-type neuro-fuzzy network

Gaussian membership function of the jth term of the ith input variable x_i, respectively.

The output of the neuro-fuzzy network is computed by:

$$y = u^{(5)} = \frac{\sum_{j=1}^{M} u_j^{(4)}}{\sum_{j=1}^{M} u_j^{(3)}} = \frac{\sum_{j=1}^{M} u_j^{(3)} \left(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i\right)}{\sum_{j=1}^{M} u_j^{(3)}} \tag{3}$$

where $u^{(5)}$ is the output of 5th layer; w_{ij} is the weighting value with ith dimension and jth rule node; and M is the number of fuzzy rule.

3 Data mining–based hierarchical cooperative coevolutionary algorithm

The learning process of DMHCCA is shown in Fig. 2. As shown in this figure, DMHCCA involves two major evolutions: neuro-level evolution and network-level evolution. The blocks of inserting good networks and inserting good neurons are the connection between the neuro and network-level evolution. These two operations indicate that good evolved results in one level evolution would be transferred to another level evolution. Once receiving good neurons or networks, the received chromosomes would be mated with other old chromosomes to yield some new offspring. Therefore, by exchanging the good information between two levels of evolution, we have more chance to find the global optimal solution.

3.1 Neuro-level evolution

In this subsection, we will discuss the neuro-level evolution (NULE). To consider the structure of TNFN, NULE adopts the variable length of a combination of chromosomes with RLS method to construct a TNFN. The structure of chromosomes to construct TSK-type neuro-fuzzy networks (TNFNs) in NULE is shown in Fig. 3. In this figure, each antecedent part of a fuzzy rule represents a chromosome selected from a group, P_{size} denotes that there are P_{size} groups in a population, and M_k indicates that there are M_k rules used in TNFN construction.

After discussing the structure of chromosomes to construct TNFNs, details of the coding step for NULE and RLS method are described as follows:

(1) *Coding Step*: The coding structure of chromosomes in the proposed NULE is shown in Fig. 4. This figure describes an antecedent part of a fuzzy rule that has the form in Eq. (1), where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation of ith dimension and jth rule node, respectively. Besides, a pair of (m, σ) indicates a neuron in Layer 2 of a TNFN. Evolving an antecedent part of a fuzzy rule is likely to evolve a neuron. Thus, the evolution of this level is called a neuro-level evolution.

(2) *RLS method*: Assume a TSK-type neural fuzzy model composed of m fuzzy rules as the following form:

$$\begin{aligned} &R_j : \text{IF } x_1 \text{ is } A_1^j \ldots \text{and } x_n \text{ is } A_n^j, \\ &\quad \text{THEN } y_j = w_o^j + w_1^j x_1 + \cdots + w_n^j x_n \end{aligned} \tag{4}$$

where $j = 1, \ldots, m$ and A_i^j is the linguistic part with respect the input i and *Rule j*. From Eq. (4), the output can be written as:

$$y = \frac{\sum_{j=1}^{m} u_j y_j}{\sum_{j=1}^{m} u_j} = \hat{u}_1 y_1 + \hat{u}_2 y_2 + \cdots + \hat{u}_m y_m, \tag{5}$$

where u_j is the firing strength of *Rule j*, and $\hat{u}_j = u_j/(u_1 + \cdots + u_m)$. Then it is possible to express the equation above into the form:

$$\begin{aligned} y &= \hat{u}_1\left(w_0^1 + w_1^1 x_1 + \cdots + w_n^1 x_n\right) \\ &\quad + \cdots + \hat{u}_m\left(w_0^m + w_1^m x_1 + \cdots + w_n^m x_n\right) = aW \end{aligned} \tag{6}$$

where $W = [W_1^T \cdots W_m^T]^T$, $W_j = [w_0^j \cdots w_n^j]^T$, $j = 1, \ldots m$, and

$$\begin{aligned} a = [&\hat{u}_1\, \hat{u}_1 x_1 \ldots \hat{u}_1 x_n \\ &\hat{u}_2\, \hat{u}_2 x_1 \ldots \hat{u}_2 x_n \\ &\qquad \vdots \\ &\hat{u}_m\, \hat{u}_m x_1 \cdots \hat{u}_m x_n]^T \end{aligned}$$

Since y and a are known values, the only unknown value is the consequent part W. Suppose a given set of training inputs and desired outputs is $\{x(t), y_d(t)\}_{t=1}^{M}$. The Eq. (6) can be rewritten as:

$$AW = Y_d \tag{7}$$

where $A = [a(1)\ a(2) \ldots a(M)]^T$.

In order to get the smooth estimation, the regularization is adopted. The approximation solution can be written as follows:

$$\hat{W} = (A^T A + \lambda I)^{-1} A^T Y_d, \tag{8}$$

where λ is a regularization parameter that adjusts the smoothness. Thus, by getting Eq. (8), we finish the estimation of the consequent part of fuzzy rules.

The learning process of NULE involves seven operators: initialization, self-organization algorithm, data mining–based selection method, fitness assignment, reproduction, crossover, mutation, and insert good networks. The whole learning process is introduced below:

a. *Initialization*: Before we start the neuro-level evolution, the initial groups of individuals should be generated. Thus, initial groups are generated randomly within a

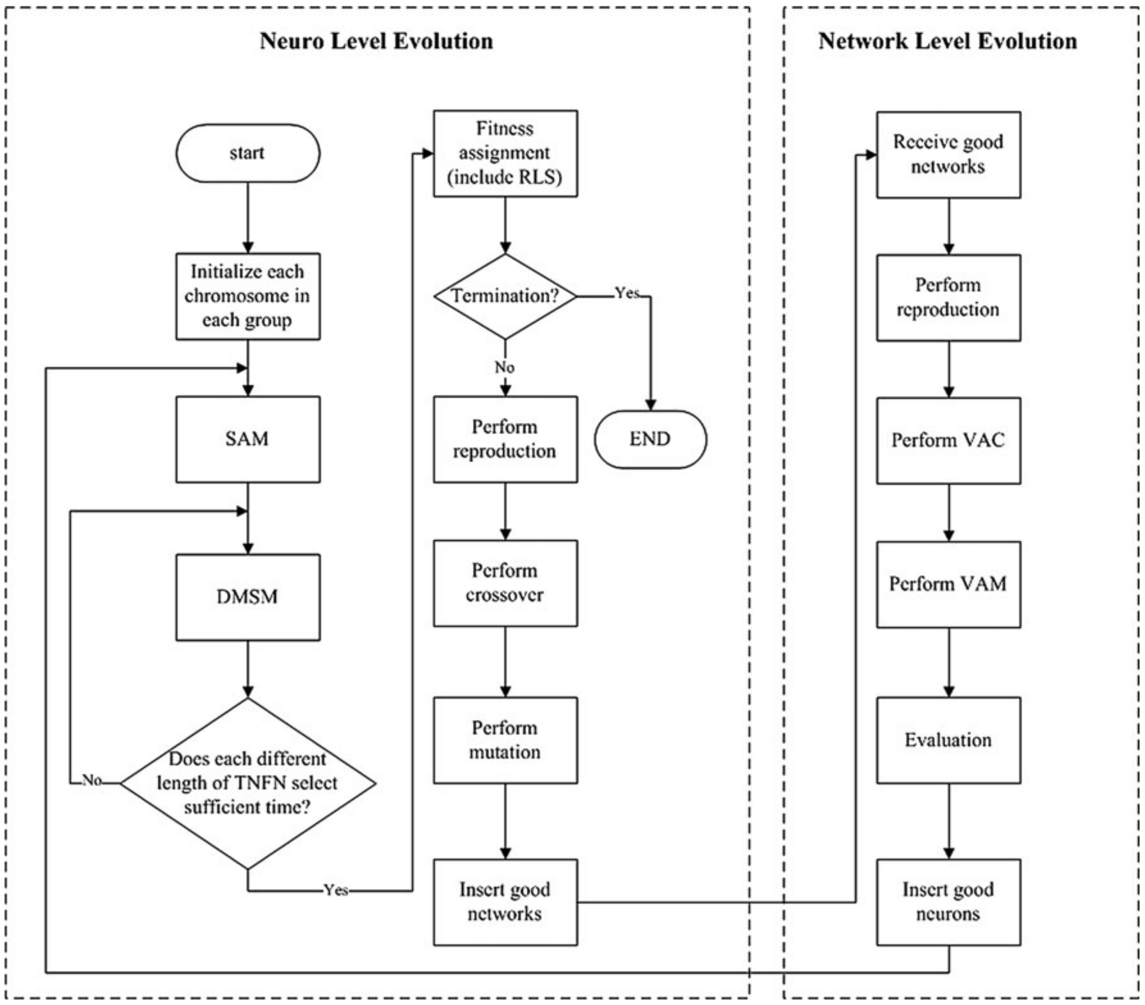

Fig. 2 Learning process of DMHCCA

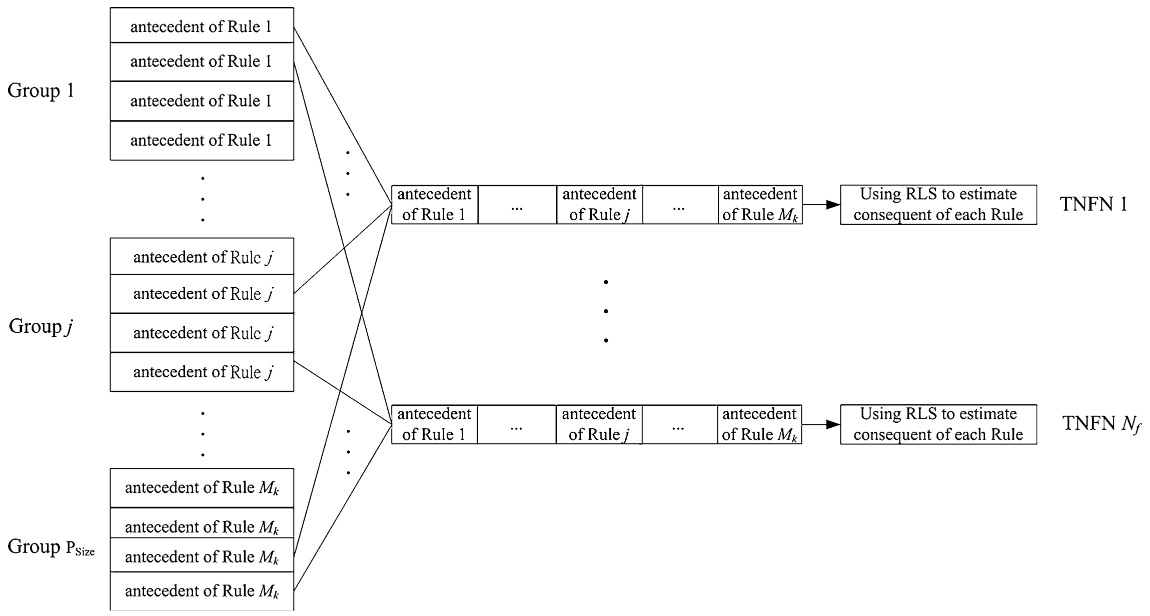

Fig. 3 Structure of chromosomes to TNFN construction in NULE

m_{1j}	σ_{1j}	m_{2j}	σ_{2j}	$\cdots$	m_{nj}	σ_{nj}

Fig. 4 Coding an antecedent part of a fuzzy rule into a chromosome in NULE

predefined range. The following formulations show how to generate the initial chromosomes in each group:

$$\begin{aligned}&\text{Deviation: } \mathrm{Chr}_{g,c}[p] = random[\sigma_{\min}, \sigma_{\max}],\\ &\quad \text{where } p = 2, 4, \ldots, 2n;\\ &\quad g = 1, 2, \ldots, P_{\text{size}}; \quad c = 1, 2, \ldots, N_C, \end{aligned} \tag{9}$$

$$\begin{aligned}&\text{Mean: } \mathrm{Chr}_{g,c}[p] = random[m_{\min}, m_{\max}],\\ &\quad \text{where } p = 1, 3, \ldots, 2n-1,\end{aligned} \tag{10}$$

where $\mathrm{Chr}_{g,c}$ represents cth chromosome in the gth group, N_C is the total number of chromosomes in each group, p represents the pth gene in a $\mathrm{Chr}_{g,c}$, and $[\sigma_{\min}, \sigma_{\max}]$, $[m_{\min}, m_{\max}]$ represent the predefined range to generate the chromosomes.

b. *Self-adaptive method (SAM)*: To select fuzzy rules automatically, the proposed DMHCCA adopts our previous research—the self-adaptive method (SAM) [37]—to determine the suitability of TNFN models with different fuzzy rules. The self-adaptive method encodes the probability vector V_{M_k} to stand for the suitability of a TNFN with M_k rules. In addition, in SAM, the minimum and maximum number of rules must be predefined to limit the number of fuzzy rules to a certain bound, that is, $[M_{\min}, M_{\max}]$. The processing steps of SAM are described as follows:

Step 1. Update the probability vectors V_{M_k} according to the following equations:

$$\begin{cases} V_{M_k} = V_{M_k} + (Upt\_value_{M_k} * \lambda), & \text{if} \quad Avg \le fit_{M_k} \\ V_{M_k} = V_{M_k} - (Upt\_value_{M_k} * \lambda), & \text{otherwise} \end{cases} \tag{11}$$

$$Avg = \sum_{M_k=M_{\min}}^{M_{\max}} fit_{M_k} / (M_{\max} - M_{\min} + 1), \tag{12}$$

$$Upt\_value_{M_k} = fit_{M_k} \Big/ \sum\nolimits_{M_k=M_{\min}}^{M_{\max}} fit_{M_k}; \tag{13}$$

$$\begin{aligned}&\text{if} \quad Fitness_{M_k} \ge (Best\_Fitness_{M_k} - ThreadFitnessvalue)\\ &\quad \text{then} \quad fit_{M_k} = fit_{M_k} + Fitness_{M_k},\end{aligned} \tag{14}$$

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

Step 2. Determine the selection times of TNFN with different rules according to the probability vectors as follows:

$$Rp_{M_k} = (Selection\_Times) * (V_{M_k} / Total\_Velocy), \tag{15}$$

$$Total\_Velocy = \sum_{M_k=M_{\min}}^{M_{\max}} V_{M_k}, \tag{16}$$

where $M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}$, $Selection\_Times$ represents the total selection times in each generation and Rp_{M_k} represents the selection times of TNFN with M_k rules in one generation.

Step 3. Accumulator calculation: If the current best combination of chromosomes does not improve, then accumulator can be computed as below:

$$\begin{aligned}&\text{if} \quad Best\_Fitness_g = Best\_Fitness,\\ &\quad \text{then}\, Accumulator = Accumulator + 1,\end{aligned} \tag{17}$$

where $Best\_Fitness_g$ represents the best fitness value of the best combination of chromosomes in the gth generation, and $Best\_Fitness$ represents the best fitness value of the best combination of chromosomes in the current generations.

c. *Data mining–based selection method (DMSM)*: This process performs the selection step, which involves the selection of groups and the selection of chromosomes.

(1) *Selection of groups*: This paper proposes DMSM to determine the suitable groups for chromosomes selection to form a TNFN. In DMSM, suitable groups are selected according to the groups, which conduct from association rules that indicate good performance. In contrast, unsuitable groups are avoided selecting according to the groups, which conduct from association rules that demonstrate bad performance. To perform DMSM, we use a transaction-built action and an association rule mining action to select the well-performing groups. The details of these two actions are described as follows.

Action1: Transaction-built action.

The aims of this action are twofold: accumulate the transaction set and select groups. Regarding the accumulation of transaction set, the transactions are built using the following equations:

$$\begin{aligned}&\text{if} \quad Fitness_{M_k} \ge (Best\_Fitness_{M_k} - ThreadFitnessvalue)\\ &\quad Transaction_j[i] = TFCRuleSet_{M_k}[i]\\ &\quad \text{then}\\ &\quad PerformanceIndex = g,\end{aligned} \tag{18}$$

$$\begin{aligned}&\text{if} \quad Fitness_{M_k} < (Best\_Fitness_{M_k} - ThreadFitnessvalue)\\ &\quad Transaction_j[i] = TFCRuleSet_{M_k}[i]\\ &\quad \text{then}\\ &\quad PerformanceIndex = b,\end{aligned} \tag{19}$$

where $i = 1, 2, \ldots, M_k$, $M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}$, $j = 1, 2, \ldots, TransactionNum$, the $Fitness_{M_k}$ is the fitness value of TNFN with M_k rules, $ThreadFitnessvalue$ is a

Table 1 Transactions in the DMSM

Transaction index	Groups	Performance index
1	1, 4, 8	g
2	2, 4, 7, 10	b
…	…	…
TransactionNum	1, 3, 4, 6, 8, 9	G

predefined value, *TransactionNum* is the total number of transactions, $Transaction_j[i]$ is the ith item in the jth transaction, $TFCRuleSet_{M_k}[i]$ is the ith group in the M_k groups used for chromosomes selection, and $Performance\ Index = g$ and $Performance\ Index = b$ represent the good and bad performance, respectively. Hence, transactions have the form shown in Table 1. As shown in Table 1, the first transaction indicates that the three-rule TNFN formed by the first, fourth, and eighth groups has "good" performance. In contrast, the second transaction indicates that the four-rule TNFN formed by the second, fourth, seventh, and the tenth groups has "bad" performance.

Regarding the group selection, DMSM selects groups using the following equation:

$$\begin{aligned}&\text{if}\quad Accumulator \le NormalTimes\\&\text{then}\quad GroupIndex[i] = Random[1, P_{\text{Size}}],\end{aligned}\tag{20}$$

where $i = 1, 2, \ldots, M_k$, $M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}$, *Accumulator* is used to determine which action should be adopted, $GroupIndex[i]$ is the selected ith group of the M_k groups, and P_{Size} indicates that there are P_{Size} groups in a population. If the best fitness value does not improve for a sufficient number of generations (*NormalTimes*), then DMSM selects groups according to the association rule mining action.

Action 2. Association rule mining action.

In the association rule mining action, suitable groups are selected according to the association rules. To produce the association rules with good performance, the *frequent groups* must be found in advance. Thus, we adopt FP-growth method described in [33] to find the *frequent groups*. Then, the found *frequent groups* are compared with the groups owing bad performance shown in Table 1 to count the confidence degree, which can be computed by the following formula:

$$\begin{aligned}&confidence(frequent\ groups \Rightarrow good)\\&= P(good|frequent\ groups)\\&= \frac{supp\ (frequent\ groups\ \cup\ good)}{supp\ (frequent\ groups\ \cup\ good) + supp\ (frequent\ groups\ \cup bad)},\end{aligned}\tag{21}$$

where $P(good|frequent\ groups)$ is the conditional probability, *frequent groups* $\cup$ *good* or *bad* is the union of *frequent groups* and good or bad performance, and *supp* (*frequent groups* $\cup$ *good* or *bad*) is the counts of *frequent groups* with good or bad performance occurring in transactions. Then, the rule is valid if

$$confidence(frequent\ groups \Rightarrow good) \ge minconf,\tag{22}$$

where *minconf* is the minimal confidence given by a user or an expert. Hence, we can infer that if a rule satisfies Eq. (22), then the *frequent groups* can be considered as the suitable groups. For example, if the confidence of {2, 5, 8} ⇒ {g} is larger than the minimum confidence, we produce this association rule, which indicates that the combination of the second, fifth, and eighth groups have "good" performance. After doing so, the frequent groups are conducted to produce association rules and generate the *AssociatedGoodPool*, which contains all frequent groups that satisfy Eq. (22).

After the association rules are constructed, DMSM selects groups according to the association rules. The group indexes are selected from the associated good groups according to the following equations:

$$\begin{aligned}&\text{if}\quad NormalTimes < Accumulator \le ExploreTimes\\&\quad\text{then}\quad GroupIndex[i] = w,\\&\quad\text{where}\quad w = GoodItemSet[q]\\&\quad = Random[AssociatedGoodPool],\end{aligned}\tag{23}$$

where $q = 1, 2, \ldots, AssociatedGoodPoolNum$, $i = 1, 2, \ldots, M_k$, $M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}$, *ExploreTimes* is a predefined value that judge to perform the association rule mining action, *AssociatedGoodPool* is the sets of good item set obtained from the association rules, *AssociatedGoodPoolNum* is the total number of sets in *AssociatedGoodPool* and $GoodItemSet[i]$ presents a good item set randomly selected from *AssociatedGoodPool*. In the Eq. (23), if M_k is greater than the size of *GoodItemSet*, the remaining groups are selected using Eq. (20). If the best fitness value does not improve for a sufficient number of generations (*ExploreTimes*), DMSM selects groups based on the transaction-built action and sets $Accumulator = 0$.

(2) *Selection of chromosomes*: After the M_k groups are selected, M_k chromosomes are selected from M_k groups as follows:

$$ChromosomeIndex[i] = q,\tag{24}$$

where $q = Random[1, N_c]$, $i = 1, 2, \ldots, k$, N_c is the total number of chromosomes in each group, and $ChromosomeIndex[i]$ is the index of a chromosome that is selected from the ith group.

d. *Fitness assignment*: To assign the fitness value of an individual, the following detailed steps in the fitness value assignment are performed:

Step 1. Choose M_k antecedent part of fuzzy rules using RLS method to construct a TNFN Rp_{M_k}times from M_k

groups with size N_C. The M_k groups are obtained from the DMSM.

Step 2. Evaluate every TNFN that is generated from Step 1 to obtain a fitness value. In this paper, the fitness value is designed according to the following formulation:

$$\text{Fitness Value} = 1/(1 + E(y, \bar{y})), \tag{25}$$

where

$$E(y, \bar{y}) = \sum_{i=1}^{N} (y_i - \bar{y}_i)^2, \tag{26}$$

where y_i and $\bar{y}_i$ represents the desired and predicted values of the ith output, respectively, $E(y, \bar{y})$ is an error function and N represents the number of the training data in each generation.

Step 3. Divide the fitness value by M_k and accumulate the divided fitness value to the selected antecedent part of fuzzy rules with their fitness value records.

Step 4. Divide the accumulated fitness value of each chromosome from M_k groups by the number of times that it has been selected.

e. *Reproduction*: To perform reproduction, elite-based reproduction strategy (ERS) [29] is adopted. In ERS, every chromosome with the best performance is kept. In the remaining chromosomes in each group, the roulette-wheel selection method [38] is adopted for proceeding with the reproduction process. Then the well-performed chromosomes in the top half of each group [21] 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 [38] 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 that 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 that would prevent from falling into the local minimal solution. In the mutation step, uniform mutation [39] is adopted, and the mutated gene is drawn randomly from the domain of the corresponding variable.

h. *Insert good networks*: Since there are "*Selection_Times*" networks constructed in every generation, the fitness value of each network is recorded and compares it with the network evolution level. If the fitness of the network is better than the worst network in the network evolution level, then this network is inserted into the network evolution level.

If the number of generations reaches a predefined maximal iteration value or the best fitness value is greater than a fitness threshold, DMHCCA is terminated, and output the final results.

3.2 Network-level evolution

In this subsection, the network-level evolution (NWLE) is discussed. The main processes of NWLE involve six operations: receive good networks, reproduction, variable antecedent-part crossover, variable antecedent-part mutation, evaluation, and insert good neurons. The details of these operations are described as follows:

a. Receive good networks: Before the network evolution starts, we receive N well-performed networks from neuro-level evolution to be chromosomes. The coding structure of chromosomes in the network-level evolution is shown in Fig. 5. In this figure, each block of a chromosome describes an antecedent part of a fuzzy rule that has the form in Eq. (4), where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation of ith dimension and jth rule node, respectively. The consequent part of a fuzzy rule is skipped to encode into chromosomes since regularized least square is proposed to estimate the consequent part. After that, we sort the chromosomes to prepare for performing reproduction.
b. Reproduction: Reproduction is a process in which string are copied according to their fitness value. In this step, roulette-wheel selection method is adopted for the reproduction process. The well-performed chromosomes in the top half of each group proceed to the next generation. The other half is generated by executing variable two-part and variable two-part operations on chromosomes in the top half of the parent individuals.
c. Variable antecedent-part crossover: In the network-level evolution, the variable antecedent-part crossover (VAC) is proposed to perform crossover. In VAC, two parents are selected by using the roulette-wheel selection method [38]. Because the selected parents may be with different length, the misalignment of individuals must be avoided in the crossover operation. Thus, antecedent-part crossover is proposed to address

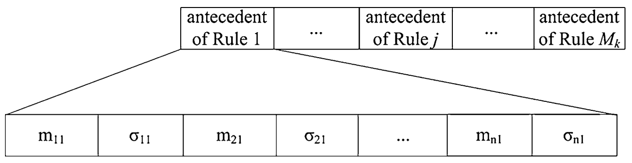

Fig. 5 The coding the antecedent part of fuzzy rules into a chromosome in the network-level evolution

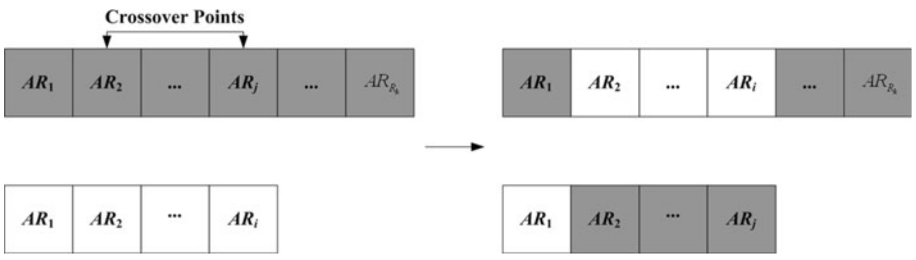

Fig. 6 The variable antecedent-part crossover operation in the network-level evolution

this problem. The antecedent part means that only the antecedent of fuzzy rule is performed crossover operation. In VAC, two-point crossover [38] is adopted to execute crossover. Thus, new individuals are generated by exchanging the site's values between the selected sites of the parents' individuals. In VAC, to avoid the misalignment of individuals in the crossover, the selection of the crossover points would not exceed the shortest length chromosome of two parents. Two individuals with different lengths using VAC operation are shown in Fig. 6 where AR_j represents the parameters of the antecedent part of the jth rule in the TNFN, and R_k represents there are k fuzzy rules in a TNFN. After performing the VAC, the new offspring can replace the individuals with poor performance.

d. Variable antecedent-part mutation: The mutation operator can randomly alter the allele of a gene. It provides new information to every population at the site of an individual. In the network-level evolution, the variable antecedent-part mutation (VAM) is adopted to perform the mutation operation. The benefit of VAM is to be applied to different length of chromosomes. The VAM operation of each individual is shown in Fig. 7 where AR indicates antecedent part of fuzzy rule In VAM, uniform mutation [39] is adopted, and the mutated gene is drawn randomly from the domain of the corresponding variable.

e. Evaluation: The evaluating step is to evaluate the fitness of each chromosome that has not already been evaluated in a population. The higher a fitness value indicates, the better the performance. Since each chromosome only includes the antecedent part of fuzzy rules, the consequent part of fuzzy rules is not defined. Thus, similar to the fitness assignment in NULE, RLS method is used to estimate the consequent part of fuzzy rules.

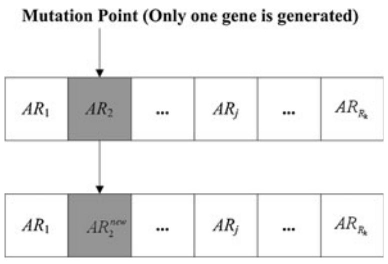

Fig. 7 The variable antecedent-part mutation operation in the network-level evolution

f. Insert good neurons: After the evaluation operation, if a network has a higher fitness value than the best network in the neuro level, insert the neurons into the corresponding groups of subpopulation in the neuro-level evolution.

In short, the purpose of NWLE is to reserve the good combinations of fuzzy rules produced by NULE and evolve the structure of the produced neural fuzzy networks. Thus, the utility of NWLE is to fine tune the evolved results of NULE. To this end, NULE would be a major evolution to evolve TNFNs and it affects the effectiveness of the proposed DMHCCA model.

4 Illustration examples

To verify the proposed DMHCCA, two examples are discussed in this section. The first one is a prediction of Mackey–Glass time series. The second one is a three-dimensional (3D) surface alignment task. Based on these examples, this study compares DMHCCA with that of others methods. The initial parameters for the two examples are given in Table 2. The initial parameters of the proposed DMHCCA are determined by parameter exploration in [40], which was the first study in parameter exploration. As shown in [40], a small population size is good for the initial performance, and a large population size is good for long-term performance. Moreover, a low mutation rate is good for online performance, and a high mutation rate is good for off-line performance. Thus, we adjust parameters of DMHCCA according to the criterion mentioned in parameter exploration method.

4.1 Example 1: Prediction of Mackey–Glass time series

The Mackey–Glass time series is a common benchmark for testing different learning algorithms. Thus, we utilize such chaotic time series to perform an extensive analysis on our proposed algorithm and other evolutionary algorithms

Table 2 Initial parameters of DMHCCA before training

Parameters	Value		Parameters	Value	
	NULE	NWLE		NULE	NWLE
P_{size}	40	20	Mutation rate	0.2	0.1
N_c	20	None	$[M_{\min}, M_{\max}]$	[3, 25]	None
Selection_Times	50	None	$[m_{\min}, m_{\max}]$	[−10, 10]	[−10, 10]
NormalTimes	10	None	$[\sigma_{\min}, \sigma_{\max}]$	[1, 15]	[1, 15]
ExploreTimes	15	None	*minconf*	60 %	None
Crossover rate	0.6	0.7	RLS parameter (λ)	0.003	0.003

The Mackey–Glass time series is generated from the following delay differential equation:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{27}$$

Crowder [41] extracted 1,000 input–output data pairs $\{x, y^d\}$, which consisted of four past values of $x(t)$, that is

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)] \tag{28}$$

where $\tau = 17$ and $x(0) = 1.2$. There are four inputs to DMHCCA, corresponding to these values of $x(t)$, and one output representing the value $x(t + \Delta t)$, where Δt is a time prediction into the future. The first 500 pairs [from $x(1)$ to $x(500)$] are the training data set, and the remaining 500 pairs [from $x(501)$ to $x(1{,}000)$] are the testing data set used for validating the proposed method. The values are floating-point numbers assigned using the DMHCCA initially. The fitness function in this case is defined in Eqs. (25) and (26) to train the neural fuzzy network. The evolution learning processes 500 generations; it is repeated 50 times. For comparative analysis, the present study adopts the root mean square error (RMSE), which is defined as follows:

$$\text{RMSE} = \left[\frac{1}{N_t}\sum_{l=1}^{N_t}\left(Y_l(t+6) - Y_l^d(t+6)\right)^2\right]^{1/2}, \tag{29}$$

where N_t is the number of testing data, $Y_l^d(t+6) = x(t+6)$ is the desired value, and $Y_l(t+6)$ is the predicted value by the model with four inputs and one output.

To compare with other algorithms, in this example, according the parameter exploration method [40], 12, 12, 10, and 12 fuzzy rules are set for hierarchical enforced subpopulations (HESP) [27], enforced subpopulations (ESP) [26], traditional symbiotic evolution (TSE) [42], and traditional genetic algorithm (TGA) [19], respectively. In addition, the population size has the range of 10–250 in increments of 10, the crossover rate has the range of 0.25–1 in increments of 0.05, and the mutation rate has the range of 0–0.3 in increments of 0.01. Toward this end, the other parameters setting for HESP, ESP, TSE, and TGA are as follows: (1) the population sizes are 30, 30, 200, and 50, respectively; (2) the crossover rates are 0.6, 0.6, 0.4, and 0.7, respectively; (3) the mutation rate of the four methods are 0.04, 0.05, 0.05, and 0.04, respectively. In addition, as same with DMHCCA method, the evolution learning of each method processes for 500 generations and is repeated 50 times.

Table 3 Performance comparison of various existing models

Method	RMSE			
	Best	Mean	Worst	STD
DMHCCA	0.0032	0.0048	0.0082	0.0011
HESP	0.0076	0.0092	0.0012	0.0014
ESP	0.0092	0.011	0.015	0.0016
TSE	0.015	0.019	0.024	0.0024
TGA	0.021	0.029	0.064	0.013

Table 4 Comparison of the running time of various algorithms

Method	Best (s)	Worst (s)	Mean (s)
DMHCCA	6.55	57.26	23.39
HESP	15.36	107.86	56.25
ESP	18.76	128.43	66.19
TSE	24.48	192.71	152.75
TGA	47.36	228.59	158.66

Table 3 lists the generalization capabilities of the proposed DMHCCA, HESP [27], ESP [26], TSE [42], and TGA [19]. Clearly, as shown in Table 3, DMHCCA obtains a lower RMSE than other methods.

Furthermore, this case also compares the running time of DMHCCA with that of other methods. The running time defined in this case is used to measure the time when the fitness of the algorithm converges to the predefined value. The results of four algorithms over 50 runs are reported in Table 4. As shown in this table, the proposed DMHCCA is faster than ESP, TSE, and TGA.

4.2 Example 2: 3D surface alignment task

In this example, we apply DMHCCA to a 3D surface alignment task. The example of 3D surface alignment is a real problem that aims to align two surfaces. Figure 8

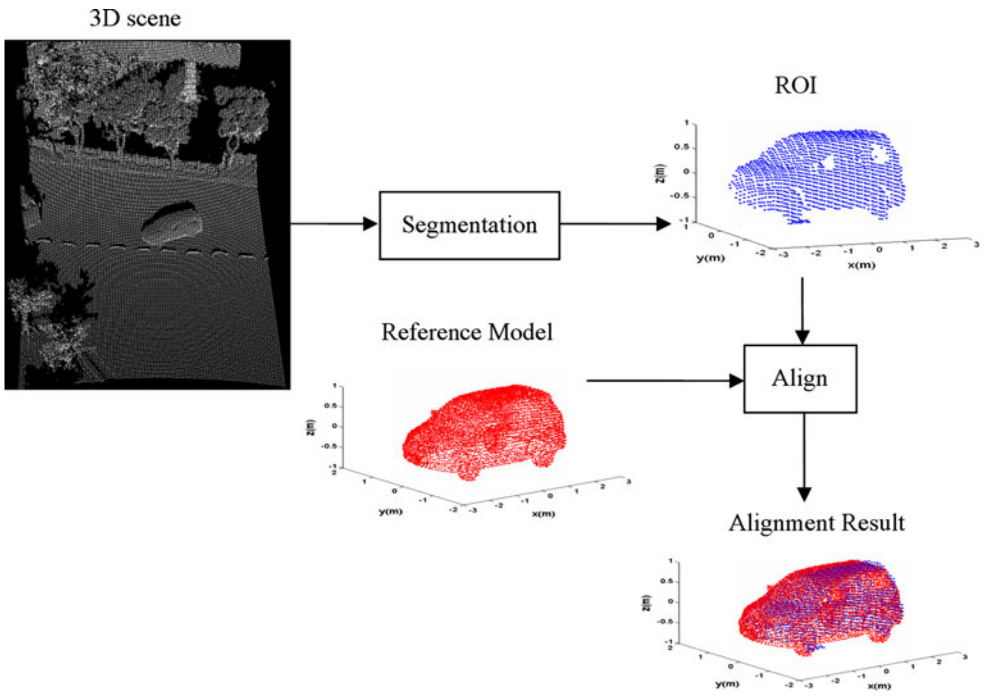

Fig. 8 Procedure of a 3D surface alignment task

illustrates the procedure of a 3D surface alignment task. From this figure, the 3D scene is scanned by a 3D imaging laser scanner where the size of the scanned scene is 256 × 256 with 20° field of view. Each pixel in the range image reflects a range data that indicates a distance from the sensed point to the scanner. In other words, the range data can be considered as a 3D point with respect to the scanner. Thus, the scanner can be a center of a coordinate system to represent each sensed range data. To this end, the 3D position of each pixel is created by transforming range data to Cartesian coordinate. The region of interest (ROI) is extracted by using the segmentation algorithm described in [43]. The reference model is a target 3D surface that the ROI wants to align with. Thus, the purpose of the 3D surface alignment task is to align the ROI with the reference model.

The problem of 3D surface alignment has been implemented by several methods [44–48]. Among them, a coarse-to-fine technique is a useful way for performing 3D surface alignment [44, 45]. Coarse alignment provides an approximate transformation for aligning two surfaces. Such alignment must be efficient and accurate. Fine alignment takes the initial gauss of transformation given by coarse alignment as a starting point to iteratively minimize the distance between the input surface and the destination surface. Thus, this study utilizes a TNFN-based coarse-to-fine method to perform 3D surface alignment.

Regarding coarse alignment of 3D surface, this study adopts a TNFN-based coarse alignment approach. Such approach captures VFH of multi-views of a 3D object as the input of a TNFN. The desired output of the TNFN is the corresponding pose of the captured feature. Thus, the desired output and the feature input can be performed for using DMHCCA to train a TNFN. Once the TNFN has been trained, input of the VFH of an arbitrary view of an object into the trained TNFN can yield an estimated pose. Then, we can utilize the estimated output pose to recover the input point clouds to coarsely align with the reference model.

Regarding fine alignment of 3D surface, similar to the neural network method (NNM) [45], a TNFN-based fine alignment approach is used to combine the DMHCCA trained TNFN-based surface modeling with the downhill simplex optimization method to iteratively reduce the distance from the input 3D surface to the reference surface.

To examine the alignment accuracy, 2000 synthesized point cloud sets are generated randomly within the range described in Table 5. For training the TNFN, 70 % of point clouds (1,400) are prepared for training data set and the remaining 30 % of point clouds (600) are prepared for testing data set. The initial parameters used by DMHCCA for the TNFN training are defined in Table 2.

Table 5 Range of 3D rigid transformation parameters

3D rigid transformation parameter	The range of affine transformation parameter
ϕ (degree), for roll	[−10, 10]
φ (degree), for yaw	[−90, 90]
θ (degree), for pitch	[0, 90]
$x(m)$	[−0.2, 0.2]
$y(m)$	[−0.2, 0.2]
$z(m)$	[−0.2, 0.2]

For setting the parameters of NNM, according to [45], a 2-layer neural network is used to model the vertebral surface model where the first layer has 20 nodes and the second layer has 10 nodes. In this example, by practical experimentations, the first layer setting for 30 nodes and the second layer setting for 20 nodes would have good results for modeling the surface of the reference model. In addition, the back-propagation algorithm is used for training the neural network, and the training process stops as the error between the output of the neural network and the desire distance value is less than 0.001 or the iterations reach 1000. Thus, NNM adopts the above parameters to train a neural network to model the reference surface.

Since the execution time and alignment accuracy are two major issues for a surface alignment system, we take them as the evaluation conditions to examine the propose alignment system.

4.2.1 Alignment accuracy

To evaluate the alignment accuracy, we compare the proposed method with NNM [45] and iterative closest point (ICP) [48]. About the stopping criterion, to compare all alignment methods, this paper sets the same criterion for each alignment method. To this end, the alignment procedure of each method is terminated when the number of iteration reaches 100 or the alignment error is less than 0.0005. Therefore, based on the 600 testing sets of point clouds, the alignment error is listed in Table 6 where RMSE indicates the root mean square error. From this table, the proposed method exhibits the lowest coarse and fine alignment error than other systems. Figure 9a–c presents a real alignment example (ROI is extracted by Fig. 8) of the proposed TNFN-based method, NNM, and ICP where the blue and red point clouds represent the testing and reference model data, respectively. From this figure, the fine alignment error of the proposed method, NNM, and ICP are 0.0558, 0.1121, and 0.0569 m, respectively. This result indicates that the proposed TNFN-based method can achieve high accuracy in real 3D point cloud data. Furthermore, regarding the alignment speed, the execution time of the proposed system, NNM, and ICP are 1.71, 2.13, and 7.93 s, respectively. Therefore, the proposed method demonstrates the higher alignment speed than NNM and ICP.

Table 6 Results of alignment accuracy and execution time

Method	Average RMSE (m)	Average execution Time (s)
TNFN-based coarse-to-fine alignment	0.0651	3.19
NNM	0.1357	4.21
ICP	0.0667	46.26

4.2.2 Alignment speed

In consideration of alignment speed, we calculate the average execution time of aligning 600 testing sets of point clouds. The results of the alignment speed are also listed in Table 6. As shown in this table, the execution time of the

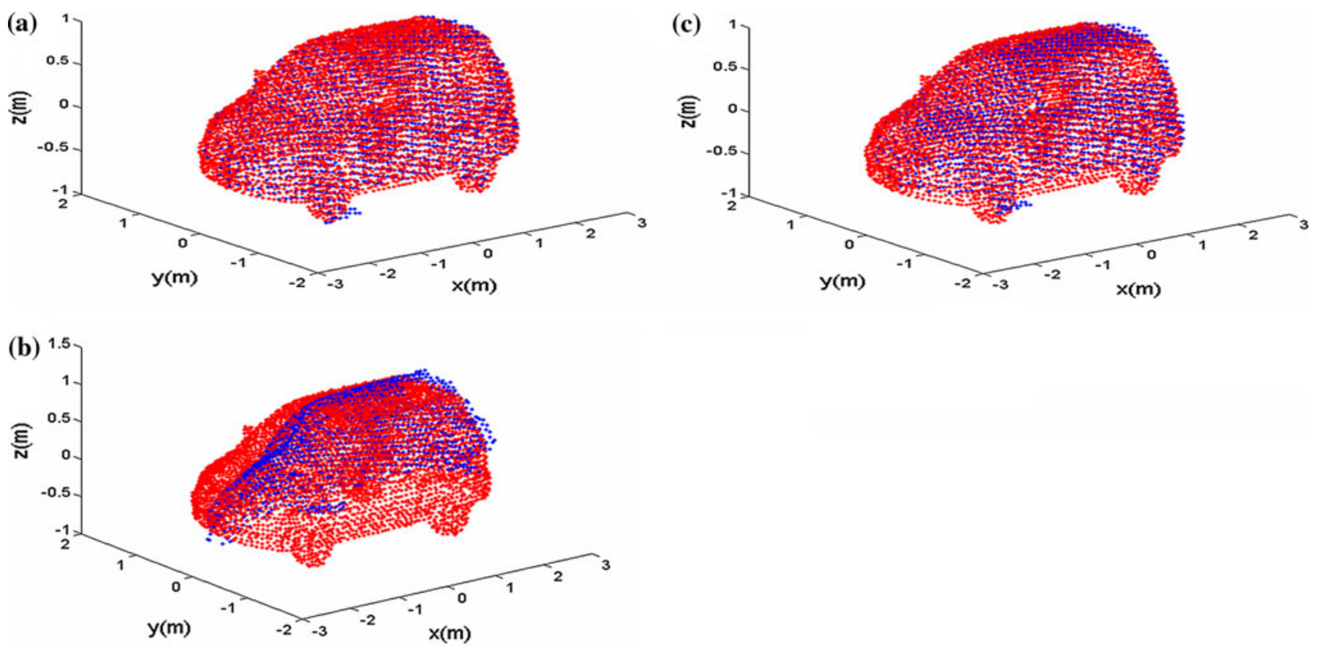

Fig. 9 Alignment results **a** proposed TNFN-based method, **b** NNM, **c** ICP

proposed TNFN-based method is shorter than those from NNM and ICP.

5 Conclusion

In this paper, DMHCCA is proposed for designing TNFNs. The proposed DMHCCA involves two-level evolutions: NULE and NWLE. NULE combines DELA and RLS to not only choose the group of fuzzy rules systematically but also increase the rate of convergence. NWLE proposed VAC and VAM to enable the mating and mutating of the variable length of chromosomes. The mutual evolution of NULE and NWLE would make the neurons and structure of network to be evolved locally and globally, respectively. According to the simulation results on benchmark, the proposed DMHCCA exhibits better performance than other learning methods. Besides, a 3D surface alignment task is utilized to examine the learning ability of DMHCCA. The experimental results show that the DMHCCA trained TNFN-based alignment method is superior to other alignment systems.

Although DMHCCA can get better results in comparison with other learning algorithms, it still has a limitation. Specifically, the number of hierarchical level is only two to execute the training of structure and parameters of neural fuzzy networks. As the application problem become more complex, there is a need to increase the hierarchical level to match the complex problem. Thus, in the future work, the multi-hierarchical level is taken into consideration of further investigation of how to cooperate these hierarchical levels to adapt the model to a complex problem.

Acknowledgments This work was funded by Contract NSC 99-2623-E-009-006-D from the National Science Council, Taiwan, R.O.C.

Appendix

The following contents list the abbreviations used in this paper.

Data mining based hierarchical cooperative coevolutionary algorithm, DMHCCA; neuro-level evolution, NULE; network-level evolution, NWLE; data mining–based evolutionary learning algorithm, DELA; three-dimensional, 3D; back-propagation, BP; genetic algorithm, GA; enforced subpopulations, ESP; multi-groups cooperation-based symbiotic evolution, MGCSE; TSK-type neuro-fuzzy network, TNFN; regularized least square, RLS; variable antecedent-part crossover, VAC; variable antecedent-part mutation, VAM; hierarchical enforced subpopulations, HESP; self-adaptive method, SAM; data mining–based selection method, DMSM; elite-based reproduction strategy, ERS; traditional symbiotic evolution, TSE; traditional genetic algorithm, TGA; region of interest, ROI; neural network method, NNM; iterative closest point, ICP.

References

1. Lin CT, Lee CSG (1996) Neural fuzzy systems: a neuro-fuzzy synergism to intelligent system. Prentice-Hall, Englewood Cliffs, NJ
2. Towell GG, Shavlik JW (1993) Extracting refined rules from knowledge-based neural networks. Mach Learn 13:71–101
3. Lin CJ, Lin CT (1997) An ART-based fuzzy adaptive learning control network. IEEE Trans Fuzzy Syst 5(4):477–496
4. Wang LX, Mendel JM (1992) Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 22(6): 1414–1427
5. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern 15:116–132
6. Juang CF, Lin CT (1998) An on-line self-constructing neural fuzzy inference network and its applications. IEEE Trans Fuzzy Syst 6(1):12–31
7. Riemiller M, Braun H (1993) A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: Proceeding of IEEE international conference on neural networks, pp 586–591
8. Lin FJ, Lin CH, Shen PH (2001) Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. IEEE Trans Fuzzy Syst 9(5):751–759
9. Takagi H, Suzuki N, Koda T, Kojima Y (1992) Neural networks designed on approximated reasoning architecture and their application. IEEE Trans Neural Netw 3(5):752–759
10. Mizutani E, Jang J-SR (1995) Coactive neural fuzzy modeling. In: Proceeding of IEEE international conference on neural networks, pp 760–765
11. Lin C-J, Chin C-C (2004) Prediction and identification using wavelet-based recurrent fuzzy neural networks. IEEE Trans Syst Man Cybern Part B 34(5):2144–2154
12. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1:4–27
13. Juang CF, Lin CT (1999) A recurrent self-organizing neural fuzzy inference network. IEEE Trans Neural Netw 10(4):828–845
14. Mastorocostas PA, Theocharis JB (2002) A recurrent fuzzy-neural model for dynamic system identification. IEEE Trans Syst Man Cybern 32(2):176–190
15. Goldberg DE (1989) Genetic algorithms in search optimization and machine learning. Addison-Wesley, Reading, MA
16. Koza JK (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, MA
17. Fogel LJ (1994) Evolutionary programming in perspective: the top-down view. In: Zurada JM, Marks RJ II, Goldberg C (eds) Computational intelligence: imitating life. IEEE Press, Piscataway, NJ
18. Li M, Wang Z (2009) A hybrid coevolutionary algorithm for designing fuzzy classifiers. Inf Sci 179(12):1970–1983
19. Karr CL (1991) Design of an adaptive fuzzy logic controller using a genetic algorithm. In: Proceeding of the 4th international conference on genetic algorithms, pp 450–457
20. Lin CT, Jou CP (2000) GA-based fuzzy reinforcement learning for control of a magnetic bearing system. IEEE Trans Syst Man Cybern Part B 30(2):276–289

21. Juang CF, Lin JY, Lin CT (2000) Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. IEEE Trans Syst Man Cybern Part B 30(2):290–302
22. Bandyopadhyay S, Murthy CA, Pal SK (2000) VGA-classifier: design and applications, IEEE Trans. Syst Man Cybern Part B 30(6):890–895
23. Carse B, Fogarty TC, Munro A (1996) Evolving fuzzy rule based controllers using genetic algorithms. Fuzzy Sets Syst 80(3): 273–293
24. Chen CH, Lin CJ, Lin CT (2009) Using an efficient immune symbiotic evolution learning for compensatory neuro-fuzzy controller. IEEE Trans Fuzzy Syst 17(3):668–682
25. Lin CJ, Chen CH, Lin CT (2011) An efficient evolutionary algorithm for fuzzy inference systems. Evol Syst 2(2):83–99
26. Gomez FJ (2003) Robust non-linear control through neuroevolution, Ph. D. Disseration, The University of Texas at Austin
27. Gomez F, Schmidhuber J (2005) Co-evolving recurrent neurons learn deep memory POMDPs. In: Proceeding of conference on genetic and evolutionary computation, pp 491–498
28. Lin CJ, Hsu YC (2007) Reinforcement hybrid evolutionary learning for recurrent wavelet-based neuro-fuzzy systems. IEEE Trans Fuzzy Syst 15(4):729–745
29. Hsu YC, Lin SF, Cheng YC (2010) Multi groups cooperation based symbiotic evolution for TSK-type neuro-fuzzy systems design. Exp Syst Appl 37(7):5320–5330
30. Lee JT, Wu HW, Lee TY, Liu YH, Chen KT (2009) Mining closed patterns in multi-sequence time-series database. Data Knowl Eng 68:1071–1090
31. Tanbeer SK, Ahmed CF, Jeong BS (2009) Parallel and distributed algorithm for frequent pattern mining in large database. IETE Tech Rev 26:55–65
32. Agrawal R, Srikant R (1994) Fast algorithm for mining association rules. In: Proceeding of the international conference on VLDB, pp 487–499
33. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceeding of ACM-SIGMOD, Dallas, Tx, pp 1–12
34. Wu YT, An YJ, Geller J, Wu YT (2006) A data mining based genetic algorithm. In: Proceeding of IEEE workshop SEUS-WCCIA, pp 27–28
35. Shankar S, Purusothaman T (2009) Utility sentient frequent itemset mining and association rule mining: a literature survey and comparative study. Int J Soft Comput Appl 4:81–95
36. Sugeno M, Tanaka K (1991) Successive identification of a fuzzy model and its applications to prediction of a complex system. Fuzzy Sets Syst 42(3):315–334
37. Lin SF, Cheng YC (2010) Two-strategy reinforcement evolutionary algorithm using data-mining based crossover strategy with TSK-type fuzzy controllers. Int J Innovat Comput Control 6(9):3863–3885
38. Cordon O, Herrera F, Hoffmann F, Magdalena L (2001) 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
39. Cox E (2005) Fuzzy modeling and genetic algorithms for data mining and exploration, 1st edn. Morgan Kaufman Publications, San Francisco, USA
40. De Jong KA (1975) Analysis of the behavior of a class of genetic adaptive systems, Ph. D. Dissertation, Dep. Computer and Communication Sciences, Univ. Michigan, Ann Arbor, MI
41. Cowder RS (1990) Predicting the mackey-glass time series with cascade-correlation learning. In: Proceedings of the 1990 connectionist models summer school, pp. 117–123
42. Moriarty DE, Miikkulainen R (1996) Efficient reinforcement learning through symbiotic evolution. Mach Learn 22:11–32
43. Rabbani T, van den Heuvel FA, Vosselmann G (2006) Segmentation of point clouds using smoothness constraint. Proc ISPRS 35:248–253
44. Liu H, Yan J, Zhang D (2006) Three-dimensional surface registration: a neural network strategy. Neurocomputing 70:597–602
45. Zhang J, Ge Y, Ong SH, Chui CK, Teoh SH, Yan CH (2008) Rapid surface registration of 3D volumes using a neural network approach. Image Vis Comput 26:201–210
46. Chetverikov D, Stepanov D, Kresk P (2005) Robust Euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm. Image Vis Comput 23:299–309
47. Liu YH (2004) Improving ICP with easy implementation for free-form surface matching. Pattern Recogn 37:211–226
48. Besl P, Mckay N (1992) A method for registration of 3-D shapes. IEEE Trans Pattern Anal Mach Intell 14(2):239–256