Neurocomputing 149 (2015) 1003–1013

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Novel adaptive hybrid rule network based on TS fuzzy rules using an improved quantum-behaved particle swarm optimization

Lin Lin *, Feng Guo, Xiaolong Xie, Bin Luo

School of Mechatronics Engineering, Harbin Institute of Technology, Harbin, China

ARTICLE INFO

Article history:
Received 7 November 2013
Received in revised form
11 May 2014
Accepted 13 July 2014
Communicated by A.M. Alimi
Available online 4 August 2014

Keywords:
Adaptive hybrid rule network
Opinion leader-based quantum-behaved particle swarm optimization
Composed particle
Chaotic time series prediction

ABSTRACT

A novel adaptive hybrid rule network (AHRN) based on Takagi–Sugeno (TS) fuzzy rules is proposed to resolve chaotic system prediction problems. This model automatically adjusts its structure and dynamically establishes rule sets (apart from statically) to adapt in learning new samples. For the learning process, the opinion leader-based quantum-behaved particle swarm optimization (OLB-QPSO) algorithm is proposed. This algorithm uses composed particles generated according to AHRN and emphasizes the importance of the composed particle with the highest fitness based on a social communication law. To improve the chance of finding the best global solution, the movement of the composed particle is affected by the subparticles as inner factors and by the swarm as outer factor.

Three chaotic time series experiments are performed to validate the proposed method. Results show that AHRN that uses the OLB-QPSO with composed particles can effectively provide the appropriate rules to search for solutions in a wide space and significantly improve the probability of obtaining the optimal global solution.

1. Introduction

Fuzzy theories have been widely applied in many fields, such as computer-aided design [1–4], system identification [5], automatic control [6,7], pattern recognition [6], data mining [8], and prediction [9], because of the ability to handle the complex problems with strong nonlinearity or high degree of uncertainty. Therefore, the fuzzy model has been proposed and proven as powerful in complex system modeling [10]. Two kinds of fuzzy models currently exist: the Mamdani [11] and the Takagi–Sugeno (TS) fuzzy models [5]. To date, the use of the TS fuzzy model to predict chaotic time series problems has gained increasing attention, and research on building the TS fuzzy rule has become a key point in practical applications.

Formulating fundamental TS rules is based on structure and parameter identification. The structure used to determine the antecedent and the consequent parts of a rule can be identified through heuristics [5], neural networks [12], fuzzy clustering methods [13], and so on. Meanwhile, the existing parameters in the rule can be identified through the least-square method [5], gradient descent [14], genetic algorithm (GA) [15], and so on. The structure and parameters, as well as the number of rules can influence prediction. An improper number of rules can neither improve the accuracy nor clear the explanation; thus, classic methods, such as increasing or merging clusters and mountain or subtractive clustering, are used. Simultaneously, new methods have also been proposed to optimize the model and improve the prediction performance, such as the TS-group method of data handling algorithm [16], the incremental smooth support vector regression algorithm [17], and the habitually linear evolving TS fuzzy model [18]. However, these methods still cannot guarantee that the accurate partition of the input samples and the correct number of rules are employed to solve chaotic time series prediction problems. Therefore, a novel rule-based fuzzy model, called adaptive hybrid rule network (AHRN) is proposed in this study to establish the solution model. AHRN is mainly established by nodes that represent fuzzy subspaces (i.e., different clusters of input data) and that are linked to others to generate rules. In the learning process, AHRN can automatically partition input samples according to their characteristics; thus, the structure can be updated adaptively. Meanwhile, the dynamic rule selection mechanism (DRSM) is used to establish the rule set according to the number of rules (RL) and rule similarity (RS). Through this mechanism, AHRN can build the most suitable set of rules. However, implementing AHRN is another key issue. Experience proves that an evolution algorithm is a better choice after several experiments that implement different kinds of algorithms have been performed in this research.

An evolutionary algorithm is a self-organizing and adaptive artificial intelligence technology that solves problems by

* Corresponding author.
E-mail address: waiwaiyl@163.com (L. Lin).

http://dx.doi.org/10.1016/j.neucom.2014.07.033
0925-2312/

simulating a biological evolution process and mechanism. GA [19], ant colony optimization [20], and particle swarm optimization (PSO) [21,22] are typical examples of evolutionary algorithm. PSO has better computational efficiency, i.e., it requires less memory space and lesser CPU speed, and less number of parameters to adjust [23]. As research develops further, new algorithms based on standard PSO have been proposed, such as elite PSO with mutation [24], group decision PSO [25], multi-grouped PSO [26], optimization algorithm based on the TS fuzzy model of self-adaptive disturbed PSO and neural network [27], and quantum-behaved PSO (QPSO) [28]. QPSO has lesser parameters to control and better search capability than standard PSO [23]. QPSO has been increasingly used in chaotic prediction problems [29,30]; hence, new algorithms based on standard QPSO have been proposed. The stochastic coefficient is one of the key factors that influences the particle movement in classic QPSO; thus, Coelho [31] generated random numbers by using Gaussian distribution sequences with zero mean and unit variance for the stochastic coefficient. This method may allow particles to move away from the current point and escape from local minima. The mean best position (mbest) is another key factor; thus, Xi [32] proposed the weighted QPSO algorithm. This algorithm modifies the calculation method for the mbest by assigning each particle with a weight coefficient that linearly decreases with a particle's rank; this method promises considerable influence on the movement of the particle with high fitness. To further improve performance, other valuable studies have been done. Sun [33] incorporated the improved heuristic strategies into QPSO; and the proposed method does not require using penalty functions. Moreover, it explores the optimum solution at a low computational effort. Pan [34] introduced the chaos theory into QPSO; and the proposed method uses a logistic map to generate a set of chaotic offsets and produces multiple positions around every local optimal position of the particle, and thus, convergence accuracy is better than that in typical QPSO. Sun [35] proposed modified QPSO, which substitutes the global best position (gbest) by a personal best position (pbest) of a randomly selected particle, thus exhibiting stronger global search capability than QPSO and PSO. In the present study, an improved algorithm called opinion leader-based QPSO (OLB-QPSO) is proposed with a new type of particle. This method modifies the calculation method for the mbest and uses the composed particles with multiple structures.

This study investigates (1) how the fuzzy map of data can be identified based on the correct data partition, (2) how the explanation of fuzzy rules can be improved to enhance robustness, and (3) how the local optimal solution can be avoided.

This paper is organized as follows. In Section 2, the construction of AHRN, including structure and DRSM analyses, is described. In Section 3, several key factors required to implement AHRN are discussed in detail. These factors include QPSO, OLB-QPSO, and the compose particle. In Section 4, three experiments are presented. Box–Jenkins gas furnace data and the Mackey–Glass chaotic time series are used to validate prediction capability, whereas exhausted gas temperature (EGT) samples are employed to demonstrate the performance of AHRN that uses OLB-QPSO with composed particles in actual engineering. Finally, the conclusions are provided in Section 5.

2. Construction of AHRN

The samples used in AHRN are as follows:

$$\begin{cases} P=[X_1X_2\ldots X_iY] & (i=1,2,\ldots,n) \\ X_i=[x_{i1}x_{i2}\ldots x_{ij}]^T & (i=1,2,\ldots,n;j=1,2,\ldots,m) \\ Y=[y_1y_2\ldots y_j] & (j=1,2,\ldots,m) \end{cases}, \quad (1)$$

where n is the dimension of the input vector of a sample, m is the capacity of the samples, x_{ij} is the ith variable in the input vector of the jth sample, and y_j is the output vector of the jth sample.

The output of a sample in AHRN can be obtained by the weighted sum of different TS fuzzy rules because the TS fuzzy rules model is suitable to for solving nonlinear prediction problems. The TS fuzzy rule is represented as follows:

$$R_t: \text{If } x_1 \text{ is } A_1^t \text{ and } x_2 \text{ is } A_2^t \text{ and } \cdots \text{ and } x_i \text{ is } A_i^t$$
$$\text{Then } y_t = a_1^t x_1 + \cdots + a_i^t x_i + b^t, \quad (2)$$

where R_t is the tth fuzzy rule, A_i^t is the fuzzy set of x_i in rule R_t, y_t is the output of R_t, a_i^t is the real coefficient corresponding to x_i, and b^t is a compensation value. Therefore, the process of constructing rules is a key issue in this study. This issue is related to the structure of AHRN and the rule selection mechanism.

2.1. Structure of AHRN

AHRN is a network model that consists of a network structure and DRSM. The network structure shows the topology structure of the network, whereas DRSM is used to build the fuzzy rules. The network structure contains many nodes linked in a certain direction. The typical AHRN structure is shown in Fig. 1.

In this figure, A_i^{ki} is the kith distribution interval of the ith dimension of the sample; $f_i^{ki}(x)$ is the membership function on interval A_i^{ki}; ω_i^{ki} is the weight used to adjust the membership degree calculated by $f_i^{ki}(x)$; a_i^{ki} is a real coefficient corresponding to interval A_i^{ki}, which is similar to that of a_i^n in Formula (2); and B is a compensation matrix, the elements of which are equivalent to b^t in Formula (2).

Fig. 1 shows an interval in the node, which is used to establish different fuzzy intervals. Two kinds of intervals are included in AHRN. The first is real interval (RI), which represents a distribution interval for a dimension of the current sample. The second is computing interval (CI), which is generated based on RI and is used to build the membership degree. The relationship between RIs and CIs is shown in Fig. 2.

In this figure, (a) is the minimal value of certain dimensions of a sample, (b) is the maximal value, (c) is the middle value between (a) and (b), (e) is the middle value between (a) and (c), (f) is the

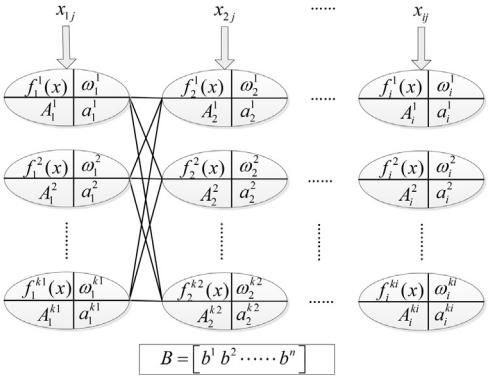


Fig. 1. Network structure of AHRN.

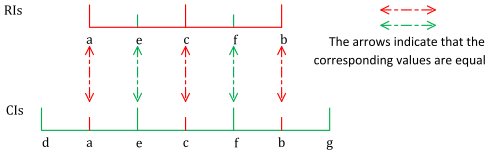


Fig. 2. Relationship between RIs and CIs.

middle value between (c) and (b), (a) is also the middle value between (d) and (e), (b) is also the middle value between (f) and (g), and the long vertical lines are the interval lines. Therefore, RIs are (a) and (c) and (c) and (b), whereas CIs are (d) and (e), (e) and (f) and (f) and (g). That is, an RI is divided into two parts, wherein each part is used to form a neighbor CI, as shown in Fig. 2.

Any kind of distribution can be approximated by using the linear weighted sum obtained by the finite number of normal distributions. Moreover, the diversity of sample distribution should be considered. Therefore, the sample distribution for a CI should be obtained by the linear weighted sum based on multiple normal distributions. In addition, sample distribution in AHRN should be obtained by the normal distribution of the current CI and its adjacent CIs. Therefore, the membership function $f_i^k(x)$ adopts the normal distribution model, which is represented as follows:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right). \tag{3}$$

Three normal distributions are typically used to obtain the membership degree for a CI to reduce computation, as shown as Formula (4). If a corresponding CI does not exist, then two CIs are available at least.

$$S_i^{ki}=\omega_i^{ki-1}f_i^{ki-1}(x_i)+\omega_i^{ki}f_i^{ki}(x_i)+\omega_i^{ki+1}f_i^{ki+1}(x_i), \tag{4}$$

where x_i is the ith dimension of the sample; S_i^{ki} is the final membership degree, called available membership degree (AMD) on the kith CI; $f_i^{ki}(x_i)$ is the normal distribution model on the kith CI; ω_i^{ki} is the weight on the kith CI.

As shown in Fig. 1, weight ω_i^{ki} is used to adjust the influence degree of adjacent CIs for the membership degree of the current CI. In the learning process, the number of sample points with different colors falling into a particular CI is different; thus, the influence degree of different CIs varies, as shown as Fig. 3.

As shown in Fig. 3, the number of samples for a dimension that falls into the left CI (interval (a) and (c)) is more than the other two (intervals (b) and (d) and (c) and (e)). That is, the left CI is more important than the others in obtaining the membership degree for the current CI. Therefore, the weight of the left CI is set by a large value.

The real coefficient a_i^{ki} is used to build a rule. AHRN can select different real coefficients. By selecting different ki, the rules built in AHRN may cover the entire rule space, thus improving the diversity of rules.

Initial AHRN can be constructed according to the maximum number of RIs. Thus, the maximum number of CIs can be obtained according to Fig. 2. In the learning process, the actual number of CIs is updated adaptively by adjusting the center of the CI. The AHRN structure is equivalent to the adjustment of the distribution interval for the samples. Thus, in the most situations, $k1$, $k2$,…, and ki in Fig. 1 do not have to be equal to one another because they are updated adaptively.

To provide enough compensation values, the amount of elements in B is equal to the maximum RLs in the rule set. Thus, the compensation value according to the index n should be provided. For example, the tth rule uses the tth element.

This structure has the following benefits. (1) The range and number of CIs can be adjusted automatically; thus, the best suitable distribution can be determined. (2) The distribution adjustment for all dimensions of the sample results in the optimal structure. (3) The optimal rule can be set to avoid the disadvantage of the fixed rule set established by experts.

2.2. DRSM of AHRN

The pattern of the rule used in AHRN is similar to that in the typical TS fuzzy model. In the typical TS fuzzy model, the rules are determined by the knowledge of experts or by certain methods for future samples. However, the model may be unsuitable for all samples. AHRN can establish the optimal rule set for all samples.

AHRN adopts DRSM to establish the rule set. RL and RS are the two important parameters that can be obtained by the algorithm described in Section 3. RL and RS can be used to guide nodes to build different rules. The optimal RL can prevent redundant or inadequate rules, whereas the optimal RS can prevent excessive homogeneity and heterogeneity. After RL and RS are adjusted, the process of establishing the rule set for a sample is described as follows:

Step 1: Calculate the AMDs for different CIs of each variable in the input vector. Organize the AMDs of each variable as the membership degree ordered vector (MDOV) of each variable. MDOV is the vector that is composed of membership degrees in descending order from different CIs of the variable in the input vector.

Step 2: Select the CIs and the corresponding real coefficients that are contained in the nodes, the AMD of which is initially positioned in each MDOV (i.e., the AMD is the maximal one) to build the first rule.

Step 3: Build the tth ($1<t<\mathrm{RL}+1$) rule based on the previous rule, which includes two stages. The first stage involves selecting the nodes with AMDs that are positioned at tth in each MDOV. The second stage involves replacing unsuitable nodes with suitable ones for each dimension according to the RS, as described in Formula (5):

$$A_i^m \vee A_i^n=\begin{cases} A_i^m, (S_i^m-S_i^n)/S_i^m > RS \\ A_i^n, (S_i^m-S_i^n)/S_i^m \le RS \end{cases}, \tag{5}$$

where $\vee$ is the selection operator; A_i^m and A_i^n are the CIs of the sample, S_i^m and S_i^n are the AMDs on the CIs.

An example is provided to illustrate DRMS clearly. The hypotheses are formulated. (1) The current sample is $[x_{1j}\ x_{2j}\ x_{3j}\ x_{4j}]$, with a dimension of four. (2) Two rules must be built; (3) b^1 and b^2 are the compensation values for the two rules. (4) The corresponding relationships are shown in Table 1, where j is the sample index, and the other symbol is shown in Fig. 1.

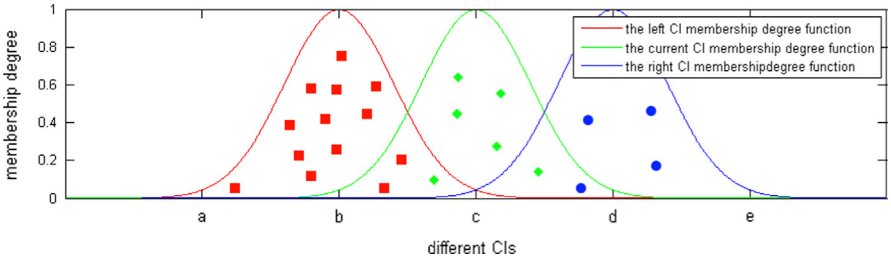


Fig. 3. Sample distribution.

Table 1
Corresponding relationships.

	Sample variable	Corresponding MDOV	Corresponding CIs
1	x_{1j}	$[\omega_1^1 f_1^1(x_{1j})\ \omega_1^2 f_1^2(x_{1j})\ \omega_1^3 f_1^3(x_{1j})]$	A_1^1, A_1^2, and A_1^3
2	x_{2j}	$[\omega_2^3 f_2^3(x_{2j})\ \omega_2^2 f_2^2(x_{2j})\ \omega_2^1 f_2^1(x_{2j})]$	A_2^3, A_2^2, and A_2^1
3	x_{3j}	$[\omega_3^3 f_3^3(x_{3j})\ \omega_3^1 f_3^1(x_{3j})\ \omega_3^2 f_3^2(x_{3j})]$	A_3^3, A_3^1, and A_3^2
4	x_{4j}	$[\omega_4^2 f_4^2(x_{4j})\ \omega_4^1 f_4^1(x_{4j})\ \omega_4^3 f_4^3(x_{4j})]$	A_4^2, A_4^1, and A_4^3

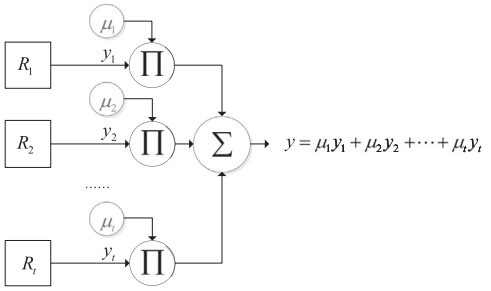


Fig. 4. Output of a sample.

According to Step 2, the first rule can be represented as follows:

$$R_1: \text{If } x_{1j} \text{ is } A_1^1 \text{ and } x_{2j} \text{ is } A_2^3 \text{ and } x_{3j} \text{ is } A_3^3 \text{ and } x_{4j} \text{ is } A_4^2$$
$$\text{Then } y_1 = a_1^1\omega_1^1 f_1^1(x_{1j}) + a_2^3\omega_2^3 f_2^3(x_{2j}) + a_3^3\omega_3^3 f_3^3(x_{3j}) + a_4^2\omega_4^2 f_4^2(x_{4j}) + b^1, \tag{6}$$

For the jth sample, when another relationship exists as Formula (7), then, the second rule is represented as Formula (8), where the symbol is shown in Formula (5).

$$\begin{cases} (S_1^1 - S_1^2)/S_1^1 \le RS \\ (S_2^3 - S_2^2)/S_2^3 > RS \\ (S_3^3 - S_3^1)/S_3^3 \le RS \\ (S_4^2 - S_4^1)/S_4^2 > RS \end{cases}, \tag{7}$$

$$R_2: \text{If } x_{1j} \text{ is } A_1^2 \text{ and } x_{2j} \text{ is } A_2^3 \text{ and } x_{3j} \text{ is } A_3^1 \text{ and } x_{4j} \text{ is } A_4^2$$
$$\text{Then } y_2 = a_1^2\omega_1^2 f_1^2(x_{1j}) + a_2^3\omega_2^3 f_2^3(x_{2j}) + a_3^1\omega_3^1 f_3^1(x_{3j}) + a_4^2\omega_4^2 f_4^2(x_{4j}) + b^2, \tag{8}$$

Step 3 can be repeated to build additional rules. After the rule set is established, the output of a sample can be obtained, as shown in Fig. 4.

In the figure, R_t is the tth rule; y_t is an output obtained by R_t; μ_t is the weight for y_t, which is obtained by Formula (9); and y is the final output for the sample.

$$\mu_t = \prod S, S \subset R_t, \tag{9}$$

where R_t is the tth rule, S is the AMD of the nodes that are building R_t.

To evaluate the performance of AHRN, least mean square, mean relative estimation error, and root-mean-square error (RSME) can be used to construct the fitness function based on Fig. 4.

3. Learning process of AHRN

AHRN provides an approximation model by using several algorithms for model learning. For example, evolutionary algorithms can be used, but swarm intelligence algorithms may be better choices. Among swarm intelligence algorithms, the OLB-QPSO with composed particles is selected for AHRN in this study. This algorithm is an improved QPSO.

3.1. QPSO algorithm

The basic concept of PSO is that the potential solution for every optimization problem is represented as a particle flying in the search space, wherein all particles taken as a swarm are optimized according to the corresponding fitness value, and each particle follows the optimal particle in the swarm to search for the best solution in the search space. The search model uses a velocity vector that consists of flying direction and distance. However, the optimal process is implemented by tracking movement. The velocity (including direction and magnitude) causes the particles to fly in a finite search space instead of in the entire search space, and thus, certain PSO algorithms cannot obtain the global optimal solution. QPSO has been proposed to avoid such problems.

In QPSO, quantum theory is applied in the searching process. QPSO states that the movement of a particle is affected by bound states generated from attractive potentials. The particle affected by a certain bound state can appear in any position in the search space with a certain probability; thus, the particle that represents a solution can reach the best position in the feasible search space instead of diverging to infinity.

Three import positions exist in QPSO: pbest, gbest, and mbest. pbest represents the best solution obtained by the particle in the history of movement, and can be regarded as the best individual knowledge. gbest represents the best solution obtained by all particles in the history of movement and can be regarded as the best social knowledge. mbest represents the mean level of all particles in the current iteration and can be regarded as the mainstream thought. In the learning process, the best social knowledge is the optimal solution at the current iteration, whereas the movement of the particle is affected by both best individual knowledge and mainstream thought.

Given that $x_i(t)$ represents the ith dimension of the particle at time t; $pbest_i$ represents the ith dimension of pbest; and $mbest_i$ represents the ith dimension of mbest. The particle moves according to the iterative Formula (10):

$$\begin{cases} x_i(t+1) = pbest_i + \beta \times |mbest_i - x_i(t)| \times \ln(1/u), & \text{if } k \ge 0.5 \\ x_i(t+1) = pbest_i - \beta \times |mbest_i - x_i(t)| \times \ln(1/u), & \text{if } k < 0.5 \end{cases}, \tag{10}$$

where β represents the contraction–expansion coefficient, and u and k in range [0, 1] are the values generated by the uniform probability distribution functions. When n represents the capacity of the swarm, the *mbest* can be obtained by Formula (11):

$$mbest_i = \frac{1}{n}\sum_{i=1}^{n} pbest_i(t). \tag{11}$$

3.2. OLB-QPSO

As discussed in the introduction, many improved algorithms based on QPSO have been introduced. To use the advantage of the fuzzy rule, valuable studies have been conducted. For example, Zhai [36] proposed a non-singleton interval type-2 fuzzy logic system for mixed Gaussian and impulse noise removal, and quantitatively and visually obtained excellent results because the system was designed based on QPSO. In the current study, OLB-QPSO is proposed based on the communication effects study in the mass communication field. In this field, the opinion leader is the individual who typically provides information, points of view, and advices to others, and thus, can influence the decisions of others. Before information reaches the masses, opinion leaders initially obtain and interpret information based on their knowledge, and then allow the information to flow to others. Therefore, the opinion leader can influence the understanding of others

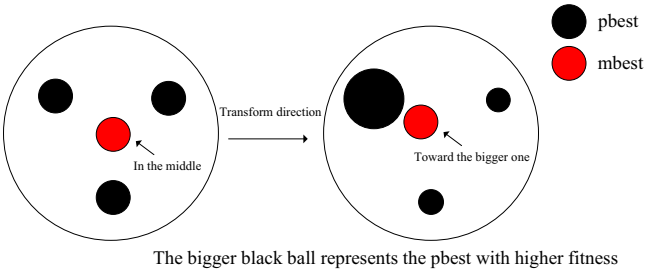


Fig. 5. Evolution of the mbest.

regarding the information. In QPSO, the aforementioned phenomenon shows that the construction of mainstream thought is relayed to the particle with the highest fitness compared with the others. The mbest in OLB-QPSO can be obtained by

$$mbest_i = \alpha \times pbest_i^{ol}(t) + (1-\alpha)/(n-1) \times \left(\sum_{c=1}^{n} pbest_i^{c}(t) - pbest_i^{ol}(t) \right), \quad (12)$$

where $pbest_i^c(t)$ is the pbest of the cth particle, $pbest_i^{ol}(t)$ is the pbest acting as an opinion leader, α is the weight of the opinion leader, and n is the capacity of the swarm. Experience has shown that an improved result is obtained when α is set to a value within the range of [0.3, 0.5].

Supposing that the capacity of the swarm is three, the evolution of the mbest from Formulas (11) to (12) is shown in Fig. 5. Therefore, the mbest can approach the suitable pbest.

In summary, the proposed method for generating the mbest is derived from the weighted QPSO [32], but emphasizes the importance of the particle with the highest fitness, and is in accordance with the social communication law on mainstream thought. Therefore, this method is suitable for generating the mbest.

3.3. Structure of the composed particle

In the classic QPSO model, particle movement is determined by individual knowledge and mainstream thought. Individual knowledge is determined by the fitness function and the variable taken by the particle. The fitness function is determined by the actual problem, whereas the variable organization in the particle (called the particle structure) is typically a 1D array, as shown in Fig. 6. Therefore, only one attractor is used to influence the appearance position of the current particle. Moreover, the flexibility of motivation is not fully reflected. To avoid such problems and improve the performance of searching for the global optimal solution, a composed particle with a multiple structure is proposed.

In this study, two kinds of parameters are included in AHRN. The first is used to determine the model structure (i.e., structure parameters), and the second is used to establish the rule set (i.e., rule parameters). Both parameters may affect appearance position. In addition, auxiliary information is needed, such as fitness value and the structure of the currently obtained model. Therefore, three subparticles exist in the composed particle, as shown in Fig. 7. The first is the structure subparticle (SSP), which considers the structure parameters. The second is the rule subparticle (RSP), which considers the rule parameters. The third is the auxiliary information subparticle (AISP), which considers auxiliary information. The details of each subparticle in the learning process are given in the subsequent paragraphs.

SSP and RSP evolve respectively; thus, two attractors exist. The first attractor, called structure attractor (SA), is generated according to all SSPs in the swarm. The second attractor, called the rule attractor (RA), is generated according to all RSPs in the swarm. SA

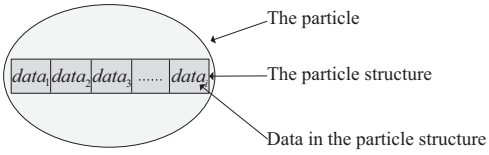


Fig. 6. Classic particle structure.

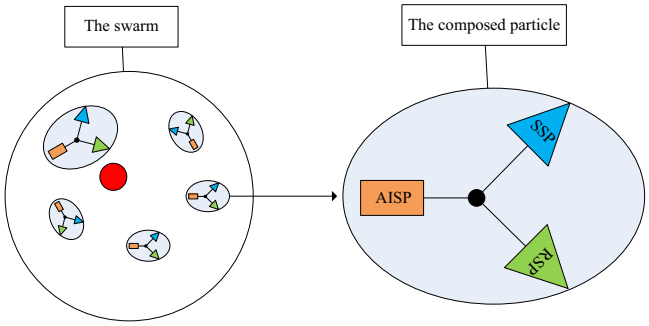


Fig. 7. The swarm and one of the composed particles.

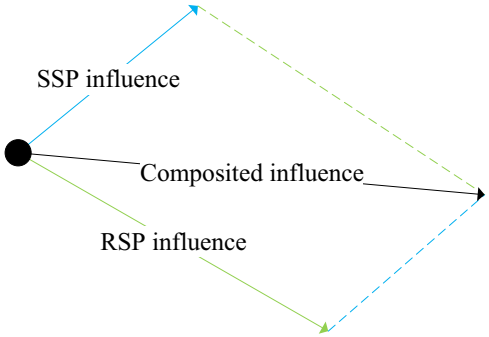


Fig. 8. Composited influence.

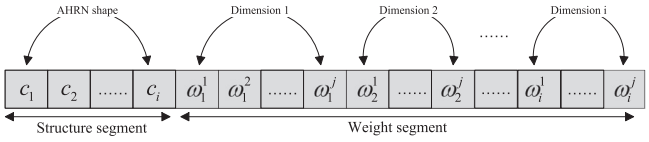


Fig. 9. Array in SSP.

is used to optimize the structure of AHRN, whereas RA is used to optimize the rule set of AHRN. Consequently, the appearance position of a composed particle is composited influence, as illustrated is shown in Fig. 8.

As shown in Fig. 8, the composed particle jumps from one position to another under composited influence. Composited influence is formed according to the parallelogram law, and the two edges are SSP and RSP influences. However, updating the structure in AHRN can impose on the establishment of a rule set. That is, SSP influence can impose on RSP influence. This complicated relationship can force the composed particle to search for the gbest with a high possibility, thus increasing the chance of obtaining the optimal solution. The contents taken by the three subparticles in the learning process of AHRN are introduced as follows.

SSP considers the structure parameters, including (1) the numbers of nodes for each dimension of the sample, wherein all numbers are used to construct the structure segment; and (2) every weight of possible nodes, wherein all weights are used to construct the weight segment. Both segments comprise the array in SSP, as shown in Fig. 9, where c_i is the number of nodes of

the ith dimension, and ω_i^j is the weight of the jth node of the ith dimension. In the learning process, AHRN selects the front c_i weights of dimension i as the available weights of the ith dimension, and then uses all available weights of different dimensions to establish the rule set.

RSP considers the rule parameters, including (1) RL and RS, which are used to construct the instructing segment; (2) every coefficient of possible nodes, wherein all coefficients are used to construct the coefficient segment; and (3) compensation values, which are used to construct the compensation segment. All three segments comprise the array in RSP, as shown in Fig. 10, where *RL* is the number of rules in rule set, *RS* is the rule similarity used in DRSM, a_i^j is the coefficient of the jth node of the ith dimension, b^n is the compensation value shown in Fig. 1, and n is typically set to the maximum *RL*. In the learning process, the method of selecting coefficients is the same as that of selecting weights, AHRN selects front *RL* compensation values to build the corresponding rules.

AISP considers the current structure of AHRN generated according to SSP, as well as the fitness value of the current particle. AISP cannot directly influence movement, but can indirectly influence it because the composed particle with the highest fitness, which functions as the opinion leader, has the greatest influence in OLB-QPSO.

Therefore, the appearance position of the composed particle is influenced by two inner factors that are generated by SSP and RSP, and by one outer factor that is generated by the fitness status in the swarm. The outer factor can improve the composited influence, as shown in Fig. 8. Experience shows that the particle of this kind can improve the performance in obtaining the best position because they can search for the optimal solution in a wide feasible space.

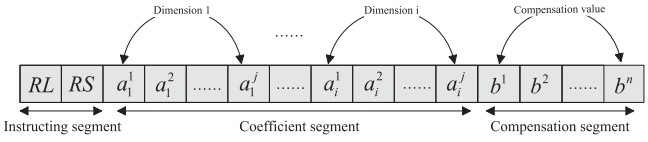


Fig. 10. Array in RSP.

3.4. Description of the entire algorithm

The key description of the implementing algorithm for AHRN is as follows.

Step 1: All parameters that need confirmation should be confirmed, such as RI number (RIN), convergence condition (CC), maximum iterations (MI), particle number (PN), and coefficient range (CR). All the samples should be normalized when needed; and the current number of iteration should set to be 0.

Step 2: All composed particles in the swarm should be initialized (for each particle, initialize SSP and RSP and then update AHRN to obtain AISP). All the pbests and the gbest should be updated.

Step 3: When the fitness of the gbest satisfies the requirement of the convergence condition, then the optimal solution should updated to be the gbest and the algorithm stops. Otherwise, the algorithm continues.

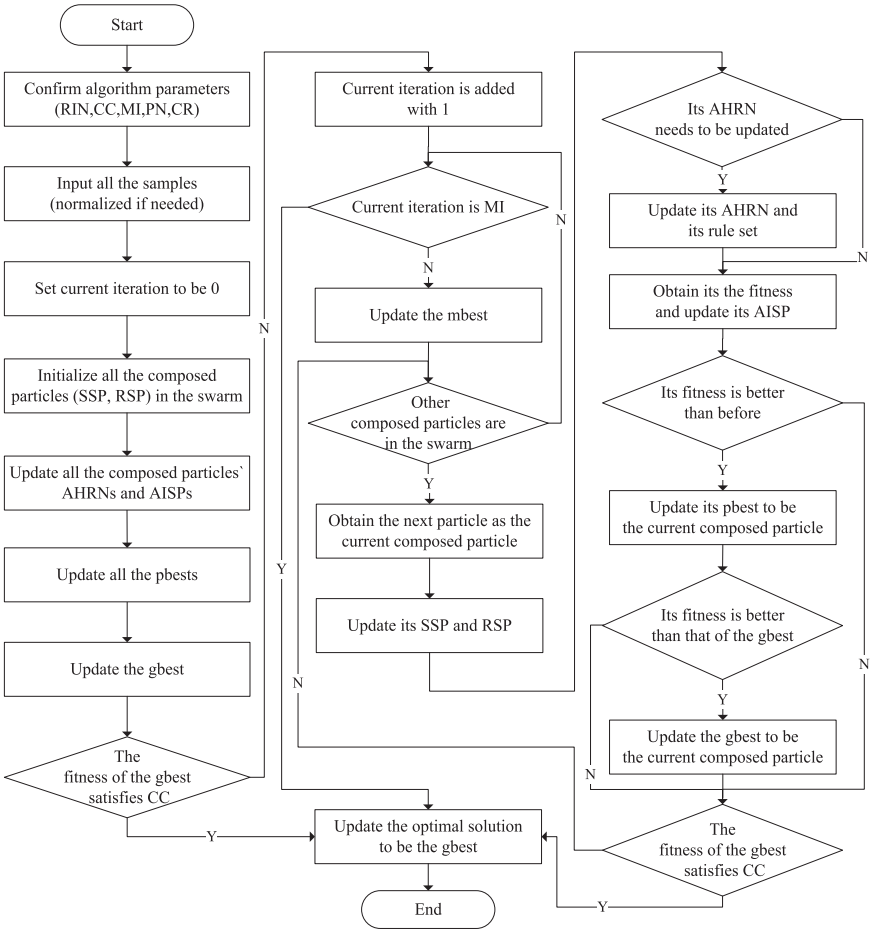


Fig. 11. Algorithm flowchart.

Step 4: The learning process proceeds into iterations, and the current number of iterations is added with 1. However, when the maximum number of iterations is reached, the optimal solution is updated to be the gbest and the algorithm stops.

Step 4.1: The mbest is updated by using Formula (12).

Step 4.2: For each composed particle in the swarm:

Step 4.2.1: Update SSP and RSP by using Formula (10).

Step 4.2.2: Update AHRN when necessary.

Step 4.2.3: Update the rule set according to DRSM when AHRN is updated.

Step 4.2.4: Obtain the fitness and update AISP.

Step 4.2.5: When the fitness is better than before, update the pbest to be the current composed particle.

Step 4.2.6: When the fitness is better than that of the gbest, update gbest to be the current composed particle.

Step 4.2.7: When the fitness of the gbest satisfies the requirement of CC, update the optimal solution to be the gbest and the algorithm stops. Otherwise, the algorithm continues.

As known from the description, the number of function evolutions (NFEs) is equal to the swarm capacity in the initialization process, whereas NFEs is equal to the swarm capacity multiplied by the number of iterations in the iteration process. Therefore, in the entire learning process, the NFEs can be estimated approximately according to Formula (13):

$$NFEs = S \times (1+I), \tag{13}$$

where S is the swarm capacity, and I is the number of iterations.

The algorithm flowchart is shown in Fig. 11 to illustrate the algorithm in detail.

In each iteration process: The time to update AHRN and implement fitness function is needed to calculate the fitness of a particle. In the process of updating AHRN, each dimension of an sample generates CI nodes, and thus, the time is mainly related to $n \times CI \times D$, where D is the sample dimension and n is the capacity of the samples. In the process of implementing the fitness function, RL rules are built, and each dimension is calculated by RL times, and thus, the time is mainly related to $n \times RL \times D$ (RL is typically set to be less than RI to avoid the excessively similar rules). Therefore, for the swarm, the time is mainly related to $n \times S \times (CI \times D + RL \times D)$, where S is the swarm capacity.

In the entire learning process: The time for the initialization and all iterations is needed. During the initialization, all samples are calculated once, and thus, the time is mainly related to $n \times S \times (CI \times D + RL \times D)$. However, during all iterations, all samples are calculated by I times, where I is the numbers of the iterations, and thus, the time is mainly related to $n \times S \times (CI \times D + RL \times D) \times I$. Consequently, the entire learning time is mainly related to $n \times S \times (CI \times D + RL \times D) \times (1+I)$, whereas $CI = RI + 1$ and $RL \leq RI$; and thus, the entire learning time is less than $n \times S \times D \times (2RI+1) \times (1+I)$.

Because S, D, and RI is typically far less than n and I, the time complexity of the algorithm is: when $n >> i$ or $i >> n$, $T(n) = O(n)$; otherwise, $T(n) = O(n^2)$. And because the number of the auxiliary variables is mainly related to the swarm capacity, the space complexity of the algorithm is $S(n) = O(n)$.

4. Experiments and analyses

AHRN can be employed to predict a chaotic time series. This study investigated the prediction capability of AHRN by using Box–Jenkins gas furnace data and the Mackey–Glass chaotic time series, and obtained excellent results. To demonstrate the practicability of AHRN in the engineering field, the EGT of a particular aero engine was used. To simplify experiment manipulation, the interface of an application programmed for the experiments is

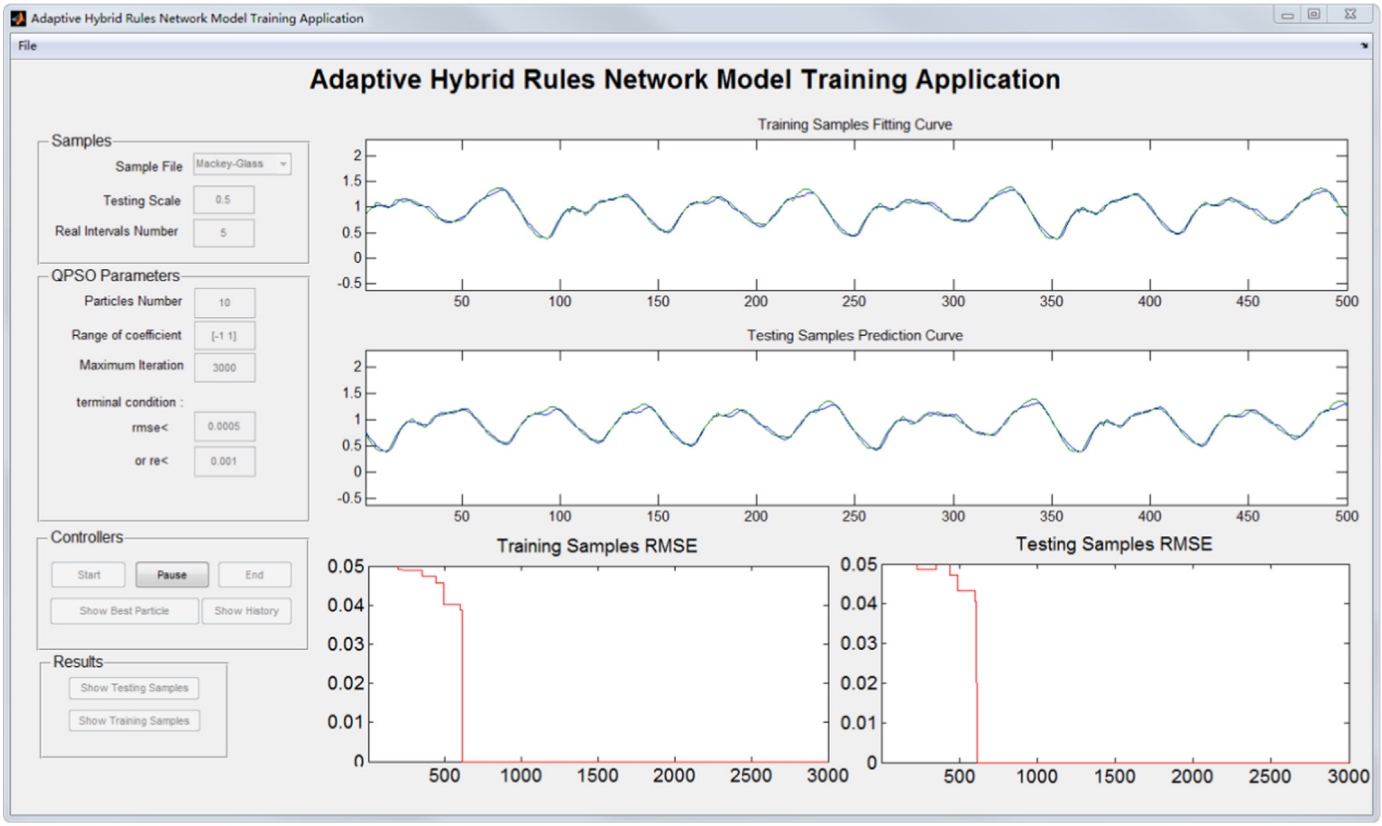


Fig. 12. Application interface.

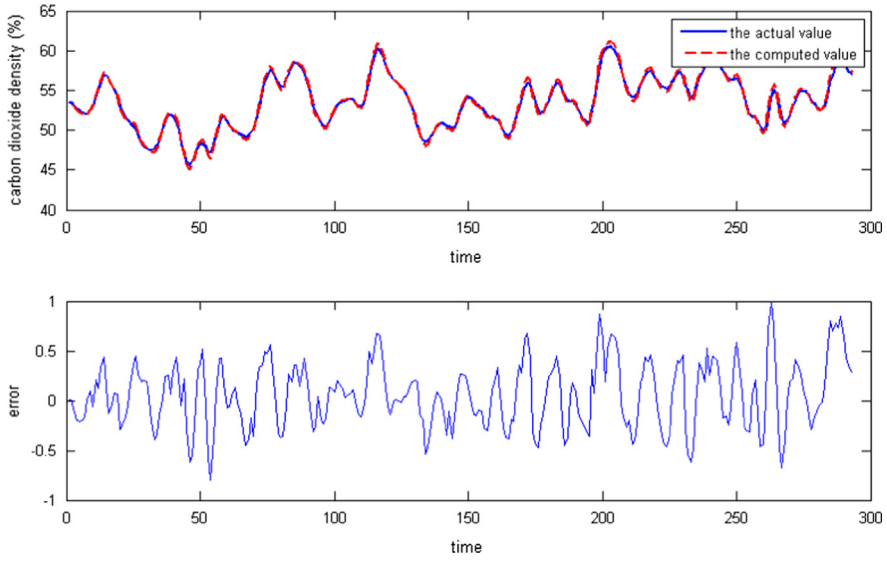


Fig. 13. Result for Box–Jenkins gas furnace data experiment.

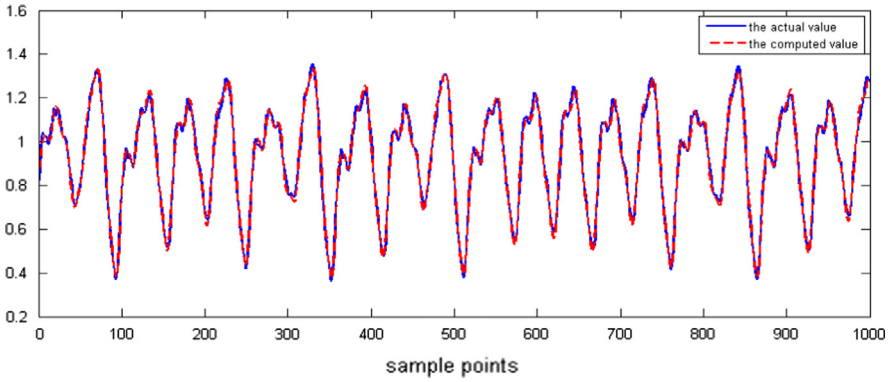


Fig. 14. Result for the Mackey–Glass experiment.

shown in Fig. 12. All experiments were run by MATLAB R2010b, and the computer system comprised an Intel Core i3 (2.4 GHz) CPU with 2 GB RAM and Windows 7 OS.

4.1. Experiment on Box–Jenkins gas furnace data

Box–Jenkins gas furnace data are a classic benchmark in the chaotic time series field. The data set originally consists of 296 input–output pairs [u(t), y(t)] collected from a furnace with a sampling time of 9 s. The input $u(t)$ is the gas flow rate into the furnace, whereas the output $y(t)$ is CO_2 concentration in the outlet gases. The objective is to predict the output $y(t)$ by using the other variables. Among all data points, $u(t)$, $t(t-1)$, $u(t-2)$, $y(t-1)$, $y(t-2)$, and $y(t-3)$ were used to construct the input vector; $y(t)$ was used to construct the output vector; and the input and output vectors were grouped as one sample. The capacity of the samples was 290, and the first 70% of the samples were used as training samples, whereas the remaining 30% were used as testing samples. Based on several experiments, the effective key parameter setting was RIN=5 and PN=30. The result is shown in Fig. 13. The average execution time is 1.373 s in each iteration process.

AHRN exhibits a reasonable capability in terms of predicting Box–Jenkins gas furnace data, and the errors are relatively stable. The performance can be improved after the algorithm is further optimized.

4.2. Experiment on the Mackey–Glass chaotic time series

The Mackey–Glass chaotic time series was also employed to validate the prediction capability of AHRN that uses the OLB-QPSO with composed particles. The time series was generated by the following differential delay equation:

$$x(t+1)=0.9x(t)+\frac{0.2x(t-17)}{1+x^{10}(t-17)}, \tag{14}$$

Among the points generated, $x(t-18)$, $x(t-12)$, $x(t-6)$, and $x(t)$ were used to construct the input vector; $x(t+6)$ was used to construct the corresponding output vector; and the input and output vectors were grouped as one sample. The capacity of the samples was 1000, and the first 500 samples were used as training samples, whereas the remaining 500 samples were used as testing samples. Based on several experiments, the effective key parameter setting was RIN=5 and PN=30. The result is shown in Fig. 14. The average execution time is 2.653 s in each iteration process.

The results obtained by applying different algorithms were compared and shown in Table 2.

The results obtained by using different algorithms were compared and shown in Table 3.

AHRN demonstrate a reasonable capability in predicting the Mackey–Glass chaotic time series problem. To validate the capability of robustness in AHRN, the samples should be noised. Noising of the original samples is shown in Table 4. For example,

Table 2
Results from different algorithms.

Algorithm	MSE	Average relative errors
Proposed by Kang et al. [37]	0.161	–
Proposed by Kukolj and Levi [37]	0.129	–
Proposed by Pomares et al. [37]	0.363	–
DE/QDE [37]	0.112	-
The present study	0.111	0.09428%

Note: "–" indicates that relevant studies did not provide the corresponding information.

Table 3
Results from different algorithms.

Algorithm	Training samples		Testing samples	
	RSME	Average relative errors	RSME	Average relative errors
Choi et al. [38]	0.000430	–	0.000410	–
Maguire et al. [39]	0.014000	–	0.009000	–
Duan et al. [40]	0.024000	–	0.025300	–
The present study	0.019138	1.6984%	0.020895	1.9141%

Note: "–" indicates that relevant studies did not provide the corresponding information.

Table 4
Data with noise.

Original data	Noised data	Original data	Noised data	Original data	Noised data
1.081111	1.090000	0.752387	0.760000	0.869055	0.857000
0.982946	0.975000	0.520449	0.530000	0.376558	0.362000
1.001452	1.010000	0.628194	0.622000	1.224462	1.210000
0.762473	0.757000	0.522397	0.512000	0.856656	0.847000

Table 5
First 20 original data points.

Time (s)	EGT (°C)	Time (s)	EGT (°C)	Time (s)	EGT (°C)	Time (s)	EGT (°C)
$t(1)$	533	$t(6)$	563	$t(11)$	548	$t(16)$	526
$t(2)$	538	$t(7)$	525	$t(12)$	556	$t(17)$	541
$t(3)$	538	$t(8)$	528	$t(13)$	521	$t(18)$	526
$t(4)$	536	$t(9)$	582	$t(14)$	543	$t(19)$	523
$t(5)$	537	$t(10)$	545	$t(15)$	546	$t(20)$	526

data in the original sample is 1.081111, but the corresponding data with noise is 1.090000.

To guarantee the consistency of the experiment parameter setting, RIN=5 and PN=30. The result is as follows. For the training samples, RSME was 0.019532 and the average relative error was 1.7274%. For the testing samples, RSME was 0.019127 and the average relative error was 1.7%. Therefore, reasonable generalization capability was still reflected although the samples were noised.

4.3. Experiment on EGT

EGT is an important parameter that characterizes aero engine health status. This parameter determines the availability of the engine, and thus, it has considerable practical application value in accurately predicting aircraft engine exhaust temperature and processing subsequent design activities. Data in this experiment were a chaotic time series composed of EGT from a particular aero engine from the China International Airlines Company. The first 20 original data points are shown in Table 5.

Among the data, $x(t-4)$, $x(t-3)$, $x(t-2)$, and $x(t-1)$ were used to construct the input vector; $x(t+1)$ was used to construct the output vector; and the input and output vectors were grouped as one sample. The capacity of the samples was 234. The first 192 samples were used as training samples, whereas the remaining 42 samples were used as the testing samples. Based on several experiments, the effective key parameter setting was RIN=5 and PN=30. The result is shown in Fig. 15. The average execution time is 0.936 s in each iteration process.

The noise should be removed before training samples in this experiment. However, the data curve of the result was smoother than the original, thus indicating that AHRN that uses OLB-QPSO with composed particles combines the two operations (i.e., removing noise and training samples), as known in Fig. 15. Therefore, the time to removing noise was reduced, and the prediction performance was improved. The results obtained by using different algorithms are shown in Table 6.

Table 6
Results from different algorithms.

Algorithm	RSME of testing data	Average relative error of testing data (%)
Shisheng et al. [41]	-	1.820
The present study	14.8089	2.070
Standard BP neural network	20.2072	2.879

Note: "–" indicates that relevant studies did not provide the corresponding information.

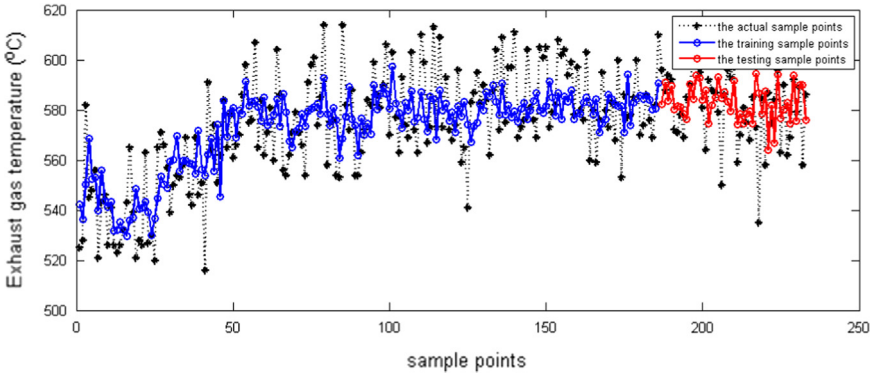


Fig. 15. Result for EGT experiment.

After comparing with other algorithms, AHRN that uses OLB-QPSO with composed particles was found to exhibit certain advantages over other algorithms with regard to predicting EGT of a particular aero engine from the China International Airlines Company.

5. Conclusion

AHRN has been introduced in this study. The structure of AHRN can be updated adaptively according to actual samples, with DRSM providing the most suitable rule set to obtain the optimal solution. Several algorithms can be applied to implement the learning process of AHRN; however, OLB-QPSO with composed particles is selected in this research. This algorithm obtains the mbest based on the opinion leader, and the structure of the composed particle is composed of SSP, RSP, and AISP. The influencing factors for the movement of the composed particle include the composited influence generated by SSP and RSP, as well as the fitness status (which occurs in AISP) in the swarm. The experiments have shown that AHRN that uses OLB-QPSO with composed particles exhibits powerful generalization capability and good performance in chaotic time series prediction, as well as high practical application value when used with the aid of computers.

Acknowledgments

The authors are grateful to the anonymous reviewers for their very helpful comments and constructive suggestions with regard to this paper. This paper is supported by National Natural Science Foundation of China (Grant no. 51075083), the major project of National Defense Foundation of China, and also by the advance research project of General Armament Department.

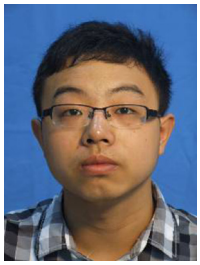
References

[1] F. Gao, G. Xiao, J.-j. Chen, Product interface reengineering using fuzzy clustering, Comput.-Aided Des. 40 (2008) 439–446.
[2] S.F. Qin, I.N. Jordanov, D.K. Wright, Freehand drawing system using a fuzzy logic concept, Comput.-Aided Des. 31 (1999) 359–360.
[3] S.F. Qin, D.K. Wright, I.N. Jordanov, From on-line sketching to 2D and 3D geometry: a system based on fuzzy knowledge, Comput.-Aided Des. 32 (2000) 851–866.
[4] C.C.L. Wang, T.K.K. Chang, M.M.F. Yuen, From laser-scanned data to feature human model: a system based on fuzzy logic concept, Comput.-Aided Des. 35 (2003) 241–253.
[5] T. TAKAGI, M. SUGENO, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. 15 (1985) 116–132.
[6] L.X. Wang, Adaptive Fuzzy Systems and Control: Design Stability Analysis, Prentice Hall Professional Technical Reference, 1994.
[7] J.J. Buckley, Sugeno type controllers are universal controllers, Fuzzy Sets Syst. 53 (1993) 299–303.
[8] P.P. Angelov, X. Zhou, Evolving fuzzy-rule-based classifiers from data streams, IEEE Trans. Fuzzy Syst. 16 (2008) 1462–1475.
[9] Y. Gao, M.J. Er, NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches, Fuzzy Sets Syst. 150 (2005) 331–350.
[10] L. Wang, R. Langari, Complex systems modeling via fuzzy logic, IEEE Trans. Syst. Man Cybern. Part B Cybern. 26 (1996) 100–106.
[11] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, Int. J. Man Mach. Stud. 7 (1975) 1–15.
[12] M.F. Azeem, M. Hanmandlu, N. Ahmad, Structure identification of generalized adaptive neuro-fuzzy inference systems, IEEE Trans. Fuzzy Syst. 11 (2003) 666–681.
[13] N. Wang, Y. Yang, A fuzzy modeling method via enhanced objective cluster analysis for designing TSK model, Expert Syst. Appl. 36 (2009) 12375–12382.
[14] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (1993) 665–685.
[15] A. Bastian, Identifying fuzzy models utilizing genetic programming, Fuzzy Sets Syst. 113 (2000) 333–350.
[16] B. Zhu, C.-Z. He, P. Liatsis, X.-Y. Li, A GMDH-based fuzzy modeling approach for constructing TS model, Fuzzy Sets Syst. 189 (2012) 19–29.
[17] R. Ji, Y. Yang, W. Zhang, Incremental smooth support vector regression for Takagi–Sugeno fuzzy modeling, Neurocomputing (2013).
[18] A. Kalhor, B.N. Araabi, C. Lucas, A new systematic design for habitually linear evolving TS fuzzy model, Expert Syst. Appl. 39 (2012) 1725–1736.
[19] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, U Michigan Press, 1975.
[20] M. Dorigo, G. Di Caro, T. Stützle, Ant Algorithms, Future Generation Computer Systems, 16, 2000, v-vii.
[21] J. Kennedy, R.C. Eberhart, Y. Shi, Chapter seven—the particle swarm, in: J. Kennedy, R.C. Eberhart, Y. Shi (Eds.), Swarm Intelligence, Morgan Kaufmann, San Francisco, 2001, pp. 287–325.
[22] R.C. Eberhart, Y. Shi, Chapter four—evolutionary computation implementations, in: R.C. Eberhart, Y. Shi (Eds.), Computational Intelligence, Morgan Kaufmann, Burlington, 2007, pp. 95–143.
[23] A. Khare, S. Rangnekar, A review of particle swarm optimization and its applications in Solar Photovoltaic system, Appl. Soft Comput. 13 (2013) 2997–3006.
[24] J. Wei, L. Guangbin, L. Dong, Elite particle swarm optimizaion with mutation, system simulation and scientific computing, 2008. ICSC 2008, in: Seventh International Conference on Asia Simulation Conference; 2008, pp. 800–803.
[25] L. Wang, Z. Cui, J. Zeng, Hybrid intelligent systems, 2009, in: Ninth International Conference on HIS'09. 1(2009), pp. 388–393.
[26] J.-H. Seo, C.-H. Im, C.-G. Heo, J.-K. Kim, H.-K. Jung, C.-G. Lee, Multimodal function optimization based on particle swarm optimization, IEEE Trans. Magn. 42 (2006) 1095–1098.
[27] W. Jianfang, L. Weihua, Optimization algorithm based on T–S fuzzy model of self-adaptive disturbed particle swarm optimization and neural network, in: 2009 Ninth International Conference on Hybrid Intelligent Systems, 2009, pp. 457–461.
[28] H. Liu, S. Xu, X. Liang, A modified quantum behaved particle swarm optimization for constrained optimization, in: International Symposium on Intelligent Information Technology Application Workshops, 2008. IITAW'08. 2008, pp. 531–534.
[29] V.C. Mariani, A.R.K. Duck, F.A. Guerra, L.d.S. Coelho, R.V. Rao, A chaotic quantum-behaved particle swarm approach applied to optimization of heat exchangers, Appl. Therm. Eng. 42 (2012) 119–128.
[30] F. Liu, H. Duan, Y. Deng, A chaotic quantum-behaved particle swarm optimization based on lateral inhibition for image matching (International Journal for Light and Electron Optics), Optik 123 (2012) 1955–1960.
[31] L.d.S. Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, Expert Syst. Appl. 37 (2010) 1676–1683.
[32] M. Xi, J. Sun, W. Xu, An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position, Appl. Math. Comput. 205 (2008) 751–759.
[33] C. Sun, S. Lu, Short-term combined economic emission hydrothermal scheduling using improved quantum-behaved particle swarm optimization, Expert Syst. Appl. 37 (2010) 4232–4241.
[34] D. Pan, J. Li, An Improved quantum-behaved particle swarm optimization algorithm based on chaos theory exerting to local optimal position, Inf. Comput. Sci. 10 (2013) 1819–1828.
[35] J. Sun, C.H. Lai, W. Xu, Z. Chai, A Novel and More Efficient Search Strategy of Quantum-behaved Particle Swarm Optimization, Adaptive and Natural Computing Algorithms, Springer (2007) 394–403.
[36] D. Zhai, M. Hao, J.M. Mendel, A non-singleton interval type-2 fuzzy logic system for universal image noise removal using quantum-behaved particle swarm optimization, in: Fuzzy Systems (FUZZ), 2011 IEEE International Conference on, (IEEE2011), pp. 957–964.
[37] H. Su, Y. Yang, Differential evolution and quantum-inquired differential evolution for evolving Takagi–Sugeno fuzzy models, Expert Syst. Appl. 38 (2011) 6447–6451.
[38] J.-N. Choi, S.-K. Oh, W. Pedrycz, Identification of fuzzy models using a successive tuning method with a variant identification ratio, Fuzzy Sets Syst. 159 (2008) 2873–2889.
[39] L.P. Maguire, B. Roche, T.M. McGinnity, L.J. McDaid, Predicting a chaotic time series using a fuzzy neural network, Inf. Sci. 112 (1998) 125–136.
[40] J.-C. Duan, F.-L. Chung, Multilevel fuzzy relational systems: structure and identification, Soft Comput. 6 (2002) 71–86.
[41] Z. Shisheng, L. Da, D. Gang, Convolution sum discrete process neural network and its application in aeroengine exhausted gas temperature prediction, Acta Aeronaut. Astronaut. Sin. 32 (2012) 438–455.

Lin Lin received the Ph.D degree in Mechanical Design from Harbin Institute of Technology, Harbin, China, in 2003. She is currently a professor in the School of Mechatronics Engineering in Harbin Institute of Technology. Her research interests include time series prediction, neural networks, fuzzy modeling, and mechanical product sheme intelligent design.

Feng Guo received the B.A. and M.A degrees from the Newmedia technology and Art Department in Harbin Institute of Technology, Harbin, China, in the year 2006 and 2008 respectively. He is currently working towards the Ph.D. degree in School of Mechatronics Engineering in Harbin Institute of Technology. His research interests include fuzzy modeling and control, particle swarm optimization, and mechanical product sheme intelligent design.

Xiaolong Xie received the M.E. degrees in Mechanical Design & Theory from Harbin Institute of Technology, Harbin, China, in 2011. He is currently working towards the Ph.D. degree in School of Mechatronics Engineering in Harbin Institute of Technology. His research interests include computer-aided design, and knowledge-based engineering.



Bin Luo received the B.A degrees from the School of Mechatronics Engineering in Harbin Engineering University. He is currently working towards the M.A degree in School of Mechatronics Engineering in Harbin Institute of Technology. His research interests include Multi-objective optimization and intelligent design.