Applied Soft Computing 11 (2011) 5463–5476

Contents lists available at ScienceDirect

Applied Soft Computing

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

Nonlinear system control using self-evolving neural fuzzy inference networks with reinforcement evolutionary learning

Cheng-Jian Lin[a,*], Cheng-Hung Chen[b]

[a] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan, ROC
[b] Department of Electrical Engineering, National Formosa University, No. 64, Wunhua Rd., Huwei Township, Yunlin County 632, Taiwan, ROC

A R T I C L E   I N F O

Article history:
Received 17 November 2008
Received in revised form 9 September 2010
Accepted 1 May 2011
Available online 14 May 2011

Keywords:
Neural fuzzy inference network
Reinforcement learning
Particle swarm optimization
Cultural algorithm

A B S T R A C T

This study presents a reinforcement evolutionary learning algorithm (REL) for the self-evolving neural fuzzy inference networks (SENFIN). By applying functional link neural networks (FLNN) as the consequent part of the fuzzy rules, the proposed SENFIN model combines orthogonal polynomials and linearly independent functions in a functional expansion of the FLNN. The SENFIN model can generate the consequent part of a nonlinear combination of the input variables. An efficient reinforcement evolutionary learning algorithm (REL), which consists of structure learning and parameter learning, is also presented. The structure learning is to determine the number of fuzzy rules. It adopts a subgroup symbiotic evolution to yield several variable fuzzy systems and uses an elite-based structure strategy to find the suitable number of fuzzy rules for solving a specific problem. The parameter learning is to adjust parameters of the SENFIN. It is a hybrid evolutionary algorithm, i.e., combining the cooperative particle swarm optimization and the cultural algorithm, called the cultural cooperative particle swarm optimization (CCPSO). As the result, the CCPSO approach can increase the global search capacity by using the belief space. In this paper the proposed NFIN with an efficient reinforcement evolutionary learning algorithm had been evaluated by two reinforcement learning applications, i.e., to balance the *cart–pole* system and the *ball and beam* system. Experimental results have demonstrated that the proposed approach performs well in reinforcement learning problems.

1. Introduction

With the ability to combine the low-level learning and computational power of neural networks and the high-level human-like thinking and reasoning of fuzzy systems, neural fuzzy inference networks (NFINs) have become a popular research topic, see [1–9] for example. In traditional approaches, the inference processes of if-then rules often rely on a substantial amount of heuristic observations in order to express knowledge of proper strategies. It is obvious that, for human experts, it is extremely difficult to examine all the input–output data from a complex system for the purpose of finding the proper rules for a NFIN. To cope with this difficulty, several approaches to generating if-then rules from numerical data have been proposed [4–9]. However, these approaches were developed for supervised learning; that is, the correct “target” output values should be specified for each corresponding input pattern in order to guide the network's whole learning procedure.

In general, parameter training is the main issue in designing a NFIN. Backpropagation (BP) training is commonly adopted to solve this problem, while it is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent approach is used in BP training to minimize the error function, the BP-based algorithms may reach the local minima very quickly and never find the global solution. Additionally, the performance of BP training highly depends on the initial values of the system parameters. In general, for different networking topologies one has to derive new mathematical expressions for each network layer. If precise training data can be easily obtained, the supervised learning algorithm may be efficient enough in many applications. Unfortunately, for some real-world applications precise training data are usually difficult and expensive to obtain. Hence, there has been a growing interest in reinforcement learning applications [10–18], while the relative training data are very rough and coarse and only “evaluative”, when compared with the “instructive” feedback in the supervised learning problem.

As to a favorable NFIN topology, technologies that can be used to train the system parameters and find the global solution while optimizing the overall structure are required. Many recent developments in evolutionary algorithms have provided such strategies for

* Corresponding author.
E-mail address: cjlin@ncut.edu.tw (C.-J. Lin).

1568-4946/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2011.05.012

NFINs design. Moriarty and Miikkulainen [17] proposed a reinforcement learning method called symbiotic, adaptive neural evolution that evolves a population of neurons through genetic algorithms to form a neural network. The genetic algorithm adopted by Juang et al. [18] is based upon symbiotic evolution which, when applied in fuzzy controller design, complements the local mapping property of a fuzzy inference rule. Lin et al. [19] presented a symbiotic evolution learning method that uses group-based population to assess fuzzy rules locally, whereas each group that represents a set of the individuals performs the evolution process in each generation. In addition, the compact genetic algorithm that represents a population as a probability distribution over the set of solutions and uses probability vectors to estimate the gene is suitable to 1 or 0 was applied by Harik et al. [20]. Lin and Xu [21] proposed a hybrid evolutionary learning algorithm that uses probability vectors to keep that the best group can be reproduced many times in each generation. Deriving from the mentioned works, a novel strategy of the structure learning is proposed in this study. The strategy adopts a subgroup symbiotic evolution (SSE) to yield several varied fuzzy systems and applies an elite-based structure strategy (ESS) to find the most suitable number of fuzzy rules by using their probability values.

Simultaneously, a new optimization algorithm, called particle swarm optimization (PSO), appears to be better than the BP algorithm. It is an evolutionary computation technique that was developed by Kennedy and Eberhart in 1995 [22,23]. The underlying motivation for the development of the PSO algorithm is the social behavior of animals, such as bird flocking, fish schooling and swarm theory. The PSO has been successfully applied to many optimization problems, such as control problems [24,25] and feedforward neural network design [26,27]. However, the PSO suffers from the burden of high dimensions, such that its performance falls as the dimensionality of the search space increases. Bergh et al. [28] then proposed a cooperative approach that employs cooperative behavior, called CPSO, which uses multiple swarms to improve the drawback of the traditional PSO. However, the CPSO still uses the traditional PSO formula, i.e., the local best position of each particle and global best position in the swarm. Therefore, the CPSO may only find a suboptimal solution. As the result, the parameter learning strategy, called cultural cooperative particle swarm optimization (CCPSO), which combines the cooperative PSO and cultural algorithm, to increase global search capacity, is proposed herein to avoid trapping in a suboptimal solution and to ensure that a nearby global optimal solution can be found.

This study proposes an efficient reinforcement evolutionary learning (REL) algorithm for the self-evolving NFIN (SENFIN). The SENFIN is based on our previous work [9], which combines a NFIN with a functional link neural network (FLNN). The consequent part of the fuzzy rules that corresponds to an FLNN comprises the functional expansion of input variables. The advantages of the proposed SENFIN-REL approach are as follows: (1) the proposed REL algorithm can dynamically determine the number of fuzzy rules and adjust parameters of a SENFIN, (2) the consequent of the fuzzy rules of SENFIN model involves a nonlinear combination of input variables, and (3) as demonstrated in Section 6 of this paper, the SENFIN-REL method is more effective than the other compared methods, in terms of relative shorter values of required balanced time.

The rest of this paper is organized as follows. Section 2 describes the basic concept of cultural algorithm and Section 3 proposes the structure of the SENFIN. Next, Section 4 presents the efficient evolutionary learning algorithm, while Section 5 introduces the reinforcement evolutionary learning algorithm used for constructing the SENFIN model. After exhibited the experimental results of the system-balancing simulations in Section 6, conclusions are drawn in the last section.

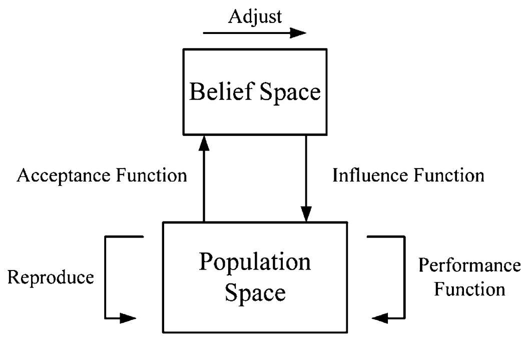

Fig. 1. Framework of cultural algorithm.

2. Basic concept of the cultural algorithm

The cultural algorithms [29–31] involve acquiring the belief space from the evolving population space and then exploiting that information to guide the search. Fig. 1 presents the cultural algorithm components. Cultural algorithms can be described in terms of two basic components—belief space and the population space. The belief space is the information repository in which the individuals can store their experiences for other individuals to learn from them indirectly. In cultural algorithms, the information acquired by an individual can be shared with the entire population, unlike in most evolutionary techniques, in which the information can be shared only with the offspring of individual. The population space comprises a set of possible solutions to the problem, and can be modeled using any population-based approach. The belief space and the population space are linked using a scheme that states rules that govern the individuals of the population space that can contribute to the belief space based on its experiences (according to the acceptance function), and the belief space can influence the new individuals of the population space (according to the influence function).

3. Structure of self-evolving neural fuzzy inference networks

This section describes the SENFIN model, which uses a nonlinear combination of input variables. Each fuzzy rule corresponds to a sub-FLNN [32,33], comprising a functional link. The SENFIN model is based on our previous research [9]. Fig. 2 presents the structure of the proposed SENFIN model. Nodes in layer 1 are input nodes, which represent input variables. Nodes in layer 2 are called membership function nodes and act as membership functions, which express the input fuzzy linguistic variables. Nodes in this layer are adopted to determine Gaussian membership values. Each node in layer 3 is called a rule node. Nodes in layer 3 are equal to the number of fuzzy sets that correspond to each external linguistic input variable. Links before layer 3 represent the preconditions of the rules, and links after layer 3 represent the consequences of the rule nodes. Nodes in layer 4 are called consequent nodes, each of which is a nonlinear combination of input variables. The node in layer 5 is called the output node; it is recommended by layers 3 and 4, and acts as a defuzzifier.

The SENFIN model realizes a fuzzy if-then rule in the following form [9].

Rule$_j$: IF x_1 is A_{1j} and x_2 is A_{2j} ... and x_i is A_{ij} ... and x_N is A_{Nj}

$$\text{THEN } \hat{y}_j = \sum_{k=1}^{M} w_{kj}\phi_k = w_{1j}\phi_1 + w_{2j}\phi_2 + \ldots + w_{Mj}\phi_M \tag{1}$$

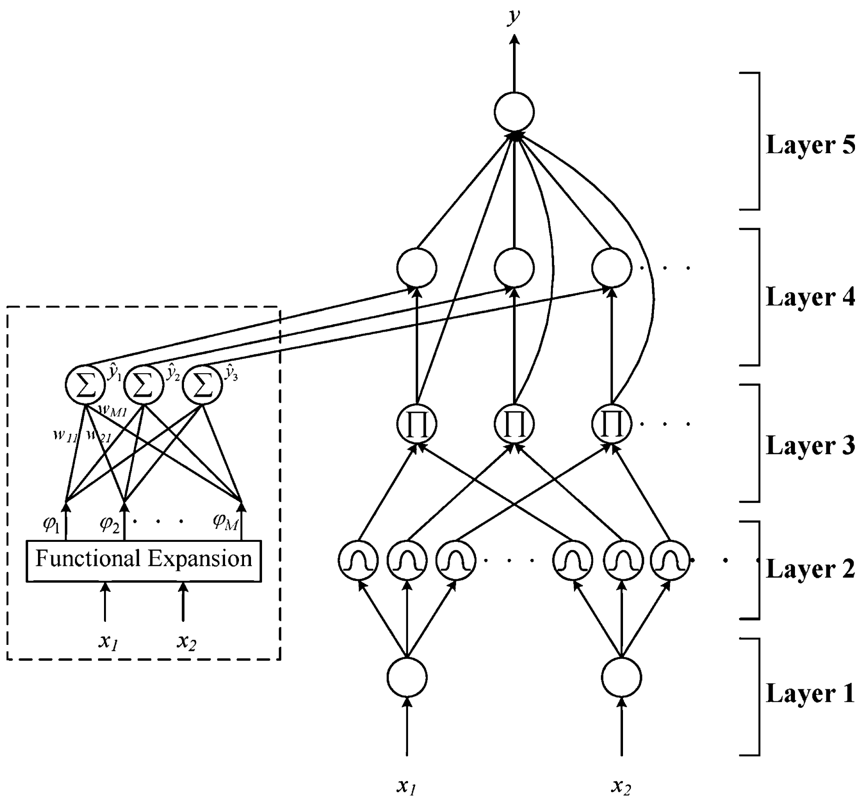

Fig. 2. Structure of proposed SENFIN model.

where x_i and $\hat{y}_j$ are the input and local output variables, respectively; A_{ij} is the linguistic term of the precondition part with Gaussian membership function; N is the number of input variables; w_{kj} is the link weight of the local output; ϕ_k is the basis trigonometric function of input variables; M is the number of basis function, and $Rule_j$ is the j-th fuzzy rule.

The operation functions of the nodes in each layer of the SENFIN model are now described. In the following description, $u^{(l)}$ denotes the output of a node in the l-th layer.

Layer 1 (Input node): No computation is performed in this layer. Each node in this layer is an input node, which corresponds to one input variable, and only transmits input values to the next layer directly:

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

Layer 2 (Membership function node): Nodes in this layer correspond to a single linguistic label of input variables in Layer 1. Therefore, the calculated membership value specifies the degree to which an input value belongs to a fuzzy set in layer 2. The implemented Gaussian membership function in layer 2 is

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

where m_{ij} and σ_{ij} are the mean and variance of the Gaussian membership function, respectively, of the j-th term of the i-th input variable x_i.

Layer 3 (Rule Node): Nodes in this layer represent the preconditioned part of a fuzzy logic rule. They receive one-dimensional membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described above is adopted to perform the IF-condition matching of the fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{4}$$

where the $\prod_i u_{ij}^{(2)}$ of a rule node represents the firing strength of its corresponding rule.

Layer 4 (Consequent Node): Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are nonlinear combinations of input variables from a functional link neural network, where the nonlinear combination function has not used the function tanh(·), as shown in Fig. 3. For such a node,

$$u_j^{(4)} = u_j^{(3)} \cdot \sum_{k=1}^{M} w_{kj}\phi_k \tag{5}$$

where w_{kj} is the corresponding link weight of functional link neural network and ϕ_k is the functional expansion of input variables. The functional expansion uses a trigonometric polynomial basis function, given by $[1, x_1, \sin(\pi x_1), \cos(\pi x_1), x_2, \sin(\pi x_2), \cos(\pi x_2), x_1x_2]$ for two-dimensional input variables. Therefore, M is the number of basis functions, $M = 3 \times N$, where N is the number of input variables.

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

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}\left(\sum_{k=1}^{M} w_{kj}\phi_k\right)}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}\hat{y}_j}{\sum_{j=1}^{R} u_j^{(3)}} \tag{6}$$

where R is the number of fuzzy rules, and y is the output of the SENFIN model. As described above, the number of tun-

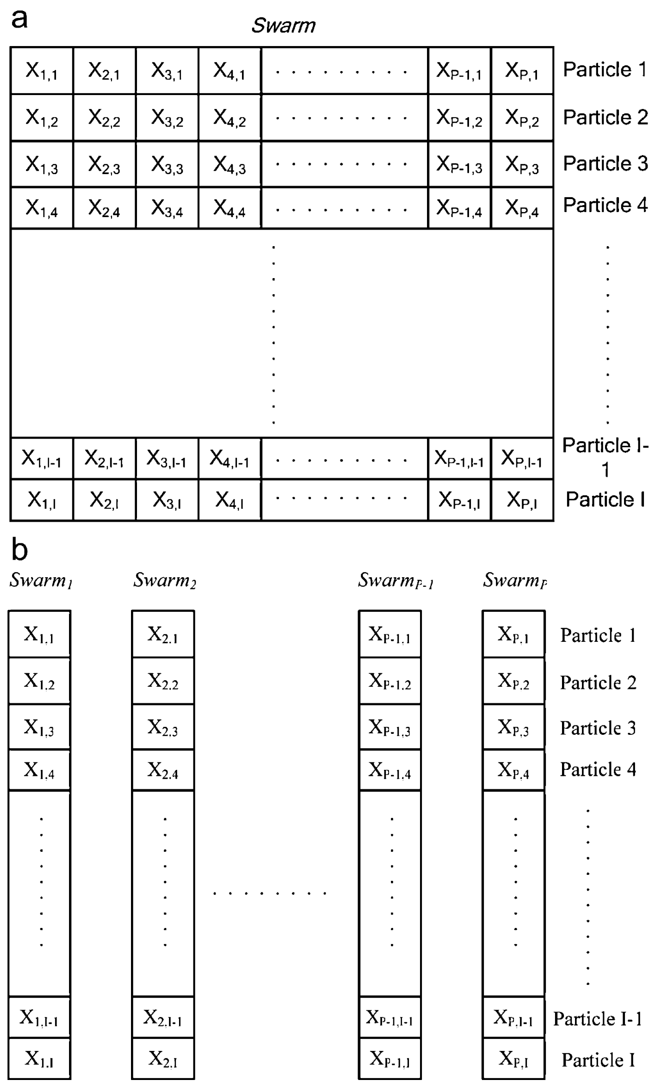

Fig. 3. Framework of the (a) PSO and (b) CPSO.

ing parameters for the SENFIN model is known to be $5 \times N \times R$, where N and R denote the number of inputs and existing rules, respectively.

4. An efficient evolutionary learning algorithm

This section describes the proposed efficient evolutionary learning algorithm. The learning algorithm comprises structure learning and parameter learning. The structure learning consists of the SSE and ESS. In the SSE, the fitness value of a rule (a sub-particle) is computed as the sum of the fitness values of all the feasible combinations of that rule with all other randomly selected rules, and then dividing this sum by the total number of combinations. The concept of ESS is similar to mimic the maturing phenomenon in society, where individuals become more suitable to the environment as they acquire more knowledge from society. The ESS can find suitable the number of rules and the combinations of rules according to probability values.

The parameter learning adopts a cultural cooperative particle swarm optimization (CCPSO) to adjust parameters of SENFIN model. The traditional PSO uses one swarm of particles defined by the P-dimension vectors to evolve. The CPSO method [28] can change traditional PSO into P swarms of one-dimension vectors, such that each swarm represents a dimension of the original problem. Fig. 3(a) and (b) shows the framework of the traditional PSO and CPSO method. The key point is that, instead of using one swarm (of I particles) to find the optimal P-dimension vector, the vector is split into its components so that P swarms (of I particles each) optimize a one-dimension vector. Notably, the function that is being optimized still requires a P-dimension vector to be evaluated. Additionally, each swarm aims to optimize a single component of the solution vector essentially solving a one-dimension optimization problem. Unfortunately, the CPSO still employs just the local best position and the global best position of the traditional PSO to evolution process. Therefore, the CPSO may fall into a suboptimal solution. The CCPSO method, which combines the cooperative particle swarm optimization and the cultural algorithm to increase the global search capacity, is proposed to avoid trapping in a suboptimal solution and to ensure the ability to search for a near global optimal solution.

The CCPSO method is characteristic of the cooperative particle swarm optimization and cultural algorithm. Fig. 4 shows the framework of the proposed CCPSO method, which is based on a CPSO all of whose parameters are simultaneously tuned using the belief space of the CA. The CCPSO method can strengthen the global search capability. If 50-dimension vectors are used in the original PSO, then the vectors in CCPSO can be changed into 50 swarms of one-dimension vectors. In the original PSO, the particle can Exhibit 50 variations in each generation, whereas the CCPSO offers $50 \times 50 = 2500$ different combinations in each generation. Additionally, each position of the CCPSO can be adjusted not only using the belief space which stores the paragons of each swarm, but also by searching around the local best solution and the global best solution. In the aforementioned scheme, the proposed CCPSO method can avoid falling into a suboptimal solution and ensure that the approximate global optimal solution can be found.

The detailed flowchart of the proposed efficient evolutionary learning algorithm is presented in Fig. 5. The foremost step in the proposed efficient evolutionary learning algorithm is the coding of a fuzzy rule into a sub-particle. Fig. 6 shows an example of the coding of parameters of a fuzzy rule into a sub-particle where i and j represent the i-th input variable and the j-th rule, respectively. In this study, a Gaussian membership function is adopted with variables that represent the mean and deviation of the membership function. Fig. 6 represents a fuzzy rule given by Eq. (1), where m_{ij} and σ_{ij} are the mean and deviation of a Gaussian membership function, respectively, and w_{kj} represents the corresponding link weight of the consequent part that is connected to the j-th rule node. In this study, a real number represents the position of each sub-particle.

The learning algorithm process is described step-by-step below.

Step 1. Create initial swarms

Before the proposed efficient evolutionary learning algorithm is applied, every position $x_{p,i}(t)$ must be created randomly in the range [0,1] in each subgroup, where $p = 1, 2, \ldots, P$ represents the p-th swarm, $i = 1, 2, \ldots, I$ represents the i-th particle, and t denotes the t-th generation.

Step 2. Create initial belief space

The belief space is the information repository in which the particles can store their experiences for other particles to learn from them indirectly. Create P belief space, B_p $(p = 1, 2, \ldots, P)$. Each initial B_p is defined as an empty set.

Step 3. Subgroup symbiotic evolution (SSE)

In order to keep the same number of each fuzzy system in each group, this size α needs to be defined in each group, that is, the size of each group is α. Therefore, the size of group must be set to $\alpha \cdot (R_{\max} - R_{\min} + 1)$, where $R_{\max}$ and $R_{\min}$ represent the maximum number of rules and the minimum number of rules, respectively.

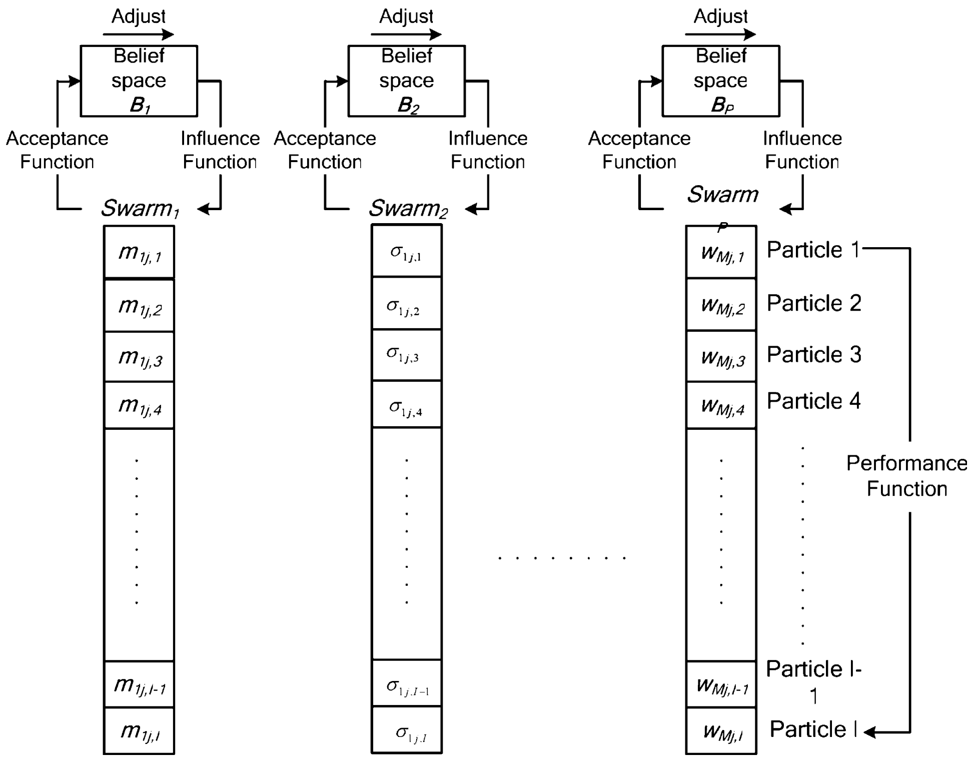

Fig. 4. Framework of proposed CCPSO.

In this step, the fitness value of a rule (a sub-particle) is computed as the sum of the fitness values of all the feasible combinations of that rule with all other randomly selected rules, and then dividing this sum by the total number of combinations. Fig. 7 shows the structure of the sub-particle in the subgroup symbiotic evolution. The stepwise assignment of the fitness value is as follows.

- *Step 3.1:* Randomly select R fuzzy rules (sub-particle) from each of the above subgroups, and compose the fuzzy system using these R rules.
- *Step 3.2:* Calculate fitness value of the particles using the SENFIN thus composed. In this study, the fitness function is used by a reinforcement signal in Eq. (14) that we will introduce in the next section.
- *Step 3.3:* Divide the fitness value by R and accumulate the divided fitness value to the fitness record of the R selected rules with their recorded fitness values initially set to zero.
- *Step 3.4:* Repeat the above steps until the space of each group has been filled a sufficient number of times, and record the number of fuzzy systems to which each sub-particle has contributed.
- *Step 3.5:* Divide the accumulated fitness of each sub-particle by the number of times it has been selected.
- *Step 3.6:* Sort these sub-particles in each subgroup in order of increasing fitness.

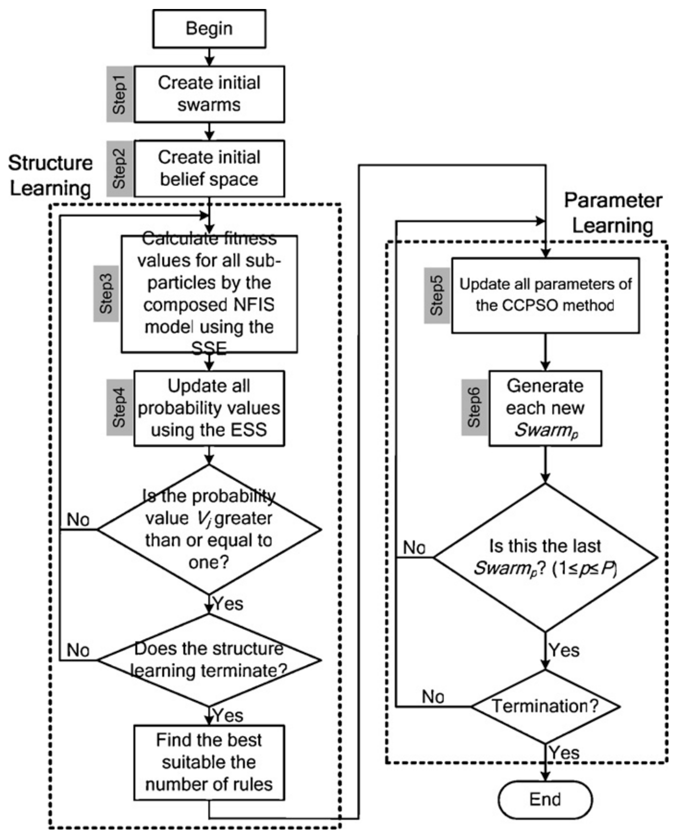
Begin
Step1
Create initial swarms
Step2
Create initial belief space
Structure Learning
Step3
Calculate fitness values for all sub-particles by the composed NFIS model using the SSE
Step4
Update all probability values using the ESS
Is the probability value V_j greater than or equal to one?
No
Yes
Does the structure learning terminate?
No
Yes
Find the best suitable the number of rules
Parameter Learning
Step5
Update all parameters of the CCPSO method
Step6
Generate each new $Swarm_p$
Is this the last $Swarm_p$? (1≤p≤P)
No
Yes
Termination?
No
Yes
End

Fig. 5. Flowchart of the proposed efficient evolutionary learning algorithm.

Step 4. Elite-based structure strategy (ESS)

The foremost step in ESS is the coding of the probability value V_j into building blocks (BBs), as shown in Fig. 8, where each probability value represents the suitability of rules of a fuzzy system. In Fig. 8, $R_{\max}$ and $R_{\min}$ are predefined to prevent less or more fuzzy rules from being generated in a fuzzy system. According to the results of the ESS, suitable the number of rules and combination of rules can be found. The details of ESS are as follows.

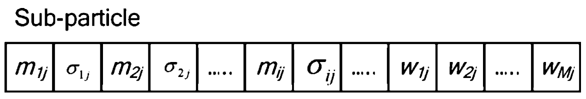

Fig. 6. Coding a fuzzy rule into a sub-particle.

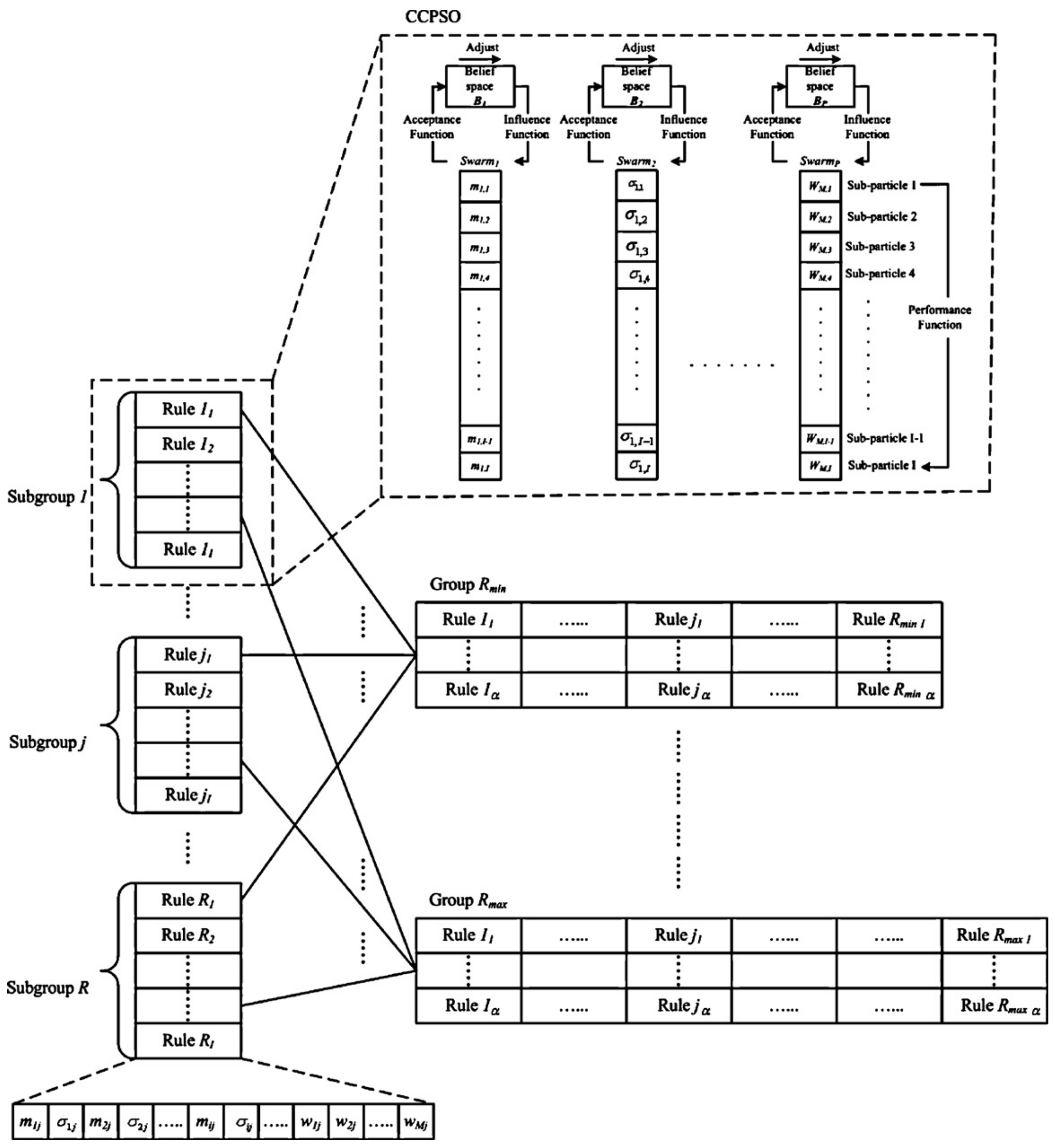
CCPSO
Adjust
Belief space B_1
Acceptance Function
Influence Function
Swarm$_1$
Belief space B_2
Swarm$_2$
Belief space B_P
Swarm$_P$
$m_{1,1}$ $m_{1,2}$ $m_{1,3}$ $m_{1,4}$ $m_{1,I-1}$ $m_{1,I}$
$\sigma_{1,1}$ $\sigma_{1,2}$ $\sigma_{1,3}$ $\sigma_{1,4}$ $\sigma_{1,I-1}$ $\sigma_{1,I}$
$W_{M,1}$ $W_{M,2}$ $W_{M,3}$ $W_{M,4}$ $W_{M,I-1}$ $W_{M,I}$
Sub-particle 1
Sub-particle 2
Sub-particle 3
Sub-particle 4
Performance Function
Sub-particle I-1
Sub-particle I
Subgroup 1
Rule 1_1
Rule 1_2
Rule 1_I
Subgroup j
Rule j_1
Rule j_2
Rule j_I
Subgroup R
Rule R_1
Rule R_2
Rule R_I
Group R_{min}
Rule 1_1 …… Rule j_1 …… Rule $R_{min\ 1}$
Rule 1_α …… Rule j_α …… Rule $R_{min\ \alpha}$
Group R_{max}
Rule 1_1 …… Rule j_1 …… …… Rule $R_{max\ 1}$
Rule 1_α …… Rule j_α …… …… Rule $R_{max\ \alpha}$
m_{1j} σ_{1j} m_{2j} σ_{2j} ….. m_{ij} σ_{ij} ….. w_{1j} w_{2j} ….. w_{Mj}

Fig. 7. Structure of sub-particle in subgroup symbiotic evolution.

- *Step 4.1:* Update probability values of BBs according to the following equations;

$$\begin{cases} V_j(t_s+1) = V_j(t_s) + (Upt\_value_j \cdot \lambda) & \text{, if } Avg_j \text{ is the best performance} \\ V_j(t_s+1) = V_j(t_s) & \text{, otherwise} \end{cases} \tag{7}$$

where $j = [R_{\min}, R_{\max}]$

$$Avg_j = \frac{\sum_{\alpha'=1}^{\alpha} F_{\alpha'}}{\alpha}$$

$$Upt\_value_j = \frac{\sum_{\alpha'=1}^{\alpha} F_{\alpha'}}{\sum_{R'=R_{\min}}^{R_{\max}} \sum_{\alpha'=1}^{\alpha} F_{\alpha'}}$$

where V_j is a probability value in the BBs and presents the suitability of rules of a fuzzy system; λ is a constant; Avg_j is a average fitness value in the j-th group; $F_{\alpha'}$ is a fitness value of each composed fuzzy system in each group and α is the size of each group.

- *Step 4.2:* Find the best suitable the number of rules

The probability values of BBs initially set to zero. Repeat the step 4.1 until V_j is greater than or equal to one. The finished above step is called one structure learning. Therefore, the number of structure learning must set to N_s. Accumulate probability values of each structure learning in BBs and divide the accumulated probability values by N_s to find the suitable number of rules and the combinations of rules.

Step 5. Update all parameters of the CCPSO method

- *Step 5.1:* Update local best position $L_{p,i}$ and global best position G_p

The local best position $L_{p,i}$ is the best previous position that yielded the best fitness value of the p-th swarm of the i-th parti-

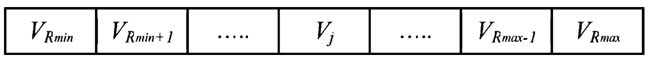

Fig. 8. Coding probability values into building blocks.

```
Procedure of the CCPSO method
Begin
    Let $k$=0;
    Repeat
        Let $j$=$R_{min}$;
        Repeat
            Calculate fitness value by SSE;
            Update $V_j$ by ESS;
            If $V_j$>=1 then
                Break;
            End if
        Until $j$=$R_{max}$;
        $k$=$k$+1;
    Until $k$=$Ns$;

    Repeat
        Let $p$=1;
        Repeat
            Update all parameters of the CCPSO method by (8) to (12) ;
            Generate each new $Swarm_p$;
        Until $p$=$P$;
    Until termination condition is reached;
End
```

Fig. 9. The pseudo code of the CCPSO method.

cle, and the global best position G_p is generated by the whole local best position. In step 5.1, the first step updates the local best position. Compare the fitness value of each current particle with that of its local best position. If the fitness value of the current particle exceeds those of its local best position, then the local best position is replaced with the position of the current particle. The second step updates the global best position. Compare the fitness value of all particles in their local best positions with that of the particle in the global best position. If fitness value of the particle in the local best position is better than those of the particles in the global best position, then the global best position is replaced with the current local best position.

$$L_{p,i}(t+1) = \begin{cases} x_{p,i}(t), & \text{if } F(x_{p,i}(t)) < F(L_{p,i}(t)) \\ L_{p,i}(t), & \text{if } F(x_{p,i}(t)) \geq F(L_{p,i}(t)) \end{cases}$$
$$G_p(t+1) = \underset{L_{p,i}}{\operatorname{argmin}} F(L_{p,i}(t+1)), \qquad 1 \leq i \leq I \tag{8}$$

- *Step 5.2:* Adjust each belief space B_p using an acceptance function

The first part of step 5.2 sorts these particles in each $Swarm_p$ in order of increasing fitness. Then, the paragon of each $Swarm_p$ is put into belief space B_p using an acceptance function. This function yields the number of particles that are used to adjust each belief space, and is as follows. The number of accepted particles decreases as the number of generations increases.

$$N_{\text{accepted}} = n\% \cdot I + \frac{n\%}{t} \cdot I \tag{9}$$

where $n\%$ is a parameter that is set by user, and must specify the top performing 20% [31]; I is the number of particles, and t represents the t-th generation. The acceptance function uses temporal information about the current number of generations to determine N_{accepted}, the number of individuals, accepted at generation (generation number) t. However, it will select more than that in the beginning of the search process and gradually reduces the number as the process proceeds. As in simulated annealing, more variability is allowed in the belief space at the beginning than at the end. The second step adjusts B_p. The interval of belief space BI_p is defined $BI_p = [l_p, u_p] = \{x | l_p \leq x \leq u_p, x \in \Re\}$, where l_p is the lower bound on belief space B_p and u_p is the upper bound on belief space B_p. Then, the position of each particle in B_p is compared with the lower bound l_p. If the position of the particle is smaller than the lower bound l_p, then the lower bound l_p is replaced with the current position. Furthermore, the position of each particle in the B_p is compared with the upper bound u_p. If the position of the particle is greater than the upper bound u_p, then the upper bound u_p is replaced with the current position. These rules are given below.

$$l_p = \begin{cases} x_{p,i} & \text{if } x_{p,i} \leq l_p \\ l_p & \text{otherwise} \end{cases}$$
$$u_p = \begin{cases} x_{p,i} & \text{if } x_{p,i} \geq u_p \\ u_p & \text{otherwise} \end{cases} \tag{10}$$

Step 6. Generate each new $Swarm_p$ using l_p, u_p, $L_{p,i}$, and G_p

In step 5.3, the first step adjusts every position of each $Swarm_p$ using an influence function Eq. (11). This step can change the direction of each particle in solution space, not easily being trapped at a local optimum. Then, the second step updates velocity and position of each particle to generate the each new $Swarm_p$ using Eqs. (12) and (13).

$$x_{p,i}(t) = \begin{cases} x_{p,i}(t) + \left|Rand()\cdot(u_p - l_p)\right| & if\ x_{p,i} < l_p \\ x_{p,i}(t) - \left|Rand()\cdot(u_p - l_p)\right| & if\ x_{p,i} > u_p \end{cases} \tag{11}$$

$$v_{p,i}(t+1) = w\cdot v_{p,i}(t) \quad +c_1\cdot Rand()\cdot[L_{p,i}(t+1) - x_{p,i}(t)] \\ +c_2\cdot Rand()\cdot[G_p(t+1) - x_{p,i}(t)] \tag{12}$$

$$x_{p,i}(t+1) = x_{p,i}(t) + v_{p,i}(t+1) \tag{13}$$

where c_1 and c_2 denote acceleration coefficients; $Rand()$ is generated from a uniform distribution in the range [0,1], and w controls the magnitude of $v_{p,i}(t)$.

The pseudo-code for the CCPSO method is listed in Fig. 9. Finally, these parameters of the proposed efficient evolutionary learning algorithm are explained as follows:

Notation	Description	Function
$R_{\min}$	The minimum number of fuzzy rules in SSE and ESS	Avoid too few fuzzy rules
$R_{\max}$	The maximum number of fuzzy rules in SSE and ESS	Avoid too many fuzzy rules
α	The size of each group in SSE and ESS	Determine the size of each group
λ	A constant in Eq. (7)	Influences the updated rate of V_j
N_s	The maximum number of generations of structure learning	Decide the number of generations of structure learning
$n\%$	A parameter of acceptance function in Eq. (9)	Determine the number of individuals that can enter belief space
w	The coefficient of the inertia term in Eq. (12)	Control the magnitude of velocity
c_1	The coefficient of the cognitive term in Eq. (12)	Acceleration for local best
c_2	The coefficient of the society term in Eq. (12)	Acceleration for global best

$R_{\min}, R_{\max}, \alpha, \lambda$, and N_s influence structure learning, and $n\%, w, c_1$, and c_2 influence parameter learning. $R_{\min}$, $R_{\max}$, α, and N_s depend on the complexity of the problem. The selection of parameter λ critically affects the analysis of V_j. The parameter λ, which uses the range (0, 1], was carefully examined in extensive experiments, and defined as [0.01,0.5]. $n\%$ is a parameter of acceptance function given by the user, in (0, 50%]; Saleem and Reynolds [31] suggests using 20%. w, c_1, and c_2 are three coefficients of updated velocity. The three coefficients are based on practical experimentation, and then defined by the user, in [0.4,0.9], [1,2], and [1,2].

5. Reinforcement evolutionary learning for a SENFIN model

Unlike the supervised learning problem, in which the correct "target" output values are given for each input pattern, the reinforcement learning problem has only very simple "evaluative" or "critical" information, rather than "instructive" information, available for learning. In the extreme case, there is only a single bit of information to indicate whether the output is right or wrong. Fig. 10 shows the efficient reinforcement evolutionary learning algorithm (REL). Its training environment interacts with reinforcement learning problems. In this study, the reinforcement signal indicates whether a success or a failure occurs.

As shown in Fig. 10, the proposed REL consists of a SENFIN model, which acts as the control network that determines the proper action to take according to the current input vector (environment state). The structure of the proposed REL is different from the actor-critic architecture of Barto et al. [10], which consists of a control network and a critic network. The input to the SENFIN model is the state of a plant, and the output is a control action of the state, denoted by f. The only available feedback is a reinforcement signal that notifies the SENFIN model only when a failure occurs.

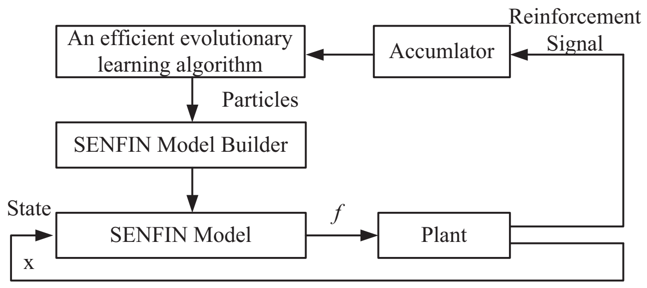

Fig. 10. Flowchart of the REL for the SENFIN model.

An accumulator plays a role which is a relative performance measure, as shown in Fig. 10. It accumulates the number of time steps before a failure occurs. In this study, the feedback takes the form of an accumulator that determines how long the experiment is still a "success"; this is used as a relative measure of the fitness of the proposed REL method. That is, the accumulator will indicate the "fitness" of the current SENFIN model. The key to the REL is formulating a number of time steps before failure occurs and using this formulation as the fitness function for the REL method. The advantage of the proposed method need not use the critical network as either a multi-step or single-step predictor.

Fig. 11 shows the flowchart of the REL method. The proposed REL method runs in a feed forward fashion to control the environment (plant) until a failure occurs. Our relative measure of the fitness function takes the form of an accumulator that determines how long the experiment is a "success". In this way, according to a defined fitness function, a fitness value is assigned to each string in the population where a high fitness value means a good fit. In this study, we use a number of time steps before failure occurs to define the fitness function. The goal of the REL method is to maximize the fitness value. The fitness function is defined by:

$$Fitness\,Function(i) = TIME\_STEP(i) \tag{14}$$

where $TIME\_STEP(i)$ represents how long the experiment is a "success" with the i-th population. Eq. (14) reflects the fact that long-time steps before failure occurs (to keep the desired control goal longer) means a higher fitness of the REL method.

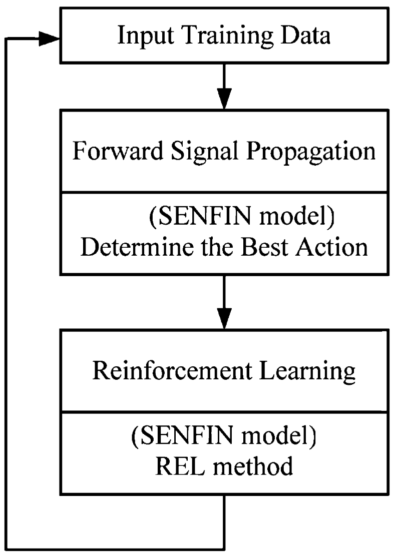
Input Training Data
Forward Signal Propagation
(SENFIN model) Determine the Best Action
Reinforcement Learning
(SENFIN model) REL method

Fig. 11. Flowchart of the REL method.

Table 1
Initial parameters before training.

Parameter	Value
w	0.4
c_1	1.6
c_2	2
$R_{\min}$	2
$R_{\max}$	12
λ	0.05
α	10
N_s	10
$n\%$	20%
Coding Type	Real Number

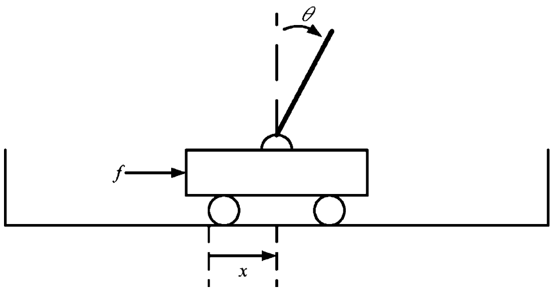

Fig. 12. The cart–pole balancing system.

6. Experimental results

In this section, we compare the performance of the SENFIN model using the REL method with some existing models for two applications. The first simulation was performed to balance the cart–pole system that was described in [33], while the second experiment was to balance the ball and beam system [34]. The initial parameters for the two simulations are given in Table 1. Note that these used initial parameters in this paper were determined by practical experimentation or trial-and-error tests.

Example 1. Control of a cart–pole balancing system

In this example, we apply the REL method to the classic control problem of a cart–pole balancing. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and classic control techniques [11,33,34], or the reinforcement learning schemes [12,18], and has been applied as a control benchmark. As shown in Fig. 12, the cart–pole balancing problem is the application of learning how to balance an upright pole. The bottom of the pole is hinged to a cart that travels along a finite-length track to its right or left. Both the cart and the pole can move only in the vertical plane; that is, each has only one degree of freedom.

There are four state variables in the system: θ, the angle of the pole from an upright position (in degrees); $\dot{\theta}$, the angular velocity of the pole (in degrees/s); x, the horizontal position of the cart's center (in meters); and $\dot{x}$, the velocity of the cart (in meters/seconds). The only control action is f, which is the amount of force (in *Newtons*) applied to the cart in order to move it toward left or right. The system fails when the pole falls past a certain angle ($\pm 12^{\circ}$ is used here) or the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track). The goal of this control problem is to determine a sequence of forces that is applied to the cart to balance the pole upright. The equations of motion that we used are:

$$\theta(t+1) = \theta(t) + \Delta\dot{\theta}(t) \tag{15}$$

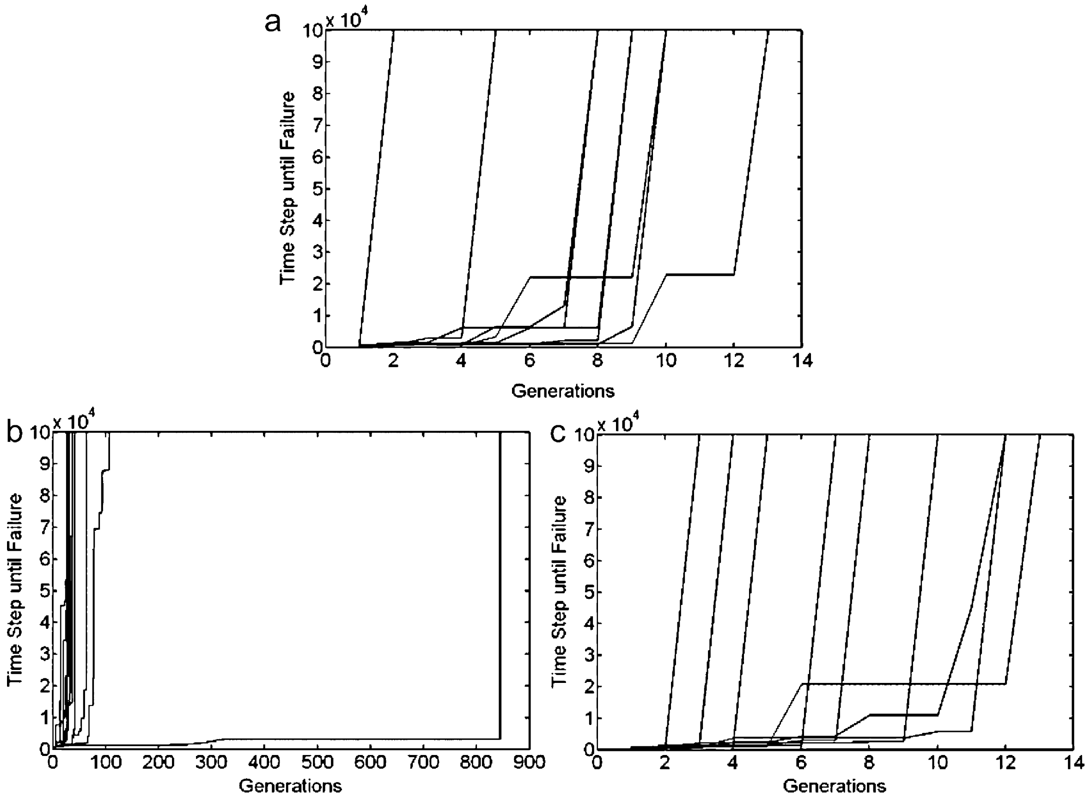

Fig. 13. The performance of (a) the REL method, (b) the R-PSO method [22], and (c) the R-CPSO method [28] on the cart–pole balancing system.

$$\dot\theta(t+1)=\dot\theta(t)+\Delta\frac{(m+m_p)g\ \sin\theta(t)}{(4/3)(m+m_p)l-m_pl\ \cos^2\theta(t)}-\frac{\cos\theta(t)[f(t)+m_pl\dot\theta(t)^2\ \sin\theta(t)-\mu_c sgn(\dot x(t))]}{(4/3)(m+m_p)l-m_pl\cos^2\theta(t)}-\frac{\mu_p(m+m_p)\dot\theta(t)/m_pl}{(4/3)(m+m_p)l-m_pl\cos^2\theta(t)} \quad (16)$$

$$x(t+1)=x(t)+\Delta\dot x(t) \quad (17)$$

$$x(t+1)=\dot x(t)+\Delta\frac{f(t)+m_pl[\dot\theta(t)^2\ \sin\theta(t)-\ddot\theta(t)\ \cos\theta(t)]}{(m+m_p)}-\frac{\mu_c sgn(\dot x(t))}{(m+m_p)} \quad (18)$$

where, l = 0.5 m, the length of the pole; m = 1.1 kg, combined mass of the pole and the cart; m_p = 0.1 kg, mass of the pole; g = 9.8 m/s, acceleration due to the gravity; μ_c = 0.0005, coefficient of friction of the cart on the track; μ_p = 0.000002, coefficient of friction of the pole on the cart; Δ = 0.02 s, sampling interval.

The constraints on the variables are $-12^\circ\le\theta 12^\circ$, $-2.4\,\text{m}\le x\le 2.4\,\text{m}$, and $-10\,\text{N}\le f\le 10\,\text{N}$. A control strategy is deemed successful if it can balance a pole for 100,000 time steps.

The four input variables $(\theta,\dot\theta,x,\dot x)$ and the output f_t are normalized between 0 and 1 over the following ranges: θ:[−12,12], $\dot\theta:[-60,60]$, x:[−2.4,2.4], $\dot x:[-3,3]$, and f_t:[−10,10]. The four normalized state variables are used as inputs to the proposed SENFIN model. The coding of a rule in a sub-particle is the form illustrated in Fig. 6. The values are floating-point numbers assigned by the REL method initially. The fitness function in this example is defined as in Eq. (14) to train the SENFIN model, where Eq. (14) also represents how long the cart–pole balancing system fails and receives a penalty signal of −1, when the beam deviates beyond a certain angle ($|\theta|>12^\circ$) and the cart runs off the bounds of its track ($|x|>2.4$ m).

In this experiment, the initial values were set to (0, 0, 0, 0). A total of ten runs were performed and each run started from the same initial state. Fig. 13(a) shows that the SENFIN model learned on average to balance the pole at the 7th generation. In this figure, each run represents that largest fitness value in the current generation being selected before the cart–pole balancing system fails. When the REL method was stopped, we chose the best strings in the population in the final generation and tested them on the cart–pole balancing system. Fig. 14 presents the results of the probability vectors in the ESS. In this figure, the final average optima number of rules is 3. The obtained fuzzy rules of the SENFIN using the REL method are exhibited as follows:

Table 2
Performance comparison of various existing models in Example 1.

Method	Mean	Best	Worst
GENITOR [11]	3268	415	18,743
TDGAR [12]	186	18	310
CQGAF [15]	133	12	288
SANE [17]	1984	46	5865
R-SE [18]	214	15	380
R-PSO [22]	125	27	845
R-CPSO [28]	8	3	13
R-GA [34]	324	26	550
REL	7	2	13

Fig. 15(a) displays the angular deviation of the pole when the cart–pole balancing system was controlled by a well-trained SENFIN model started from the initial state with $x(0)=0,\dot x(0)=0,\theta(0)=0$ and $\dot\theta(0)=0$. The simulated results show that the trained SENFIN model has good control ability for the cart–pole balancing system.

As shown in Fig. 13, the performance comparison of our proposed model with the reinforcement particle swarm optimization (R-PSO) [22] (see Fig. 13(b)) and the reinforcement cooperative particle swarm optimization (R-CPSO) [28] (see Fig. 13(c)) are demonstrated as well. In the R-PSO and R-CPSO approaches, the coefficient w, cognitive coefficient c_1, and society coefficient c_2 was set to 0.4, 1.6 and 2, respectively. Fig. 13(b) and (c) illustrate that the R-PSO and the R-CPSO methods learned to balance the pole on average at the 120th and 8th generations. Furthermore, Fig. 15(b) and (c) accordingly show the angular deviations of the pole when the cart–pole balancing system was controlled by [22,28]. As presented in Figs. 13 and 15, the control capabilities of the trained SENFIN model with the REL algorithm are much better than the ones of [22,28] in the cart–pole balancing system.

In addition, the GENITOR [11], TDGAR [12], CQGAF [15], and SANE (Symbiotic Adaptive Neural Evolution) [17] have been applied to this controlling benchmark problem, while the simulated results are summarized in Table 2, by the numbers of pole-balancing trials, i.e., to reflect the number of training episodes required). In the GENITOR method [11], the normal evolution algorithm was used to evolve the weights in a fully-connected, two-layer neural network, with additional connections from each input unit to the output layer. The TDGAR [12] learning scheme is a new hybrid GA, which integrates the TD prediction method and the GA to fulfill the reinforcement learning task. The CQGAF [15] accomplishes the GA-based fuzzy system design in a reinforcement learning environment, where only weak reinforcement signals, such as “success” and “failure” are available. The network consists of five input, five

Rule$_1$: IF x_1 is $\mu(0.284411,-0.01446)$ and x_2 is $\mu(0.727959,0.146663)$ and x_3 is $\mu(0.694891,1.43122)$ and x_4 is $\mu(0.932191,0.012762)$

$$\text{THEN }\hat y_1=-1.44628-0.13254x_1-0.04798\cos(\pi x_1)+0.177318\sin(\pi x_1)+0.382741x_2-0.79246\cos(\pi x_2)+0.911939\sin(\pi x_2)-0.17453x_3+0.263316\cos(\pi x_3)-1.05748\sin(\pi x_3)+0.763629x_4+0.336507\cos(\pi x_4)+0.19016\sin(\pi x_4)+0.776083x_1x_2x_3x_4$$

Rule$_2$: IF x_1 is $\mu(0.813759,0.033526)$ and x_2 is $\mu(-0.00915,0.961607)$ and x_3 is $\mu(0.002588,0.57906)$ and x_4 is $\mu(0.592518,0.461347)$

$$\text{THEN }\hat y_1=-0.27423-0.32326x_1+0.919114\cos(\pi x_1)+1.83531\sin(\pi x_1)-0.9622x_2+0.410156\cos(\pi x_2)+0.027012\sin(\pi x_2)+1.0483x_3+0.455301\cos(\pi x_3)+2.41398\sin(\pi x_3)+0.670275x_4+0.323742\cos(\pi x_4)-0.73245\sin(\pi x_4)-1.10604x_1x_2x_3x_4$$

Rule$_3$: IF x_1 is $\mu(1.19906,1.78198)$ and x_2 is $\mu(-0.43297,0.108291)$ and x_3 is $\mu(-0.65957,-0.40318)$ and x_4 is $\mu(0.279944,0.13257)$

$$\text{THEN }\hat y_1=-0.36624-0.13325x_1-1.8409\cos(\pi x_1)+0.235678\sin(\pi x_1)+0.062486x_2+1.30283\cos(\pi x_2)+0.881228\sin(\pi x_2)-0.15466x_3+0.485734\cos(\pi x_3)+0.720555\sin(\pi x_3)+0.560237x_4+1.2791\cos(\pi x_4)-1.44565\sin(\pi x_4)+0.771205x_1x_2x_3x_4$$

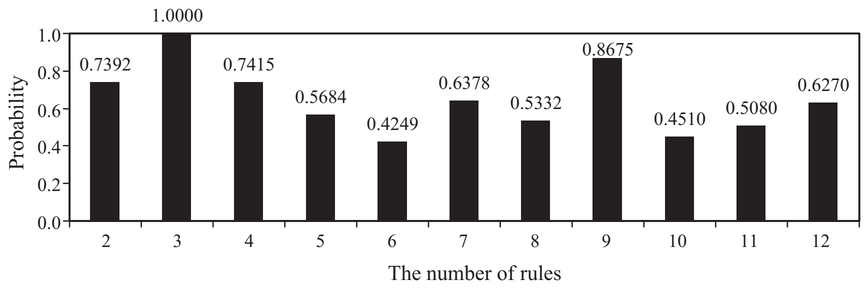

Fig. 14. The probability vectors of the ESS step in the proposed REL.

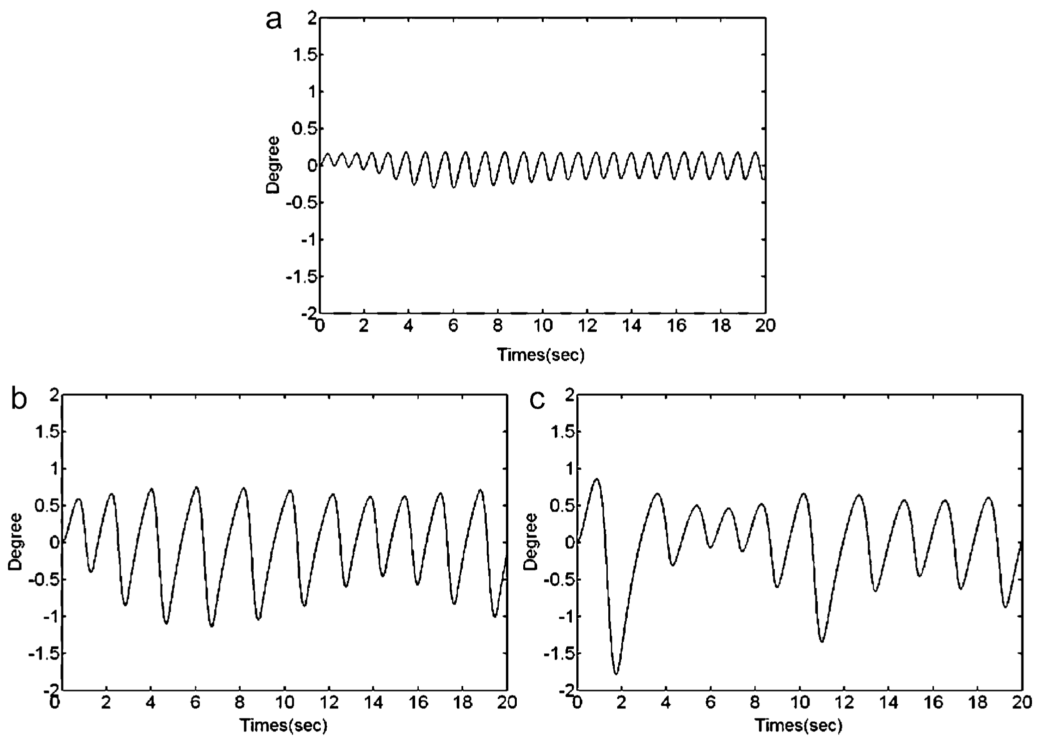

Fig. 15. Angular deviation of the pole by a trained (a) the REL method, (b) the R-PSO method [22], and (c) the R-CPSO method [28].

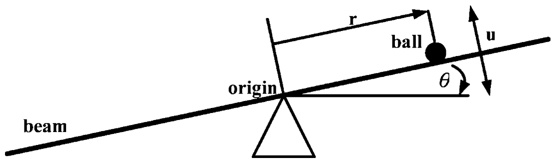

Fig. 16. The ball and beam system.

hidden, and one output units. In the SANE approach [17], the symbiotic evolution algorithm is used to evolve a two-layer neural network with five input, eight hidden, and two output units. An individual in the SANE represents a hidden unit with five specified connections to the input and output units. As indicated in Table 2, the proposed REL algorithm is feasible and effective.

Example 2. Control of a ball and beam system

A ball and beam system [34] is shown in Fig. 16. The beam is made to rotate in a vertical plane when a torque is applied at the center of rotation. The ball is free to roll along the beam, while it is required that the ball remains in contact with the beam.

The ball and beam system can be written in the state-space form as

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix}x_2\\ B(x_1x_4^2 - G\sin x_3)\\ x_4\\ 0\end{bmatrix} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix}u, \qquad (19)$$
$$y = x_1$$

where $x = (x_1, x_2, x_3, x_4)^T \equiv (r, \dot{r}, \theta, \dot{\theta})^T$ is the state of the system and $y = x_1 \equiv r$ represents the system output. The control parameter u is the angular acceleration ($\ddot{\theta}$), and the parameters $B = 0.7143$ and $G = 9.81$ are chosen for this control system. The purpose of the controller u is to determine $u(x)$ such that the closed-loop system-output y will converge to zero from different initial conditions.

According to the input/output-linearization algorithm [34], the control law $u(x)$ is determined as follows:

For state x, compute $v(x) = -\alpha_3\phi_4(x) - \alpha_2\phi_3(x) - \alpha_1\phi_2(x) - \alpha_0\phi_1(x)$, where $\phi_1(x) = x_1$, $\phi_2(x) = x_2$, $\phi_3(x) = -BG\sin x_3$, $\phi_4(x) = -BGx_4\cos x_3$, and α_i is chosen so that $s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0$ is a Hurwitz polynomial. Compute $a(x) = -BG\cos x_3$ and $b(x) = BGx_4^2 \sin x_3$. Then $u(x) = [v(x) - b(x)]/a(x)$.

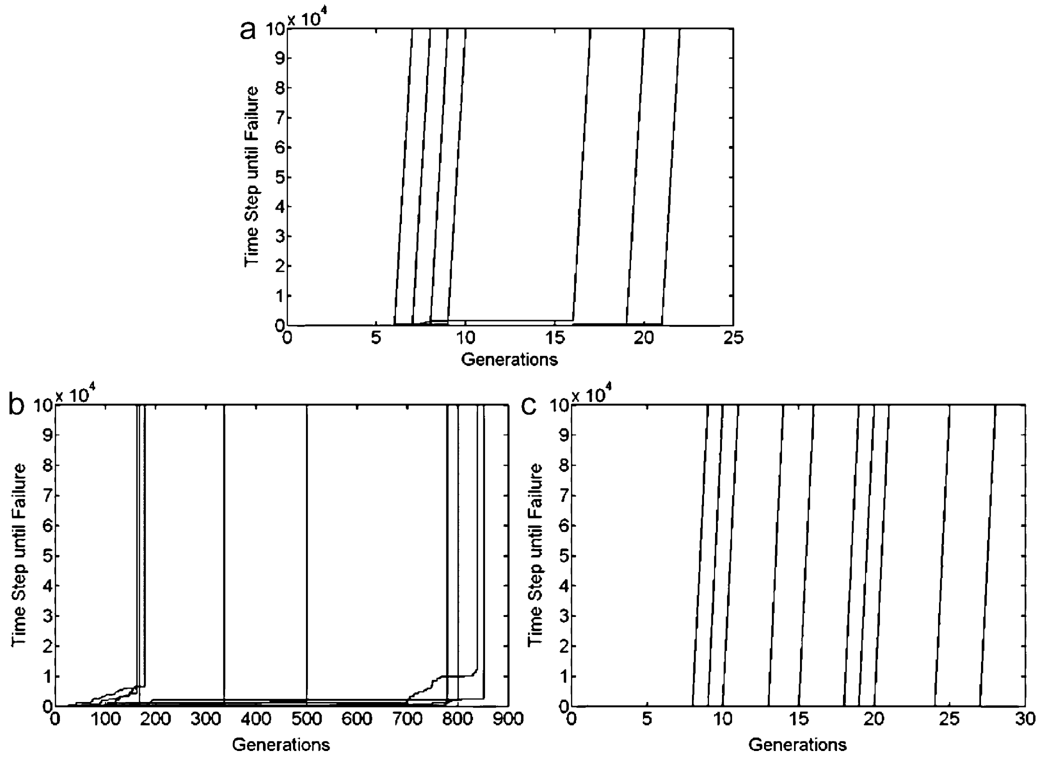

Fig. 17. The performance of (a) the REL method, (b) the R-PSO method [22], and (c) the R-CPSO method [28] on the ball and beam balancing system.

The four input variables $(r, \dot{r}, \theta, \dot{\theta})$ and the output $u(k)$ are normalized between 0 and 1 over the following ranges: r:[−5,5], $\dot{r}$: [−3, 3]:[−3,3], θ:[−1,1], $\dot{\theta}$: [−2, 2], and u:[−70,70]. As stated in the previous example, the initial floating-point values are assigned by using the REL algorithm. In the proposed REL method, the fitness function in this example is also defined in Eq. (14) to train the SENFIN model, where Eq. (14) states how long the ball and beam system fails and receives a penalty signal of −1 when the beam deviates beyond a certain angle ($|\theta| > 12^\circ$) and the ball reaches the end of the beam ($|r| > 2$ m). A ten-run simulation is conducted, while each run started from the same initialization state. Fig. 17(a) shows that the SENFIN model learned on average to balance the ball at the 12th generation. In this figure, each run results in the largest fitness value in the current generation being selected before the ball and beam system fails. After the learning process was stopped, we chose the best string in the population of the final generation and tested it on the ball and beam system. The simulated results of the probability vectors for the ESS method are presented in Fig. 18, where the final average optimal number of rules is 4. The obtained fuzzy rules of the SENFIN using the REL method are summarized as follows:

$$\begin{aligned}
\text{Rule}_1: \quad & \text{IF } x_1 \text{ is } \mu(0.284411, -0.01446) \text{ and } x_2 \text{ is } \mu(0.727959, 0.146663) \\
& \text{and } x_3 \text{ is } \mu(0.075622, 0.696157) \text{ and } x_4 \text{ is } \mu(-0.16145, 1.05776) \\
& \text{THEN } \hat{y}_1 = -0.5719 - 0.176952x_1 + 0.695774\cos(\pi x_1) + 0.53314\sin(\pi x_1) \\
& \qquad -0.25504x_2 - 1.6741\cos(\pi x_2) - 0.55762\sin(\pi x_2) \\
& \qquad -0.18066x_3 + 1.06147\cos(\pi x_3) + 0.393831\sin(\pi x_3) \\
& \qquad -0.1008x_4 + 0.322637\cos(\pi x_4) + 1.88298\sin(\pi x_4) \\
& \qquad -0.01681x_1x_2x_3x_4 \\
\text{Rule}_2: \quad & \text{IF } x_1 \text{ is } \mu(0.764218, 1.13001) \text{ and } x_2 \text{ is } \mu(-0.00824, 0.547146) \\
& \text{and } x_3 \text{ is } \mu(0.606443, 1.23632) \text{ and } x_4 \text{ is } \mu(1.23826, -1.02783) \\
& \text{THEN } \hat{y}_1 = -0.7883 - 1.11792x_1 + 1.8139\cos(\pi x_1) - 0.10719\sin(\pi x_1) \\
& \qquad +0.99488x_2 + 1.40969\cos(\pi x_2) + 1.63694\sin(\pi x_2) \\
& \qquad -0.61266x_3 + 0.878479\cos(\pi x_3) - 0.17153\sin(\pi x_3) \\
& \qquad -0.10053x_4 + 0.677443\cos(\pi x_4) - 1.41097\sin(\pi x_4) \\
& \qquad +0.905048x_1x_2x_3x_4 \\
\text{Rule}_3: \quad & \text{IF } x_1 \text{ is } \mu(1.42665, 0.373173) \text{ and } x_2 \text{ is } \mu(0.438513, -0.64736) \\
& \text{and } x_3 \text{ is } \mu(-0.17628, 1.36622) \text{ and } x_4 \text{ is } \mu(1.25773, -0.27832) \\
& \text{THEN } \hat{y}_1 = -0.50156 - 0.1015x_1 - 0.868533\cos(\pi x_1) + 0.586759\sin(\pi x_1) \\
& \qquad +0.563745x_2 + 0.376822\cos(\pi x_2) - 0.64332\sin(\pi x_2) \\
& \qquad -1.2245x_3 + 0.246941\cos(\pi x_3) + 0.1687\sin(\pi x_3) \\
& \qquad -1.08746x_4 + 0.903228\cos(\pi x_4) - 0.69598\sin(\pi x_4) \\
& \qquad -0.70899x_1x_2x_3x_4 \\
\text{Rule}_4: \quad & \text{IF } x_1 \text{ is } \mu(0.07984, 1.41799) \text{ and } x_2 \text{ is } \mu(1.88053, 0.191622) \\
& \text{and } x_3 \text{ is } \mu(0.769444, -0.42783) \text{ and } x_4 \text{ is } \mu(0.780801, 0.532372) \\
& \text{THEN } \hat{y}_1 = 2.09824 + 0.664375x_1 + 1.02052\cos(\pi x_1) + 0.469765\sin(\pi x_1) \\
& \qquad +1.77397x_2 + 0.310921\cos(\pi x_2) + 1.31612\sin(\pi x_2) \\
& \qquad -1.18918x_3 + 0.119524\cos(\pi x_3) + 0.56419\sin(\pi x_3) \\
& \qquad +0.796049x_4 + 0.738078\cos(\pi x_4) - 0.237018\sin(\pi x_4) \\
& \qquad +0.538203x_1x_2x_3x_4
\end{aligned}$$

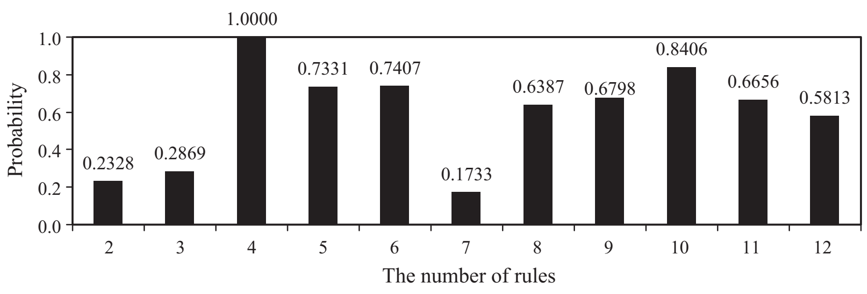

Fig. 18. The probability vectors of the ESS step in the proposed REL.

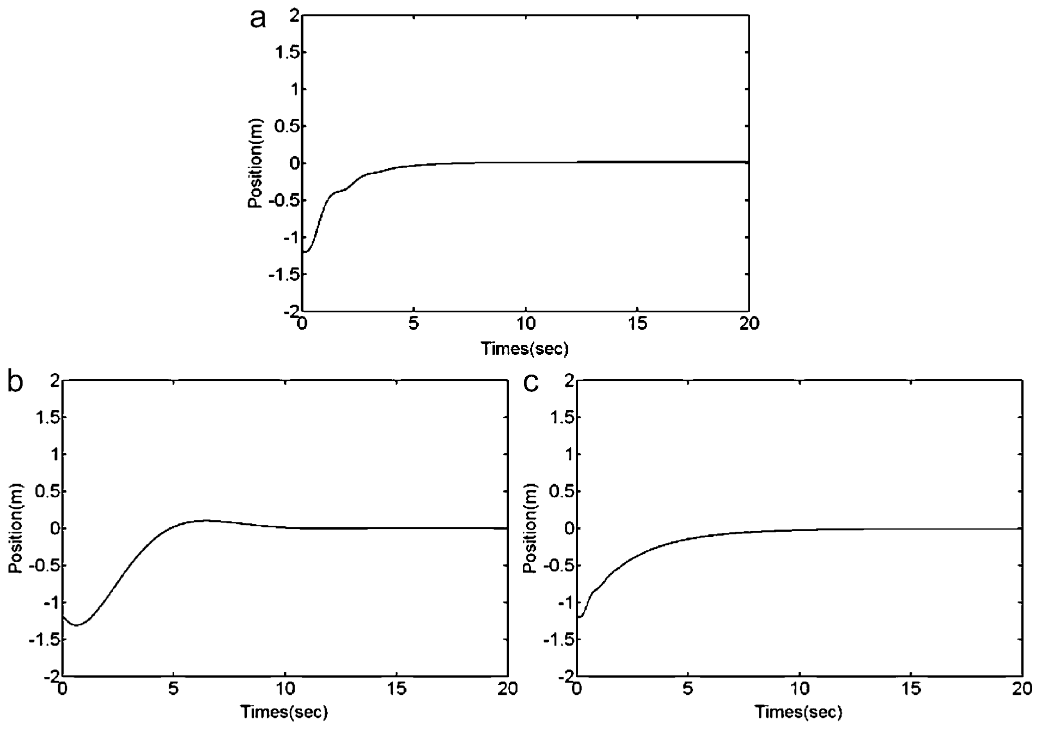

Fig. 19. Position deviation of the ball by a trained (a) the REL method, (b) the R-PSO method [21], and (c) the R-CPSO method [28].

Fig. 19(a) demonstrates the position deviation of the ball, when the ball and beam system was controlled by the well-trained SENFIN model starting at the initial state that $x(0) = -1.2$, $\dot{x}(0) = -0.01$, $\theta(0) = 0.58$, and $\dot{\theta}(0) = 0.58$. In this figure, the position of the ball decays to zero gradually. Consequently, the experimental results prove that the trained SENFIN model has great capability of controlling the ball and beam balancing system.

As well as in Example 1, in this example we compared the performances of our method with the R-PSO method [22] and the R-CPSO method [28], while the same settings as used in Example 1 are applied. Fig. 17(b) and (c) illustrate that the R-PSO and R-CPSO methods learned, on average, to balance the ball in the 478-th and 17-th generations, respectively. Fig. 19(b) and (c) present the position deviations of the ball, when the ball and beam system was controlled by the R-PSO and the R-CPSO methods, starting by the initial state with $r(0) = -1.2$, $\dot{r}(0) = -0.01$, $\dot{\theta}(0) = 0.58$, and $\theta(0) = 0.58$. As exhibited in Figs. 17 and 19, the control capabilities of the trained SENFIN model using the REL algorithm are also better than those in [22,28] in the ball and beam balancing system. Table 3 summarizes a performance comparison of various existing models ([11,12,17,18,22,28,15,34]) in Example 2. As the result, the performance indices (i.e., mean, best, and worst generations) of the proposed learning method outperform the methods in [11,12,17,18,22,28,15,34].

Table 3
Performance comparison of various existing models in Example 2.

Method	Mean	Best	Worst
GENITOR [11]	4982	551	19,853
TDGAR [12]	210	30	324
CQGAF [15]	187	23	298
SANE [17]	2287	150	6217
R-SE [18]	274	32	492
R-PSO [22]	478	163	852
R-CPSO [28]	17	9	28
R-GA [34]	466	97	678
REL	12	7	22

7. Conclusion and future works

This study proposes an efficient reinforcement evolutionary learning algorithm (REL) for NFINs (SENFIN). The proposed REL consists of structure learning that can determine the number of fuzzy rules and parameter learning that can adjust parameters of SENFIN model. The structure learning adopts a subgroup symbiotic evolution (SSE) to yield several variable fuzzy systems and uses an elite-based structure strategy (ESS) to find suitable the number of fuzzy rules. The parameter learning uses CCPSO that employs cooperative behavior among multiple swarms can increase the global search capacity using the belief space. The advantages of the proposed SENFIN-REL method are as follows; (1) the proposed REL can automatically construct SENFIN and adjust parameters of SENFIN; (2) the consequent of the fuzzy rules of SENFIN model involves a nonlinear combination of input variables. The experimental results demonstrate that the proposed SENFIN-REL performs better than the other methods for solving reinforcement learning problems. One advanced topics for the proposed REL method should be addressed in future research. The computational complexity of REL method should be decreased in subgroup symbiotic evolution. The crowding distance operator [35] presented in NSGA-II, can be adopted like fast non-dominated sorting in the solution space according to each objective function.

References

[1] C.T. Lin, C.S.G. Lee, Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems, Prentice-Hall, NJ, May 1996.
[2] N. Kasabov, Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, MIT Press, Cambridge, MA, 1996.
[3] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (3) (1993) 665–685.
[4] C.F. Juang, C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Syst. 6 (February (1)) (1998) 12–31.
[5] N.K. Kasabov, Q. Song, DESENFIN: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Trans. Fuzzy Syst. 10 (April (2)) (2002) 144–154.
[6] F. Sun, Z. Sun, L. Li, H.X. Li, Neuro-fuzzy adaptive control based on dynamic inversion for robotic manipulators, Fuzzy Sets Syst. 134 (February (1)) (2003) 117–133.
[7] C. Li, C.Y. Lee, Self-organizing neuro-fuzzy system for control of unknown plants, IEEE Trans. Fuzzy Syst. 1 (February (1)) (2003) 135–150.
[8] C.J. Lin, C.C. Chin, Prediction and identification using wavelet-based recurrent fuzzy neural networks, IEEE Trans. Syst. Man Cybern. Part B 34 (October (5)) (2004) 2144–2154.
[9] C.H. Chen, C.J. Lin, C.T. Lin, A functional-link-based neuro-fuzzy network for nonlinear system control, IEEE Trans. Fuzzy Syst. 16 (October (5)) (2008) 1362–1378.
[10] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuron like adaptive elements that can solve difficult learning control problem, IEEE Trans. Syst. Man Cybern. 13 (5) (1983) 834-847.
[11] D. Whitley, S. Dominic, R. Das, C.W. Anderson, Genetic reinforcement learning for neuro control problems, Mach. Learn. 13 (1993) 259–284.
[12] C.T. Lin, C.P. Jou, GA-based fuzzy reinforcement learning for control of a magnetic bearing system, IEEE Trans. Syst. Man Cybern. Part B 30 (April (2)) (2000) 276–289.
[13] Y.H. Kim, L. Lewis, Reinforcement adaptive learning neural-net-based friction compensation control for high speed and precision, IEEE Trans. Control Syst. Technol. January (8) (1) (2000) 118–126.
[14] H. Pingan, S. Jagannathan, Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints, IEEE Trans. Syst. Man Cybern. Part B 37 (April (2)) (2007) 425–436.
[15] C.F. Juang, Combination of online clustering and Q-value based GA for reinforcement fuzzy system design, IEEE Trans. Fuzzy Syst. 13 (June (3)) (2005) 289–302.
[16] D. Zhao, J. Yi, D. Liu, Particle swarm optimized adaptive dynamic programming, in: Proc. of IEEE Int. Symposium on Approximate Dynamic Programming and Reinforcement Learning, April, 2007, pp. 32–37.
[17] D.E. Moriarty, R. Miikkulainen, Efficient reinforcement learning through symbiotic evolution, Mach. Learn. 22 (1996) 11–32.
[18] C.F. Juang, J.Y. Lin, C.T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design, IEEE Trans. Syst. Man Cybern. Part B 30 (April (2)) (2000) 290–302.
[19] C.J. Lin, Y.J. Xu, A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications, Fuzzy Sets Syst. 157 (April (8)) (2006) 1036–1056.
[20] G.R. Harik, F.G. Lobo, D.E. Goldberg, The compact genetic algorithm, IEEE Trans. Evol. Comput. 3 (November (4)) (1999) 287–297.
[21] C.J. Lin, Y.J. Xu, A hybrid evolutionary learning algorithm for TSK-type fuzzy model design, Math. Comput. Model. 43 (March) (2006) 563–581.
[22] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. of IEEE Int. Conf. on Neural Networks, vol. 4, December, 1995, pp. 1942–1948.
[23] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proc. of the Sixth International Symposium on Micro Machine and Human Science, October, 1995, pp. 39–43.
[24] Z.L. Gaing, A particle swarm optimization approach for optimum design of PID controller in AVR system, IEEE Trans. Energy Convers. 19 (June (2)) (2004) 384–391.
[25] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Trans. Power Syst. 15 (November (4)) (2000) 1232–1239.
[26] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern. Part B 34 (April (2)) (2004) 997–1006.
[27] R. Mendes, P. Cortez, M. Rocha, J. Neves, Particle swarms for feedforward neural network training, in: Proc. of IEEE Int. Joint Conf. on Neural Networks, 2002, pp. 1895–1899.
[28] F. van den Bergh, A.J.T.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (June (3)) (2004) 225–239.
[29] R.G. Reynolds, An introduction to cultural algorithms, in: Proc. of the Third Annual Conf. on Evolution Programming, February, 1994, pp. 131–139.
[30] X. Jin, R.G. Reynolds, Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach, in: Proc. of IEEE Congress on Evolutionary Computation, vol. 3, July, 1999, pp. 1672–1678.
[31] S. Saleem, R. Reynolds, Cultural algorithms in dynamic environments, in: Proc. of IEEE Congress on Evolutionary Computation, vol. 2, July, 2000, pp. 1513–1520.
[32] J.C. Patra, R.N. Pal, B.N. Chatterji, G. Panda, Identification of nonlinear dynamic systems using functional link artificial neural networks, IEEE Trans. Syst. Man Cybern. Part B 29 (April (2)) (1999) 254–262.
[33] J.C. Patra, R.N. Pal, A functional link artificial neural network for adaptive channel equalization, Signal Process. 43 (May) (1995) 181–195.
[34] C.L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, in: Proc. of the Fourth Int. Conf. on Genetic Algorithms, vol. 1, 1991, pp. 450–457.
[35] K. Deb, A. Pratap, S. Agrawal, T. Meyarian, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.