Applied Soft Computing 11 (2011) 4847–4858

Contents lists available at ScienceDirect

Applied Soft Computing

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

A Rule-Based Symbiotic MOdified Differential Evolution for Self-Organizing Neuro-Fuzzy Systems

Miin-Tsair Su [a], Cheng-Hung Chen [b], Cheng-Jian Lin [c,*], Chin-Teng Lin [a]

[a] Department of Electrical Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, ROC
[b] Department of Electrical Engineering, National Formosa University, Yunlin County 632, Taiwan, ROC
[c] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan, ROC

ARTICLE INFO

Article history:
Received 9 March 2009
Received in revised form 24 March 2011
Accepted 26 June 2011
Available online 8 July 2011

Keywords:
Neuro-fuzzy systems
Symbiotic evolution
Differential evolution
Entropy measure
Control

ABSTRACT

This study proposes a Rule-Based Symbiotic MOdified Differential Evolution (RSMODE) for Self-Organizing Neuro-Fuzzy Systems (SONFS). The RSMODE adopts a multi-subpopulation scheme that uses each individual represents a single fuzzy rule and each individual in each subpopulation evolves separately. The proposed RSMODE learning algorithm consists of structure learning and parameter learning for the SONFS model. The structure learning can determine whether or not to generate a new rule-based subpopulation which satisfies the fuzzy partition of input variables using the entropy measure. The parameter learning combines two strategies including a subpopulation symbiotic evolution and a modified differential evolution. The RSMODE can automatically generate initial subpopulation and each individual in each subpopulation evolves separately using a modified differential evolution. Finally, the proposed method is applied in various simulations. Results of this study demonstrate the effectiveness of the proposed RSMODE learning algorithm.

1. Introduction

Neuro-fuzzy systems (NFSs) [1–3] have been demonstrated to solving many engineering problems. They combine the capability of neural networks to learn from processes and the capability of fuzzy reasoning under linguistic information pertaining to numerical variables. On the other hand, recent development in genetic algorithms (GAs) has provided a method for neuro-fuzzy system design. Genetic fuzzy systems (GFSs) [4–6] hybridize the approximate reasoning of fuzzy systems with the learning capability of genetic algorithms.

GAs represent highly effective techniques for evaluating system parameters and finding global solutions while optimizing the overall structure. Thus, many researchers have developed GAs to implement fuzzy systems and neuro-fuzzy systems in order to automate the determination of structures and parameters [7–16]. Carse et al. [7] presented a GA-based approach to employ variable length rule sets and simultaneously evolves fuzzy membership functions and relations called Pittsburgh-style fuzzy classifier system. Herrera et al. [8] proposed a genetic algorithm-based tuning approach for the parameters of membership functions used to define fuzzy rules. This approach relied on a set of input–output training data and minimized a squared-error function defined in terms of the training data. Homaifar and McCormick [9] presented a method that simultaneously found the consequents of fuzzy rules and the center points of triangular membership functions in the antecedent using genetic algorithms. Velasco [10] described a Michigan approach which generates a special place where rules can be tested to avoid the use of bad rules for online genetic learning. Ishibuchi et al. [11] applied a Michigan-style genetic fuzzy system to automatically generate fuzzy IF-THEN rules for designing compact fuzzy rule-based classification systems. The genetic learning process proposed is based on the iterative rule learning approach and it can automatically design fuzzy rule-based systems by Cordon et al. [12]. A GA-based learning algorithm called structural learning algorithm in a vague environment (SLAVE) was proposed in [13]. SLAVE used an iterative approach to include more information in the process of learning one individual rule.

Moreover, a very interesting algorithm was proposed by Russo in [14] which attempted to combine all good features of fuzzy systems, neural networks and genetic algorithm for fuzzy model derivation from input–output data. Chung et al. [15] adopted both neural networks and GAs to automatically determine the parameters of fuzzy logic systems. They utilized a feedforward neural network for realizing the basic elements and functions of a fuzzy controller. In [16], a hybrid of evolution strategies and simulated annealing algorithms is employed to optimize membership function parameters and rule numbers which are combined with genetic parameters.

* 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.2011.06.015

In the aforementioned literatures, it has been fully demonstrated that GAs are very powerful in searching for the true profile. However, the search is extremely time-consuming, which is one of the basic disadvantages of all GAs. Although the convergence in some special cases can be improved by hybridizing GAs with some local search algorithms, it is achieved at the expense of the versatility and simplicity of the algorithm. Similar to GAs, differential evolution (DE) [17–19] also belongs to the broad class of evolutionary algorithms, but DE has many advantages such as the strong search ability and the fast convergence ability over GAs or any other traditional optimization approach, especially for real valued problems [19]. In addition, the DE algorithm has gradually become more popular and has been used in many practical areas, mainly many researches [20–24] demonstrated that DE is robust, simple in implementation and use, easy to understand, and requires only a few control parameters for particle swarm optimization (PSO), original GA and some modified GAs.

This study proposes a RSMODE for a SONFS. The neuro-fuzzy system is based on our previous research [25], and combines a fuzzy system with a functional link neural network (FLNN) [26]. The consequent part of the fuzzy rules that corresponds to an FLNN comprises the functional expansion of input variables.

The proposed RSMODE learning algorithm consists of structure learning to generate initial rule-based subpopulation, and parameter learning to adjust the SONFS parameters. The structure learning can determine whether or not to generate a new rule-based subpopulation which satisfies the fuzzy partition of input variables. Initially, there is not any subpopulation. The rule-based subpopulation is automatically generated from training data by entropy measure. The parameter learning combines two strategies including a subpopulation symbiotic evolution (SSE) and a modified differential evolution (MODE). The SSE in which each individual represents a single fuzzy rule differs from original symbiotic evolution [27]. Each subpopulation allows the rule itself to evolve. The MODE adopts a method to effectively search between the best individual and randomly chosen individuals.

This study is organized as follows. Section 2 describes the structure of the Self-Organizing Neuro-Fuzzy system. Section 3 presents the Rule-Based Symbiotic MOdified Differential Evolution. Next, Section 4 presents the results of simulations of various problems. Finally, the last section draws conclusions.

2. Structure of SONFS

This subsection describes the SONFS [25], which uses a nonlinear combination of input variables (FLNN) [26]. Each fuzzy rule corresponds to the FLNN, comprising a functional expansion of input variables. The SONFS model realizes a fuzzy if-then rule in the following form.

$Rule_j$: IF $\hat{x}_1$ is A_{1j} and $\hat{x}_2$ is A_{2j}... and $\hat{x}_i$ is A_{ij}... and $\hat{x}_N$ is A_{Nj}

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

where $\hat{x}_i$ and $\hat{y}_j'$ are the input and local output variables, respectively; A_{ij} is the linguistic term of the precondition part; 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 jth fuzzy rule.

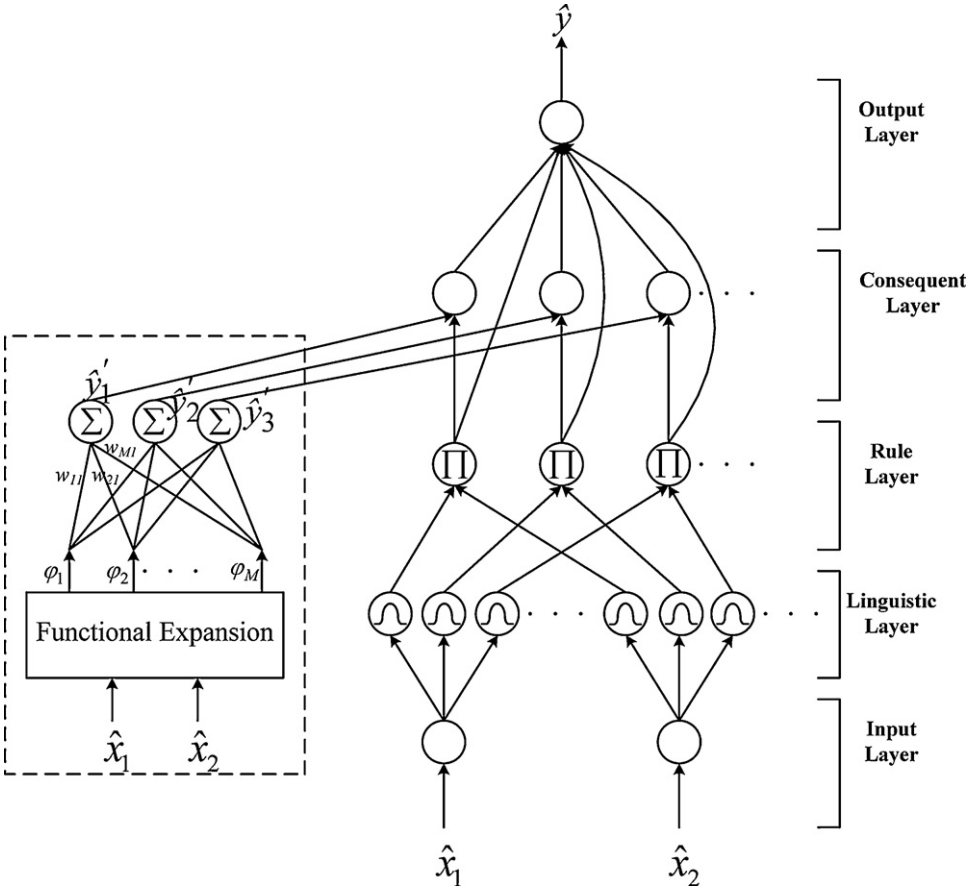

Fig. 1. Structure of the SONFS model.

The structure of the SONFS model is shown in Fig. 1, in which linguistic layer by a Gaussian-type membership function, $\mu_{A_{ij}}(\hat{x}_i)$, defined by

$$\mu_{A_{ij}}(\hat{x}_i) = \exp\left(-\frac{[\hat{x}_i - m_{ij}]^2}{\sigma_{ij}^2}\right) \tag{2}$$

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

The collection of fuzzy sets $\mathbf{A}_j = \{A_{1j}, ..., A_{Nj}\}$ pertaining to the precondition part of $Rule_j$ forms a fuzzy region that can be regarded as a multi-dimensional fuzzy set whose membership function is determined by

$$\mu_{A_j}(\hat{x}) = \prod_{i=1}^{N} \mu_{A_{ij}}(\hat{x}_i) \tag{3}$$

Nodes in consequent layer only receive the signal, which are the output from rule layer and a functional link neural network. Finally, the output node integrates all of the actions recommended by rule layer and consequent layer and acts as a defuzzifier with,

$$\hat{y} = \frac{\sum_{j=1}^{R} \mu_{A_j}(\hat{x})\hat{y}_j'}{\sum_{j=1}^{R} \mu_{A_j}(\hat{x})} = \frac{\sum_{j=1}^{R} \mu_{A_j}(\hat{x}) \left(\sum_{k=1}^{M} w_{kj}\phi_k\right)}{\sum_{j=1}^{R} \mu_{A_j}(\hat{x})} \tag{4}$$

where R is the number of fuzzy rules, $\hat{y}$ is the output of the SONFS model, w_{kj} is the corresponding link weight of functional link neural network, and ϕ_k is the functional expansion of input variables [26]. The functional expansion uses a trigonometric polynomial basis function, given by $[\hat{x}_1 \ \sin(\pi\hat{x}_1) \ \cos(\pi\hat{x}_1) \ \hat{x}_2 \ \sin(\pi\hat{x}_2) \ \cos(\pi\hat{x}_2)]$ 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.

3. A rule-based symbiotic modified differential evolution for the SONFS model

This section represents the proposed RSMODE for the SONFS. The RSMODE comprises structure learning and parameter learning. The structure learning uses the entropy measure that determines proper input space partitioning and finds the mean and variance of the Gaussian membership function and the number of rules. Next, the initial rule-based subpopulation is created according to a range of the mean and variance of the membership function. The parameter learning consists of a subpopulation symbiotic evolution (SSE) and a modified differential evolution (MODE). Each individual in each subpopulation evolves separately using a modified differential evolution. But in order to evaluate each individual, the individual is composed a fuzzy system using other individuals (rules) in other subpopulations. The detailed flowchart of the proposed RSMODE learning algorithm is presented in Fig. 2.

3.1. Structure learning

In this study, we can finish the structure learning from training data in the first generation. This subsection introduces the production of initial rule-based subpopulation, covering the coding and initialization steps. The coding step involves the membership functions and the fuzzy rules of a fuzzy system that represent individuals and are suitable for subpopulation symbiotic evolution. The initialization step assigns the number of subpopulation before the evolution process begins.

3.1.1. Coding step

The first step in RSMODE learning algorithm is the coding of a fuzzy rule into an individual. Fig. 3 shows an example of a fuzzy rule coded into an individual where i and j are the ith dimension and the jth rule. Fig. 3 describes a fuzzy rule given by Eq. (1), where m_{ij} and σ_{ij} are the mean and variance of a Gaussian membership function, respectively, and w_{kj} represents the corresponding link weight of the consequent part that is connected to the jth rule node. In this study, a real number represents the position of each individual.

3.1.2. Initialization step

For training data, finding the optimal solution is difficult because the range of training data is wide. Therefore, the data must be normalized. Let training data be transformed to the interval of [0,1]:

$$\hat{x}_i = \frac{\hat{x}_i' - \hat{x}_{i\_\min}'}{\hat{x}_{i\_\max}' - \hat{x}_{i\_\min}'} \tag{5}$$

where $\hat{x}_i$ is the value after normalization; $\hat{x}_i'$ is the vector of the ith dimension to be normalized; $\hat{x}_{i\_\min}'$ is the minimum value of vector $\hat{x}_i'$; $\hat{x}_{i\_\max}'$ is the maximum value of vector $\hat{x}_i'$.

Before the RSMODE method is designed, the individuals that will constitute R initial subpopulation must be created. The first step in structure learning is to create the initial first individual in each subpopulation to satisfy the fuzzy rule partition of input variables. The fuzzy rule partition strategy can determine whether a new rule should be extracted from the training data and determine the number of fuzzy rules in the universal of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule. For each incoming data $\hat{x}_i'$, the rule firing strength can be regarded as the degree to which the incoming data belongs to the corresponding cluster. Entropy measure between each data point and each membership function is calculated based on a similarity measure. A data point of closed mean will has lower entropy. Therefore, the entropy values between data points and current membership functions are calculated to determine whether or not to add a new rule into the initial first individual and create a new rule-based subpopulation space. For computational efficiency, the entropy measure can be calculated using the firing strength from $\mu_{A_{ij}}(\hat{x}_i)$ as follows:

$$EM_j = -\sum_{i=1}^{N} D_{ij} \ \log_2 \ D_{ij} \tag{6}$$

where $D_{ij} = \exp(u_{A_{ij}}(\hat{x}_i)^{-1})$ and $EM_j \in [0, 1]$. According to Eq. (6), the measure is used to generate a new fuzzy rule and new functional link bases for new incoming data is described as follows. The maximum entropy measure

$$EM_{\max} = \max_{1 \le j \le R} EM_j \tag{7}$$

is determined, where R is the number of existing rules. If $EM_{\max} \le \overline{EM}$, then a new rule and a new rule-based subpopulation space are generated, where $\overline{EM} \in [0, 1]$ is a prespecified threshold.

Once a new rule has been generated, the next step is to assign the initial first individual in the new rule-based subpopulation by the initial mean and variance to the new membership function and the corresponding link weight. Hence, the mean, variance and weight for the new rule are set as follows:

$$m_{ij} = \hat{x}_i \tag{8}$$

$$\sigma_{ij} = \sigma_{\text{init}} \tag{9}$$

$$w_{kj} = \text{random}[-1, \ 1] \tag{10}$$

where $\hat{x}_i$ is the current input data and σ_{init} is a prespecified constant.

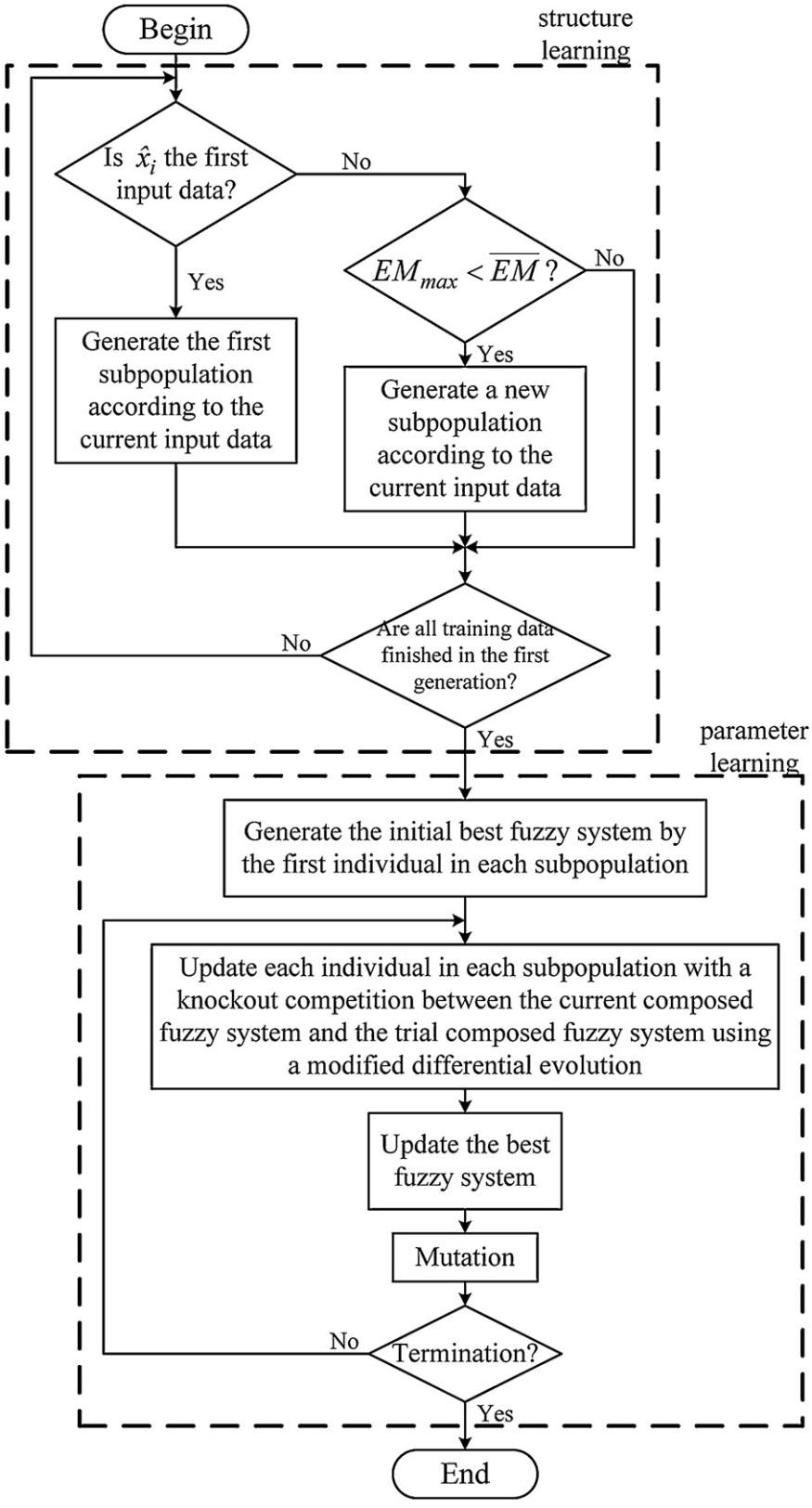

Fig. 2. Flowchart of the proposed RSMODE method.

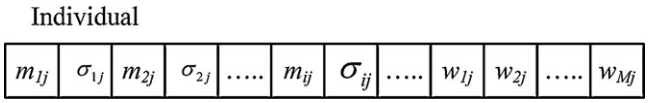

Fig. 3. Coding a fuzzy rule into an individual in the proposed RSMODE method.

The second step is to create other individuals in each subpopulation according to a range of the initial first individual. The following formulations show the production of the other individuals.

$$\text{Mean}: \quad \text{Individual}\ [d] = m_{ij} + \text{random}\ [0, 1] \times \sigma_{ij},$$
$$\text{where}\ \ d = 1, 3, \ldots, 2 \times N - 1 \tag{11}$$

$$\text{Variance}: \quad \text{Individual}\ [d] = 2 \times \text{random}\ [0, 1] \times \sigma_{ij},$$
$$\text{where}\ \ d = 2, 4, \ldots, 2 \times N \tag{12}$$

$$\text{Other parameters}: \quad \text{Individual}\ [d] = \text{random}\ [-1,\ 1],$$
$$\text{where}\ \ d > 2 \times N \tag{13}$$

where d is the site of each individual and m_{ij} and σ_{ij} are the corresponding mean and variance, respectively, of the initial first individual.

3.2. Parameter learning

The parameter learning combines two strategies including a subpopulation symbiotic evolution (SSE) and a modified differential evolution (MODE). Each subpopulation allows the individual (rule) itself to evolve by evaluating the composed fuzzy system. Fig. 4 shows the structure of the individual in the RSMODE. The parameter learning process is described step-by-step below.

Step 1: Generate the initial best Fuzzy system

In this step, we orderly select the first individual from each subpopulation, and compose a fuzzy system as the initial best fuzzy system.

Step 2: Update each individual in each subpopulation using MODE

In order to update each individual in each subpopulation, we use a modified differential evolution to select the better individual to the next step. Fig. 5 gives an example of the MODE process. Hence, this step comprises of three components – parent choice phase, offspring generation phase, and survivor selection phase.

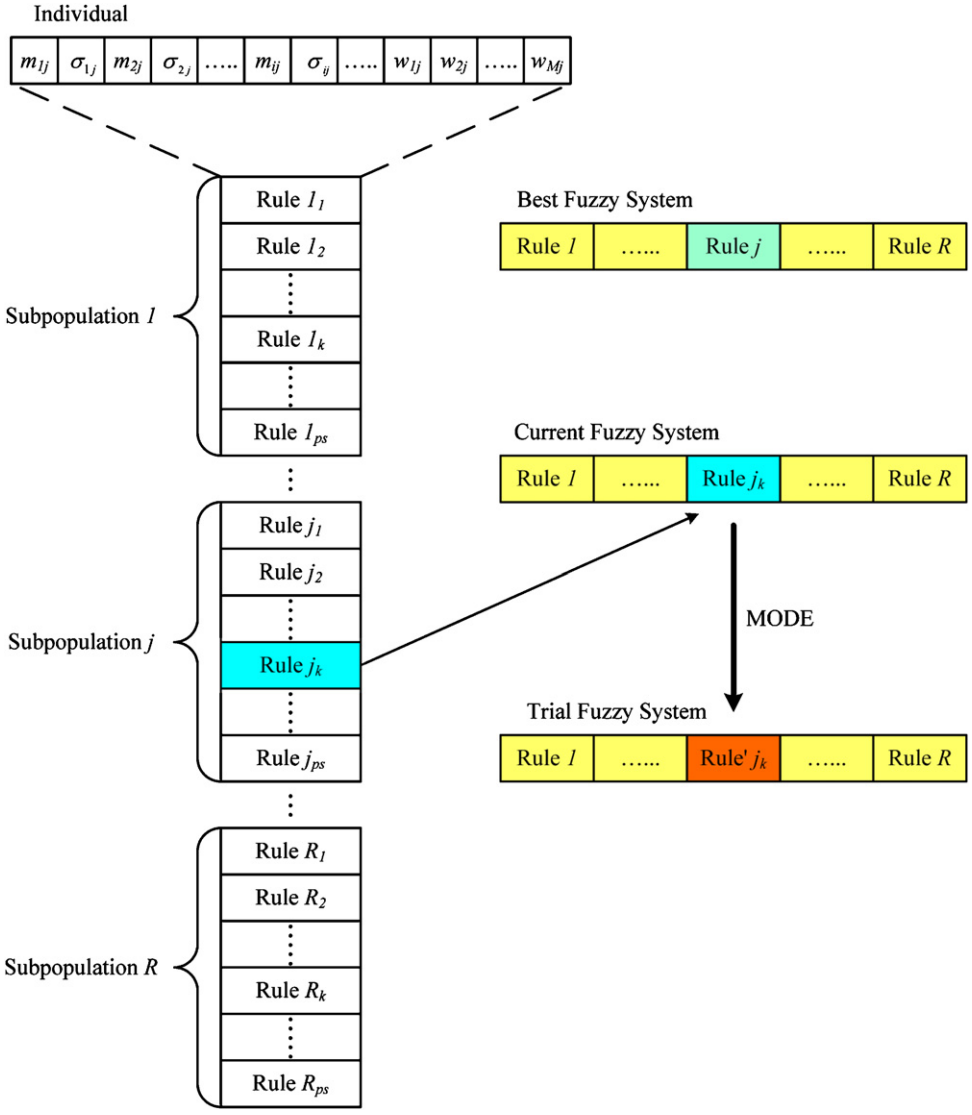

Fig. 4. Structure of the individual in the RSMODE.

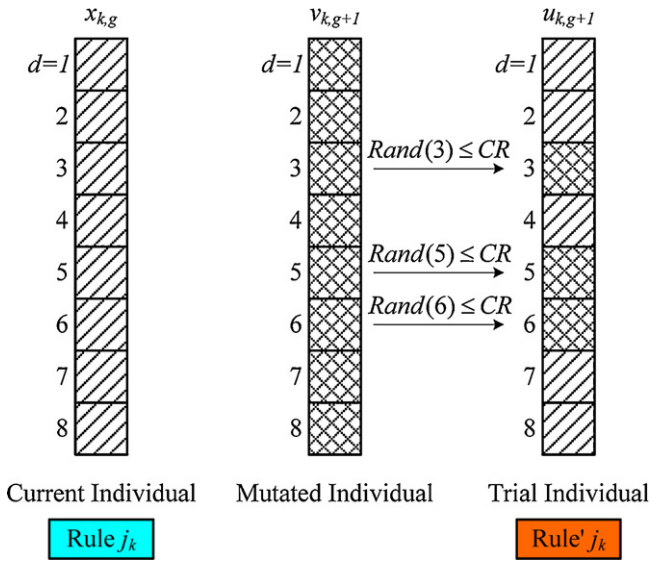

Fig. 5. Illustration of the MODE process for 8-dimensional vector.

Step 2.1: Parent choice phase

Each individual in the current generation is allowed to breed through mating with other randomly selected individuals from the subpopulation. Specifically, for each current individual $x_{k,g}$, $k = 1, 2, \ldots, PS$, where g denotes the current generation and PS denotes the population size, three other random individuals $x_{r_1,g}$, $x_{r_2,g}$ and $x_{r_3,g}$ are selected from the subpopulation such that r_1, r_2, and $r_3 \in \{1, 2, \ldots PS\}$ and $k \neq r_1 \neq r_2 \neq r_3$. This way, a parent pool of four individuals is formed to breed an offspring.

Step 2.2: Offspring generation phase

After choosing the parents, MODE applies a differential operation to generate a mutated individual $v_{k,g+1}$, according to the following equation:

$$v_{k,g+1} = x_{r_1,g} + (1 - F) \cdot (x_{r_2,g} - x_{r_3,g}) + F \cdot (x_{best} - x_{r_1,g}) \quad (14)$$

where F, commonly known as scaling factor, is defined as g/G to control the rate at which the subpopulation evolves, g denotes the current generation, G is the maximum number of generations, and x_{best} is the corresponding parameter of the current best fuzzy system. To complement the differential operation search strategy, then uses a crossover operation, often referred to as discrete recombination, in which the mutated individual $v_{k,g+1}$ is mated with $x_{k,g}$ and generates the offspring $u_{k,g+1}$. The element of trial individual $u_{k,g+1}$ are inherited from $x_{k,g}$ and $v_{k,g+1}$, determined by a parameter called crossover probability ($CR \in [0, 1]$), as follows:

$$= \begin{cases} v_{kd,g+1}, & \text{if } Rand(d) \leq CR \\ x_{kd,g}, & \text{if } Rand(d) > CR \end{cases} \quad (15)$$

where $d = 1, 2, \ldots, D$ denotes the dth element of individual vectors. $Rand(d) \in [0, 1]$ is the dth evaluation of a random number generator. For searching in nonseparable and multimodal landscapes, $CR = 0.9$ is a good choice [19] in this study.

Step 2.3: Survivor selection phase

MODE applies selection pressure only when selecting survivors. First, the current composed fuzzy system embeds the current individual $x_{k,g}$ into the best fuzzy system and the trial composed fuzzy system embeds the trial individual $u_{k,g+1}$ into the best fuzzy system. Second, a knockout competition is played between the current composed fuzzy system and the trial composed fuzzy system, and the corresponding individual of the winner is selected deterministically based on objective function values and promoted to the next phase. In this study, we adopt a fitness function (i.e., objective function) to evaluate the performance of these composed fuzzy systems. The fitness function is defined as follows.

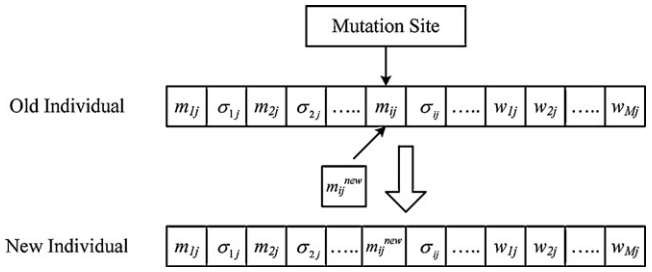

Fig. 6. A mutation operation in the RSMODE.

$$F = \frac{1}{1 + \sqrt{1/N_t \sum_{l=1}^{N_t} (y_l - \bar{y}_l)^2}} \quad (16)$$

where y_l represents the model output of the lth data; $\bar{y}_l$ represents the desired output of the lth data, and N_t represents the number of the training data.

Step 3: Update the best fuzzy system

Compare the fitness value of the current composed fuzzy system, the trial composed fuzzy system and the best fuzzy system. If the fitness value of the current composed fuzzy system exceeds those of the best fuzzy system, then the best fuzzy system is replaced with the current composed fuzzy system. If the fitness value of the trial composed fuzzy system exceeds those of the best fuzzy system, and then the best fuzzy system is replaced with the trial composed fuzzy system.

Step 4: Mutation

After the above process yielded offspring, no new information is introduced to the each subpopulation at the site of an individual. As a source of new sites, mutation should be used sparingly because it is a random search operator. In the following simulations, a mutation rate was set to $1/(2^*N + M)$, meaning that, on average, only one trial parameter is mutated, where N is the number of input variables, M is the number of basis function of SOFNS, and $2^*N + M$ is the length of each individual. Mutation is an operator that randomly alters the allele of an element. The mutation adopted in MODE to yield diversity. The individual suffers from a mutation to avoid falling in a local optimal solution and to ensure the searching capacity of approximate global optimal solution. Fig. 6 shows the mutation of an individual. The mutation value is generated according to Eqs. (11)–(13), where m_{ij} and σ_{ij} are the corresponding mean and variance, respectively, of the current individual. Following the mutation step, a new individual can be introduced into the each subpopulation.

4. Simulation results

This study evaluated the performance of the proposed RSMODE for a SONFS to control nonlinear systems. This section presents several examples and compares the performance with that of other methods. In the nonlinear system control problems,

Table 1
Initial parameters before learning.

Parameter	Value
Population size	50
Maximum number of generation	2000
Crossover rate	0.9
Mutation rate	$1/(2\times N+M)$
Coding type	Real number

SONFS–RSMODE is adopted to design controllers in three simulations – control of water bath temperature system [28], control of the ball and beam system [29], and control of backing up the truck [30]. Table 1 presents the initial parameters before learning used in the three computer simulations.

Example 1. Control of water bath temperature system

The goal of this section is to elucidate the control of the temperature of a water bath system according to,

$$\frac{dy(t)}{dt} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{T_R C} \tag{17}$$

where $y(t)$ is the output temperature of the system in °C; $u(t)$ is the heat flowing into the system; Y_0 is room temperature; C is the equivalent thermal capacity of the system, and T_R is the equivalent thermal resistance between the borders of the system and the surroundings.

T_R and C are assumed to be essentially constant, and the system in Eq. (17) is rewritten in discrete-time form to some reasonable approximation. The system

$$y(k+1) = e^{-\alpha Ts}y(k) + \frac{(\delta/\alpha)(1-e^{-\alpha Ts})}{1+e^{0.5y(k)-40}}u(k) + [1-e^{-\alpha Ts}]y_0 \tag{18}$$

is obtained, where α and δ are some constant values of T_R and C. The system parameters used in this example are $\alpha = 1.0015e^{-4}$, $\delta = 8.67973e^{-3}$ and $Y_0 = 25.0$ (°C), which were obtained from a real water bath plant considered elsewhere [28]. The input $u(k)$ is limited to 0, and 5 V represent voltage unit. The sampling period is $Ts = 30$.

The conventional online training scheme is adopted for online training. Fig. 7 presents a block diagram for the conventional online training scheme. This scheme has two phases – the training phase and the control phase. In the training phase, the switches S1 and S2 are connected to nodes 1 and 2, respectively, to form a training loop. In this loop, training data with input vector $I(k) = [y_p(k+1)\ y_p(k)]$ and desired output $u(k)$ can be defined, where the input vector of the SONFS controller is the same as that used in the general inverse modeling [31] training scheme. In the control phase, the switches S1 and S2 are connected to nodes 3 and 4, respectively, forming a control loop. In this loop, the control signal $\hat{u}(k)$ is generated according to the input vector $I'(k) = [y_{ref}(k+1)\ y_p(k)]$, where y_p is the plant output and y_{ref} is the reference model output.

A sequence of random input signals $u_{rd}(k)$ limited to 0 and 5 V is injected directly into the simulated system described in Eq. (18), using the online training scheme for the SONFS–RSMODE controller. The 120 training patterns are selected based on the input–outputs characteristics to cover the entire reference output. The temperature of the water is initially 25 °C, and rises progressively when random input signals are injected.

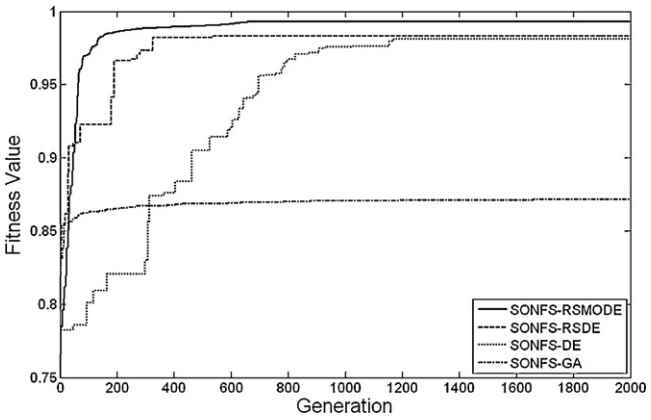

Fig. 8. Learning curves of best performance of the SONFS–REMODE, SONFS–RSDE, SONFS–DE and SONFS–GA in Example 1.

In initialization phase, four subpopulations are generated. This dissertation compares the SONFS–RSMODE controller to the SONFS–RSDE controller, the SONFS–DE controller and the SONFS–GA controller. Each of these controllers is applied to the water bath temperature control system. The performance measures include the set-points regulation, the influence of impulse noise, and a large parameter variation in the system, and the tracking capability of the controllers. Fig. 8 plots the learning curves of the best performance of the SONFS–RSMODE controller for the fitness value, the SONFS–RSDE controller, the SONFS–DE controller and the SONFS–GA controller, after the learning process of 2000 generations.

The first task is to control the simulated system to follow three set-points.

$$y_{\text{ref}}(k) = \begin{cases} 35\ ^\circ\text{C}, & \text{for } k \le 40 \\ 55\ ^\circ\text{C} & \text{for } 40 < k \le 80 \\ 75\ ^\circ\text{C}, & \text{for } 80 < k \le 120. \end{cases} \tag{19}$$

Fig. 9(a) presents the regulation performance of the SONFS–RSMODE controller. The regulation performance was also tested using the SONFS–RSDE controller, the SONFS–DE controller and the SONFS–GA controller. Fig. 9(b) plots the error curves of the SONFS–RSMODE controller, the SONFS–RSDE controller, the SONFS–DE controller, and the SONFS–GA controller. In this figure, the SONFS–RSMODE controller obtains smaller errors than

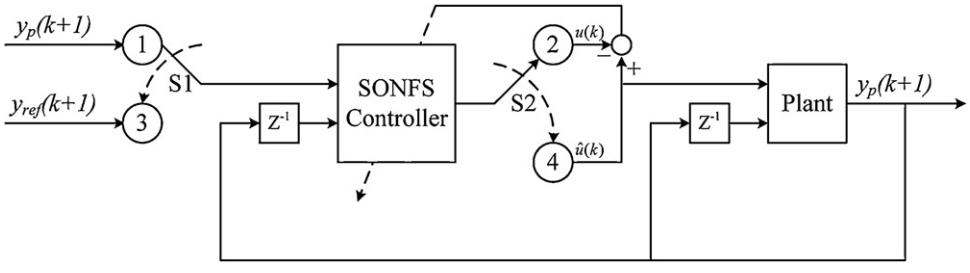

Fig. 7. Conventional online training scheme.

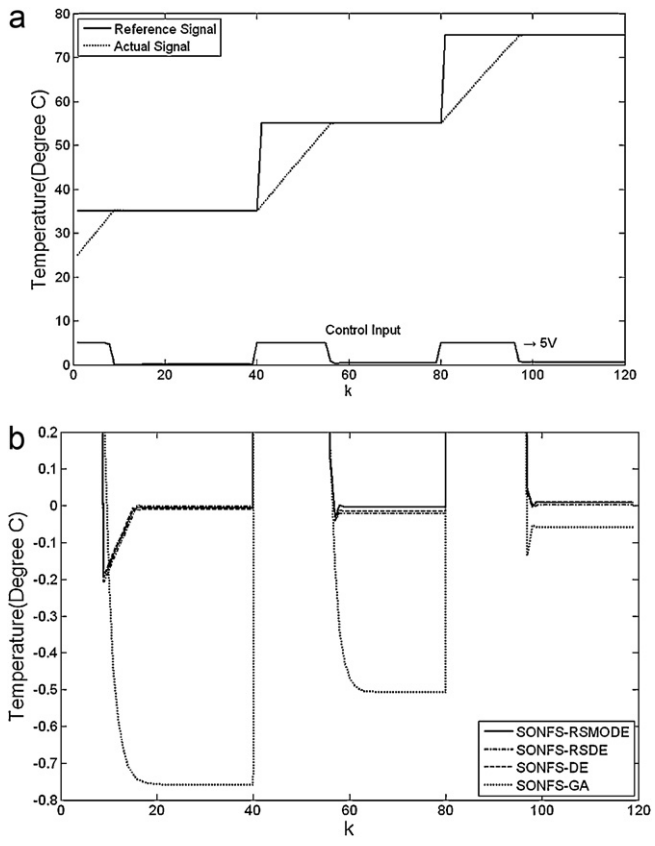

Fig. 9. (a) Final regulation performance of SONFS–RSMODE controller in water bath system. (b) Error curves of the SONFS–RSMODE controller, SONFS–RSDE controller, the SONFS–DE controller, and SONFS–GA controller.

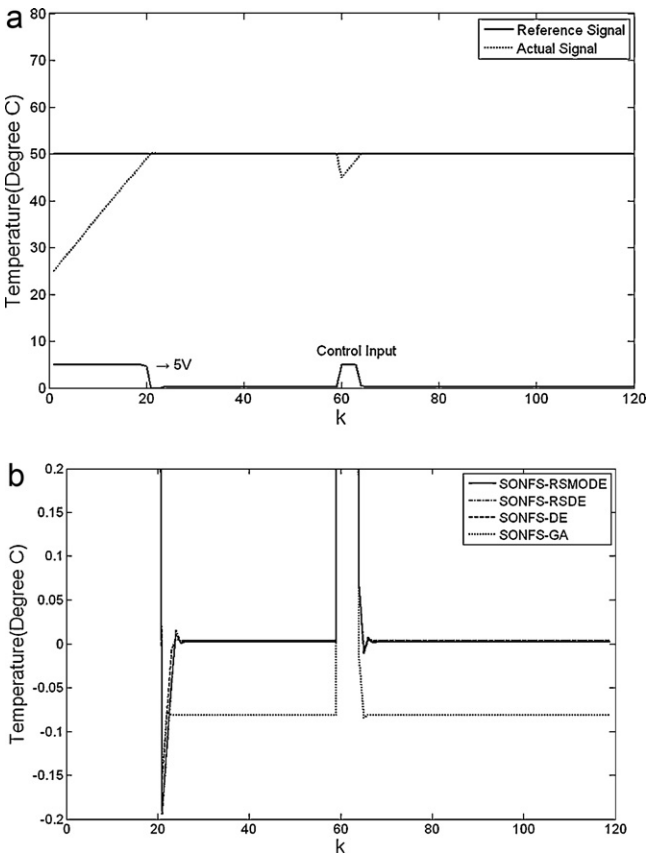

Fig. 10. (a) Behavior of SONFS–RSMODE controller under impulse noise in water bath system. (b) Error curves of SONFS–RSMODE controller, SONFS–RSDE controller, the SONFS–DE controller and SONFS–GA controller.

the other three controllers. To test their regulation performance, a performance index, the sum of absolute error (SAE), is defined by

$$\mathrm{SAE} = \sum_{k} \left| y_{\mathrm{ref}}(k) - y(k) \right| \tag{20}$$

where $y_{\mathrm{ref}}(k)$ and $y(k)$ are the reference output and the actual output of the simulated system, respectively. The SAE values of the SONFS–RSMODE controller, the SONFS–RSDE controller, the SONFS–DE controller, and the SONFS–GA controller are 352.66, 352.81, 352.91, and 372.85, which values are given in the second row of Table 2. The proposed SONFS–RSMODE controller has a much better SAE value of regulation performance than the other controllers.

The second set of simulations is performed to elucidate the noise-rejection ability of the five controllers when some unknown impulse noise is imposed on the process. One impulse noise value $-5\,^{\circ}$C is added to the plant output at the 60th sampling instant. A set-point of $50\,^{\circ}$C is adopted in this set of simulations. For the SONFS–RSMODE controller, the same training scheme, training data and learning parameters as were used in the first set of simulations. Fig. 10(a) and (b) presents the behaviors of the SONFS–RSMODE controller, the SONFS–RSDE controller, the SONFS–DE controller and the SONFS–GA controller under the influence of impulse noise, and the corresponding errors, respectively. The SAE values of the SONFS–RSMODE controller, the SONFS–RSDE controller, the SONFS–DE controller, and the SONFS–GA controller are 270.46, 270.76, 270.65, and 282.21, which are shown in the third row of Table 2. The SONFS–RSMODE controller performs quite well. It recovers very quickly and steadily after the occurrence of the impulse noise.

One common characteristic of many industrial-control processes is that their parameters tend to change in an unpredictable way. The value of $0.7 \times u(k-2)$ is added to the plant input after the 60th sample in the third set of simulations to test the robustness of the five controllers. A set-point of $50\,^{\circ}$C is adopted in this set of simulations. Fig. 11(a) presents the behaviors of the SONFS–RSMODE controller when in the plant dynamics change. Fig. 11(b) presents the corresponding errors of the SONFS–RSMODE controller, the SONFS–RSDE controller, the SONFS–DE controller and the SONFS–GA controller. The SAE values of the SONFS–RSMODE controller, the SONFS–RSDE controller,

Table 2
Comparison of performance of various controllers to control of water bath temperature system.

$\mathrm{SAE} = \sum_{k=1}^{120} \lvert y_{\mathrm{ref}}(k) - y(k) \rvert$	SONFS–RSMODE controller	SONFS–RSDE controller	SONFS–DE controller	SONFS–GA controller
Regulation performance	352.66	352.81	352.91	372.85
Influence of impulse noise	270.46	270.76	270.65	282.21
Effect of change in plant dynamics	262.63	263.21	263.25	270.66
Tracking performance	41.73	42.56	42.92	62.02

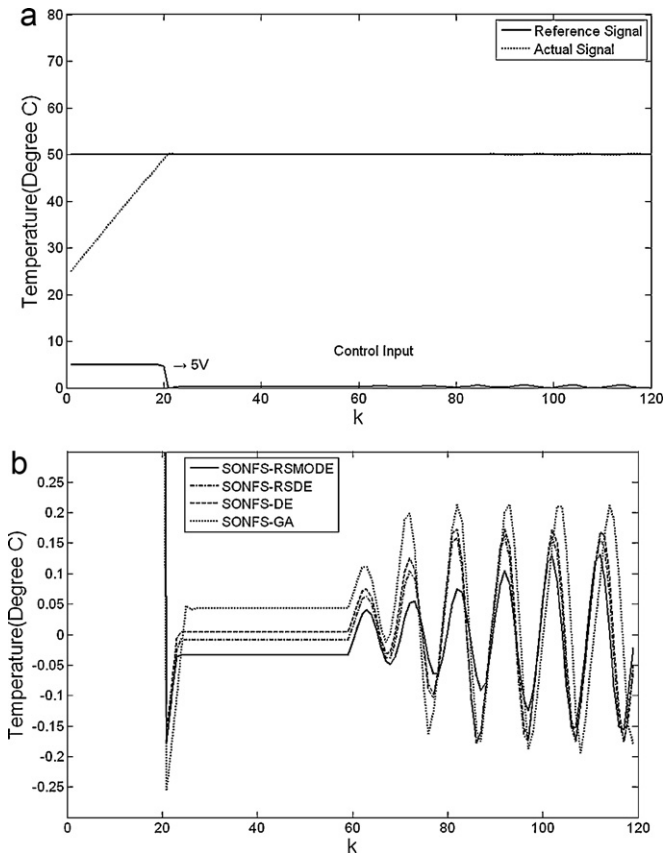

Fig. 11. (a) Behavior of SONFS–RSMODE controller when a change occurs in the water bath system. (b) Error curves of SONFS–RSMODE controller, SONFS–RSDE controller, the SONFS–DE controller, and SONFS–GA controller.

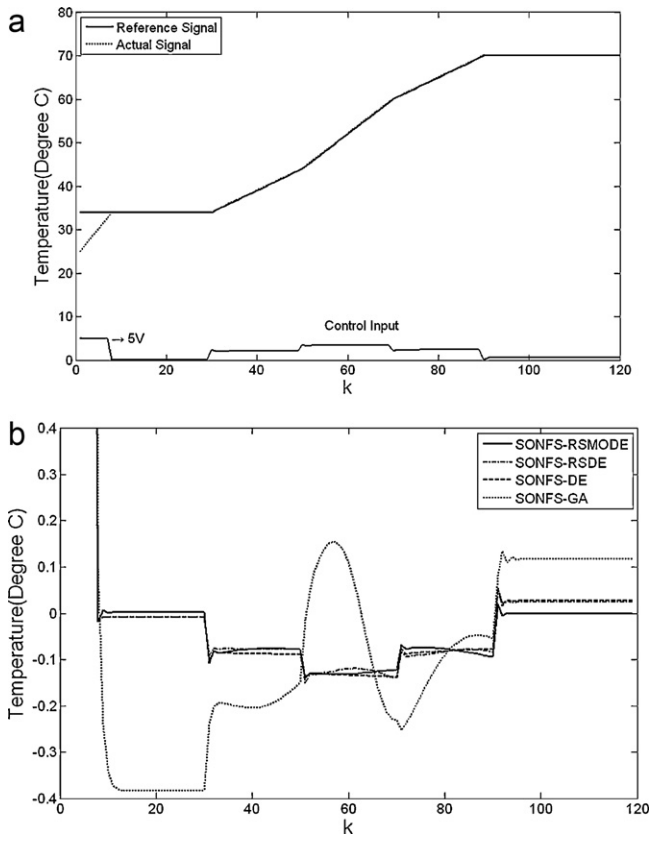

Fig. 12. (a) Tracking of SONFS–RSMODE controller when a change occurs in the water bath system. (b) Error curves of SONFS–RSMODE controller, SONFS–RSDE controller, the SONFS–DE controller, and SONFS–GA controller.

the SONFS–DE controller, and the SONFS–GA controller are 262.63, 263.21, 263.25, and 270.66, which values are shown in the fourth row of Table 2. The results present the favorable control and disturbance rejection capabilities of the trained SONFS–RSMODE controller in the water bath system.

In the final set of simulations, the tracking capability of the SONFS–RSMODE controller with respect to ramp-reference signals is studied. Define

$$y_{\mathrm{ref}}(k) = \begin{cases} 34\ ^{\circ}\mathrm{C} \quad \text{for} \quad k \le 30 \\ (34 + 0.5(k-30))\ ^{\circ}\mathrm{C} \quad \text{for} \quad 30 < k \le 50 \\ (44 + 0.8(k-50))\ ^{\circ}\mathrm{C} \quad \text{for} \quad 50 < k \le 70 \\ (60 + 0.5(k-70))\ ^{\circ}\mathrm{C} \quad \text{for} \quad 70 < k \le 90 \\ 70\ ^{\circ}\mathrm{C} \quad \text{for} \quad 90 < k \le 120 \end{cases} \tag{21}$$

Fig. 12(a) presents the tracking performance of the SONFS–RSMODE controller. Fig. 12(b) presents the corresponding errors of the SONFS–RSMODE controller, the SONFS–RSDE controller, the SONFS–DE controller, and the SONFS–GA controller. The SAE values of the SONFS–RSMODE controller, the SONFS–RSDE controller, the SONFS–DE controller, and the SONFS–GA controller are 41.73, 42.56, 42.92, and 62.02, which are shown in the fifth row of Table 2. The results present the favorable control and disturbance rejection capabilities of the trained SONFS–RSMODE controller in the water bath system. The aforementioned simulation results, presented in Table 2, demonstrate that the proposed SONFS–RSMODE controller outperforms other controllers.

Example 2. Control of the Ball and Beam System

Fig. 13 shows the ball and beam system [29]. The beam is made to rotate in the vertical plane by applying a torque at the center of rotation and the ball is free to roll along the beam. The ball must remain in contact with the beam.

Ball and beam system can be written in state space form as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B(x_1 x_4^2 - G \sin x_3) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u, \tag{22}$$

$$y = x_1$$

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 $y = x_1 \equiv r$ is the output of the system. The control u is the angular acceleration ($\ddot{\theta}$) and the parameters $B = 0.7143$ and $G = 9.81$ are set in this system. The purpose of control 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 [29], 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 $\phi_1(x) = x_1$,

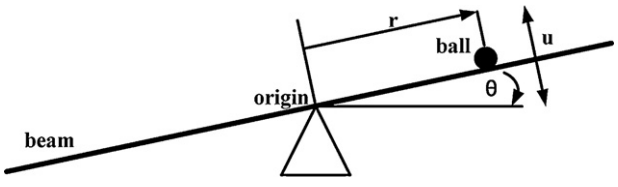

Fig. 13. Ball and beam system.

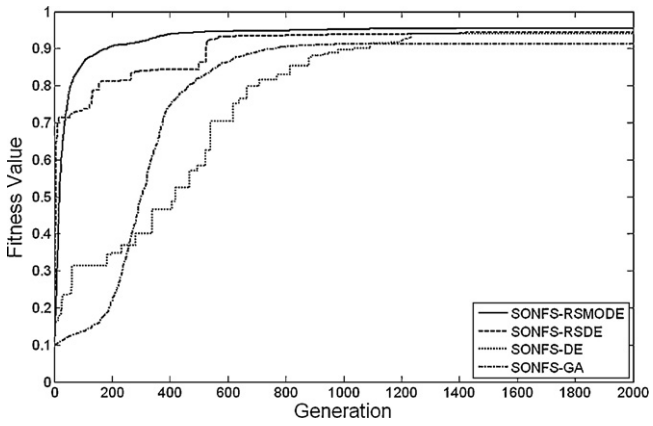

Fig. 14. Learning curves of best performance of the SONFS–RSMODE, SONFS–RSDE, SONFS–DE, and SONFS–GA in Example 2.

$\phi_2(x) = x_2$, $\phi_3(x) = -BG \sin x_3$, $\phi_4(x) = -BGx_4 \cos x_3$, and α_i are 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 $u(x) = [v(x) - b(x)]/a(x)$.

In the simulation herein, the differential equations are solved using the second/third-order Runge–Kutta method. The SONFS is trained to approximate the aforementioned conventional controller of a ball and beam system. $u(x) = [v(x) - b(x)]/a(x)$ is used to generate the input/output train pair with x obtained by randomly sampling 200 points in the region $U = [-5,5] \times [-3,3] \times [-1,1] \times [-2,2]$. In initialization phase, 14 subpopulations are generated. This example was simulated 30 times. Fig. 14 plots the learning curves of the best performance of the SONFS–RSMODE controller for the fitness value, the SONFS–RSDE controller, the SONFS–DE controller and the SONFS–GA controller, after the learning process of 2000 generations. The SONFS–RSMODE controller after learning was tested under the following four initial conditions; $x(0) = [2.4, -0.1, 0.6, 0.1]^T$, $[1.6, 0.05, -0.5, -0.05]^T$, $[-1.6, -0.05, 0.5, 0.05]^T$ and $[-2.4, 0.1, -0.6, -0.1]^T$. Fig. 15 plots the output responses of the closed-loop ball and beam system controlled by the SONFS–RSMODE controller and the SONFS–RSDE controller. These responses approximate those of the controller under the four initial conditions. In this figure, the curves of the SONFS–RSMODE controller tend quickly to stabilize. Fig. 16 also shows the behavior of the four states of the ball and beam system, starting for the initial condition $[-2.4, 0.1, -0.6, -0.1]^T$. In this figure, the four states of the system decay gradually to zero. The results

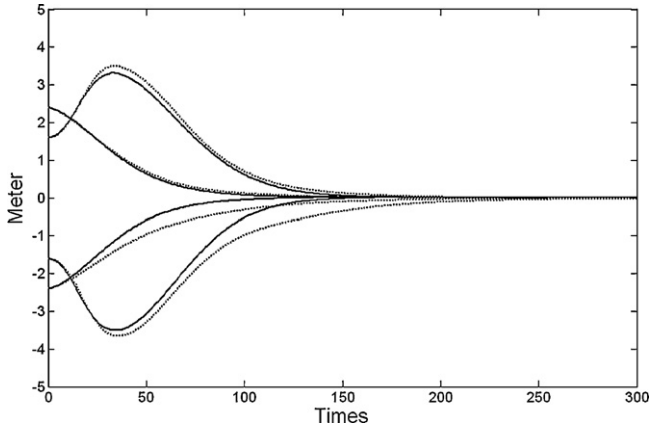

Fig. 15. Responses of ball and beam system controlled by SONFS–RSMODE controller (solid curves) and SONFS–RSDE controller (dotted curves) under four initial conditions.

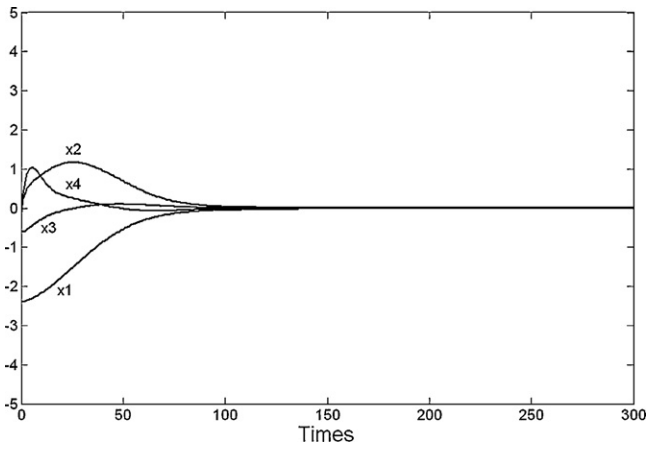

Fig. 16. Responses of four states of the ball and beam system under the control of the SONFS–RSMODE controller.

Table 3
Comparison of performance of various controllers to control of ball and beam system.

Method	SONFS–RSMODE	SONFS–RSDE	SONFS–DE	SONFS–GA
Fitness value (Avg)	0.9041	0.8737	0.8516	0.8287
Fitness value (Best)	0.9653	0.9447	0.9441	0.9131

show the perfect control capability of the trained SONFS–RSMODE controller. The performance of the SONFS–RSMODE controller is compared with that of the SONFS–RSDE controller, the SONFS–DE controller and the SONFS–GA controller. Table 3 presents the comparison results. The results demonstrate that the proposed SONFS–RSMODE controller outperforms other controllers.

Example 3. Control of backing up the truck

Backing a truck into a loading dock is difficult. It is a nonlinear control problem for which no traditional control method exists [30]. Fig. 17 shows the simulated truck and loading zone. The truck position is exactly determined by three state variables ϕ, x and y, where ϕ is the angle between the truck and the horizontal, and the coordinate pair (x, y) specifies the position of the center of the rear of the truck in the plane. The steering angle θ of the truck is the controlled variable. Positive values of θ represent clockwise rotations of the steering wheel and negative values represent counterclockwise rotations. The truck is placed at some initial position and is backed up while being steered by the controller. The objective of this control problem is to use backward only motions of the truck to make the truck arrive in the desired loading dock $(x_{desired}, y_{desired})$ at a right angle ($\phi_{desired} = 90^\circ$). The truck moves backward as the

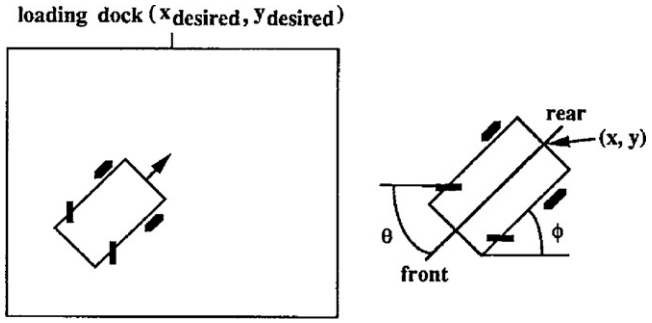

Fig. 17. Diagram of simulated truck and loading zone.

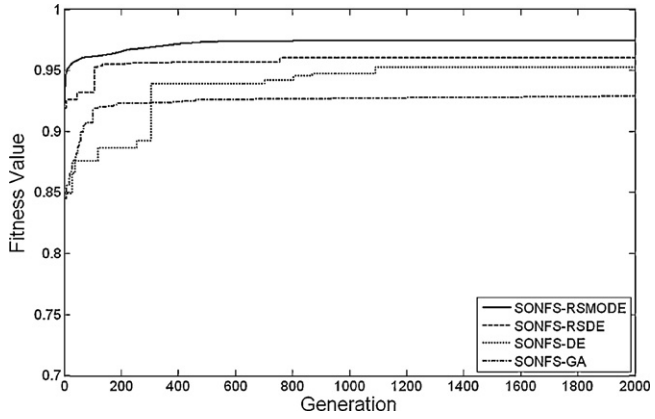

Fig. 18. Learning curves of best performance of the SONFS–RSMODE, SONFS–RSDE, SONFS–DE and SONFS–GA in Example 3.

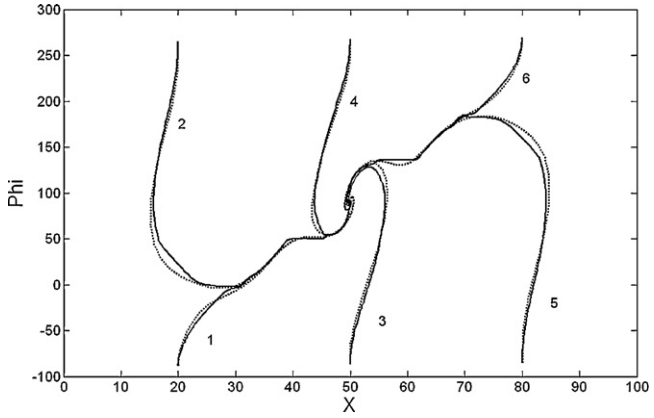

Fig. 19. The moving trajectories of the truck where the solid curves represent the six sets of training trajectories and the dotted curves represent the moving trajectories of the truck under the SONFS–RSMODE controller.

steering wheel moves through a fixed distance (d_f) in each step. The loading region is limited to the plane $[0,100] \times [0,100]$.

The input and output variables of the SONFS–RSMODE controller must be specified. The controller has two inputs, truck angle ϕ and cross position x. When the clearance between the truck and the loading dock is assumed to be sufficient, the y coordinate is not considered as an input variable. The output of the controller is the steering angle θ. The ranges of the variables x, ϕ and θ are as follows.

$$0 \le x \le 100 \tag{23}$$

$$-90^\circ \le \phi \le 270^\circ \tag{24}$$

$$-30^\circ \le \theta \le 30^\circ \tag{25}$$

The equations of backward motion of the truck are,

$$\begin{aligned} x(k+1) &= x(k) + d_f \cos\theta(k) + \cos\phi(k) \\ y(k+1) &= y(k) + d_f \cos\theta(k) + \sin\phi(k) \\ \phi(k+1) &= \tan^{-1}\left[\frac{l \sin\phi(k) + d_f \cos\phi(k) \sin\theta(k)}{l \cos\phi(k) - d_f \sin\phi(k) \sin\theta(k)}\right] \end{aligned} \tag{26}$$

where l is the length of the truck. Eq. (26) yields the next state from the present state.

Learning involves several attempts, each starting from an initial state and terminating when the desired state is reached; the

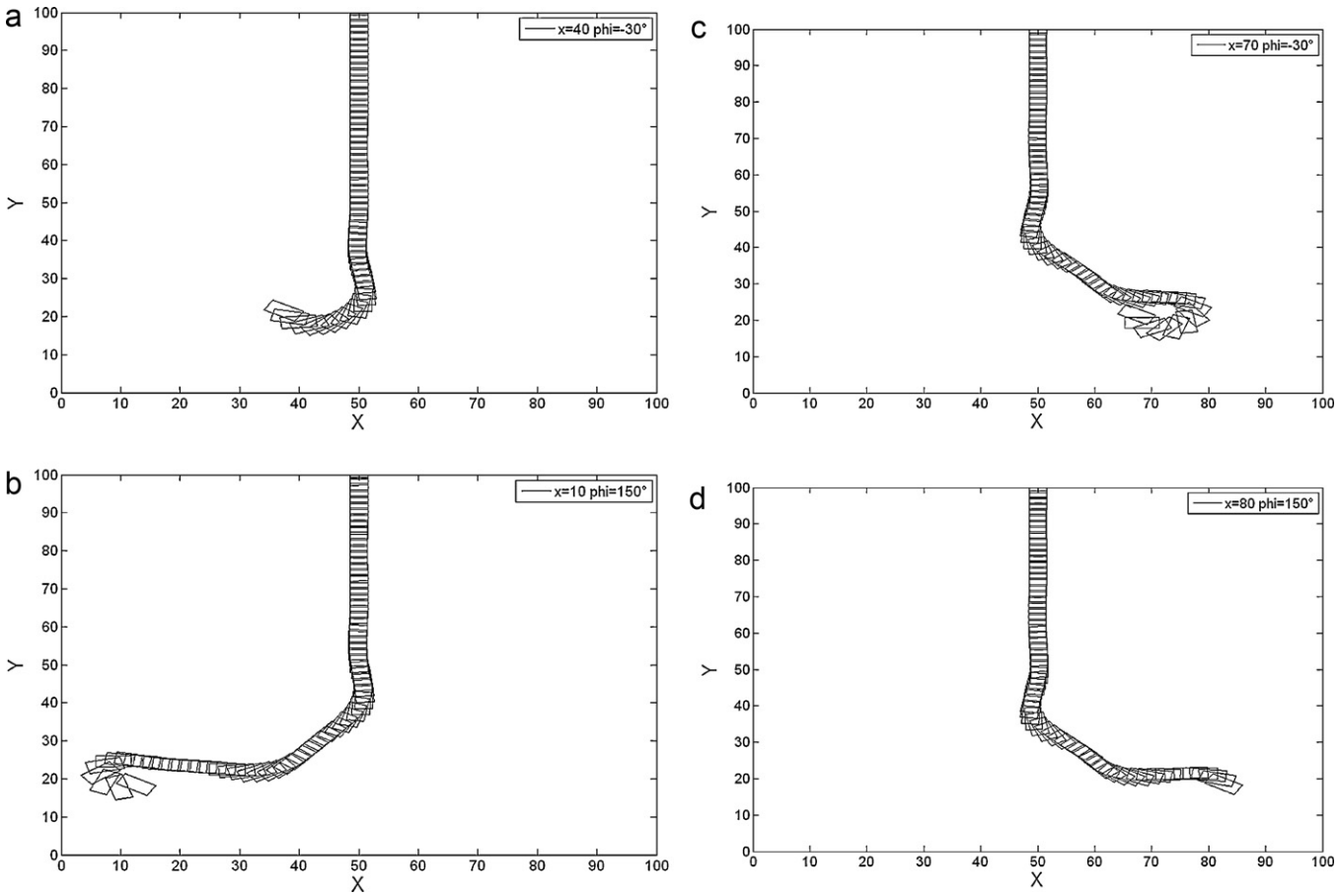

Fig. 20. Trajectories of truck, starting at four initial positions under the control of the SONFS–RSMODE after learning using training trajectories.

Table 4
Comparison of performance of various controllers to control of backing up the truck.

Method	SONFS–RSMODE	SONFS–RSDE	SONFS–DE	SONFS–GA
Fitness value (Avg)	0.9110	0.8939	0.8846	0.8421
Fitness value (Best)	0.9746	0.9604	0.9527	0.9286

SONFS is thus trained. In initialization phase, 7 subpopulations are generated. This example was simulated 30 times. The fitness value of the SONFS–RSMODE is approximately 0.9746 and the learning curve of SONFS–RSMODE is compared with those obtained using the SONFS–RSDE, SONFS–DE, and SONFS–GA, as shown in Fig. 18. In Fig. 19, the solid curves are the training paths and the dotted curves are the paths that the tuck runs under the control of the proposed controller. As this figure shown, the SONFS–RSMODE controller can smooth the training paths. Fig. 20(a)–(d) plots the trajectories of the moving truck controlled by the SONFS–RSMODE controller, starting at initial positions (x, y, ϕ) = (a) (40, 20, −30°), (b) (10, 20, 150°), (c) (70, 20, −30°) and (d) (80, 20, 150°), after the training process has been terminated. The considered performance indices include the best fitness and the average fitness value. Table 4 compares the results. According to these results, the proposed SONFS–RSMODE controller outperforms various existing methods.

5. Conclusion

This study proposes a RSMODE for a SONFS. The proposed RSMODE learning algorithm consists of structure learning to generate initial rule-based subpopulation, and parameter learning to adjust the SONFS parameters. The proposed RSMODE learning algorithm allows that each individual in each subpopulation evolves separately using a modified differential evolution. The experimental results demonstrate that the proposed RSMODE can obtain a better performance than other existing methods under some circumstances.

References

[1] C.T. Lin, C.S.G. Lee, Neural Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent System , Prentice-Hall, NJ, 1996.
[2] D. Nauck, F. Klawoon, R. Kruse, Foundations of Neuro-fuzzy Systems , John Wiley, New York, 1997.
[3] R. Fuller, Introduction to Neuro-fuzzy Systems , Physica-Verlag, New York, 1999.
[4] E. Sanchez, T. Shibata, L.A. Zadeh, Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives , World Scientific, Singapore, 1997.
[5] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, Genetic Fuzzy Systems-evolutionary Tuning and Learning of Fuzzy Knowledge Bases , World Scientific, Singapore, 2001.
[6] P.P. Angelov, Evolving Rule-based Models: A Tool for Design of Flexible Adaptive Systems , Physica-Verlag, Heidelberg, 2002.
[7] B. Carse, T.C. Fogarty, A. Munro, Evolving fuzzy rule based controllers using genetic algorithms , Fuzzy Sets Syst. 80 (June) (1996) 273–293.
[8] F. Herrera, M. Lozano, J.L. Verdegay, Tuning fuzzy logic controllers by genetic algorithms , Int. J. Approx. Reas. 12 (April–May) (1995) 299–315.
[9] A. Homaifar, E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms , IEEE Trans. Fuzzy Syst. 3 (1995 May) 129–139.
[10] J. Velasco, Genetic-based on-line learning for fuzzy process control , Int. J. Intell. Syst. 13 (1998) 891–903.
[11] H. Ishibuchi, T. Nakashima, T. Murata, Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems , IEEE Trans. Syst. Man Cybern. B: Cybern. 29 (1999) 601–608.
[12] O. Cordon, M.J. del Jesus, F. Herrera, M. Lozano, MOGUL: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach , Int. J. Intell. Syst. 14 (1999) 1123–1153.
[13] A. Gonzalez, R. Perez, SLAVE: a genetic learning system based on an iterative approach , IEEE Trans. Fuzzy Syst. 27 (April) (1999) 176–191.
[14] M. Russo, FuGeNeSys: a fuzzy genetic neural system for fuzzy modeling , IEEE Trans. Fuzzy Syst. 6 (1998) 373–388.
[15] I.F. Chung, C.J. Lin, C.T. Lin, A GA-based fuzzy adaptive learning control network , Fuzzy Sets Syst. 112 (1) (2000) 65–84.
[16] G. Alpaydin, G. Dandar, S. Balkir, Evolution-based design of neural fuzzy networks using self-adapting genetic parameters , IEEE Trans. Fuzzy Syst. 10 (2) (2002) 211–221.
[17] R. Storn, K.V. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces , J. Global Opt. 11 (December (4)) (1997) 341–359.
[18] R. Storn, System design by constraint adaptation and differential evolution , IEEE Trans. Evol. Comput. 3 (April (1)) (1999) 22–34.
[19] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization , Springer-Verlag, Germany, 2005.
[20] Z. Yang, K. Tang, X. Yao, Differential evolution for high-dimensional function optimization , IEEE Congress on Evolutionary Computation (September) (2007) 3523–3530.
[21] H.R. Cai, C.Y. Chung, K.P. Wong, Application of differential evolution algorithm for transient stability constrained optimal power flow , IEEE Trans. Power Syst. 23 (May (2)) (2008) 514–522.
[22] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution , IEEE Trans. Evol. Comput. 12 (February (1)) (2008) 107–125.
[23] S.L. Cheng, C. Hwang, Optimal approximation of linear systems by a differential evolution algorithm , IEEE Trans. Syst. Man Cybern. A 31 (November (6)) (2001) 698–707.
[24] R. Joshi, A.C. Sanderson, Minimal representation multisensor fusion using differential evolution , IEEE Trans. Syst. Man Cybern. A 29 (January (1)) (1999) 63–76.
[25] C.H. Chen, C.T. Lin, C.J. Lin, A functional-link-based fuzzy neural network for temperature control , in: 2007 IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 53–58.
[26] 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. 29 (April (2)) (1999) 254–262.
[27] D.E. Moriarty, R. Miikkulainen, Efficient reinforcement learning through symbiotic evolution , Mach. Learn. 22 (1996) 11–32.
[28] J. Tanomaru, S. Omatu, Process control by on-line trained neural controllers , IEEE Trans. Ind. Electron. 39 (1992) 511–521.
[29] J. Hauser, S. Sastry, P. Kokotovic, Nonlinear control via approximate input–output linearizion: The ball and beam example , IEEE Trans. Autom. Control. 37 (March) (1992) 392–398.
[30] D. Nguyen, B. Widrow, The truck backer-upper: an example of self-learning in neural network , IEEE Conf. Syst. Mag. 10 (3) (1990) 18–23.
[31] D. Psaltis, A. Sideris, A. Yamamura, A multilayered neural network controller , IEEE Contr. Syst. 8 (1988) 17–21.