

# Improve the Accuracy of Recurrent Fuzzy System Design Using an Efficient Continuous Ant Colony Optimization

Chi-Chung Chen<sup>1</sup> · Li Ping Shen<sup>1</sup>

Received: 16 April 2017/Revised: 15 December 2017/Accepted: 1 February 2018/Published online: 13 February 2018  
© Taiwan Fuzzy Systems Association and Springer-Verlag GmbH Germany, part of Springer Nature 2018

**Abstract** This paper proposes a new population-based evolutionary optimization algorithm, elite-mixed continuous ant colony optimization with central initialization (EMCACO-C), for improving the accuracy of Takagi–Sugeno–Kang-type recurrent fuzzy network (TRFN) designs. The EMCACO-C is a stochastic search algorithm. The EMCACO-C initializes the ant solutions on concentrative region around the center of the search range followed by a new designed elite-mixed continuous ant colony optimization to generate new solutions. The EMCACO-C mixes the few best elites to generate the directional solutions for guiding and exploring possible promising regions. Then the EMCACO-C employs the Gaussian random sampling to exploit further the directional solutions for finding better solutions. The methodology similarities and differences between the EMCACO-C and genetic algorithm are analyzed. The performances of the EMCACO-C for TRFN designs are verified in the simulations of five application examples including dynamic system control, dynamic system identification, and chaotic series prediction. The EMCACO-C performance is also compared with other swarm-based evolutionary algorithms in the simulations.

**Keywords** Continuous ant colony optimization · Gaussian sampling · Recurrent fuzzy systems · Dynamic system control · Dynamic system identification · Chaotic series prediction

## 1 Introduction

Recurrent fuzzy systems have been designed and aimed for the processing of nonlinear dynamic systems. Contrasted with feed-forward fuzzy systems, recurrent fuzzy systems are characterized by having feedback loops in their structure. In processing the dynamic systems whose current outputs depend explicitly on the delayed past outputs, delayed past inputs or both, recurrent fuzzy systems have been demonstrated to be particularly useful due to its feedback structure [1–10]. If a feed-forward fuzzy system is employed to deal with such temporal characteristic problems, the number of lagged inputs and outputs of the dynamic model should be acquired in advance. Unfortunately, however, the exact order of the real-life dynamic systems to be processed usually is unknown. Moreover, these time-delayed input/output values provide the necessary information and thus have to be fed as the inputs to the feed-forward fuzzy system. This will increase the input dimension yielding less compact fuzzy system models and perhaps making the design more difficult. On the contrary, the recurrent fuzzy systems are able to store information from the past experiences due to the internal memory inherited in recurrent models and thus do not require necessarily such information from the external inputs. Therefore, the recurrent fuzzy systems are probably more suitable than the feed-forward fuzzy systems on processing dynamic systems with temporal characteristics.

The proposed recurrent fuzzy systems in the studies [1–10] differ mainly on the feedback structure. The examples of recurrent fuzzy systems in [1–4] feed the outputs of the fuzzy system back to its inputs. The recurrent fuzzy systems proposed in [5–10] use feedback loops from the internal state variables instead of the fuzzy outputs, which are further differentiated according to the recurrence

---

✉ Chi-Chung Chen  
chichung@mail.ncyu.edu.tw

<sup>1</sup> Department of Electrical Engineering, National Chiayi University, 300 Syuefu Road, Chiayi City 60004, Taiwan

property: local and global feedbacks. The recurrent systems in the studies [5, 6] feed the output of each membership function locally back to itself, so each membership value is only affected locally by its past values. In [7], a recurrent self-evolving fuzzy neural network with local feedback (RSEFNN-LF) feeds the output of temporal firing strength locally back to itself, so the temporal firing strength is affected locally by the current and past states. The examples of recurrent fuzzy systems with the global recurrence property are recurrent self-organizing neural fuzzy inference networks (RSONFIN) [8], Takagi–Sugeno–Kang (TSK)-type recurrent fuzzy networks (TRFN) [9], and interactively recurrent self-evolving fuzzy neural network (IRSFNN) [10]. In the TRFN, all rule firing strengths are linearly summed according to their weights and then fed back as internal network input variables. As a result, the firing strength of each rule depends not only on its own previous value but also on all other rules, making the global recurrence across all rules. Due to such global recurrence, the performance superiority of TRFN to recurrent fuzzy neural networks (RFNN) [5] was shown in [9]. However, given the fuzzy rules, the accurate design of fuzzy system, equivalent to the identification of the parameters in fuzzy systems, is always a time-consuming task. This is particularly difficult for recurrent fuzzy systems. Therefore, this study will aim on improving the accuracy of TRFN designs.

In the past decades, the designs of fuzzy systems have adopted the parameter learning methods used for the learning of artificial neural network to reduce the difficulty. This was accomplished by transforming such design task into solving an optimization problem of the parameters. One major approach for solving the optimization problems is the gradient-based learning algorithm in which the search direction is derived from the gradient of the objective function with respect to the parameters [1–3, 5–10]. The gradient-based learning requires the training input–output data pairs of the fuzzy system for calculating the gradient. However, this usually does not fit for control problems because the desired control outputs corresponding to the inputs are usually not directly available in advance. Moreover, the derivation of the gradient computation is usually complicated and is especially more complex for recurrent fuzzy systems. Another problem for the gradient-based method is that it is easily trapped in local minima/maxima, which is always an issue for the optimization problems where there are multiple minima/maxima in the search space.

To avoid issues above in gradient-based learning methods, derivative-free population-based evolutionary algorithms via computation techniques, such as genetic algorithms (GAs) [9, 11–15], particle swarm optimization (PSO) [16–20], and ant colony optimization (ACO) [21–33], have been proposed and investigated extensively

for optimizing the parameters of fuzzy systems in the past. These evolutionary algorithms apply stochastic procedures on generating new solutions and evaluate many candidate solutions simultaneously, so they are more likely in probability to find the relatively better solution.

Genetic algorithm [11] was developed based on the principle of survival of fittest. The GA uses the crossover operation for exchanging the information between two randomly selected parent solutions, and the mutation technique for exploiting the solutions further to generate the offspring with optimistically better performance. In [13], the GA was employed to design fuzzy controller for the mobile robots. The design of neural fuzzy system optimized by the GA for temperature control was demonstrated in [14]. In [9], an elite genetic algorithm was employed to identify optimally the parameters of a TRFN for control problems. Particle swarm optimization (PSO) was invented based on the understanding of the social behavior of animals such as bird flocking, fish schooling [16]. For each particle in the PSO, its locally own and globally neighboring experiences stochastically determine its velocity for movement. The swarm of particles moves in the search space to find better solutions. In [17], the PSO was used for the automatic construction of fuzzy systems. The hybrid of GA and PSO (HGAPSO) was proposed to achieve better accuracy of the TRFN designs [19]. The study in [20] proposed to use interval type-2 fuzzy logic system for dynamic parameter adaption in PSO to improve the convergence and diversity of the swarm.

Inspired by the observations on real ant colony, the ACO technique was proposed and originally used for solving discrete combinatorial optimization problems (COP) [21–24]. The original ACO and its variants have been successfully applied to optimization problems and have good performances [25–28]. A rank-based ACO with the algorithm parameters being dynamically adjusted by type-1 fuzzy logic [25] or interval type-2 fuzzy logic [26] was proposed to optimize the membership functions of trajectory fuzzy controller for mobile robot. The design of a fuzzy controller for the ball and beam system using a modified ACO with ant set partition strategy was proposed in [27]. The proposed approach in [27] optimized in a systematic and hierarchical way the type of membership functions, the parameters of the membership function, and the fuzzy rules of a fuzzy controller. In the studies [25–27], the parameters of the membership functions of the fuzzy controller were discretized such that the discrete ACO variants can be applied to identify them optimally.

By extending the discrete ACO to continuous domains, a continuous ACO algorithm for solving continuous optimization problems with real-value parameters to be determined was proposed and named as  $ACO_R$  [29]. The  $ACO_R$  demonstrated good performances for continuous

optimization in [29], and later became useful for the high-precision demanded problems such as the design of fuzzy controllers for dynamic systems. Motivated by the  $ACO_R$ , several  $ACO_R$  variants have been proposed to design fuzzy systems for accuracy-oriented problems [30–33]. A cooperative continuous ACO (CCACO) [30] and an assimilation–accommodation mixed continuous ant colony optimization (ACACO) [31] inspired by the cognitive psychology concepts were proposed for the designs of feed-forward fuzzy systems. The elite-guided continuous ACO (ECACO) as claimed to be the first application of continuous ACO to recurrent fuzzy system design was proposed in [32]. These  $ACO_R$  variants achieved good performances on the designs of fuzzy systems. However, as far as accuracy of the fuzzy system design is concerned, there is still room for further improvement over the existing  $ACO_R$  approaches in terms of convergence and accuracy of the optimization algorithms. Based on the  $ACO_R$  framework, this paper proposes a new learning algorithm for TRFN design to improve further the accuracy.

The main contributions of this paper are twofold. First, a new continuous ACO, elite-mixed continuous ACO with central initialization (EMCACO-C) is proposed for TRFN designs. The EMCACO-C is developed based on the framework of the  $ACO_R$  and can be regarded as a new population-based evolutionary optimization algorithm. The EMCACO-C proposes a centrally distributed initialization of the ant solutions for fuzzy system designs followed by a new approach for new solution generation at each iteration. The performance superiority of the EMCACO-C to the  $ACO_R$ , PSO, and ACACO algorithms is demonstrated in five simulation examples of TRFN designs. Second, the advantage on the accuracy improvement for the TRFN designs gained by applying the central initialization to population-based algorithms is also demonstrated in this study. To the best of our knowledge, this is the first time to be studied.

This paper is organized as follows. The next section reviews the structure and the detailed operation of the TRFN. Section 3 proposes the EMCACO-C for TRFN design and provides the comparison of the algorithm with GA. The simulation results of the TRFNs designed by the EMCACO-C and other algorithms for several application problems are presented and discussed in Sect. 4. The application problems simulated in this paper include dynamic system control, chaotic series prediction, and dynamic system identification. Ultimately, Sect. 5 concludes this paper.

## 2 TSK-Type Recurrent Fuzzy Network

The TSK-type recurrent fuzzy network (TRFN) was proposed in [9], and its structure is shown in Fig. 1. Referring to [9], the detailed operation is reviewed as follows. Nodes

in layer 1 are input nodes. The sole function of the input nodes is to transmit the input variables to layer 2. The nodes in layer 2 are input term nodes and they act as membership functions to express the input fuzzy linguistic variables. Two types of membership functions are employed in this layer. The first one is the Gaussian membership function, which is used for the external network input variable  $x$  and affects the fuzzy output based on the local rules. The second one is a global membership function, the sigmoid function, which is used for the internal network input variable  $h$ . Each node in layer 3 is a rule node and is to calculate the firing strength of a rule. The nodes in layer 4 are consequent nodes. Each consequent node corresponding to each rule node performs a TSK-type consequence which is the linearly weighted combination of the input variables  $x$  and  $h$  plus a constant. Context nodes in layer 5 operate as the defuzzifier for the inference output  $h$ . The link weights for inference output in the consequent part of the rules are represented with singleton values. The number of the internal variables  $h$  in this layer is the same as the ones of the rules nodes and the consequent nodes. The node in layer 6 is a defuzzification node and computes the output of the TRFN.

The memory mechanism behaviour in the TRFN is explained as follows. In Fig. 1, the value of the internal variable  $h_i$  is derived from the firing strengths of all fuzzy rules and then its delayed version is fed back to layer 1 as the input for the corresponding rule. The sigmoid membership of each internal variable  $h_i$  corresponding to each rule can be regarded as the influence degree of the temporal history to the current rule. Thus, these internal network variables can memorize the temporal characteristics of the TRFN.

To be specific, a TRFN is represented by a set of the recurrent if–then fuzzy rules, which is described as follows:

$$\begin{aligned} \text{Rule } i : & \text{ If } x_1(t) \text{ is } A_{i1}, \dots, \text{ and } x_n(t) \text{ is } A_{in}, \text{ and } h_i(t) \text{ is } G, \\ & \text{ Then } y(t+1) \text{ is } a_{i0} + a_{i1}x_1(t) + \dots + a_{in}x_n(t) \\ & \quad + a_{i(n+1)}h_i(t), \\ & \text{ and } h_1(t+1) \text{ is } v_{i1}, \dots, \text{ and } h_r(t+1) \text{ is } v_{ir}, \end{aligned} \quad (1)$$

where  $A_{ij}$  and  $G$  are the fuzzy sets, and the variables  $x_i$  and  $h_i$  are the external network inputs and internal network inputs to the TRFN, respectively. The values of  $a_{ij}$  and  $v_{ij}$  are the consequent parameters for the external output  $y$  and internal inference outputs  $h$ , respectively. The external output  $y$  inferred from the TRFN with  $r$  fuzzy rules is calculated by

$$y(t+1) = \frac{\sum_{i=1}^r \phi_i(\vec{x}(t), h_i(t)) \cdot f_i(t)}{\sum_{i=1}^r \phi_i(\vec{x}(t), h_i(t))}, \quad (2)$$

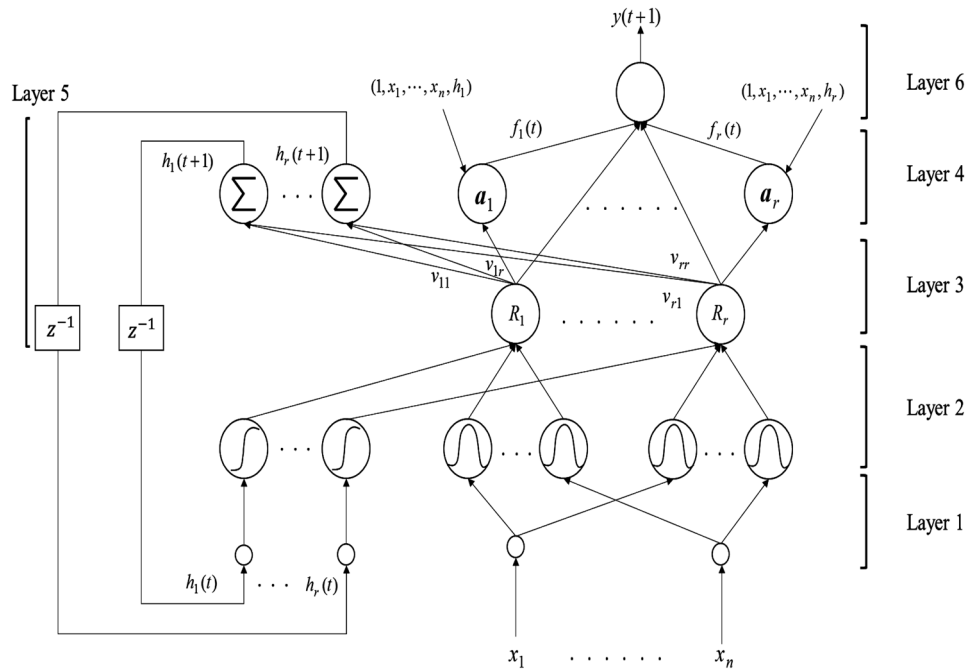


Fig. 1 Structure of the TSK-type recurrent fuzzy network [9]

and the internal inference output  $h_i$  is computed by

$$h_i(t + 1) = \sum_{j=1}^r \phi_j(\vec{x}(t), h_j(t)) \cdot v_{ji}, \quad i = 1, 2, \dots, r, \quad (3)$$

where

$$f_i(t) = a_{i0} + \sum_{j=1}^n a_{ij}x_j(t) + a_{i(n+1)}h_i, \quad (4)$$

$$\phi_i(\vec{x}(t), h_i(t)) = \frac{1}{1 + \exp(-h_i(t))} \cdot \exp\left\{-\sum_{j=1}^n \left(\frac{x_j(t) - m_{ij}}{b_{ij}}\right)^2\right\}. \quad (5)$$

where the values of antecedent parameters  $m_{ij}$  and  $b_{ij}$  are the centers and the widths of the Gaussian membership functions for the external input  $x_i$ , respectively.

Therefore, given the fuzzy rules, the design of TRFN actually is to determine the parameters  $m_{ij}$ ,  $b_{ij}$ ,  $a_{ij}$ , and  $v_{ij}$  as used in (3), (4) and (5). For a TRFN in (1) with  $n$  external input variables,  $r$  fuzzy rules, and a single output, all of the free parameters (decision variables) of the total number  $D = r \cdot (r + 3n + 2)$  can be represented as follows:

$$\vec{s} = [m_{11}, b_{11}, \dots, m_{1n}, b_{1n}, a_{10}, a_{11}, \dots, a_{1(n+1)}, \dots, m_{r1}, b_{r1}, \dots, m_{rn}, b_{rn}, a_{r0}, a_{r1}, \dots, a_{r(n+1)}, v_{11}, v_{12}, \dots, v_{1r}, \dots, v_{r1}, \dots, v_{rr}] \equiv s [1, s^2, \dots, s^D]. \quad (6)$$

The design of a TRFN can be accomplished by applying the optimization algorithm to determine the decision variables in (6) by minimizing or maximizing the value of the task-dependent objective function when the TRFN is employed for achieving such task.

### 3 Elite-Mixed Continuous ACO with Central Initialization (EMCACO-C) for TRFN Design

This section presents the proposed elite-mixed continuous ACO with central initialization (EMCACO-C) which will be used for TRFN designs. Since the proposed EMCACO-C is inspired by the  $ACO_R$ , which is actually extended from the discrete ACO framework, we firstly review the operation of the original ACO and the  $ACO_R$  in detail for easier explanation of the EMCACO-C later. The  $ACO_R$  will also be used as the benchmark for comparison.

#### 3.1 Discrete ACO Framework

The discrete ACO framework outlines a class of swarm-based evolutionary algorithms, which were inspired by the behavior of real ant colonies and now are largely applied to solve discrete combinatorial optimization problems. Ants deposited the pheromone trail when they foraged for food. Since the pheromone evaporates with the time, the path with a higher pheromone level guiding other ants is usually a shorter one to the food source. The first and best-known

discrete ACO, named as ant system (AS) [21], was proposed and applied to solve the classical traveling salesman problem (TSP). In ant system for the TSP problem, each ant chooses probabilistically the next city to visit when it moves. At the time  $t$ , the transition probability with which ant  $k$  in city  $i$  chooses to move to city  $j$  is defined as

$$P_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^a (\eta_{ij})^b}{\sum_{l \in U_i^k} (\tau_{il}(t))^a (\eta_{il})^b} & \text{if } j \in U_i^k, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

The parameters in (7) are described as follows.  $\tau_{ij}(t)$  is the pheromone trail on the edge  $(i, j)$  at the time  $t$ .  $\eta_{ij}$  is a prior heuristic information for moving from city  $i$  to city  $j$ .  $U_i^k$  is the set of allowed neighborhood cities of city  $i$  for the ant  $k$ . The values of  $a$  and  $b$  determine the relative importance of pheromone trail at time  $t$  and heuristic information. Each ant constructs a feasible solution after it visited all cities only once. All feasible solutions the ant colony built by completing their visits are collected for the discrete ACO algorithm to update the pheromone information for the next ant cycle. Many variants of discrete ACO algorithms, such as ant colony system (ACS) [22] and max–min ant system (MMAS) [23] differing on the update rules for pheromone values, were proposed.

### 3.2 ACO for Continuous Domains (ACO<sub>R</sub>)

Many continuous ACO algorithms for continuous optimization problems were proposed and studied after the proposal of the discrete ACO. In particular, ACO for continuous domain (denoted as ACO<sub>R</sub>) proposed in [29] directly extended the discrete probability distributions (7) used in the discrete ACO to the continuous probability density functions (PDFs). In the ACO<sub>R</sub>, these PDFs are the Gaussian functions and can be derived from a maintained solution archive sketched in Fig. 2 for sampling values to generate new candidate solutions.

In the ACO<sub>R</sub>, each row vector in the solution archive represents a feasible solution to the optimization problem to be solved. The quality of each feasible solution  $\vec{s}_l$  is determined by the value of the objective function  $E(\vec{s}_l)$ . All

the feasible solutions in the archive are sorted and ranked from the best quality to the worst one in the table so that the solution  $\vec{s}_l$  has rank  $l$ . These ranks are used to determine the probabilities of each ant solution being selected in the next ant cycle to generate the new solutions. The procedure of the ACO<sub>R</sub> is detailed as follows.

Initially, in the archive of  $N$  solutions, each  $D$ -dimensional row vector  $\vec{s} = [s^1, s^2, \dots, s^D]$  with the continuous-valued components as in (6) is randomly generated within the specified search space. All solutions then are evaluated, sorted, and kept in the solution archive. Each solution  $\vec{s}_l$  (with rank  $l$ ) in the sorted archive has associated with a weight  $w_l$ :

$$w_l = \frac{1}{qN\sqrt{2\pi}} \exp\left\{-\frac{(l-1)^2}{2q^2N^2}\right\}, \quad (8)$$

where  $q$  is a parameter of the ACO<sub>R</sub>.

The ACO<sub>R</sub> generates a new candidate solution in two phases. In the phase one, the ACO<sub>R</sub> chooses probabilistically one leading solution  $\vec{s}_l$  from the archive according to the following probability distribution:

$$p_l = \frac{w_l}{\sum_{m=1}^N w_m}, \quad m = 1, 2, \dots, N. \quad (9)$$

It is inferred from (8) and (9) that the solution with the better quality has the higher chances being chosen. Once a leading solution is chosen, a new candidate solution is constructed by completing the second phase. In the phase two, the new value for each variable  $s^j$  of a new candidate solution is sampled around the  $j$ th component value of the leading solution based on the Gaussian PDF  $g_l^j(s^j; \mu_l^j, \sigma_l^j)$  with the mean  $\mu_l^j = s_l^j$  and the standard deviation  $\sigma_l^j$

$$g_l^j(s^j; \mu_l^j, \sigma_l^j) = \frac{1}{\sigma_l^j \sqrt{2\pi}} \exp\left\{-\frac{(s^j - \mu_l^j)^2}{2(\sigma_l^j)^2}\right\}, \quad (10)$$

$$\sigma_l^j = \varepsilon \sum_{m=1}^N \frac{|s_m^j - s_l^j|}{N-1}, \quad (11)$$

where the pheromone evaporation rate  $\varepsilon > 0$  is a parameter of the ACO<sub>R</sub>. For constructing a new candidate solution, a leading solution  $\vec{s}_l$  from all archive solutions is chosen only once and the Gaussian PDFs  $g_l^j(s^j; \mu_l^j, \sigma_l^j)$  are sampled in sequence for  $j = 1, 2, \dots, D$ . Therefore,  $L$  new candidate solutions can be generated for consideration by repeating this two-phase process  $L$  times. These  $L$  new candidate solutions, together with the original  $N$  solutions in the previous ant cycle, are sorted again according to their values of objective function  $E(\cdot)$ . At the end of each ant cycle, only the  $N$  top-best-performance solutions over the total  $(N + L)$  solutions are reserved and stored in the archive for next cycle.

$\vec{s}_1$	$s_1^1$	...	$s_1^j$	...	$s_1^D$	$E(\vec{s}_1)$	$w_1$
$\vec{s}_2$	$s_2^1$	...	$s_2^j$	...	$s_2^D$	$E(\vec{s}_2)$	$w_2$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$	$\vdots$
$\vec{s}_l$	$s_l^1$	...	$s_l^j$	...	$s_l^D$	$E(\vec{s}_l)$	$w_l$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$	$\vdots$
$\vec{s}_N$	$s_N^1$	...	$s_N^j$	...	$s_N^D$	$E(\vec{s}_N)$	$w_N$

**Fig. 2** Solutions archive in the ACO<sub>R</sub>, where the weights  $w_1 \geq w_2 \geq \dots \geq w_N$  and the objective values  $E(\vec{s}_1) \leq E(\vec{s}_2) \leq \dots \leq E(\vec{s}_N)$  [29]



### 3.3 EMCACO-C

The proposed algorithm, elite-mixed continuous ACO with central initialization (EMCACO-C), for improving the accuracy of TRFN designs differs from the  $ACO_R$  on the initialization of ant solutions and the method of generating new candidate solution. The superiority of the proposed EMCACO-C for TRFN designs will be demonstrated in Sect. 4. The operation of the EMCACO-C is detailed as follows.

#### 3.3.1 Central Initialization

The initial guesses of ant solutions in the EMCACO-C for TRFN design are not the same as most popular swarm-based algorithms, such as GA, PSO, and  $ACO_R$ . Those well-known popular algorithms usually generate the initial population of solutions randomly and uniformly over the entire search space because of no a prior information in the beginning. In the EMCACO-C for TRFN designs, however, all the initial solutions are guessed uniformly within the certain width around the center point of the search space. Our idea for this is simply based on the case observations of some manually designed fuzzy systems. In some observed fuzzy systems, especially accuracy-oriented systems, one of the locations of the membership functions for each input in its antecedent part is usually designed around the input value which arises more frequently or is most concerned. The other membership functions for each input could be located in the positions symmetrically extending from that point. Therefore, we consider the central initialization for the archive solutions in the proposed EMCACO-C for fuzzy system designs. The centrally distributed initial ant solutions will be further optimized by the proposed algorithm.

The specific procedure for the initial guess of solutions in the EMCACO-C is as follows. In the first step, the entire search range for each decision variable to be optimized is normalized to the fixed range  $[0, 1]$ , respectively. In fact, such normalization procedure for the decision variables to be learned is very common in the field of computational intelligence because it makes the learning rates for all the decision variables uniform. Then an initial ant solution in the archive is formed by generating random numbers uniformly over the range  $[0.5 - \alpha/2, 0.5 + \alpha/2]$  and scaling back to the original search range for each decision variable. The parameter  $\alpha$  is defined as the guess width for initializing population solutions, and its value is a parameter of the EMCACO-C. The effects of the guess width  $\alpha$  for initializing solutions on the performance of TRFN designs will be discussed through simulation results.

#### 3.3.2 Elite-Mixed $ACO_R$

In the  $ACO_R$ , all  $N$  solutions in the archive are the candidates for being chosen as the leading solution. Around the leading solution, a series of  $D$  (the number of decision variables) corresponding Gaussian samplings are followed to generate a new solution. The probability of a solution to be chosen in (8) and (9) is inversely in exponential way proportional to its corresponding rank and depends on the value of the parameter  $q$  in (8). When  $q$  is small, the best-ranked solutions are intensely favored and thus the algorithm is easier to be stuck in one of the local optima while it converges faster. On the contrary, when  $q$  is too large, the probabilities for all archive solutions being selected become more uniform, and the algorithm has better chance to find the global optima while it converges more slowly.

The EMCACO-C *mixes* in scanning manner starting from the first solution component, the fixed number of the few best elites uniformly in probability to form a directional solution. The Gaussian random sampling procedure for exploiting further the directional solution then is employed to generate a new candidate solution. The entire process of the mixing operation followed by the Gaussian sampling is denoted as the elite-mixed  $ACO_R$  to differentiate from the original  $ACO_R$ . For example, we consider the EMCACO-C in which the  $Q$  best-ranked elites are involved for mixing. The EMCACO-C firstly constructs the directional solution, whose  $j$ th component  $s_{d(j)}^j$  is chosen identically and independently from the component set  $\{s_1^j, s_2^j, \dots, s_Q^j\}$ . Around the constructed directional solution, a new candidate solution is generated by *sampling* the Gaussian PDF  $g_{d(j)}^j(s^j; \mu_d^j, \sigma_{d(j)}^j)$  with  $\mu_d^j = s_{d(j)}^j$  and  $\sigma_{d(j)}^j$  as defined in (11) for  $j = 1, 2, \dots, D$ . Therefore, the  $L$  new candidate solutions are generated for consideration by  $L$  repetitions of the elite-mixed  $ACO_R$ . The value of  $Q$  is a parameter of the algorithm and its effect on the performance is discussed in Sect. 4.

#### 3.3.3 Population Update

The  $L$  new candidate solutions above are evaluated by the values of the objective function  $E(\cdot)$ . These  $L$  new candidate solutions, together with the original  $N$  solutions in the previous cycle, are sorted again according to their values of objective function. Similarly, at the end of each cycle, only the  $N$  top-best-performance solutions over the total  $(N + L)$  solutions are reserved and stored in the archive for next cycle, and the rest of them are discarded.

The EMCACO-C repeats generating the  $L$  new candidate solutions followed by the population update until the termination condition for the algorithm is satisfied. The

---

**EMCACO-C Algorithm**


---

Initialize evaporation rate  $\varepsilon$ , guess width  $\alpha$ , number of elites  $Q$ , archive size  $N$ .  
 Generate  $N$  solutions of the form (6) uniformly using the guess width  $\alpha$  around the center.  
 Evaluate and sort the initial  $N$  solutions.  
**while** the number of ant cycles is less than prescribed value **do**  
   For  $k = 1 : L$  (generate candidate solutions  $\vec{s}_{N+k}$ )  
     For  $j = 1 : D$  (generate solution components  $s_{N+k}^j$ )  
       choose  $d(j)$  from  $\{1, 2, \dots, Q\}$  uniformly  
        $s_{N+k}^j = \text{sampling } g_{d(j)}^j(s^j; \mu_d^j, \sigma_{d(j)}^j)$  with  $\mu_d^j = s_{d(j)}^j$  and  $\sigma_{d(j)}^j$  in (11)  
     End  
   End  
 Evaluate  $L$  new candidate solutions and sort  $(N+L)$  solutions.  
 Update population to keep  $N$  solutions in the archive.  
**end while**

---

**Fig. 3** EMCACO-C algorithm

pseudo code of the EMCACO-C algorithm is shown in Fig. 3.

### 3.3.4 Qualitative Comparison of the EMCACO-C with GA

Though the EMCACO-C is inspired from the ACO<sub>R</sub>, its operation is more like the Genetic algorithms (GA). Referring to [32], the methodology similarities and differences between the EMCACO-C and GA are compared. In the EMCACO-C, the mixing operation of the elite-mixed ACO<sub>R</sub> can be regarded as the multi-parent scanning crossover in the GAs [34]. In general, the GA usually selects and recombines two parents to generate new solutions. The basic principle for selecting parents is that the better-quality parents are possibly selected with higher probability. The techniques of the tournament and the roulette wheel are two of the examples. The general recombination techniques include one-point, two-point, or arithmetic crossover operations. However, the mixing operation of the elite-mixed ACO<sub>R</sub> in the EMCACO-C selects multiple parents. These parents are the top-performed  $Q$  elites in the archive. Each of these  $Q$  elites is selected uniformly in probability. The scanning crossover used in the mixing operation of the elite-mixed ACO<sub>R</sub> essentially is  $(D - 1)$ -point crossover. Intuitively, the multi-parent scanning crossover probably could provide more diverse candidate solutions than the two-point crossover by two parents. However, the mixing operation of the elite-mixed ACO<sub>R</sub> does not handle the correlation information between each optimized variable in the TRFN [29], and thus, a larger value of  $Q$  may slow down the convergence speed.

In the elite-mixed ACO<sub>R</sub>, the Gaussian sampling around the directional solution plays the similar role as the mutation operation in the GA. In general GA mutation operation, the mutation probability is usually constant and small, and thus, at each iteration some offspring solutions might not be mutated. The deviation range in the GA mutation for each offspring solution component usually is set to be fixed and identical. Furthermore, the mutation in GA usually is implemented by sampling the uniform density function. On the contrary, the sampling in the EMCACO-C is based on Gaussian density function and is always realized for generating new candidate solution. Moreover, since the deviation range for the sampling in the EMCACO-C depends on the separate convergence status of each solution component in the archive, it is not fixed.

## 4 Simulation Results and Discussions

### 4.1 The Values of $\alpha$ and $Q$

To determine suitable values of  $\alpha$  and  $Q$  in the EMCACO-C for TRFN designs, a validation procedure by employing various combinations of  $\alpha$  and  $Q$  has been conducted to optimize a TRFN. The TRFN for validation is used to produce the control input for controlling the nonlinear dynamic plant output to track the reference trajectory as detailed in Example 1 below. The computation environment is a personal computer with Intel Core™ i5-4570 CPU@3.2 GHz and memory capacity of 4 GB running the program compiled by dev-C++ on Windows 7. In the EMCACO-C, the population size  $N$  is set to 20, the number

of new solution candidates  $L$  is set to 20, and the parameter  $\varepsilon$  is set to 0.85 as that used in [29].

*Example 1* (MISO dynamic plant control) As the same as that used in [9, 19, 32], the nonlinear dynamic plant to be controlled is given by

$$y(k + 1) = \frac{y(k)y(k - 1)(y(k) + 2.5)}{1 + y^2(k) + y^2(k - 1)} + \lambda(k), \quad (12)$$

where  $\lambda(k)$  is the control input to the plant and is limited within the range  $[-4, 4]$  in this study. The system output in (12) depends on two previous outputs and is controlled by a control input. The objective is to design a TRFN for controlling the system output, initialized at  $y(0) = y(1) = 0.0$ , to track the reference trajectory  $y_d(k)$ , which is guided by the reference input  $r(k)$  and is described as follows:

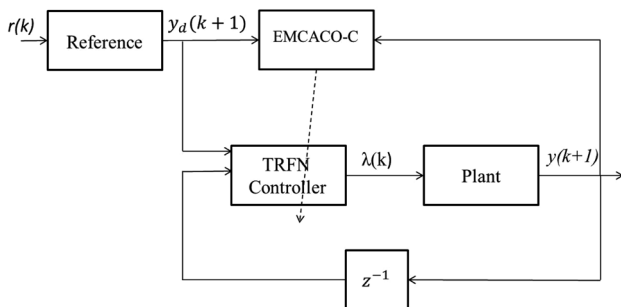
$$y_d(k + 1) = 0.6y_d(k) + 0.2y_d(k - 1) + r(k),$$

$$r(k) = 0.5 \cdot \sin\left(\frac{2\pi k}{45}\right) + 0.2 \cdot \sin\left(\frac{2\pi k}{15}\right) + 0.2 \cdot \sin\left(\frac{2\pi k}{90}\right), \quad (13)$$

where the initial reference states  $y_d(0)$  and  $y_d(1)$  are assumed to be 2.4 and 1.9, respectively.

The control configuration of the TRFN controller and the nonlinear plant is presented in Fig. 4 where the input–output variables of the TRFN are also illustrated. The only input variables to the TRFN are the desired plant output  $y_d(k + 1)$  and the current plant state  $y(k)$ . Excited by these two external inputs as well as the internal states within the TRFN, the TRFN produces the output  $\lambda(k)$  to control the nonlinear dynamic plant governed by (12). The performance of the designed TRFN controller is evaluated by the root-mean-square error (RMSE) between the plant output  $y(k)$  and the reference trajectory  $y_d(k)$  over the 250 time steps and is calculated by

$$\text{RMSE} = \left( \frac{\sum_{k=1}^{250} (y_d(k + 1) - y(k + 1))^2}{250} \right)^{\frac{1}{2}}. \quad (14)$$



**Fig. 4** Configuration of evolutionary learning of TRFN for the control of nonlinear plant in Example 1

This RMSE error is defined as the value of the objective function  $E(\cdot)$  for use in the EMCACO-C and other compared evolutionary algorithms. We firstly designed a TRFN with four fuzzy rules. The free parameters  $m_{ij} \in [-3, 3]$ ,  $b_{ij} \in [0, 3]$ ,  $a_{ij} \in [-4, 4]$ ,  $v_{ij} \in [-4, 4]$  as described in (6) are searched for finding a better controller so as to determine the suitable values of  $\alpha$  and  $Q$ . The 30,000 evaluations were performed in a single run of the optimization process to build a TRFN controller. With 100 independent runs, the validation results of the average, the standard deviation (SD), the best (minimum), and the worst (maximum) of RMSE errors are shown in Table 1. The results clearly indicate that the values of  $\alpha$  and  $Q$  should be selected as 0.2 and 2, respectively, for the EMCACO-C. These two values will be also adopted by the EMCACO-C to conduct other simulations later.

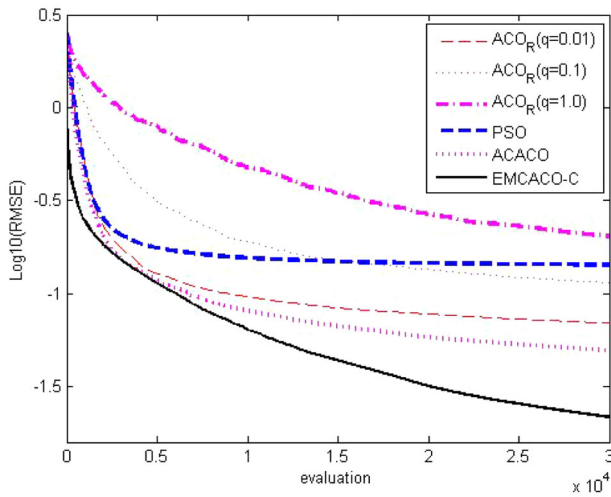
To validate the performance superiority of the proposed EMCACO-C using the determined values of  $\alpha$  and  $Q$ , some population-based optimization algorithms including the ACO<sub>R</sub> associated with several different  $q$  values, PSO, and assimilation–accommodation mixed continuous ant colony optimization (ACACO) [31] were also applied to the same TRFN design problem for comparison. The settings of parameters used in these algorithms are described as follows. In the ACO<sub>R</sub>, the population size of  $N = 50$ , the number of new candidate solutions  $L = 20$  at each ant cycle, and the parameter of  $\varepsilon = 0.85$  are set. In the PSO, the number of particles, the inertia coefficient for previous velocity, the acceleration coefficient toward the globally best-known solution, and the acceleration coefficient

**Table 1** Cross-validation results for different values of  $\alpha$  and  $Q$  in the EMCACO-C for the four-rule TRFN in Example 1

$Q$	$\alpha$	0.04	0.1	0.2	0.3	0.4	0.5
1	Avg.	0.1367	0.1128	0.1013	0.1022	0.1029	0.1121
	SD	0.0272	0.0219	0.0227	0.0289	0.0290	0.0319
	Best	0.0604	0.0505	0.0490	0.0512	0.0418	0.0443
	Worst	0.2342	0.1715	0.1528	0.2116	0.2043	0.2080
2	Avg.	0.0341	0.0298	<b>0.0281</b>	0.0329	0.0435	0.0575
	SD	0.0192	0.0153	0.0122	0.0170	0.0246	0.0396
	Best	0.0127	0.0100	<b>0.0088</b>	0.0098	0.0125	0.0144
	Worst	0.1076	0.1029	<b>0.0706</b>	0.1328	0.1557	0.2196
3	Avg.	0.0377	0.0327	0.0305	0.0371	0.0463	0.0564
	SD	0.0199	0.0169	0.0124	0.0228	0.0228	0.0298
	Best	0.0096	0.0110	0.0133	0.0149	0.0164	0.0119
	Worst	0.0974	0.0960	0.0740	0.0999	0.1405	0.1432
4	Avg.	0.0521	0.0461	0.0433	0.0473	0.0438	0.0587
	SD	0.0252	0.0265	0.0253	0.0305	0.0223	0.0315
	Best	0.0168	0.0154	0.0140	0.0150	0.0129	0.0168
	Worst	0.1374	0.1324	0.1356	0.2055	0.1210	0.2061

Bold value indicates the minimum value in the same comparison item





**Fig. 5** Average best-so-far RMSEs at each performance evaluation for the evolutionary TRFN controllers optimized by the EMCACO-C and different population-based optimization algorithms in Example 1 (eight rules)

toward the locally own best solution are set to 50, 0.5, 2.0, and 1.0, respectively. In the ACACO, the population size of  $N = 50$  and the number of new candidate solutions  $L = 20$  at each ant cycle are set. The ACACO set group size  $N_1 = 2$ ,  $N_2 = 8$ , and  $N_3 = 40$  for group 1, group 2, and group 3, respectively, as those used in [31].

In addition to the TRFNs with four rules, we also compared the TRFNs composed of eight fuzzy rules. Figure 5 and Table 2 show the learned results of the TRFN systems optimized by the EMCACO-C,  $ACO_R$ , PSO, and ACACO. The simulation results performed by the  $ACO_R$ , PSO, and ACACO were obtained when the decision variables were initialized uniformly over the entire search space as usual, that is  $\alpha = 1.0$ . Figure 5 shows the average best-so-far RMSE at each performance evaluation of each

algorithm for the TRFN with eight rules. The numerical results in Table 2 demonstrate that the average error of the EMCACO-C is smaller than the ones of the  $ACO_R$ , PSO, and ACACO for TRFNs with the same rule number. To verify the statistically significant difference between the EMCACO-C and the algorithms used for comparison, the nonparametric Wilcoxon signed-rank tests [35] for two related samples of RMSE errors were performed. For each algorithm, the null hypothesis claims that its average error is less than or equal to that of the EMCACO-C. The performance of the EMCACO-C is compared against each of the algorithms, and the resulting  $p$  value is presented. The  $p$  values in Table 2 indicate that the null hypothesis is rejected at the significant level of 5% for each comparison.

Moreover, we also observed that the performance of the TRFN with eight rules outperforms that with four rules when they were optimized by the EMCACO-C. However, this observation does not hold for the  $ACO_R$  with larger  $q$  values. The possible reason for this is that the  $ACO_R$  converges slower if the  $q$  value is larger. The best four-rule TRFN controller designed by the EMCACO-C is shown in Table 3, and the resulting control results are shown in Fig. 6. The results show that the controlled plant output is very close to the reference output except at the first few time steps because the initial conditions of the controlled plant are different from the reference.

**4.2 Application Examples**

*Example 2* (MIMO dynamic plant control) The nonlinear multiple-input multiple-output (MIMO) plant to be controlled by the TRFN is the same as that used in [19, 32] and is given by

$$y_1(k + 1) = 0.5 \cdot \left[ \frac{y_1(k)}{1 + y_2^2(k)} + \lambda_1(k - 1) \right] \tag{15}$$

**Table 2** Performances of the TRFN controllers designed by various population-based optimization algorithms for the MISO plant control problem in Example 1

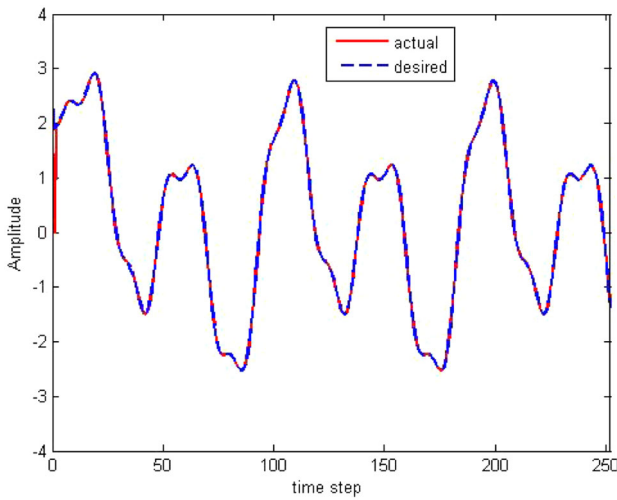
Algorithms	$ACO_R (q = 1.0)$	$ACO_R (q = 0.1)$	$ACO_R (q = 0.02)$	$ACO_R (q = 0.01)$	PSO	ACACO	EMCACO-C
Rule number				4			
RMSE avg.	0.0991	0.0973	0.0914	0.0928	0.1391	0.0769	<b>0.0281</b>
RMSE SD	0.0261	0.0454	0.0462	0.0549	0.0938	0.0552	0.0122
RMSE (best)	0.0589	0.0369	0.0189	0.0194	0.0305	0.0248	<b>0.0088</b>
$p$ value	1.9e-18	1.9e-18	4.7e-18	1.6e-17	1.9e-18	2.8e-17	-
Rule number				8			
RMSE Avg.	0.2030	0.1143	0.0675	0.0691	0.1425	0.0493	<b>0.0216</b>
RMSE SD	0.1276	0.0376	0.0235	0.0359	0.1841	0.0325	0.0082
RMSE (Best)	0.0886	0.0387	0.0233	0.0169	0.0399	0.0172	<b>0.0067</b>
$p$ value	1.9e-18	1.9e-18	2.2e-18	8.7e-18	1.9e-18	1.3e-16	-
CPU time (s)	14.8	14.5	14.8	14.7	12.5	47.1	14.7

Bold value indicates the minimum value in the same comparison item

**Table 3** Obtained TRFN controller composed of four rules optimized by the EMCACO-C. The values are truncated to two fractional digits

Rule $i$	$m_{i1}$	$b_{i1}$	$m_{i2}$	$b_{i2}$	$a_{i0}$	$a_{i1}$	$a_{i2}$	$a_{i3}$	$v_{1i}$	$v_{2i}$	$v_{3i}$	$v_{4i}$
1	2.37	1.09	- 0.62	1.07	3.19	- <b>0.14</b>	- <b>0.75</b>	3.11	- 4.00	- <b>0.41</b>	- 2.12	- 4.00
2	<b>0.19</b>	2.28	<b>0.04</b>	1.17	- 1.17	1.05	- <b>0.52</b>	1.58	4.00	<b>0.56</b>	1.48	- 4.00
3	<b>0.16</b>	0.65	- <b>0.20</b>	<b>1.52</b>	- <b>0.54</b>	0.92	- <b>0.17</b>	1.05	- 4.00	- 0.99	1.02	- 3.99
4	- 3.00	0.00	- 3.00	0.00	- 4.00	3.99	4.00	- 4.00	4.00	4.00	- 4.00	3.99

Bold value indicates the minimum value in the same comparison item



**Fig. 6** Control results of the TRFN controller optimized by the EMCACO-C in Example 1 (four rules)

$$y_2(k + 1) = 0.5 \cdot \left[ \frac{y_1(k)y_2(k)}{1 + y_2^2(k)} + \lambda_2(k - 1) \right]. \tag{16}$$

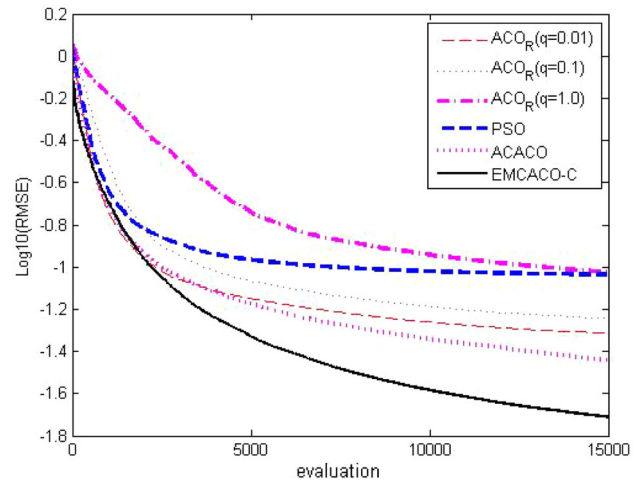
The objective is to generate two sequences of control inputs  $\lambda_1(k)$  and  $\lambda_2(k)$  such that the controlled plant outputs  $y_1(k), y_2(k)$ , initialized at  $y_1(0) = 0.2$  and  $y_2(0) = 0.0$ , follow the two desired sinusoidal reference outputs  $y_{d1}(k)$  and  $y_{d2}(k)$ , respectively, as follows:

$$y_{d1}(k) = \sin\left(\frac{k\pi}{45}\right) \tag{17}$$

$$y_{d2}(k) = \cos\left(\frac{k\pi}{45}\right). \tag{18}$$

Similarly, the only input variables to the TRFN are the current plant states  $y_1(k), y_2(k)$  and the desired outputs  $y_{d1}(k + 1), y_{d2}(k + 1)$ . Excited by these inputs, the TRFN produces two control inputs  $\lambda_1(k)$  and  $\lambda_2(k)$ . The two delayed TRFN outputs  $\lambda_1(k - 1)$  and  $\lambda_2(k - 1)$  are used to control the plant governed by (15) and (16). The RMSE of the two tracking data over the 250 time steps is defined by

$$RMSE = \left( \sum_{k=1}^{250} \frac{(y_{d1}(k+1) - y_1(k+1))^2 + (y_{d2}(k+1) - y_2(k+1))^2}{250} \right)^{\frac{1}{2}}. \tag{19}$$



**Fig. 7** Average best-so-far RMSEs at each performance evaluation for the evolutionary TRFN controllers optimized by the EMCACO-C and different population-based optimization algorithms in Example 2 (eight rules)

In this example, we designed the TRFN systems with two different numbers of rules as studied in [32]: six rules and eight rules. The free parameters  $m_{ij} \in [-1, 1]$ ,  $b_{ij} \in [0, 1]$ ,  $a_{ij} \in [-2, 2]$ ,  $v_{ij} \in [-2, 2]$  are searched for finding a better-performed TRFN controller. The 15,000 evaluations were performed in a single run of the optimization process. With 100 independent runs of simulations, the average best-so-far RMSE at each performance evaluation of each algorithm for the eight-rule TRFN is shown in Fig. 7.

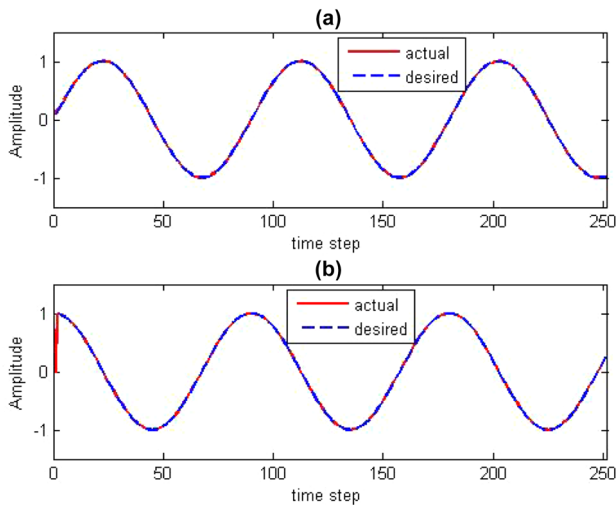
The learned results of the TRFNs with six and eight rules are shown in Table 4. The results show that the average error of the EMCACO-C is smaller than the ones of the ACO<sub>R</sub>, PSO, and ACACO for TRFNs with the same rule number. The  $p$  values in Table 4 indicate that the null hypothesis is rejected at the significance level of 5% for each comparison using the Wilcoxon test. Similarly, the performance of the TRFN with eight rules outperforms that with six rules when they were designed by the EMCACO-C. The control results of the TRFN controller optimized by the EMCACO-C are shown in Fig. 8. The results demonstrate that two controlled plant outputs are very close to the desired ones except at the first few time steps.

*Example 3* (Continuously stirred tank reactor control) In this example, we use a TRFN system to control a

**Table 4** Performances of the TRFN controllers designed by various population-based optimization algorithms for the MIMO plant control problem in Example 2

Algorithms	ACO <sub>R</sub> ( $q = 1.0$ )	ACO <sub>R</sub> ( $q = 0.1$ )	ACO <sub>R</sub> ( $q = 0.02$ )	ACO <sub>R</sub> ( $q = 0.01$ )	PSO	ACACO	EMCACO-C
Rule number	6						
RMSE avg.	0.0864	0.0595	0.0576	0.0590	0.1096	0.0408	<b>0.0242</b>
RMSE SD	0.0120	0.0089	0.0178	0.0209	0.0704	0.0156	0.0136
RMSE (best)	0.0650	0.0390	0.0208	0.0145	0.0333	0.0134	<b>0.0058</b>
$p$ value	1.9e-18	6.5e-18	1.3e-17	2.0e-17	1.9e-18	1.3e-11	-
Rule number	8						
RMSE avg.	0.0943	0.0569	0.0473	0.0486	0.0921	0.0362	<b>0.0195</b>
RMSE SD	0.0237	0.0101	0.0172	0.0198	0.0582	0.0156	0.0100
RMSE (best)	0.0629	0.0285	0.0102	0.0136	0.0259	0.0107	<b>0.0043</b>
$p$ value	1.9e-18	2.1e-18	7.1e-17	5.8e-17	2.2e-18	2.3e-12	-

Bold value indicates the minimum value in the same comparison item



**Fig. 8** Control results of the TRFN controller optimized by the EMCACO-C in Example 2 (six rules)

continuously stirred tank reactor (CSTR) in which the first-order, irreversible, exothermic reaction  $A \mapsto B$  is carried out. At the input, the fresh feed of chemical A is mixed with the recycle stream of unreacted material A with a recycle rate. Let  $t$  be the moment of time the output exits the CSTR. According to [36], the CSTR material and energy balance equations in dimensionless variables are

$$\begin{aligned} \dot{x}_1(t) &= f_1(\bar{x}(t)) + \left(\frac{1}{\lambda} - 1\right)x_1(t - \tau) \\ \dot{x}_2(t) &= f_2(\bar{x}(t)) + \left(\frac{1}{\lambda} - 1\right)x_2(t - \tau) + \beta u, \end{aligned} \tag{20}$$

where

$$\begin{aligned} f_1(\bar{x}(t)) &= \frac{-1}{\lambda}x_1(t) + D_a(1 - x_1(t))\exp\left(\frac{x_2(t)}{1 + x_2(t)/\gamma_0}\right) \\ f_2(\bar{x}(t)) &= -\left(\frac{1}{\lambda} + \beta\right)x_2(t) + HD_a(1 - x_1(t))\exp\left(\frac{x_2(t)}{1 + x_2(t)/\gamma_0}\right), \end{aligned} \tag{21}$$

where  $x_i(t) = \theta_i$  for  $t \in [-\tau, 0], i = 1, 2$ , and  $-1 \leq u(t) \leq 1$  is the control input. The state  $x_1(t)$  represents the conversion rate of the reaction,  $0 \leq x_1(t) \leq 1$ , and  $x_2(t)$  is the dimensionless temperature. As the same as that used in the previous study [32], the constant parameters are given as  $\gamma_0 = 20, H = 8, \beta = 0.3, D_a = 0.0072, \lambda = 0.8, \tau = 2$ , and the initial values of  $x_i(t)$  are  $\theta_1 = 0.144$  and  $\theta_2 = 0.8862$ . The objective is to control the conversion rate  $x_1(t)$  to follow the pre-defined set points  $x_d(t)$  as given by

$$x_d(t) = \begin{cases} 0.4, & t \leq 35, \\ 0.6, & 35 < t \leq 70, \\ 0.8, & 70 < t \leq 105, \\ 0.6, & 105 < t \leq 150. \end{cases} \tag{22}$$

The TRFN controller to be optimized is composed of six fuzzy rules and its inputs are the current states  $x_1(t), x_2(t)$ , and the set point  $x_d(t)$ . The free parameters  $m_{ij} \in [0, 1], b_{ij} \in [0, 0.5], a_{ij} \in [-1, 1], v_{ij} \in [-1, 1]$  are searched for finding a better-performed TRFN controller. In the simulation, the controller commands a new control input  $u$  every 0.5 time interval. The simultaneous differential equations in (20) are solved by Euler method for simulations. The RMSE error for training is defined by

$$\text{RMSE}_{\text{train}} = \left( \sum_{k=1}^{300} \frac{W(k)(x_1(k) - x_d(k))^2}{300} \right)^{\frac{1}{2}}, \tag{23}$$

where the time step  $k$  is set to  $2t$  and  $W(k)$  is defined by

$$W(k) = \begin{cases} 1, & 70n + 1 \leq k \leq 70n + 5, \quad n = 0, 1, 2, 3, \\ 5, & \text{otherwise.} \end{cases} \tag{24}$$

In the  $\text{RMSE}_{\text{train}}$ , when the desired conversion rate changes from one set point to another, the regulation errors in the initial five steps of the transition state are weighted with

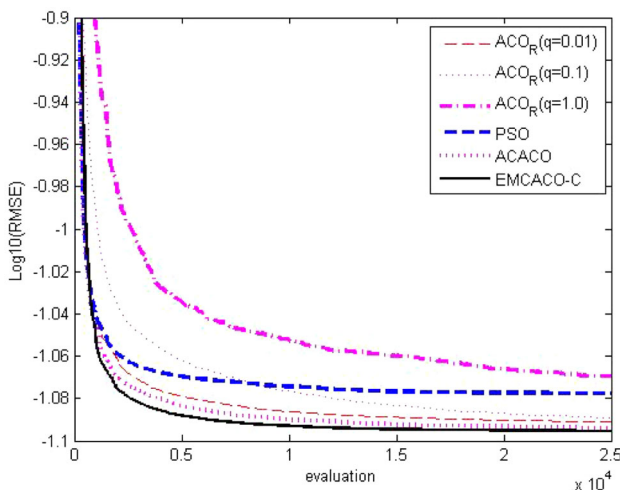
$W = 1$ , and the following errors in the steady state are weighted with  $W = 5$ . The 25,000 evaluations were performed in a single run of the training process. As considered in [32], by neglecting the initial transition errors, the evaluation RMSE for the obtained controller is defined by

$$RMSE_{eva} = \left( \sum_{k=21}^{300} \frac{W(k) (x_1(k) - x_d(k))^2}{280} \right)^{\frac{1}{2}} \quad (25)$$

where

$$W(k) = \begin{cases} 0, & 70n + 1 \leq k \leq 70n + 5, \quad n = 1, 2, 3, \\ 1, & \text{otherwise.} \end{cases} \quad (26)$$

With 100 independent runs of simulations, the average best-so-far training RMSE at each performance evaluation of each algorithm for the TRFN is shown in Fig. 9. Table 5 shows the learned statistical results of training and evaluation RMSEs. The results show the average training error  $RMSE_{train}$  of the EMCACO-C is smaller than the ones of the  $ACO_R$ , PSO, and ACACO. Moreover, the average



**Fig. 9** Average best-so-far training RMSEs at each performance evaluation for the evolutionary TRFN CSTR controllers optimized by the EMCACO-C and different population-based optimization algorithms in Example 3

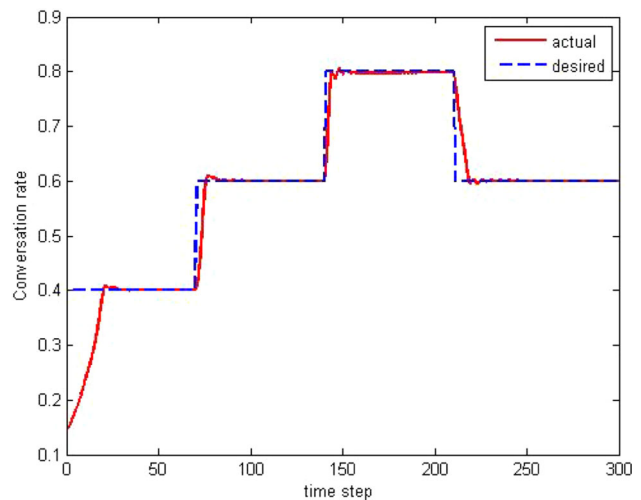
evaluation error  $RMSE_{eva}$  of the EMCACO-C is also smaller than the ones of the population-based algorithms in comparison. The  $p$  values in Table 5 indicate that the null hypothesis is rejected at the significance level of 5% for each comparison using the Wilcoxon test for the samples of evaluation errors  $RMSE_{eva}$ . The control results of the TRFN optimized by the EMCACO-C are shown in Fig. 10. The results demonstrate that the controlled plant output is very close to the desired set points except at the initial transition time steps.

**Example 4** (Mackey–Glass Chaotic Series Prediction)

This example considers the prediction of chaotic time series using a TRFN as studied in [7]. The time series to be predicted is generated by the well-known Mackey–Glass chaotic system, which is described by the delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)} - 0.1x(t), \quad (27)$$

where  $\tau$  is set to 17, and the initial value  $x(0)$  is 1.2 with  $x(t) = 0$  for  $t < 0$ . The objective is to design a TRFN using the four past values of  $x(t - 24)$ ,  $x(t - 18)$ ,  $x(t - 12)$  and  $x(t - 6)$  as the external inputs for predicting  $x(t)$ . The



**Fig. 10** CSTR control result of the TRFN controller optimized by the EMCACO-C in Example 3

**Table 5** Performances of the TRFN controllers designed by various population-based optimization algorithms for the CSTR control problem in Example 3

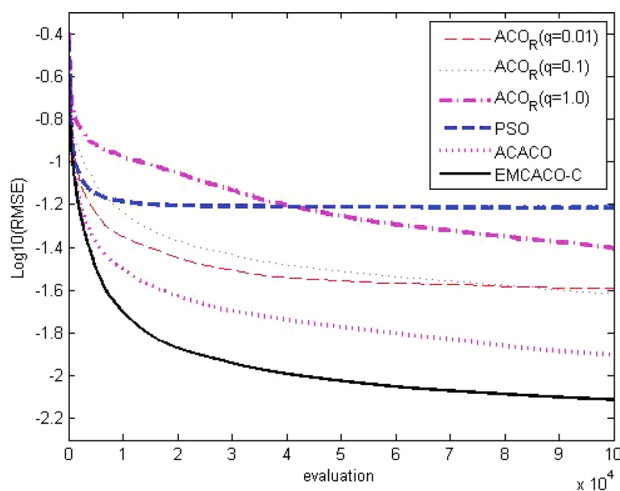
Algorithms	$ACO_R (q = 1.0)$	$ACO_R (q = 0.1)$	$ACO_R (q = 0.02)$	$ACO_R (q = 0.01)$	PSO	ACACO	EMCACO-C
$RMSE_{train}$ avg.	0.0852	0.0814	0.0805	0.0811	0.0836	0.0805	<b>0.0803</b>
$RMSE_{train}$ SD	0.0019	0.0009	0.0005	0.0025	0.0071	0.0016	0.0007
$RMSE_{eva}$ avg.	0.0137	0.0074	0.0054	0.0063	0.0096	0.0053	<b>0.0047</b>
$RMSE_{eva}$ SD	0.0026	0.0019	0.0013	0.0036	0.0076	0.0021	0.0013
$p$ value	2.1e-18	4.1e-16	1.4e-07	1.1e-08	2.4e-13	1.1e-04	–

Bold value indicates the minimum value in the same comparison item

objective function is defined as the RMSE error between the sampled output of the system governed by (27) and the TRFN output.

In this example, the TRFN predictor to be optimized is composed of five fuzzy rules. The free parameters  $m_{ij} \in [0.4, 1.4]$ ,  $b_{ij} \in [0, 0.5]$ ,  $a_{ij} \in [-1.4, 1.4]$ ,  $v_{ij} \in [-1.4, 1.4]$  are searched for finding a better-performed TRFN predictor. To evaluate the performances, the following experiment was conducted: First, the time series data governed by (27) were generated using Runge–Kutta numerical method. As discussed in [7], 500 data patterns from  $t = 124$  to 623 extracted from the time series are used for training the TRFN and the other 500 data patterns from  $t = 624$  to 1123 are for testing. The 100,000 evaluations were performed in a single run of the training process.

With 50 independent runs of simulations, the average best-so-far training RMSE at each performance evaluation of each algorithm for the TRFN is shown in Fig. 11. Table 6 presents the learned results of training errors  $RMSE_{train}$  (the first 500 data) and testing errors  $RMSE_{test}$  (the second 500 data). The results show that the training error  $RMSE_{train}$  and testing error  $RMSE_{test}$  of the EMCACO-C are smaller than



**Fig. 11** Average best-so-far training RMSEs at each performance evaluation for the evolutionary TRFN predictors optimized by the EMCACO-C and different population-based optimization algorithms in Example 4

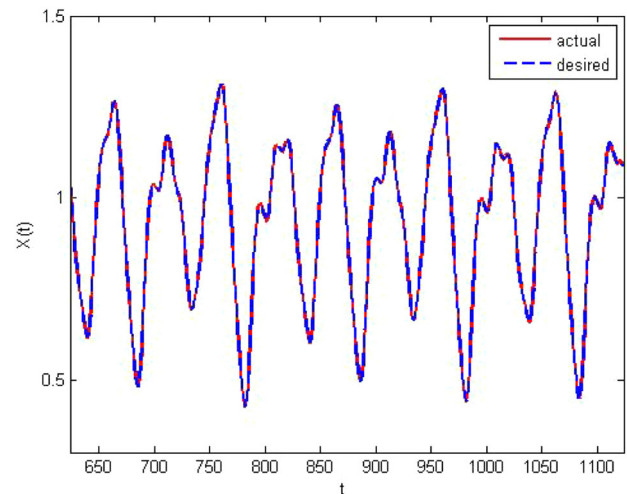
the ones of the ACO<sub>R</sub>, PSO, and ACACO. The  $p$  values in Table 6 indicate that the null hypothesis is rejected at the significance level of 5% for each comparison using the Wilcoxon test for the samples of testing errors  $RMSE_{test}$ . In [7], the TRFN was trained by the gradient-based supervised learning (TRFN-S) for predicting the same chaotic time series. The testing  $RMSE_{test}$  using the TRFN-S is 0.012, which is larger than that by the EMCACO-C. The prediction outputs from  $t = 624$  to 1123 of the optimized TRFN by the EMCACO-C are demonstrated in Fig. 12. The results show nearly undistinguishable difference between the actual chaotic series and the predicted series.

*Example 5* (Dynamic system identification) This example as taken from [9] uses a TRFN to identify a nonlinear dynamic system with multiple time delays. The nonlinear plant is described as follows:

$$y_p(k + 1) = f(y_p(k), y_p(k - 1), y_p(k - 2), \lambda(k), \lambda(k - 1)) \tag{28}$$

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2}. \tag{29}$$



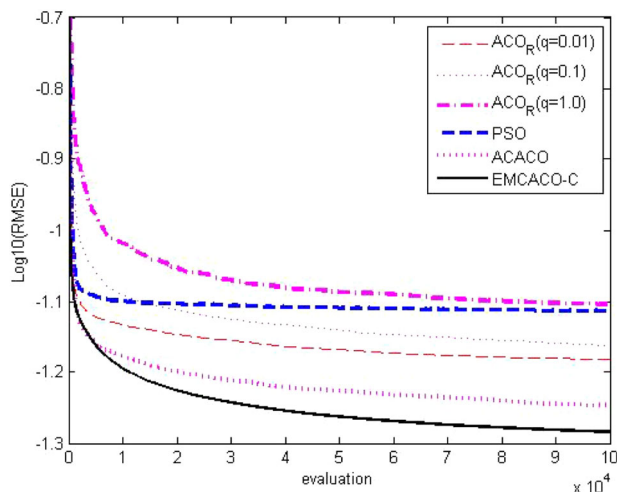
**Fig. 12** Test results of the TRFN predictor optimized by the EMCACO-C in Example 4

**Table 6** Performances of the TRFN designed by various population-based optimization algorithms for the prediction of nonlinear chaotic time series in Example 4

Algorithms	ACO <sub>R</sub> ( $q = 1.0$ )	ACO <sub>R</sub> ( $q = 0.1$ )	ACO <sub>R</sub> ( $q = 0.02$ )	ACO <sub>R</sub> ( $q = 0.01$ )	PSO	ACACO	EMCACO-C
RMSE <sub>train</sub> avg.	0.0393	0.0241	0.0252	0.0256	0.0612	0.0125	<b>0.0077</b>
RMSE <sub>test</sub> avg.	0.0388	0.0240	0.0250	0.0280	0.0603	0.0126	<b>0.0078</b>
RMSE <sub>test</sub> SD	0.0118	0.0055	0.0139	0.0238	0.0278	0.0038	0.0020
RMSE <sub>test</sub> (best)	0.0251	0.0139	0.0087	0.0082	0.0190	0.0058	<b>0.0042</b>
$p$ value	3.8e−10	3.8e−10	4.0e−10	4.8e−10	3.8e−10	8.9e−09	–

Bold value indicates the minimum value in the same comparison item

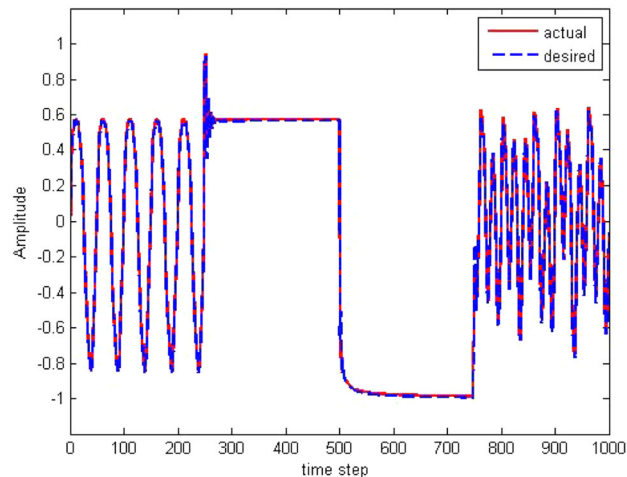




**Fig. 13** Average best-so-far training RMSEs at each performance evaluation for the evolutionary TRFN models optimized by the EMCACO-C and different population-based optimization algorithms in Example 5

The current output  $y_p(k)$  and the control input  $\lambda(k)$  are the inputs of the TRFN model, and the value of  $y_p(k+1)$  is the target output for the TRFN. The objective function is defined as the RMSE between the plant output in (28) and the TRFN output.

The TRFN identification model to be optimized is composed of three fuzzy rules. The free parameters  $m_{ij} \in [-1.5, 1.5]$ ,  $b_{ij} \in [0, 1.5]$ ,  $a_{ij} \in [-1.5, 1.5]$ ,  $v_{ij} \in [-1.5, 1.5]$  are searched for finding a better-performed TRFN model. The training data for the TRFN are the control inputs of 900 time steps and the corresponding outputs produced by the plant. The first 450 control inputs are independently and identically distributed uniform random sequences over  $[-1, 1]$ . The remaining half control inputs are generated by a sinusoid  $1.05\sin(\pi k/45)$ . The 100,000 evaluations were performed in a single run of the optimization process. To see the identification performance, the following guiding input signal is used for test:



**Fig. 14** Test results of the TRFN model optimized by the EMCACO-C in Example 5

$$\lambda(k) = \begin{cases} \sin(\pi k/25), & k < 250 \\ 1.0, & 250 \leq k < 500 \\ -1.0, & 500 \leq k < 750 \\ 0.3\sin(\pi k/25) + 0.1\sin(\pi k/32) + 0.6\sin(\pi k/10), & 750 \leq k < 1000. \end{cases} \quad (30)$$

With 50 independent runs of simulations, the average best-so-far training RMSE at each performance evaluation of each algorithm for the TRFN is shown in Fig. 13. Table 7 shows the learned numerical results of statistical training and testing RMSEs. The results show that the training error  $\text{RMSE}_{\text{train}}$  and testing error  $\text{RMSE}_{\text{test}}$  of the EMCACO-C are smaller than the ones of the  $\text{ACO}_R$ , PSO, and ACACO. The  $p$  values in Table 7 indicate that the null hypothesis is rejected at the significance level of 5% for each comparison using the Wilcoxon test for the samples of testing errors  $\text{RMSE}_{\text{test}}$ . The same problem was also studied using TRFN-S [9]. The TRFN-S reached the testing error  $\text{RMSE}_{\text{test}}$  of 0.0346, which is larger than that by the EMCACO-C. The outputs of the optimized TRFN by the EMCACO-C for the test input in (30) are demonstrated in Fig. 14. The results show TRFN output is very close to the output of dynamic plant.

**Table 7** Performances of the TRFN designed by various population-based optimization algorithms for the system identification problem in Example 5

Algorithms	$\text{ACO}_R (q = 1.0)$	$\text{ACO}_R (q = 0.1)$	$\text{ACO}_R (q = 0.02)$	$\text{ACO}_R (q = 0.01)$	PSO	ACACO	EMCACO-C
$\text{RMSE}_{\text{train}}$ avg.	0.0787	0.0687	0.0640	0.0658	0.0775	0.0567	<b>0.0520</b>
$\text{RMSE}_{\text{test}}$ avg.	0.0618	0.0503	0.0444	0.0452	0.0698	0.0433	<b>0.0286</b>
$\text{RMSE}_{\text{test}}$ SD	0.0089	0.0125	0.0139	0.0139	0.0175	0.0124	0.0067
$\text{RMSE}_{\text{test}}$ (best)	0.0444	0.0228	0.0218	0.0223	0.0515	0.0224	<b>0.0216</b>
$p$ value	$3.8e-10$	$4.5e-10$	$7.7e-08$	$1.4e-08$	$3.8e-10$	$3.8e-09$	–

Bold value indicates the minimum value in the same comparison item

### 4.3 Analysis of the EMCACO-C

#### 4.3.1 Analysis of Computational Complexity

The time complexity of the EMCACO-C as applied to TRFN design is proportional to the number  $N_{cycles}$  of ant cycles (iterations), the number  $L$  of new candidate solutions, the runtime  $T_{gen}$  for generating a new ant, and the evaluation time  $T_{eva}$  for a new solution, and can be computed as follows:

$$N_{cycles} \times (L \times (T_{gen} + T_{eva})). \tag{31}$$

In the simulation results of Example 1 as an example, the computational time of each evolutionary learning for TRFN design for tracking control was also reported in Table 2. The results indicate the time the EMCACO-C consumed is comparable to that of the  $ACO_R$ , which implies that in this example the runtime  $T_{gen}$  in the EMCACO-C is not much different from that in the  $ACO_R$  or both of them are negligible compared to the evaluation time  $T_{eva}$  for a new solution.

#### 4.3.2 Performance Analysis of the Central Initialization on TRFN Designs

In addition to the results in Table 1, a cross-validation on three of the simulated examples has been conducted for discussing the advantages of the central initialization on the experimented TRFN designs. The validation results for the EMCACO-C, PSO, and  $ACO_R$  are presented in Table 8. The results show that for each algorithm itself, the central initialization helps achieve the smaller error than the blind

uniform initialization. This demonstrates that the proposed central initialization can help the investigated swarm-based algorithms including the EMCACO-C achieve the better performance on the TRFN designs for application examples.

Furthermore, to learn whether the proposed central initialization ( $\alpha = 0.2$ ) confines the EMCACO-C convergence to the initial local guesses around or not, we need to examine the parameters of the designed TRFN. The parameters of the best designed TRFN controller optimized by the EMCACO-C in Example 1 are shown in Table 3. We found that 25% (12/48) of the parameters fall in the initially guessed region, which is slightly more than the case (20%) of the blind uniform guess. We also observed that some of the parameter values in the designed TRFN are equal or very close to the specified limit for search, which has demonstrated that if the proper guess width is used, the central initialization does not necessarily confine the search only toward the initial local guess.

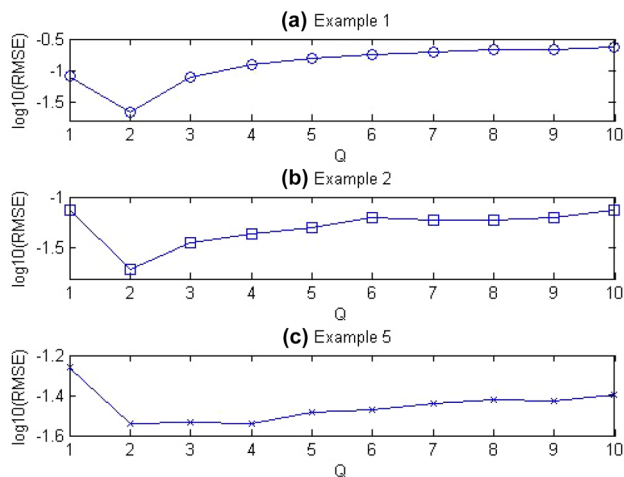
#### 4.3.3 Analysis of the Number of Elites and the Evaporation Rate

The value of 2 for  $Q$  currently used in the EMCACO-C for applications simulations was determined simply based on the cross-validation results for the four-rule TRFN in Example 1 when the evaporation rate  $\varepsilon$  is 0.85 as that used in [29]. To learn more about the influences of  $Q$  value on the performances of the EMCACO-C for TRFN designs, more experiments on Examples 1, 2, and 5 using the values of  $Q$  ranging from 1 to 10 were simulated. The results in Fig. 15 show that the best performances are achieved when

**Table 8** Performance comparisons of the EMCACO,  $ACO_R$ , and PSO as applied to the TRFN designs by using the different guess widths for initialization

Algorithms guess width $\alpha$	PSO		$ACO_R$ ( $q = 0.02$ )		EMCACO-C	
	1.0	0.2	1.0	0.2	1.0	0.2
Example 1 (4 rules)						
RMSE avg.	0.1391	0.0552	0.0914	0.0344	0.0963	<b>0.0281</b>
RMSE (best)	0.0305	0.0180	0.0189	0.0114	0.0189	<b>0.0088</b>
Example 1 (8 rules)						
RMSE avg.	0.1425	0.0400	0.0675	0.0464	0.0624	<b>0.0216</b>
RMSE (best)	0.0399	0.0136	0.0233	0.0145	0.0223	<b>0.0067</b>
Example 2 (6 rules)						
RMSE avg.	0.1096	0.0394	0.0576	0.0367	0.0534	<b>0.0242</b>
RMSE (best)	0.0333	0.0137	0.0208	0.0104	0.0150	<b>0.0058</b>
Example 2 (8 rules)						
RMSE avg.	0.0921	0.0351	0.0473	0.0328	0.0449	<b>0.0195</b>
RMSE (best)	0.0259	0.0165	0.0102	0.0072	0.0108	<b>0.0043</b>
Example 5						
RMSE <sub>test</sub> avg.	0.0698	0.0613	0.0444	0.0291	0.0430	<b>0.0286</b>
RMSE <sub>test</sub> (best)	0.0515	0.0511	0.0218	0.0217	<b>0.0205</b>	0.0216

Bold value indicates the minimum value in the same comparison item



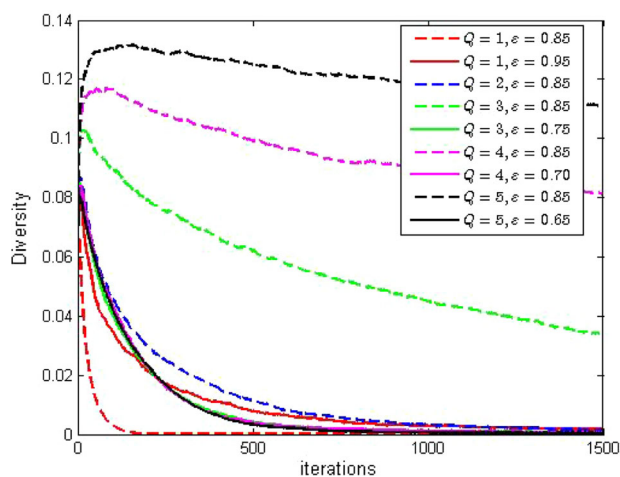
**Fig. 15** Comparison about the performance achieved by the EMCACO-C with respect to different values of  $Q$

two elites are used for mixing operation. It is also observed that the error decreased significantly when the value of  $Q$  was changed from one to two. When the value of  $Q$  is one, the EMCACO-C actually did not mix elites, and thus, the Gaussian sampling was the only operation in the algorithm, which is the possible reason for being inferior. Furthermore, the average error increases roughly with the value of  $Q$  starting from 2. One of the possible reasons is that a higher value of  $Q$  provides more exploration in search thus lower convergence speed and the worse performance.

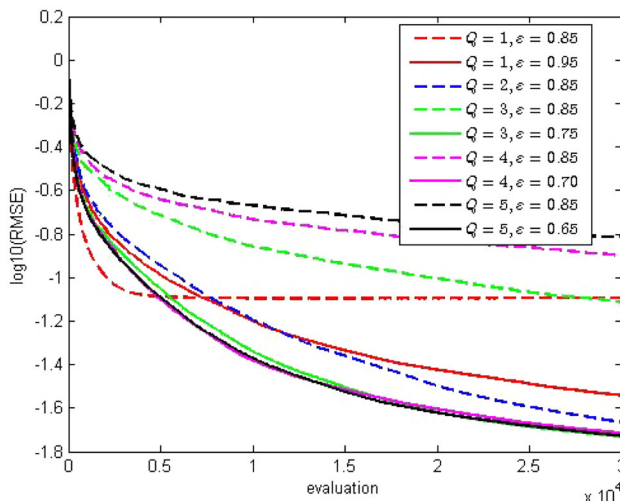
To verify the above reason, the normalized diversity of all ant solutions is defined to indicate the convergence of learning and is computed as follows:

$$\text{Diversity}(t) = \frac{1}{N-1} \sum_{m=1}^N \frac{1}{D} \sum_{j=1}^D \frac{|s_m^j(t) - s_1^j(t)|}{L^j}, \quad (32)$$

where  $t$  is the iteration index and  $L^j$  is the search length for the parameter in the  $j$ th dimension. With 100 independent runs, the average normalized diversity of the EMCACO-C with each different  $Q$  value in Example 1 as an example is shown in Fig. 16. The results show that when  $\epsilon$  is fixed to 0.85, the EMCACO-C with a higher value of  $Q$  converges slower than that with a lower value of  $Q$ . However, because a higher value of  $\epsilon$  can provide more exploration in search and a lower value of  $\epsilon$  can provide more exploitation in search, different values of  $\epsilon$  can be used to respond to the selected  $Q$  value for balancing the exploration and exploitation based on the following guideline: the higher the value of  $Q$  is selected, the lower the value of  $\epsilon$  is used. The diversity results using the different values of  $\epsilon$  to respond to the selected  $Q$  value are also shown in Fig. 16 for comparison. The corresponding learning curves of performance evaluations are shown in Fig. 17, indicating



**Fig. 16** Average normalized diversity of the TRFN parameters optimized by the EMCACO-C with different values of  $Q$  and  $\epsilon$  in Example 1 (Dash lines are the cases of  $\epsilon = 0.85$ ; solid lines are the cases for different values of  $\epsilon$ )



**Fig. 17** Average best-so-far training RMSEs optimized by the EMCACO-C with different values of  $Q$  and  $\epsilon$  in Example 1 (Dash lines are the cases of  $\epsilon = 0.85$ ; solid lines are the cases for different values of  $\epsilon$ )

that the resulting performances using the adjusted value of  $\epsilon$  are much improved and some of them are even better than the case of  $Q = 2$  and  $\epsilon = 0.85$ .

### 5 Conclusions

This paper proposes an efficient continuous ACO, elite-mixed continuous ACO with central initialization (EMCACO-C), for improving the accuracy of TRFN designs. The EMCACO-C is developed based on the framework of the  $ACO_R$  and can be regarded as a new

population-based evolutionary optimization algorithm. Unlike most popular population-based algorithms, the EMCACO-C initializes the ant solutions on concentrative region around the center of the search range. The EMCACO-C mixes in scanning manner the fixed number of the few best elites uniformly in probability to form a directional solution. A new candidate solution is generated by sampling Gaussian density function around the constructed directional solution. The methodology similarities and differences between the EMCACO-C and GAs are also compared in this study.

In the TRFN designs for the problems of dynamic systems control, dynamic system identification, and chaotic series prediction, the simulation results demonstrated that the EMCACO-C achieved smaller errors than those by the algorithms of ACO<sub>R</sub>, PSO, and ACACO. The Wilcoxon signed-rank tests were conducted to verify the significant differences in comparisons.

In addition, the advantages on the accuracy improvement for TRFN designs gained by applying the central initialization to the ACO<sub>R</sub> and PSO were also demonstrated in this study. The selection guidelines for the number of elites for mixing operation and the value of the evaporation rate in the EMCACO-C for TRFN design were discussed and demonstrated by the simulation results of the application example.

In the future, the performances of the EMCACO-C for the designs of various types of recurrent and feed-forward fuzzy systems will be studied. The study of different skewed initializations and elites selection techniques for finding the performance improvement will also be considered. In addition, hybridization of other evolutionary optimization algorithms with the EMCACO-C may further improve optimization performance and thus will also be studied in the future.

**Acknowledgements** This work was supported by Ministry of Science and Technology, Taiwan under Grants MOST 104-2221-E-415-006 and MOST 105-2221-E-415-018.

## References

- Zhang, J., Morris, A.J.: Recurrent neuro-fuzzy networks for nonlinear process modeling. *IEEE Trans. Neural Netw.* **10**(2), 313–326 (1999)
- Wang, Y.C., Chien, C.J., Teng, C.C.: Direct adaptive iterative learning of control of nonlinear systems using output-recurrent fuzzy neural network. *IEEE Trans. Syst. Man Cybern. B Cybern.* **34**(3), 1348–1359 (2004)
- Mouzouris, G.C., Mendel, J.M.: Dynamic nonsingleton fuzzy logic systems for nonlinear modeling. *IEEE Trans. Fuzzy Syst.* **5**(2), 199–208 (1997)
- Savran, A.: An adaptive recurrent fuzzy system for nonlinear identification. *Appl. Soft Comput.* **7**(2), 593–600 (2007)
- Lee, C.H., Teng, C.C.: Identification and control of dynamic systems using recurrent fuzzy neural networks. *IEEE Trans. Fuzzy Syst.* **8**(4), 349–366 (2000)
- Lin, C.J., Chin, C.C.: Prediction and identification using wavelet based recurrent fuzzy neural networks. *IEEE Trans. Syst. Man Cybern. B Cybern.* **34**(5), 2144–2154 (2004)
- Juang, C.F., Lin, Y.Y., Tu, C.C.: A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing. *Fuzzy Sets Syst.* **161**(19), 2552–2568 (2010)
- Juang, C.F., Lin, C.T.: A recurrent self-organizing neural fuzzy inference network. *IEEE Trans. Neural Netw.* **10**(4), 828–845 (1999)
- Juang, C.F.: A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. *IEEE Trans. Fuzzy Syst.* **10**(2), 155–170 (2002)
- Lin, Y.-Y., Chang, J.-Y., Lin, C.-T.: Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. *IEEE Trans. Neural Netw.* **24**(2), 310–321 (2013)
- Goldberg, D.E.: *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, Reading (1989)
- Surmann, H., Maniadakis, M.: Learning feed-forward and recurrent fuzzy systems: a genetic approach. *J. Syst. Architect.* **47**(7), 649–662 (2001)
- Mucients, M., Moreno, D.L., Bugarin, A., Barro, S.: Design of a fuzzy controller in mobile robotics using genetic algorithms. *Appl. Soft Comput.* **7**(2), 540–546 (2007)
- Lin, C.J.: A GA-based neural fuzzy system for temperature control. *Fuzzy Sets Syst.* **143**(2), 311–333 (2004)
- Lin, F.J., Huang, P.K., Chou, W.D.: Recurrent-fuzzy-neural-network-controlled linear induction motor servo drive using genetic algorithms. *IEEE Trans. Ind. Electron.* **54**(3), 1449–1461 (2007)
- Eberchart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: *Proceedings International Symposium on Micro Machine and Human Science*, Nagoya, pp. 39–43 (1995)
- Juang, C.F., Chung, I.F., Hsu, C.H.: Automatic construction of feedforward/recurrent fuzzy systems by clustering-aided simplex particle swarm optimization. *Fuzzy Sets Syst.* **158**(18), 1979–1996 (2007)
- Khanesar, M.A., Shoorehdeli, M.A., Teshnehlab, M.: Hybrid training of recurrent fuzzy neural network model. In: *Proceedings of the International Conference on Mechatronics and Automation*, pp. 2598–2603 (2007)
- Juang, C.F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Trans. Syst. Man Cybern. B Cybern.* **34**(2), 997–1006 (2004)
- Olivas, F., Valdez, F., Castillo, O., Melin, P.: Dynamic parameter adaptation in particle swarm optimization using interval type-2 fuzzy logic. *Soft. Comput.* **20**(3), 1057–1070 (2016)
- Dorigo, M., Maniezzo, V., Colnani, A.: Ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. B Cybern.* **26**(1), 29–41 (1996)
- Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperating learning approach to the traveling salesman problem. *IEEE Trans. Evolut. Comput.* **1**(1), 53–66 (1997)
- Stutzle, T., Hoos, H.H.: Max–Min ant system. *Future Gener. Comput. Syst.* **16**(9), 889–914 (2000)
- Dorigo, M., Stutzle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
- Castillo, O., Neyoy, H., Soria, J., Melin, P., Valdez, F.: A new approach for dynamic fuzzy logic parameter tuning in Ant Colony Optimization and its application in fuzzy control of a mobile robot. *Appl. Soft Comput.* **28**, 150–159 (2015)

26. Olivas, F., Valdez, F., Castillo, O., Gonzalez, C.I., Martinez, G., Melin, P.: Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems. *Appl. Soft Comput.* **53**, 74–87 (2017)
27. Castillo, O., Lizárraga, E., Soria, J., Melin, P., Valdez, F.: New approach using ant colony optimization with ant set partition for fuzzy control design applied to the ball and beam system. *Inf. Sci.* **294**, 203–215 (2015)
28. Valdez, F., Castillo, O., Melin, P.: Ant colony optimization for the design of Modular Neural Networks in pattern recognition. In: *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pp. 163–168 (2016)
29. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. *Eur. J. Oper. Res.* **185**(3), 1155–1173 (2008)
30. Juang, C.F., Hung, C.W., Hsu, C.H.: Rule-based cooperative continuous ant colony optimization to improve the accuracy of fuzzy system design. *IEEE Trans. Fuzzy Syst.* **22**(4), 723–735 (2014)
31. Chen, C.C., Shen, L.P., Huang, C.F., Chang, B.R.: Assimilation-accommodation mixed continuous ant colony optimization for fuzzy system design. *Eng. Comput.* **33**(7), 1882–1898 (2016)
32. Juang, C.F., Chang, P.H.: Recurrent fuzzy system design using elite-guided continuous ant colony optimization. *Appl. Soft Comput.* **11**(2), 2687–2697 (2011)
33. Chen, C.C., Shen, L.P.: Recurrent fuzzy system design using mutation-aided elite continuous ant colony optimization. In: *Proceedings of IEEE International Conferences on Fuzzy Systems*, pp. 1642–1648 (2016)
34. Eiben, A.E., Raue, P.-E., Ruttkay, Z.: Genetic algorithms with multi-parent recombination. In: *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pp. 78–87 (1994)
35. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **1**(1), 3–18 (2011)
36. Lehman, B., Bentsman, J., Lunel, S.V., Verriest, E.I.: Vibrational control of nonlinear time lag systems with bounded delay: averaging theory, stabilizability, and transient behavior. *IEEE Trans. Automat. Control* **39**(5), 898–912 (1994)

**Chi-Chung Chen** received the B.S. and M.S. degrees in Control Engineering from National Chiao Tung University, Taiwan, in 1993 and 1995, respectively. He received the Ph.D. degree in Electrical Engineering from University of California, Los Angeles (UCLA) in 2000. In 2001–2014, he worked in the technology industry and focused on the development of wireless communication products. In 2014, he joined National Chiayi University, Taiwan, where he is currently an assistant professor of Electrical Engineering. His current research and professional interests are neural fuzzy systems, intelligent control, and evolutionary computation. He is a member of Phi Tai Phi and IEEE.

**Li Ping Shen** received the B.S. and M.S. degrees in Electrical Engineering from National Chiayi University, Taiwan, in 2014 and 2016, respectively.