A self-adaptive artificial bee colony algorithm with local search for TSK-type neuro-fuzzy system training

Kuang-Pen Chou
Institute of Computer Science and Engineering
National Chiao-Tung University
Hsinchu, Taiwan
kpchou.cs00g@nctu.edu.tw

Chin-Teng Lin
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
Chin-Teng.Lin@uts.edu.au

Wen-Chieh Lin
Department of Computer Science
National Chiao-Tung University
Hsinchu, Taiwan
wclin@cs.nctu.edu.tw

Abstract—In this paper, we introduce a self-adaptive artificial bee colony (ABC) algorithm for learning the parameters of a Takagi-Sugeno-Kang-type (TSK-type) neuro-fuzzy system (NFS). The proposed NFS learns fuzzy rules for the premise part of the fuzzy system using an adaptive clustering method according to the input-output data at hand for establishing the network structure. All the free parameters in the NFS, including the premise and the following TSK-type consequent parameters, are optimized by the modified ABC (MABC) algorithm. Experiments involve two parts, including numerical optimization problems and dynamic system identification problems. In the first part of investigations, the proposed MABC compares to the standard ABC on mathematical optimization problems. In the remaining experiments, the performance of the proposed method is verified with other metaheuristic methods, including differential evolution (DE), genetic algorithm (GA), particle swarm optimization (PSO) and standard ABC, to evaluate the effectiveness and feasibility of the system. The simulation results show that the proposed method provides better approximation results than those obtained by competitors methods.

Index Terms—Evolutionary algorithm (EA), Artificial bee colony (ABC)optimization, Neuro-fuzzy system (NFS)

I. Introduction

Dynamic systems are commonly found in many real-world applications, such as control, pattern recognition, and other engineering problems [1] [2] [3]. Especially in control areas, the goal is to develop a system to describe the time dependence of a serial of given states in a geometrical manifold. At any given time, the state of a dynamic system, which is represented by a tuple of real numbers, corresponds to a particular point in the state space. The output of a dynamic system is derived with evolution rules from past states, including both input and output before, and current states.

The integration of fuzzy logic and artificial neural network is widely applied to construct control systems for solving non-linear problems in the machine learning community [2] [3], which has advantages from both high-level reasoning ability of fuzzy logic and low-level learning power of neural network. Moreover, the neuro-fuzzy system (NFS) transforms humanlike decision-making as a family of fuzzy IF-THEN rules to express the locally linear relations between input and output. By blending linear relations via fuzzy membership functions without complicated mathematics, the overall NFS can be established to present a non-linear model.

One critical design issue while building a NFS is to deter-mine the number of fuzzy rules used to cover each input di-mension. Most researchers employ a firing strength criterion to generate fuzzy sets automatically. Numerous algorithms have been developed to find proper parameters in the learning phase. These learning methods can be categorized into two groups: 1) derivate-based and 2) metaheuristic-based methods. The derivative-based method is also known as back-propagation (BP) method [4], which aims to minimize the cost function via propagating the gradient value of output error to adjust the value of each neuron. In most cases, the BP method works efficiently to find a set of parameters. However, it may fall into locally optimal solution due to its intrinsic property to find global optimum solutions hard. In contrast, numerous metaheuristic-based methods have introduced [5], which finds global optimums via bio-inspired methods. The evolutionary algorithm (EA) is one of the representative method, which obtains global optimums out by mimicking evolutionary mech-anism and behavior in a biologic group. These nature-inspired based techniques have been widely applied in various areas, including optimization problems, identification problems and classification problems.

In comparison with the derivative-based methods, the EA commonly utilizes the information obtained from the individ-ual in human-made population without any gradient informa-tion [6]. EA has a higher chance to escape from local optimal solutions by operating different mutation mechanism. More-over, crossover mechanism promotes the information to be exchanged within the population to obtain better search results as possible, and elimination mechanism replaces unsuitable solution with a new one to improve the search effectivity.

In literature, different protocols of swarm intelligence-based algorithms, including particle swarm optimization (PSO), ge-netic algorithm (GA), differential evolution (DE) and artificial

978-1-7281-2153-6/19/$31.00 c 2019 IEEE

bee colony (ABC), have been proposed to learn the parameters of artificial neural networks [7] [5] [8]. Amongthe various algorithms mentioned above, the ABC hasshown an impres-sive performance on different constraint problems. The ABC is introduced by Karaboga [9], which imitates the foraging behavior of natural honeybees, and evaluated on numerical optimization problems [10].

In this study, we introduce a modified ABC (MABC) algorithm to train the parameters of TSK-type NFS, named NFS MABC, for solving non-linear system problems. The proposed NFS MABC comprises two parts, including the structure learning and the parameter learning. In the structure learning phase, the number of fuzzy rules for each input dimension is determined with an adaptive clustering method. Moreover, the semantic term of first-order TSK-type reasoning is regarded as the subsequent clause of each fuzzy rule. In the parameter learning phase, the proposed MABC is applied for parameter optimization by a batch learning mode. The MABC is different from the standard ABC. We introduce an adaptive mutation parameter to make the process of evolution more efficient. Moreover, ranking-based weights are utilized to keep the difference significantly between individuals. Finally, the search strategy of local neighbor is applied to avoid convergence early to trap at local optimal solutions.

The remainder of this study is organized as follows. Section II introduce the structure of the NFS. Section III presents a description of the standard ABC and the scheme of pro-posed MABC. In Section IV, numerical optimization and dynamic system identification problems are used to reveal the performance of the proposed method, respectively. Finally, conclusion is given in Section V.

II. Structure of the Neuro-Fuzzy System

In this section, we introduce the architecture of the NFS used in this study. The structure of the NFS is established according to the pairs of the input-output datums at hand with an adaptive clustering algorithm and the TSK-type fuzzy reasoning. The proposed NFS comprises five layers and nodes in each layer serve as neurons. Fig.1 depicts the overview of the NFS. In the premise part of the NFS, distinct fuzzy sets are applied to generate fuzzy rules. Moreover, the first-order TSK-type fuzzy reasoning serves as the consequent part of each fuzzy IF-THEN rule. Note that the structure of the NFS is determined in the first generation via go through all training datum pairs and obtained the initial weights. In the remaining generations, the architecture of the NFS is fixed, the learning procedure only adjusts values of parameters in the NFS. The mathematical functions of each layer in the NFS are given as

follows. The input of the NFS is defined , X = (x1, . . . ,
where is the dimenson of the input ~ .)

Layer 1 (Input Layer): Each node in layer 1 corresponds to one input variable and transmit it to next layer as input with no computation.

$$u^1_i = x_i.$$

Layer 2 (Fuzzification Layer): Each node in this layer defines a membership function that calculates the output from the previous layer into a fuzzy set.

$$u^2_{i,j} = \exp\left(-\frac{u^1_i - m_{i,j}}{\sigma_{i,j}}\right)^2. \quad (2)$$

where $m_{i,j}$ and $\sigma_{i,j}$ denotes the mean and variance of the Gaussian membership function for the jth term of the ith input variable x_i, respectively.

Layer 3 (Spatial Firing Layer): Each node in layer 3 corresponds to one fuzzy rule and performs as a spatial rule node. A spatial firing strength u_{3j} is calculated using fuzzy AND operator to perform the premise part of each rule j. The spatial firing strength is defined as the expression below.

$$u_j^3 = F^J = \prod_j u_{i,j}^2 . \quad (3)$$

Layer 4 (Consequent Layer): Each node in layer 3 has a corresponding node in this layer named as a consequent node. Nodes in layer 4 perform a linear combination of the input variables. The output of each consequent node is defined as follows:

$$u_j^4 = \sum_{i=1}^{X_n} a_{i,j} x_i(t). \quad (4)$$

where $a_{i,j}$ is a weight linking from a node in the previous layer to a node in the current layer.

Layer 5 (Output Layer): The node in layer 5 is the output node of the NFS, which performs defuzzification operation to integrate all of actions of previous layers as the expression below.

$$y = u_j^5 = \frac{\sum_{j=1}^{R} F_j \left(\sum_{i=1}^{n} a_{i,j} x_i \right)}{\sum_{j=1}^{R} J} , \quad (5)$$

where R and y are the total number of fuzzy rules and the output of the NFS, respectively.

III. Parameter Optimization in the NFS

A. The standard ABC

The ABC algorithm, introduced by Karaboga [9], imitates the foraging behavior of natural bee swarms. In comparison with other inspired-heuristic approaches, except the essential parameters, the ABC algorithm only requires one additional parameter, limit, to determine when to abandon exhausted solutions. The foraging behavior in the ABC algorithm is treated as finding an optimum solution for a particular problem. Each food source in the ABC algorithm corresponds to a possible solution for the target problem; namely, the amount of nectar contained in a food source indicates the adaptability of the solution to the problem. In the standard ABC algorithm, three kinds of artificial agents are defined, including employed bees, onlooker bees, and scout bees, and corresponding to different stages with distinct search strategies, respectively. Moreover, the number of employed bees and onlooker bees are often equal to the size of the colony. The main learning steps of the

(1) standard ABC algorithm are given as follows:

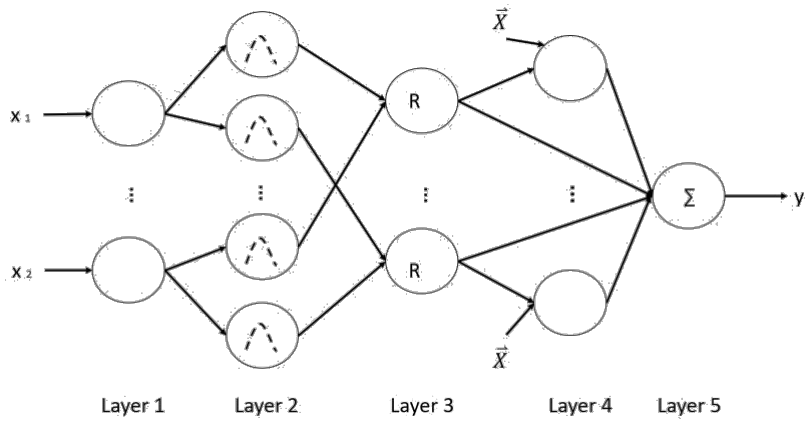

Fig. 1. Structure of the neuro-fuzzy system which is established using adaptive clustering method and TSK-type fuzzy reasoning to express the IF-THEN rules.

- Initialize the population and its corresponding parameters
- REPEAT
  - Employed bees explore the environment according to its previous memory.
  - Onlooker bees search for better sources near the hive depending on the feedback from employed bees on the dancing area.
  - Scout bees assign new positions of food sources to replace exhausted food sources.
- UNTIL (terminated condition)

At the first step, the initial population of SN food sources, $X = X_1,...,X_{SN}$ is generated randomly, where X_i is a D-dimensional vector presented as $X_i = (x_{i,1}, \ldots, x_{i,D})$ and is initialized using (6) , where max_j and min_j are the upper bound and the lower bound at dimension j, respectively.

$$x_{i,j} = min_j + rand(0, 1)_j(max_j - min_j). \quad (6)$$

In every search procedure, an employed bee or an onlooker bee randomly selects a neighbor as the reference and modifies on arbitrary one parameter to produce an offspring source as below:

$$v_j = x_{i,j} + \varphi_j(x_{i,j} - x_{k,j}), \quad (7)$$

where $k \in [1, \ldots, SN]$ but different from i and, $j \in [1, \ldots, D]$. φ_j is a scaling factor in the interval of [-1,1] to determine the offset between the current source and the candi-date source. The quality of the candidate source, fitness value, is computed according to its cost value. The greedy method is applied to the sifting mechanism; namely, a candidate solution with higher fitness than its parent solution, then the older one will be replaced.

Although the number of employed bees and onlooker bees is equal to the size of the population, the selection strategy is different in these two phases. At employed bee phase, each food source is ensured to be investigated once in each generation. In contrast to a fair selection, a food source be chosen by an onlooker bee to explore according to its fitness value. A source with higher fitness value has a higher chance to be chosen to investigate by the roulette wheel method presented as the following expression.

$$prob_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i}. \quad (8)$$

$$v_j = x_{m,j} + \varphi_j(x_{n,j} - x_{o,j}). \quad (10)$$

In the scout bee stage, food sources which can not be improved at least a predefined limit times of exploration will be abandoned by scout bees and replaced by new one using (6).

B. The proposed Modified ABC

The previous section introduced the fundamental concept of the ABC algorithm. In this section, we present a modi-fied ABC named as MABC. Since the ABC algorithm has been shown to be competitive with other conventional bio-inspired algorithms on numerical optimization problem, many researchers carry out in-depth research on ABC. Moreover, the ABC algorithm is widely used to solve engineering problems. The improvement of ABCs can be roughly divided into three categories: 1) introduce new solution search equations [11] [12] [13], 2) blend with other operations [14] [15], and 3) adopt a multi-population strategy [10].

In the standard ABC, only one dimension of each consid-ered source changes in each investigation that causes slow convergence speed on high dimension problems. The previous study introduced a mutation parameter, MR, to accelerate the evolution speed. Each position of a considered food source has a chance to change depending on the predefined mutation rate; namely, if a random number is large than the preset value, the value of that position keep. In this study, we follow the concept of mutation and integrate the trail times of each food source with an adaptive mutation scheme as shown in (9), where trails(i) is the number of exploration times for source i that cannot be improved. With the increase of the exploration

$$\text{cand}_j = \begin{cases} x^j_{i,j}. \\ v \end{cases} \quad \text{if} \quad \text{rand}(0,1)_j \geq MR \cdot \exp^{-\frac{(\text{rand}(0,1)_j \cdot \text{trials}(i))}{\text{limit}}}; \text{ otherwise,} \tag{9}$$

$$v_j = \begin{cases} x_{i,j} + \varphi_j(x_{m,j} - x_{n,j}), & fit_i \geq fit_m \text{ and } fit_i \geq fit_n; \text{ otherwise,} \\ x_{lbest,j} + \varphi_j(x_{lbest,j-x_{i,j}}), & lbest = k,\ k \in [m, n],\ \text{s.t. } fit_k \geq \max(fit_m, fit_n). \end{cases} \tag{11}$$

times that a food source cannot be improved, an artificial agent should employ mutation strategy on this food source carefully and discreetly.

Besides, it is commonly a challenging issue to get a balance between exploration and exploitation when researchers develop EA algorithms. In this study, we adopt a similar search strategy to the standard ABC, but a different formula in the employed bees stage, to maintain the ability of exploration as shown in (10), where m, n, o ∈[1, SN] and m 6= n 6= o 6= i.

The information of the local best solution is utilized as the guide term to improve the ability of exploitation in the onlooker bee stage. For each search iteration, an onlooker bee not only selects a food source according to the selection probability but also randomly picks two of its neighbors for competition to produce local best source. The one has a higher fitness value than others is regarded as the local best term and be used to dominate the search process as (11).

Moreover, we notice that the difference between food sources becomes ambiguous when they have similar fitness. The phenomenon causes that onlooker bees are hard to choose a solution with distinguished to explore; in other words, the ability of exploration in the onlooker bee stage decreases. To maintain the difference between individuals, we introduce a ranking-based weight mechanism to replace the probability calculated directly according to fitness values of each food source. In the information exchange stage, each individual in the population are sorted according to its fitness value in descending order and assigned weights according to the arrangement order by (12).

$$w_r = \frac{1}{\delta SN\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{r-1}{\delta SN}\right)^2}, \tag{12}$$

where r is the rth solution, δSN is the standard deviation, and δ is a adjustable parameter. When δ is large, the weights of individuals are near uniform; otherwise, the best-ranking soution are strongly preferred in the onlooker bee stage. The probability of each solution to be chosen is computed by (13). As mentioned before, solutions cannot be improved furthermore will be abandoned in the scout bee stage. Inspired by opposition-based learning [15], the opposite position of the poor source is evaluated to gain the opportunity to escape from a local optimum to find a better solution. Using (6) and (14) to generate two candidate solution, and the one has higher fitness will be preserved. The overview of the proposed MABC is shown as algorithm 1.

$$new\_prob_i = \frac{w_i}{\sum_{i=1}^{SN} w_i}. \tag{13}$$

$$oppo_j = \text{rand}(0, 1)_j(max_j + min_j) - x_{i,j}. \tag{14}$$

Algorithm 1: Pseudo code of the proposed MABC

1: Initialize population using (6) and the required parameters
2: Evaluate the current population
3: set F Es = SN and Gen = 0
4: repeat
5: //**Employed bee stage**//
6: Employed bees explore all food sources using (10).
7: //**Information exchange**//
8: Assign each food source a probability to be chosen using (13) with its corresponding weight depending on the ranking order as (12).
9: //**Onlooker bee stage**//
10: Onlooker bee selects a food source by roulette wheel selection.
11: Onlooker bee searches the selected source and its local neighbors using (11).
12: //**Scout bee stage**//
13: Scout bees examine food sources and replace exhausted sources by (6) and (14).
14: Record the best food so far
15: Gen = Gen + 1
16: until Gen=MaxGen
17: Return the optimal parameters

IV. Simulation results

To verify the performance of the proposed method, we engage two experiments, including numerical optimization and dynamic system identification, in this section.

A. Numerical optimization problems

We evaluate the performance of the proposed MABC on scalable numerical benchmark functions provided by CEC 2017 special session [16] and compare it with the standard ABC. Tab. I lists the function types of the benchmark func-tions. In this study, the dimensions of the benchmark functions are set to D = 30, and 50. Each function is regarded as a black box, although the values of the optimal solutions are known

TABLE I
FUNCTION TYPES OF CEC'17 BENCHMARK FUNCTIONS.

	Function Type			
	Unimodal Functions	Multimodal Functions	Hybrid Functions	Composition Functions
Function No.	f1, f3	f4, . . . , f10	f11 . . . , f20	f21, . . . , f30

TABLE II
THE RESULTS OF THE PROPOSED ABC AND THE STANDARD ABC FOR D=30.

	Dim=30	
	ABC	MABC
No.	Mean±STD	Mean±STD
f01	1.6898e+02 ± 2.1507e+02	6.1062e+03 ±5.9425e+03
f03	1.1509e+05 ± 1.4640e+04	6.7823e+04 ±1.4275e+04
f04	2.7357e+01 ± 2.8225e+01	7.8688e+01 ±1.0328e+01
f05	8.1647e+01 ± 1.0381e+01	4.1425e+01 ±2.3736e+01
f06	3.6442e-11 ± 3.4981e-11	3.5210e-08 ±1.9603e-07
f07	9.5713e+01 ± 8.7584e+00	1.0045e+02 ±2.1793e+01
f08	8.4685e+01 ± 1.3371e+01	3.2587e+01 ±2.0397e+01
f09	7.5300e+02 ± 3.6322e+02	8.4399e-02 ±1.4423e-01
f10	2.2359e+03 ± 3.0242e+02	3.8857e+03 ±4.6147e+02
f11	3.2034e+02 ± 2.6529e+02	5.0371e+01 ±2.7469e+01
f12	2.6946e+05 ± 1.2493e+05	1.8156e+05 ±1.7753e+05
f13	7.1742e+03 ± 6.4374e+03	1.8520e+04 ±1.9421e+04
f14	1.0383e+05 ± 1.0512e+05	7.6468e+03 ±4.3282e+03
f15	1.6948e+03 ± 2.1068e+03	7.2340e+03 ±8.7338e+03
f16	4.9823e+02 ± 1.5250e+02	2.2141e+02 ±9.5566e+01
f17	1.8307e+02 ± 7.3326e+01	8.2641e+01 ±2.0586e+01
f18	2.0693e+05 ± 1.2415e+05	1.6701e+05 ±9.7699e+04
f19	2.2320e+03 ± 2.5114e+03	2.7095e+03 ±3.3063e+03
f20	1.7235e+02 ± 7.3130e+01	5.3479e+01 ±3.3372e+01
f21	2.4379e+02 ± 7.3699e+01	2.4905e+02 ±2.1369e+01
f22	1.0175e+02 ± 1.6094e+00	1.0045e+02 ±1.4775e+00
f23	4.0148e+02 ± 2.2839e+01	3.7903e+02 ±8.5145e+00
f24	2.3865e+02 ± 1.2334e+02	4.9268e+02 ±1.5396e+01
f25	3.8426e+02 ± 8.3505e+01	3.8777e+02 ± 9.7607e-01
f26	2.1507e+02 ± 2.9292e+01	1.3162e+03 ±8.6547e+01
f27	5.1108e+02 ± 4.6025e+00	5.0784e+02 ±5.3992e+00
f28	4.0019e+02 ± 3.1552e+00	3.7880e+02 ±6.7878e+01
f29	5.8169e+02 ± 7.9435e+01	4.6271e+02 ±5.3510e+01
f30	5.0960e+03 ± 1.8174e+03	5.5047e+03 ±3.2918e+03
wins	13	16

in advance. Two competitive algorithms perform each function 31 runs, and the maximum function evaluations is defined as 10000 × D. The common parameters, including population size and limit, are set to 50 and 5 × D, respectively. The experimental results are shown in Tabs. II, III. The mean and standard deviation of error values, error values are calculated by $f(\sim x) - f(\sim x^*)$ as illustrated in [16], are presented in these tables. From Tabs. II, III, the proposed method achieves a lower mean of error values than the standard ABC in most cases.

B. Indentification of dynamic systems

In this section, four function approximation problems are provided to verify the feasibility and effectiveness of the proposed method. Moreover, different kinds of EAs as competitors are engaged to optimize the NFS, including DE, PSO, GA, and standard ABC. For the convenience of description, the

TABLE III
THE RESULTS OF THE PROPOSED ABC AND THE STANDARD ABC FOR D=50.

	Dim=50	
	ABC	MABC
No.	Mean±STD	Mean±STD
f01	1.6401e+03 ± 1.7755e+03	8.6269e+03 ± 9.1293e+03
f03	2.3614e+05 ± 2.2746e+04	1.8117e+05 ± 2.3514e+04
f04	2.8243e+01 ± 1.0553e+01	7.6648e+01 ± 5.7493e+01
f05	1.8860e+02 ± 2.4693e+01	1.0500e+02 ± 6.9513e+01
f06	2.6372e-11 ± 2.7327e-11	3.8014e-07 ± 1.2125e-06
f07	2.0766e+02 ± 2.0122e+01	2.6576e+02 ± 2.1530e+01
f08	2.0371e+02 ± 2.2103e+01	9.2334e+01 ± 5.6968e+01
f09	4.8024e+03 ± 1.5977e+03	6.7129e-01 ± 1.3322e+00
f10	4.1685e+03 ± 3.2987e+02	8.9147e+03 ± 7.8189e+02
f11	4.0288e+02 ± 1.3821e+02	6.7963e+01 ± 2.5719e+01
f12	2.3557e+06 ± 1.2795e+06	1.6678e+06 ± 1.2976e+06
f13	2.0207e+03 ± 1.8046e+03	3.9174e+03 ± 5.4407e+03
f14	5.0028e+05 ± 3.5159e+05	9.6200e+04 ± 6.5930e+04
f15	3.7270e+03 ± 4.9983e+03	4.3065e+03 ± 7.0365e+03
f16	1.2131e+03 ± 1.9501e+02	9.3437e+02 ± 2.6715e+02
f17	7.9360e+02 ± 1.6930e+02	6.0028e+02 ± 1.4711e+02
f18	9.1804e+05 ± 5.0590e+05	6.6607e+05 ± 3.0892e+05
f19	1.0628e+04 ± 4.5469e+03	1.6558e+04 ± 1.1175e+04
f20	6.0721e+02 ± 1.3371e+02	4.9127e+02 ± 1.8199e+02
f21	3.9717e+02 ± 2.1157e+01	3.4590e+02 ± 7.3201e+01
f22	1.7025e+03 ± 2.3647e+03	5.9079e+03 ± 4.3983e+03
f23	6.4716e+02 ± 4.7754e+01	4.9984e+02 ± 3.9660e+01
f24	9.9566e+02 ± 6.8051e+01	7.0949e+02 ± 2.3124e+01
f25	5.0962e+02 ± 1.8026e+01	5.2136e+02 ± 4.1809e+01
f26	1.0762e+03 ± 1.3560e+01	1.8690e+03 ± 1.7492e+02
f27	6.3392e+02 ± 2.3101e+01	5.9184e+02 ± 4.1107e+01
f28	4.8365e+02 ± 8.8363e+00	4.8929e+02 ± 2.0352e+01
f29	1.0526e+03 ± 1.2667e+02	5.8080e+02 ± 1.3456e+02
f30	6.8908e+05 ± 5.4594e+04	1.0297e+06 ± 2.9224e+05
wins	13	16

combinations of these EA approaches with the NFS are represented as NFS_DE, NFS_PSO, NFS_GA, and NFS_ABC, respectively. For a meaningful comparison, we set the same population size and number of generations for all competitive algorithms with 50 and 2000, respectively. Moreover, the NFS_ABC and the proposed NFS_MABC adopt the same criteria, limit is set to five times of the problem dimension, to abandon exhausted solutions. Tab. IV gives the descriptions of dynamic system processing problems used in this study, where y(t) and u(t) are current state of the dynamic systems as the input, and y(t + 1) is the desired output.

$$u(t) = \sin\left(\frac{2\pi t}{100}\right). \tag{16}$$

In problem 1, the system has one output and a control input. The training of the NFS involves 900-time steps datums generated as described in [2]. The input is i.i.d. uniformly random sequence over [−2, 2] for the first half of the 900-time steps and the remaining time steps are given by a sinusoid function $1.05\sin(\pi t/45)$. The testing signal is generated by (15). We follow the same protocols as described in [2] to obtain the training and test data. Tab. V gives a detailed description of the training and testing reports for each combination of NFS. Fig. 2 depicts the difference of identification results between the NFS_MABC and the actual system output.

$$u(t) = \begin{cases} \sin(\pi t/25), & t < 250, \\ 1, & 250 \le t < 500, \\ 1, & 500 \le t < 750, \\ 0.3\sin(\pi t/25) + 0.1\sin(\pi t/32) + 0.6\sin(\pi t/10), & 750 \le t < 1000. \end{cases} \quad (15)$$

TABLE IV
NONLINEAR DYNAMIC SYSTEMS AND CORRESPONDING PARAMETERS USED TO EVALUATE IN THIS WORK.

Problem No.	Input of NFS	No. of training/testing data	System
1	y(t), u(t)	900/1000	$y(t+1) = 0.72y(t) + 0.025y(t-1)u(t-1) + 0.01u^2(t-2) + 0.2u(t-3)$
2	y(t), u(t)	900/1000	$y(t+1) = f(y(t), y(t-1), y(t-2), u(t), u(t-1))$, where $f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1x_2x_3x_5(x_3-1)+x_4}{1+x_2^2+x_3^2}$
1	y(t), u(t)	200/300	$y(t+1) = \frac{y(t)}{y^2(t)+1} + u^3(t)$
2	x(t − 24), x(t − 18), x(t − 12), x(t − 6)	500/500	$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)$

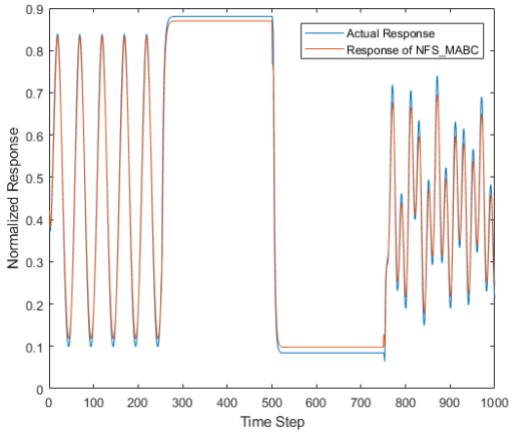

Fig. 2. Simulation result of dynamic system problem using the proposed NFS MABC with three rules in problem 1.

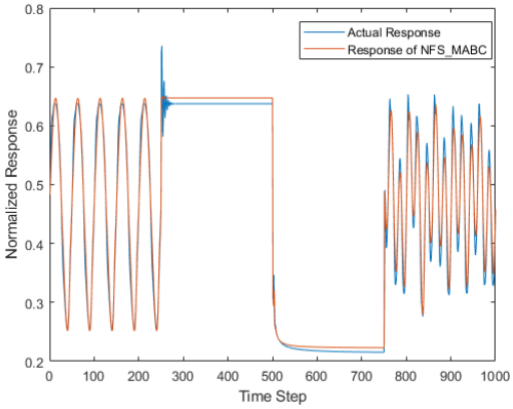

Fig. 3. Simulation result of dynamic system problem using the proposed NFS MABC with three rules in problem 2.

In problem 2, the same protocols are adopted as in Problem 1 to generate the training data and testing data sets. Tab. VI shows the result reports, including the number of rules generated, the number of free parameters and root-mean-square errors (RMSE) of all comparative models for training and testing. Fig. 3 shows the prediction results of the proposed NFS MABC.

In problem 3, we use the control input by (16) to generate the data sets for training and testing purpose and the system output y(t) is bounded within the range [−2, 2], and y(0) = 0. Moreover, the suggestion by [4] is adopted to use 200 samples as the training set. Tab. VII and Fig. 4 show the learning performance of all related NFSs and the simulation results, respectively. The prediction curve by the proposed method has a perfect match with the actual response of the system.

Problem 4, known as the Mackey-Glass chaotic time series, is a time series prediction problem, which is generated by the delay differential equation as listed in Tab. IV. In the equation, τ is set to 17, and the initial value of x is given as x(0) = 1.2. The system predicts x(t) using the tuple of four past states [x(t −24), x(t −18), x(t −12), x(t −6)]. As suggestion in [2], [4], [17], total 1000 patterns are generated from time step 124 to 1123 to form training data sets with the first 500 patterns, and the remainings for testing. The predicted result is drawn in Fig. 5, and the learning performance of competitors is summarized in Tab. VIII. The performance of the proposed NFS_MABC surpasses than other NFSs.

V. CONCLUSIONS

In this paper, a modified ABC with a self-adaptive mecha-nism is introduced to learn the parameters of a TSK-type NFS for solving the dynamic system identification problems. For constructing the NFS model, an adaptive clustering method is employed to generate fuzzy rules automatically. Moreover, the TSK-type reasoning is applied for the consequent part of each fuzzy rule. All free parameters, including the premise and subsequent reasoning parts in the NFS, are adjusted by the proposed MABC for finding better values. To verify the

TABLE V
PERFORMANCE OF COMPETITORS FOR THE PROBLEM 1.

	NFS_DE	NFS_PSO	NFS_GA	NFS_ABC	NFS_MABC
No. of rules	3	3	3	3	3
No. of parameters	21	21	21	21	21
Train RMSE (Mean±STD)	0.0937 ± 0.0009	0.0961 ± 0.0012	0.0970 ± 0.0006	0.0929 ± 0.0005	0.0924 ± 0.0003
Test RMSE (Mean ±STD)	0.0294 ± 0.0026	0.0320 ± 0.0020	0.0323 ± 0.0041	0.0302 ± 0.0038	0.0276 ± 0.0029

TABLE VI
PERFORMANCE OF COMPETITORS FOR THE PROBLEM 2.

	NFS_DE	NFS_PSO	NFS_GA	NFS_ABC	NFS_MABC
No. of rules	3	3	3	3	3
No. of parameters	21	21	21	21	21
Train RMSE (Mean±STD)	0.0660 ± 0.0053	0.0700 ± 0.0050	0.0731 ± 0.0051	0.0649 ± 0.0057	0.0635 ± 0.0065
Test RMSE (Mean ±STD)	0.0297 ± 0.0104	0.0383 ± 0.0024	0.0404 ± 0.0102	0.0222 ± 0.0400	0.0218 ± 0.0069

TABLE VII
PERFORMANCE OF COMPETITORS FOR THE PROBLEM 3.

	NFS_DE	NFS_PSO	NFS_GA	NFS_ABC	NFS_MABC
No. of rules	3	3	3	3	3
No. of parameters	21	21	21	21	21
Train RMSE (Mean±STD)	0.0137 ± 0.0087	0.0189 ± 0.0083	0.00235 ± 0.0091	0.00120 ± 0.0063	0.0094 ± 0.0064
Test RMSE (Mean ±STD)	0.0141 ± 0.0091	0.0194 ± 0.0085	0.0242 ± 0.0100	0.0123 ± 0.0066	0.0097 ± 0.0067

TABLE VIII
PERFORMANCE OF COMPETITORS FOR THE PROBLEM 4.

	NFS_DE	NFS_PSO	NFS_GA	NFS_ABC	NFS_MABC
No. of rules	3	3	3	3	3
No. of parameters	39	39	39	39	39
Train RMSE (Mean±STD)	0.0359 ± 0.0234	0.1047 ± 0.0543	0.1007 ± 0.0360	0.0019 ± 0.0088	0.0161 ± 0.0084
Test RMSE (Mean ±STD)	0.0368 ± 0.0024	0.1064 ± 0.0553	0.1025 ± 0.0361	0.0193 ± 0.0087	0.0161 ± 0.0082

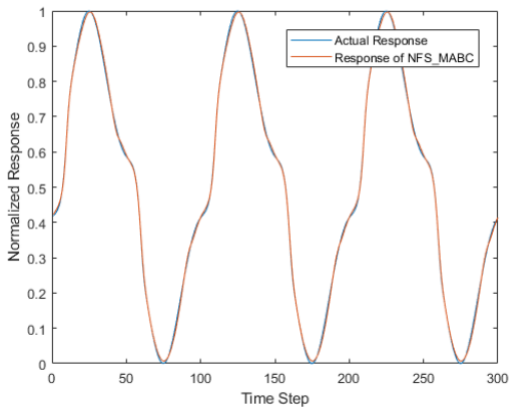

Fig. 4. Simulation result of dynamic system problem using the proposed NFS MABC with three rules in problem 3.

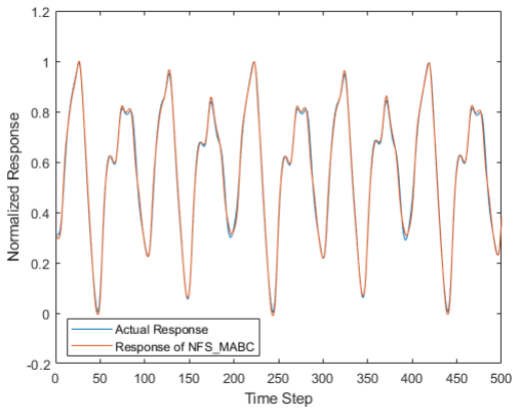

Fig. 5. Simulation result of Mackey-Glass chaotic series using the proposed NFS MABC with three rules in problem 4.

general ability of the proposed MABC, numerical optimization problems are concluded with the standard ABC algorithm on two problem sizes. The experimental results show that the proposed algorithm performs better than the standard ABC on most mathematical optimum problems. To verify the performance bring up the proposed method, several evolutionary algorithms are considered with the same NFS model. The simulation results show that the proposed MABC algorithm made the proposed NFS reaps better approximation results than other evolutional approaches.

REFERENCES

[1] M. Emami, A. A. Goldenberg, and I. Trksen, "Fuzzy-logic control of dynamic systems: from modeling to design," Engineering Applications of Artificial Intelligence, vol. 13, no. 1, pp. 47 – 69, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197699000317

[2] C.-F. Juang, Y.-Y. Lin, and C.-C. Tu, "A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing," Fuzzy Sets and Systems, vol. 161, no. 19, pp. 2552 – 2568, 2010, theme: Neural Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165011410001612

[3] K. Shihabudheen and G. Pillai, "Recent advances in neuro-fuzzy system: A survey," Knowledge-Based Systems, vol. 152, pp. 136 – 162, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705118301825

[4] C.-F. Juang and C.-T. Lin, "An online self-constructing neural fuzzy inference network and its applications," IEEE Transactions on Fuzzy Systems, vol. 6, no. 1, pp. 12–32, Feb 1998.

[5] M.-F. Han, C.-T. Lin, and J.-Y. Chang, "Differential evolution with local information for neuro-fuzzy systems optimisation," Knowledge-Based Systems, vol. 44, pp. 78–89, 05 2013.

[6] I. Boussad, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," Information Sciences, vol. 237, pp. 82 – 117, 2013, prediction, Control and Diagnosis using Advanced Neural Computations. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025513001588

[7] C.-F. Juang, "A tsk-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," IEEE Transactions on Fuzzy Systems, vol. 10, no. 2, pp. 155–170, April 2002.

[8] D. Karaboga and E. Kaya, "Training anfis using artificial bee colony algorithm for nonlinear dynamic systems identification," in 2014 22nd Signal Processing and Communications Applications Conference (SIU), April 2014, pp. 493–496.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical opti-mization, technical report - tr06," Technical Report, Erciyes University, 01 2005.

[10] M. Zhao and P. Wang, "Multi-population artificial bee colony (mpabc) algorithm for numerical optimization," IOP Conference Series: Materials Science and Engineering, vol. 452, no. 3, p. 032003, 2018. [Online]. Available: http://stacks.iop.org/1757-899X/452/i=3/a=032003

[11] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," Applied Mathematics and Com-putation, vol. 217, no. 7, pp. 3166 – 3173, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300310009136

[12] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, "The best-so-far selection in artificial bee colony algorithm," Applied Soft Computing, vol. 11, no. 2, pp. 2888 – 2901, 2011, the Impact of Soft Computing for the Progress of Artificial Intelligence. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494610003029

[13] Y. Xue, J. Jiang, B. Zhao, and T. Ma, "A self-adaptive artificial bee colony algorithm based on global best for global optimization," Soft Computing, vol. 22, no. 9, pp. 2935–2952, May 2018. [Online]. Available: https://doi.org/10.1007/s00500-017-2547-1

[14] F. Kang, J. Li, Z. Ma, and H. Li, "Artificial bee colony algorithm with local search for numerical optimization," JSW, vol. 6, pp. 490–497, 2011.

[15] S. Mahdavi, S. Rahnamayan, and K. Deb, "Opposition based learning: A literature review," Swarm and Evolutionary Computation, vol. 39, pp. 1 – 23, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650216304333

[16] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter," Nanyang Technological University, Singapore, Tech. Rep., Nov. 2016.

[17] Y. Lin, J. Chang, and C. Lin, "Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network," IEEE Transactions on Neural Networks and Learning Systems, vol. 24, no. 2, pp. 310–321, Feb 2013.