International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **11**, November **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 7793–7818

USING MULTI-ANGLES EVOLUTIONARY ALGORITHMS FOR TRAINING TSK-TYPE NEURO-FUZZY NETWORKS

Pei-Chia Hung[1], Sheng-Fuu Lin[1] and Yung-Chi Hsu[2]

[1]Department of Electrical and Control Engineering
National Chiao Tung University
No. 1001, Ta Hsueh Rd., Hsinchu 300, Taiwan
mikehome.ece93g@nctu.edu.tw; sflin@mail.nctu.edu.tw

[2]3C Software Design Center
Quanta Computer Inc.
No. 211, Wen Hwa 2nd Rd., Kueishan, Taoyuan 33377, Taiwan
ericbogi2001@yahoo.com.tw

Received July 2011; revised November 2011

Abstract. *The development of a global-based method for building robust neuro-fuzzy networks has become an interesting issue. Among the various building methods, the evolutionary algorithms provide robust ways increasing the chances of meeting the optimal solution. However, evolutionary algorithms may only use a single angle to evaluate the searching space to obtain the optimal solutions. It implies that they may slowly or even hardly meet the optimal solution. Thus, the current study provides a novel architecture that uses multiple angles for evaluating the searching space. More specifically, the novel architecture adopts multiple angles to improve the evolutionary process by dynamically adjusting the searching space. By doing so, the proposed architecture can increase the chances of meeting the optimal solution. As shown in the results, the proposed architecture outperforms other existing evolutionary algorithms. Based on the results, a framework is proposed to build a benchmark for developing evolutionary algorithms that consider the multiple angles of the solution space.*
Keywords: Neuro-fuzzy network, Evolutionary algorithm, Multiple angles

1. **Introduction.** The development of modern systems has recently geared towards solving the complex relationships between input and output patterns [1]. In other words, modern systems should have the abilities to solve diverse nonlinear problems. However, classical system designs usually require a mathematical model for solving nonlinear problems. Such design pattern may strongly depend on the mathematical modeling of plants. In other words, inaccurate mathematical modeling of plants usually degrades the performances of systems, especially for nonlinear and complex problems [2-5]. Thus, the development of a model that can efficiently solve complex and nonlinear problems has become an issue. Among various models, the neuro-fuzzy network is widely used for addressing nonlinear problems without building complex mathematical models [6]. It is due to the fact that the parameters of neuro-fuzzy network are trained based on a sequence of input and desired outcome pairs [7]. In other words, the neuro-fuzzy network structure does not need a mathematical description of the system when modeling the network. Moreover, it can present the experts' ambiguity and translate knowledge into computable numerical data [8]. Overall, the neuro-fuzzy network provides a useful design pattern for building modern systems that can face the complex and nonlinear relationships between input and output patterns. Therefore, the development of robust neuro-fuzzy networks that can

reach the global solution of various complex and nonlinear applications has become an issue.

Several studies are aimed to develop a learning structure that can build well performing neuro-fuzzy networks to address the issue above. The learning structure is mainly used to develop a suitable way for adjusting the parameters of neuro-fuzzy networks [9]. In other words, the learning structure plays an important role in influencing the performance of neuro-fuzzy networks. Thus, the development of useful learning structures is an important issue when modeling neuro-fuzzy networks. Based on this issue, the research question is "How do you develop a feasible and robust learning structure to construct the neuro-fuzzy networks?" More specifically, the research question is strongly related to the development of global-based learning structures [10] that can help neuro-fuzzy networks meet the optimal solution. Among the various global-based learning structures, the evolutionary algorithms [11], which have the ability of searching for the global solution, are widely used as the learning structures of neuro-fuzzy networks. Moreover, since evolutionary algorithms are parallel and global search techniques and they can simultaneously evaluate many points in the search space, they are more likely to converge towards the optimal solution than the other global-based learning structures.

The evolutionary algorithms seem to be useful solutions to our research question. However, traditional evolutionary algorithms only focus on a single angle when evaluating the searching space for finding the optimal solution [12]. More specifically, such algorithms may adopt the fixed range of the space to search for the solution. In other words, the searching space in such algorithms cannot adjust according to the performance of each candidate solution. For instance, if a candidate solution is near the optimal solution, the candidate solution may still need to search in the large range of space. Therefore, such algorithms may slowly meet or even hardly meet the optimal solution.

The current study proposes a novel evolutionary algorithm to provide different strategies when searching for the optimal solution and to address the issues stated above. More specifically, the current study proposes the multiple angles evolutionary algorithm (MAEA) to evaluate the searching space using multiple angles to adjust the search space. In other words, the searching space will be adjusted automatically based on the performance of each candidate solution. For instance, if a candidate solution is near the optimal solution, it can be further searched in the small range of space. On the other hand, if a candidate solution is far away from the optimal solution, it should be further searched in the large range of space. In doing so, the parameters of neuro-fuzzy networks can be trained efficiently and their outputs can have higher confidence than traditional networks. It implies that the proposed MAEA can provide benefits in searching for the optimal solution. In other words, the MAEA can consider various possible solutions in the early stages but restrict the searching space to find well performing solutions. Thus, the MAEA provides an efficient way to meet the optimal solution. Moreover, the MAEA can be useful for finding the optimal solution of complex problems (e.g., sunspot prediction [13], stock prediction [14], and users' navigation behavior classification [15]), which have large range of searching space [13]. Such searching spaces are diverse and present difficulties in meeting the optimal solution [13]. Thus, it would be better if the searched algorithms can provide multiple considerations to increase the chances of meeting the optimal solution. Thus, the proposed MAEA can be beneficial for automatically adjusting the searching space based on the performance of found candidate solutions. In other words, the MAEA can explore large spaces to find various candidate solutions in the early stages and then fine-tune the found solutions in the later stages.

In summary, the current study aims to propose a novel evolutionary algorithm that can consider multiple angles when evaluating the parameters of neuro-fuzzy networks. In

doing so, neuro-fuzzy networks can increase their chances of meeting the optimal solution. This paper is organized as follows. Section 2 introduces the related works. Section 3 describes the methodology development. Section 4 presents the illustrative results. Section 5 summarizes a framework related to the results. Finally, concluding remarks are drawn in the last section.

2. **Related Works.**

The related works in neuro-fuzzy networks and evolutionary algorithms are shown in the following subsections.

2.1. **Neuro-fuzzy networks.**

Since neuro-fuzzy networks have been successfully applied in several fields, such as control applications [2], stock prediction [3], and image processing [5], they have recently become a popular research field. The main contribution of neuro-fuzzy networks is their application in several classical applications for achieving their desired solutions. More specifically, classical applications usually need an accurate mathematical model to achieve their purposes. However, constructing a suitable mathematical model to reach the purposes of real-world applications (e.g., stock application and biometric application) is difficult. Therefore, neuro-fuzzy networks are suitable solutions for providing robust models that can address the diverse purposes of different applications because they can strongly avoid their association with mathematical models, especially for nonlinear and complex problems [16,17].

A neuro-fuzzy network is composed of a forward and backward structure and it is mainly used for constructing relationships between input and output patterns. The forward structure is mainly used for producing the output of the network. More specifically, it consists of a set of fuzzy if-then rules [18], a fuzzy inference mechanism [19], and a defuzzifier mechanism [20]. These components are shown in Table 1.

Table 1. Components of the forward structure

Component	Content
fuzzy if-then rules	They consist of an antecedent part, which represents different degrees (membership degrees). Each input pattern belongs to a membership function and its consequent part, which indicates the firing strength of the fuzzy rule.
fuzzy inference mechanism	It is mainly used to conduct fuzzy results.
defuzzifier mechanism	It is used to generate crisp outputs.

The backward structure aims to adjust the parameters of a neuro-fuzzy network (i.e., the parameters of the antecedent and consequent parts). In other words, the backward structure can be treated as the learning structure mentioned in Section 1. Several studies have recently proposed novel learning algorithms to adjust the parameters of fuzzy if-then rules automatically [21-24]. Among the proposed algorithms, the back-propagation (BP) algorithm is the most well-known learning algorithm [23,24]. More specifically, BP is used to train the parameters of neuro-fuzzy networks using the steepest descent technique to minimize the error function. Since the BP algorithm can be used to adjust the parameters of neuro-fuzzy networks, it may easily reach the local minima and even never find the global solution [25]. In addition, the performance of the BP training is mainly associated with the initial parameters. Moreover, the development of new mathematical expressions for each network layer of different neuro-fuzzy network topologies is necessary.

To address these disadvantages, the most well-known global learning structures, named evolutionary algorithms, are proposed to prevent achieving the local optimal solution [26].

More specifically, evolutionary algorithms can be used to find the optimal solution when training the parameters of neuro-fuzzy networks. Moreover, evolutionary algorithms can also be used in different neuro-fuzzy network topologies. In other words, evolutionary algorithms do not need to develop new mathematical expressions for training the parameters of the different structures of neuro-fuzzy networks. Therefore, evolutionary algorithms are better candidates than BP algorithms for training the parameters of neuro-fuzzy networks. Such arguments can also explain why our research question tends to improve evolutionary algorithms.

2.2. **Evolutionary algorithm.** Recently, several evolutionary algorithms, such as the genetic algorithm (GA) [26], genetic programming [27], evolutionary programming [28], and evolution strategies [29], have been used to train the parameters of neuro-fuzzy networks. Such algorithms not only provide parallel and global search techniques to seek the solutions but also simultaneously evaluate many points in the search space. In other words, they have more chances of converging towards the global solution than the BP algorithm. Thus, several studies have recently tried to apply evolutionary algorithms for seeking the optimal parameters of neuro-fuzzy networks (i.e., evolutionary fuzzy models) [30-35]. Among various evolutionary fuzzy models, the most well-known model is genetic fuzzy models [30-32] that used the genetic algorithms (GAs) [36] to train the parameters of fuzzy models. For instance, Karr applied GAs to designing a fuzzy controller [30]. As shown in his work, the GAs were mainly used to seek the optimal parameters of each membership function. Moreover, Lin and Xu recently applied reinforcement GAs to seeking the parameters of the TSK-type neural fuzzy controller [31]. Their work demonstrated that the trained TSK-type neural fuzzy controller could obtain better performance than BP algorithms in several control applications.

Researchers have applied GAs to stabilizing a collection of controllable linear multivariable systems. More specifically, eigenvalues are used as a fitness function to consider both the frequency and time domains. It has been shown that the GAs can outperform other existing methods, implying that GAs can be applied in various complex applications. Also, GAs has been used to construct an efficient dynamic channel allocation (DCA) scheme that mainly plays the role of dynamic inter-cell interference coordination (ICIC) in wireless communication systems. The proposed GA-based scheme can obtain better performance than other existing schemes, and can deal with many real world problems.

Even though the aforementioned genetic fuzzy models are useful for seeking the optimal solution, they still possess some challenges that need to be addressed. The major problem is the process on how to design the criteria for evaluating or generating the candidates that can be used to obtain the optimal solution. In other words, the development of a novel architecture to improve the generations of evolutionary algorithms is necessary. Thus, several studies tended to develop novel architectures to address such issue. For instance, Smith et al. proposed a symbiotic evolution to provide multiple considerations when evaluating the candidate solution [37]. More specifically, the symbiotic evolution divides the whole solution into several partial solutions, and each partial solution can be characterized as a specialization. The specialization property can ensure the diversity of evolution. In other words, such diversity can prevent a population from meeting suboptimal solutions. As shown in their work, the symbiotic evolution can obtain good performance by evaluating the partial solutions. Similar to the research, Gomez and Schmidhuber proposed the enforced sub-populations (ESP) that adopts multiple considerations to prevent a population from meeting suboptimal solutions [38]. As shown in their work, the sub-populations were used to evaluate partial solutions. Their results indicated that the ESP could obtain better performance than systems with only a single

consideration. More recently, Lin and Xu have applied GAs to seeking the parameters of the particular NFN (i.e., TSK-type NFN) [39]. Their work indicated that the well-trained TSK-type NFN have high performance in several prediction applications.

More recently, a hybrid genetic algorithm is proposed to construct the decision-making system, which can be used to make decisions for solving a series of problems. The hybrid GA not only evaluates the assembly sequences but also searches for the optimal solution for each sequence. As shown, the optimal decision could be obtained using the proposed hybrid GA, implying that the modification of the genetic algorithm can increase the chances of meeting the optimal solution. The development of multiple objects when evaluating solutions to modify the learning structure further is an issue. The multiple-objectives genetic algorithm (MOGA) has been employed to examining real-life daily pairing problems in a Taiwanese short-haul airline. The results indicated that the MOGA could easily address such multiple-objectives issues by offering robust criteria, implying that the different considerations are helpful for improving the performances of diverse problems. Moreover, Bowman et al. adopted the MOGA to design the multiple consideration base fitness functions [40]. As shown in their work, the MOGA can perform better than traditional GAs for solving the class responsibility assignment problem in object-oriented analysis.

In addition, an evolutionary algorithm has been developed to synthesize finite-state machines. More specifically, they used the proposed evolutionary algorithm not only to address the state assignment NP-complete problem but also to generate an optimal solution and implement the state machine. The results demonstrated that the evolutionary algorithm is helpful for implementing state machines using minimal requirements. Moreover, Lin and Hsu proposed the hybrid evolutionary learning algorithm (HELA) to train the parameters of wavelet neuro-fuzzy networks [41]. As shown in their work, the HELA consisted of the structure and parameter learning. The former was used to construct the number of fuzzy rules, whereas the latter was used to adjust the parameters of wavelet neuro-fuzzy networks. Their results indicated that the HELA outperforms other existing evolutionary algorithms.

More recently, evolutionary algorithms are also used to tune the parameters of neural fuzzy networks. More specifically, they proposed the modified differential evolutionary (MDE) algorithm to model the recurrent functional neural fuzzy network (RFNFN). It is shown that the new evolutionary algorithm not only speeds up the learning curve but also improves the prediction accuracy, implying that the evolutionary algorithm is helpful for efficiently adjusting the parameters of neural fuzzy networks. Furthermore, an observer-based iterative learning control with/without evolutionary programming algorithm has been designed to face the issue of efficiently converging tracking errors in nonlinear systems. The proposed evolutionary programming not only efficiently searches for the optimal solution but also reduces evolutionary time, implying that the evolutionary algorithm can improve the converged performance of nonlinear systems.

Even though the aforementioned studies demonstrated the performance of their proposed architectures, such algorithms only tended to perform the evolutionary processes (e.g., reproduction, crossover, and mutation) using a single angle. More specifically, such algorithms cannot consider different angles to let the individuals take suitable actions in searching for the solution. In other words, the searching space in such algorithms is fixed when performing the evolutionary processes. Thus, such algorithms may not offer a suitable searching space to let the well performing individuals generate better offspring in each generation. Thus, the current study uses multiple angles to improve the performance of evolutionary processes in evolutionary algorithms. In other words, the proposed

evolutionary algorithm can adjust the searching space automatically based on the performance of the individuals. In doing so, the well performing individuals can generate better offspring in a suitable searching space in each generation. In summary, the current study proposes a multi-angles evolutionary algorithm (MAEA) to adjust the parameters of neuro-fuzzy networks. The MAEA can accurately identify the relationships between input and output patterns by considering multiple angles to improve the performance of evolutionary processes.

3. Methodology.

This section introduces the methodology development of the current study. More specifically, the neuro-fuzzy network and the MAEA are described in the following subsections.

3.1. Neuro-fuzzy network.

Neuro-fuzzy networks are mainly used for representing the fuzzy if-then rules of network structures. In doing so, the well-known learning algorithms of artificial neural networks can be applied to train the fuzzy if-then rules. The main processes of a neuro-fuzzy network consist of the fuzzy rules, reasoning process, and fuzzy knowledge based. The fuzzy rules, which are defined by the antecedents and consequents, are used for modeling the relationships between control inputs and outputs. The reasoning process is mainly used for defining the means of the employed aggregation operators (i.e., fuzzy connectives and fuzzy inference method). The fuzzy knowledge based contains the definition of fuzzy sets stored in the fuzzy database. Moreover, it also contains a collection of fuzzy rules that constitute the fuzzy rule base. In general, the Mamdani-type [36] and Takagi-Sugeno-Kang (TSK) type neuro-fuzzy networks [2,28,42] are the most well-known neuro-fuzzy networks. The minimum fuzzy implication is used for performing the fuzzy reasoning of the Mamdani-type neuro-fuzzy network. Moreover, the Mamdani-type neuro-fuzzy network employs the fuzzy inference method, which uses fuzzy sets to define consequent parts. A Mamdani-type fuzzy rule is shown below.

$$\begin{aligned} &\text{IF } x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2j} \ldots \text{ and } x_n \text{ is } A_{nj} \\ &\text{THEN } y \text{ is } C_j \end{aligned} \tag{1}$$

The TSK-type neuro-fuzzy network provides more implication and aggregation methods than the Mamdani-type neuro-fuzzy network. More specifically, the first two parts of the fuzzy inference process, including fuzzifiering the inputs and applying the fuzzy operator is similar with those of the Mamdani-type neuro-fuzzy network. Moreover, the consequence of each rule is a function related to the input variables of such network. The general adopted function is the linear combination of input variables plus a constant term. A TSK-type fuzzy rule is as follows:

$$\begin{aligned} &\text{IF } x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2j} \ldots \text{ and } x_n \text{ is } A_{nj} \\ &\text{THEN } y = w_{0j} + w_{1j}x_1 + \cdots + w_n x_n \end{aligned} \tag{2}$$

The parameter w_{0j} in Equation (2) represents the constant term that sums the linear combination of input variables to generate the consequence of the jth rule node. Moreover, the parameter w_{ij} represents the ith parameter that multiplies the ith input variable to generate the linear combination of input variables. Since the consequence of a rule is crisp, the defuzzification step becomes obsolete in the TSK inference scheme. Thus, the model output is computed instead as the weighted average of the crisp rule outputs, which is computationally less expensive than calculating the center of gravity.

Recently, many studies [2,28,42] have indicated that TSK-type neuro-fuzzy networks could achieve superior performance than Mamdani-type neuro-fuzzy networks in both network size and learning accuracy. Thus, the current study adopts a TSK-type neuro-fuzzy network (TNFN) to reach its objectives. In other words, the multi-angles evolutionary

algorithm, which will be described in the following section, is used to train the parameters of the TNFN. Figure 1 shows the structure of a TNFN, which is a five-layer network structure. The function of each layer is shown below.

a. Input Layer. This layer is mainly used to collect the input values and deliver these values to the next layer. In other words, each node in this layer only transmits the input value to the next layer. The function of each node in this layer is given by

$$u_i^{(1)} = x_i, \tag{3}$$

where $u_i^{(k)}$ denotes the input value of the ith node in the kth layer and x_i denotes the ith input dimension.

b. Membership Function Layer. This layer computes for the membership degree that corresponds to each input node in the Input Layer. The membership degree of each input node that belongs to a fuzzy set [15] is calculated in this layer. Each node in this layer corresponds to a linguistic label of a particular input node in the Input Layer. The current study adopted the most well-known Gaussian membership function in this layer

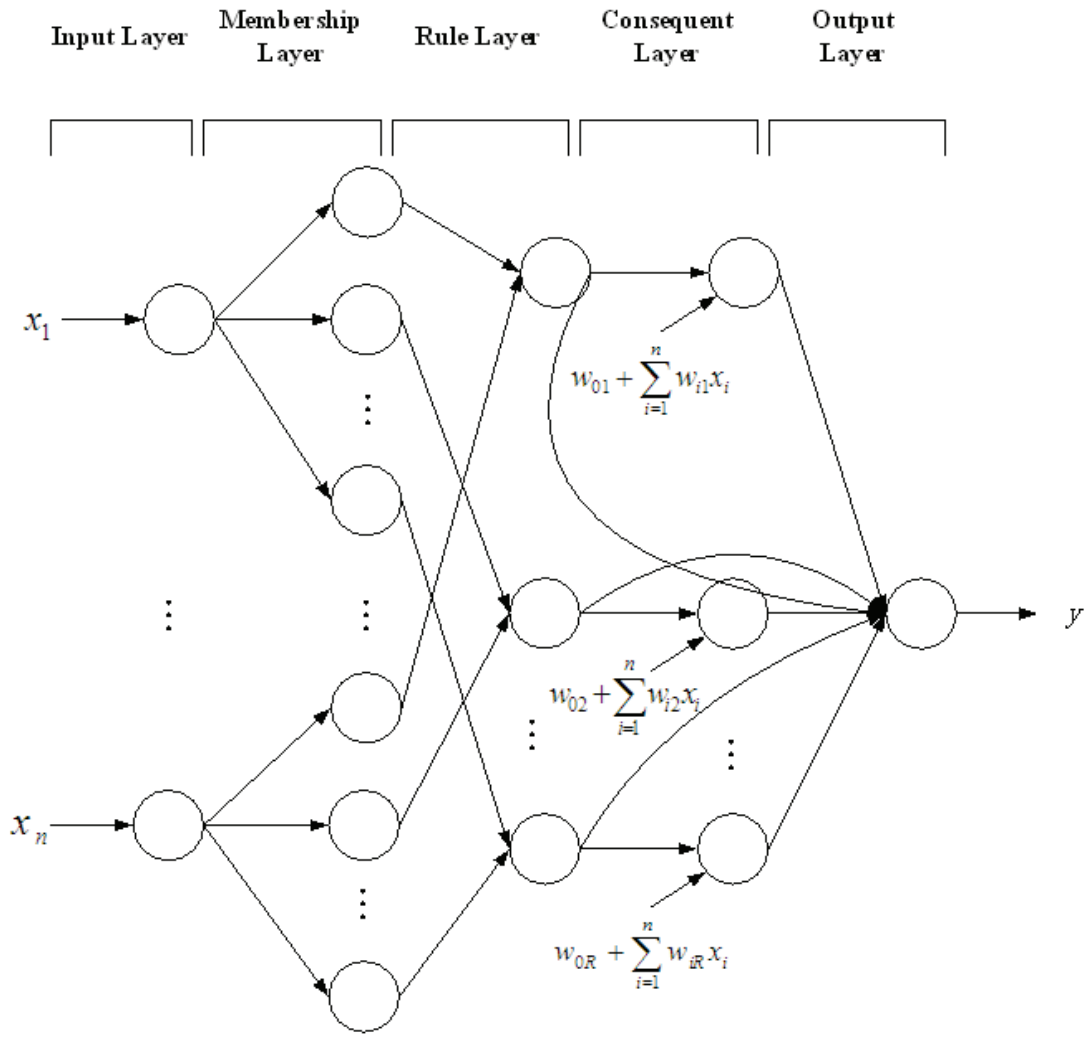


FIGURE 1. Structure of the TSK-type neuro-fuzzy network

[43]. Therefore, the function of each node in this layer is given by

$$u_{ij}^{(2)} = \exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2}\right) \tag{4}$$

where m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of the ith input node $u_i^{(k)}$ in the jth rule, respectively.

c. Rule Layer. This layer determines the firing strength of each rule. In other words, the fuzzy inference operation determines each node in this layer. The "AND" fuzzy inference operation was adopted in the current study [16], i.e., the multiplication operation was used to compute for the firing strength of each rule. The function of each rule is shown below.

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{5}$$

d. Consequent Layer. The nodes in this layer are mainly used to compute for the consequent part of each rule. In other words, each node in this layer computes based on the linear combination of input variables. More specifically, the input nodes in this layer consist of the outputs delivered from the previous layer and the input variables from the Input Layer (see Figure 1). The function of each node is given by

$$u_j^{(4)} = u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i\right) \tag{6}$$

where w_{ij} are the corresponding parameters of the consequent part.

e. Output Layer. Each node in this layer is mainly used to compute for the crisp output value. In other words, the defuzzifier operation is performed in this layer. More specifically, the outputs of the Rule Layer and Consequent Layer are used to compute for the crisp output value. The function of this layer is given by

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i\right)}{\sum_{j=1}^{R} u_j^{(3)}} \tag{7}$$

where R is the number of fuzzy rules.

3.2. **Multi-angles evolutionary algorithm (MAEA).** This section introduces the MAEA, including the learning components (see Section 3.2.1) and learning procedure (see Section 3.2.2).

3.2.1. *Learning components.* This subsection describes the main components of the proposed MAEA, including coding, fitness evaluation, reproduction, multi-angles crossover (MAC), and multi-angles mutation (MAM).

1. **Coding.** Encoding the adjustable parameters into a chromosome is the first issue of evolutionary algorithms. In other words, the parameters of the TNFN are encoded into a chromosome during coding. More specifically, the adjustable parameters of fuzzy rules are translated into a chromosome to obtain the optimal solution. According to Equation (2), the adjustable parameters of a TSK-type fuzzy rule consist of an antecedent part and a consequence part. Therefore, the parameters of the antecedent and consequence parts are encoded into a chromosome. Figure 2 shows the coding schema of a chromosome, where i represents the ith input node in the

jth rule. The coding type of the chromosomes in the current study is a float point type.

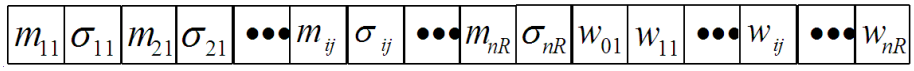


FIGURE 2. The coding schema of a chromosome

2. **Fitness Evaluation.** The fitness evaluation is mainly used for computing for the performance of each chromosome in the population. In other words, it is the main process of the evolution because the fitness value plays an important role in deciding if the optimal solution is found. A well-designed fitness value can help efficiently evaluate the population. In the current study, the most well-known root mean square (RMS) errors [44] are used to evaluate the performance of each individual because they can robustly reflect the performance of the models. In other words, the fitness function of the MAEA is designed according to the RMS errors. The fitness function designed in the current study is given by

$$FitnessValue = \frac{1}{\left(\sqrt{\frac{\sum_{i=1}^{n} |x_i - x_i'|}{n}} + 1\right)}, \tag{8}$$

where x_i and x_i', represent the ith output and the ith desired output of the TNFN, respectively. As shown in Equation (8), the high *FitnessValue* indicates that the outputs of the TNFN are close to the desired outputs.

3. **Reproduction.** A reproduction keeps the chromosomes that perform well in each generation [45]. More specifically, the chromosomes with high fitness value will have more chances of surviving in the next generation than the chromosomes with low fitness value. In general, the top half of the chromosomes in the population is used to reproduce the chromosomes in the top half of the population in the next generation, whereas the other chromosomes are generated using the crossover and mutation. The concept of the reproduction is to sum up the fitness values of the top half of chromosomes in the population and then compute for the fitness ratio of each chromosome based on the summation of the fitness values. Then, the chromosomes are reproduced according to the fitness ratio. In the current study, the roulette-wheel selection is used to select the reproduced chromosomes because it is more robust than other selection methods [45]. Then, the chromosomes with high fitness ratio will reproduce more than the chromosomes with low fitness ratio, implying that chromosomes with good performance have more chances of performing crossover and mutation to generate the good offspring in the next generation.
4. **Multi-Angles Crossover.** After performing the reproduction, the MAEA will perform the crossover to generate new chromosomes by exchanging the values between the different genes of chromosomes. Exchanging the genes between two parents to generate the offspring is the major task of the crossover [46]. In doing so, the new combination of chromosomes will be generated in the next generation. In general, the chromosomes in the top half of the population are used to perform the crossover and to generate the chromosomes in the other half of the population, implying that the crossover can seek the different combinations of the chromosomes that perform well to find the optimal solution.

   Even though the crossover offers the aforementioned benefits, only a single angle, or a single action, can be used to perform the crossover. More specifically, the

crossover in the previous evolutionary algorithms only exchanges the genes that were randomly selected [29-36,40,41]. In other words, there is no idea to consider the fitness value to provide the different actions that are used to generate the range of genes for generating the different combinations of chromosomes. More specifically, using a random process to generate the searching range of genes may be difficult to use for finding the optimal solution especially when the solution is near the optimal solution. For instance, if chromosomes are near the optimal solution, they need to search for the optimal solution in a small searching range. On the other hand, if the chromosomes are far away from the optimal solution, they need to search for the optimal solution in a large searching range. Thus, multiple strategies based on the fitness value are necessary when performing the crossover. In doing so, the evolutionary process can increase chances of finding the optimal solution.

Therefore, the proposed MAEA uses the multi-angles crossover (MAC) to consider the fitness value when performing the crossover. The consideration of the fitness value to provide different actions when performing the crossover is the main concept of the MAC. More specifically, the MAC uses the fitness value of parent chromosomes to determine the number of genes used to generate offspring in the next generation. In doing so, the parents with low fitness value may seek the optimal solution by exchanging many genes. On the other hand, the parents with high fitness value may seek the optimal solution by exchanging a few genes. The details of the MAC are as follows:

$$CrossoverSite_1 = Rand(chrLength) \tag{9}$$

$$MaxCroPoints = chrLength * (MaxFitnessvalue - Fitnessvalue) \tag{10}$$

$$\begin{aligned} &CrossoverSite_2 = CrossoverSite_1 + Rand(\pm MaxCroPoints) \\ &CrossoverSite_2 \in [1, chrLength] \end{aligned} \tag{11}$$

where

$$\begin{aligned} &CrossoverSite_1 \text{ is the first crossover site used to perform the MAC.} \\ &CrossoverSite_2 \text{ is the second crossover site used to perform the MAC.} \\ &MaxFitnessvalue \text{ is the maximal value of fitness value} \\ &\text{(In this study, the } MaxFitnessvalue \text{ is 1.00).} \\ &chrLength \text{ is the length of a chromosome.} \\ &MaxCroPoints \text{ is the range used to decide the } CrossoverSite_2. \end{aligned} \tag{12}$$

As shown in Equation (10), *MaxCroPoints* and *FitnessValue* have an inverse relationship, implying that the MAC can exchange many genes of the two parent chromosomes with low performance to generate the offspring. On the other hand, the MAC can exchange a few genes of the two parent chromosomes with high performance (see Equations (9)-(11)). The current study adopts the two-point crossover to exchange the genes between $CrossoverSite_1$ and $CrossoverSite_2$ from the two parents [47]. Figure 3 shows the two-point crossover. The MAEA can robustly produce the different combinations of the chromosomes according to their *FitnessValue* to seek the optimal solution when performing the MAC.

5. **Multi-Angles Mutation.** Even though the crossover can generate different combinations of the chromosomes by exchanging the existing genes, it cannot generate a new value. In this case, the evolution may only depend on the initial value of the chromosomes, and thus it may never find the optimal solution. Thus, attending new values in existing chromosomes to extend the searching space for seeking the

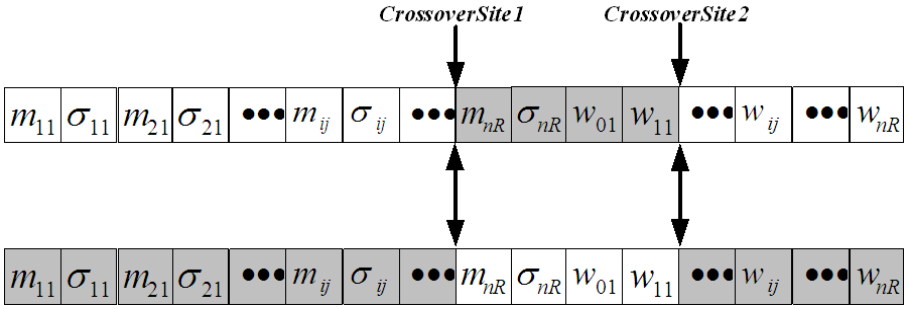


FIGURE 3. The two-point crossover

optimal solution is necessary. The mutation is a way to address such issue. The mutation is mainly used for generating a new gene that replaces the existing gene in chromosomes [48]. In doing so, the chromosomes can extend their searching space to seek the optimal solution.

Aside from the MAC, the MAEA also uses the MAM to consider the fitness value when performing the mutation and to provide an efficient way of performing the mutation. Providing multiple strategies to adjust the searching space is the main concept of the MAM. More specifically, the MAM can consider the fitness value not only to determine the number of mutation genes but also to define the range values used to update the mutation genes. In doing so, the searching space may self-adapt according to the fitness value. For example, the chromosomes with low fitness value may seek the optimal solution by considering many mutation points and updating each mutation point using a large value range. On the other hand, the chromosomes with high fitness value may seek the optimal solution by considering few mutation points and updating each mutation point using a small value range. The details of the MAM are as follows:

$$CurMaxMutPoints = MaxMutPoints \times (FitnessRange) \tag{13}$$
$$FitnessRange = MaxFitnessValue - FitnessValue \tag{14}$$
$$MutPoints = Rand(CurMaxMutPoints) \tag{15}$$
$$MutSite_p = Rand(chrLength), p = 1, 2, 3, \cdots MutPoints \tag{16}$$
$$Chr_{MutSite_p} = Chr_{MutSite_p} + MutValue \tag{17}$$
$$MutSite_p = Rand(chrLength), p = 1, 2, 3, \cdots MutPoints \tag{18}$$

where

$MaxMutPoints$ represents the predefined maximal mutation points.
$CurMaxMutPoints$ is the maximal mutation points in current generation.
$MutPoints$ is the number of mutation sites used to perform the MAM.
$MutSite_p$ is the pth mutation site of the mutated chromosome.
$Chr_{MutSite_p}$ is the $MutSite_p$th gene of the mutated chromosome.
$RangeValue$ is the predefined value related to the input space. (19)

The aforementioned equations indicate that the *MutPoints* and *FitnessValue* have an inverse relationship (see Equations (13)-(15)). Moreover, the *MutValue* and *FitnessValue* also have an inverse relationship (see Equations (16)-(18)), implying that the MAM can roughly adjust many genes of the chromosome with low performance and finely adjust few genes of the chromosomes with high performance. In doing so,

the MAEA can robustly adjust the parameters of the TNFN when performing the mutation.

In summary, the proposed MAEA can contribute in the consideration of the different angles for evaluating the chromosomes. More specifically, the crossover and mutation processes are modified in the proposed MAEA. These processes were redesigned to meet the proposed fitness value. The number of crossover points may be changed according to the fitness value of an individual. The mutation point and mutated value may be changed according to the fitness value of an individual. Details on the unique benefits of each method are shown in Table 2.

Table 2. The unique benefits of the proposed MAEA

Features	**Benefits**
MAC	1. Considers the multiple angles to perform crossover. 2. The individual with low fitness value may exchange many points to find the optimal solution. 3. The individual with high fitness value may exchange a few points to find the optimal solution.
MAE	1. Considers the multiple angles to perform mutation. 2. The individual with low fitness value may choose many points to mutate to find the optimal solution. 3. The individual with high fitness value may choose a few points to mutate to find the optimal solution. 4. The individual with low fitness value may mutate a point in a large range to find the optimal solution. 5. The individual with high fitness value may mutate a point in a small range to find the optimal solution.

3.2.2. *Learning procedure.* This section introduces the learning procedure of the MAEA (see Equation (4)). As can be seen in Figure 4, the procedure of the MAEA consists of eight steps as follows:

1. The initial population, which consists of several chromosomes, is generated based on the coding schema (see Figure 2). Each gene of a chromosome is generated randomly according to the predefined *RangeValue*.
2. The Fitness Evaluation is performed to evaluate the performance of each chromosome in the population. In other words, Equation (8) is used to compute for the fitness value of each chromosome.
3. The MAEA then judges if the evolutionary process is finished based on a predefined criterion. In general, the criterion is defined according to a predefined generation time or predefined desired fitness value [49,50]. In the current study, the predefined generation time is used to judge if the evolutionary process is finished.
4. If the evolutionary process is not yet finished, the MAEA performs the reproduction to keep the well performing chromosomes on the top half of the population.
5. After performing the reproduction, the MAEA then performs the crossover step. More specifically, a rate will be generated randomly to judge if the crossover is performed based on the predefined *CrossoverRate*. If the crossover is necessary, the evolutionary process will continue to step 6; otherwise, it will skip to step 7.

6. If the random rate is greater than *CrossoverRate*, the MAC is performed. The parents used for performing the MAC are chosen randomly from the top half of the population.
7. The MAEA then performs the mutation step. Similar to the crossover step, a rate will be generated randomly to judge if the mutation is performed based on the predefined *MutationRate*. If the mutation is necessary, the evolutionary process will go to step 8; otherwise, the evolutionary process will go back to step 2 until the predefined criterion is reached.
8. The MAM is performed in this step. More specifically, the chromosome is selected from the newly generated chromosomes randomly. Then, the MAM is performed based on the selected chromosomes. The evolutionary process then goes back to step 2 to evaluate the mutated chromosomes.

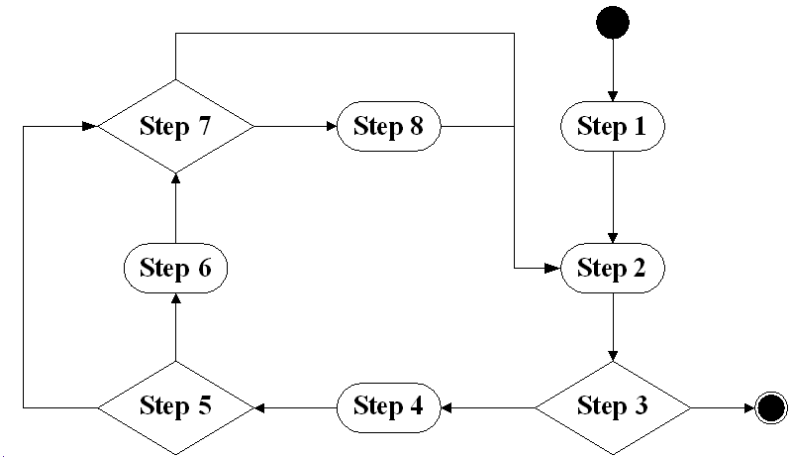


FIGURE 4. Procedures of the MAEA

4. **Illustrative Examples.** This section discusses the two simulations used for investigating the performance of the proposed MAEA. In the first example, a simple chaotic signal [51] is applied not only to investigate the benefits of the MAEA but also to demonstrate the performance of each component. In the second example, a complex chaotic time series [52] was applied to evaluate the robustness and efficiency of the proposed MAEA.

4.1. **Prediction of a simple chaotic signal.** In this example, a time series prediction problem is used to evaluate the performance of the proposed MAEA. More specifically, a simple chaotic signal related to a one-step-ahead prediction [51] is used to investigate the performance of the proposed MAEA because such chaotic signal is easy to implement and it can easily obtain a good prediction performance. Thus, the simple chaotic signal can be used as a benchmark for evaluating not only the benefits of the MAEA but also the performance of each component. The MAEA is used to train the parameters of the TNFN, which is used to predict the chaotic signal. Thus, the chaotic signal should be identified first. The function related to the chaotic signal can be described as follows [51]:

$$x(k+1) = ax(k)(1 - x(k)) \tag{20}$$

According to Equation (20), the time series is generated based on the value of the parameter T. More specifically, the time series plays an important role in influencing the generated time series. For instance, the system has a single fixed point at the origin when $T < 1$. In this case, the time series becomes constant. On the other hand, the system generates a periodic attractor when $T > 3$. Moreover, the system becomes chaotic when

$T > 3.6$. In general, this example chooses T of 3.8 for generating the chaotic signal [36]. Moreover, the first 60 pairs (i.e., $x(1)$-$x(60)$), which are generated using the initial value of 0.001 (i.e., $x(1) = 0.001$), are used as the training patterns for modeling the TNFN. The other 100 pairs (i.e., $x(1)$-$x(100)$), which are generated using the initial value of 0.9, are used as the testing data set to validate the well-trained TNFN.

In addition, deciding the predefined parameters of the MAEA has always been an issue. Thus, the current study adopts a parameter exploration, which was first proposed by De Jong [53], to decide the suitable predefined parameters. The parameter exploration is useful when the dataset is not large. Thus, it is suitable for this example because the dataset generated from Equation (20) is not large. For this reason, the parameter exploration was applied to decide the predefined parameters of the proposed MAEA. More specifically, the parameter exploration uses the different ranges of the value to evaluate the performance of the MAEA and to judge if the value is suitable. For instance, the number of fuzzy rules ranges from 3 to 15 in increments of 1, the number of chromosomes in a population (population size) ranges from 10 to 120 in increments of 10, the crossover rate ranges from 0.20 to 1.00 in increments of 0.05, and the mutation rate ranges from 0.0 to 0.3 in exponential increments. The other parameters of the MAEA are defined similar to the procedures above. The parameters are defined according to the parameters that can enable the MAEA to obtain the best fitness value in small generation times. Table 3 shows the definition of the parameters of the proposed MAEA obtained after performing parameter exploration. Even though the parameter exploration can provide a systematic way for investigating the different ranges of each parameter, it may have difficulties in deciding the benchmark of the range in each parameter. In other words, performing parameter exploration is necessary for deciding the range of the parameters of different applications. Thus, providing a self-adapting architecture can be considered in future works to decide the suitable range of each parameter in the proposed MAEA.

Table 3. The predefined parameters of the MAEA

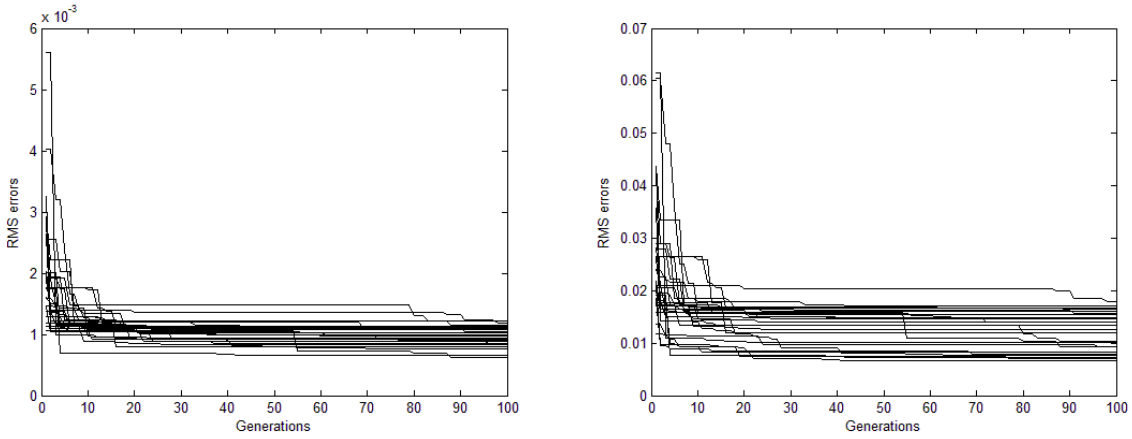
Parameters	**Value**
The number of rules	4
The number of chromosomes in a population	30
The number of generation times	100
MaxMutationNum	5
RangeValue for weight	$[1.500, -1.500]$
RangeValue for membership functions	$[3.000, -3.000]$
Mutation rate	0.3
Crossover rate	0.5

After deciding the parameters, the MAEA was then used to perform the evolutionary process. The simulation was conducted for 30 runs and each run started with the same initial parameters. The simulation was used to evaluate the performance of the proposed algorithm in 30 runs. In doing so, reliable evidence on the performance of the proposed MAEA was obtained.

Figure 5(a) shows the learning curves of the 30 runs performed using the MAEA. The fitness value was translated to represent the RMS error [44] using the following equation to easily investigate the performance of each learning curve:

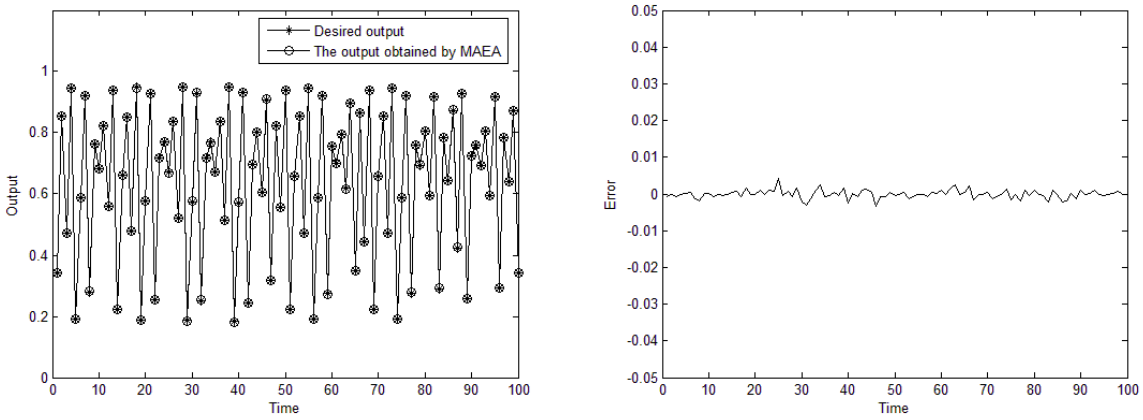
$$RMSerror = \sqrt{\frac{\sum_{i=1}^{n} |x_i - x_i'|}{n}} \tag{21}$$

As can be seen in Figure 5(a), each learning curve can reach the low RMS error, which was close to 0.002, implying that the proposed MAEA can obtain good performance in 30 runs. A tradition genetic algorithm (GA) [27] was compared with the MAEA to further demonstrate the performance of the proposed MAEA. Similar to the MAEA, the parameters of the traditional GA were defined using parameter exploration. Moreover, the traditional GA was used to perform a simulation for 30 runs. Figure 5(b) shows the 30 learning curves of the traditional GA. Comparing the learning curves in Figure 5(a) to those in Figure 5(b), the proposed MAEA was shown able to outperform the traditional GA dramatically because the proposed MAEA can consider multiple angles when performing the crossover (i.e., MAC) and mutation (i.e., MAM) processes, which can adjust the search space and range according to the fitness value of chromosomes. Thus, the MAEA can generate efficient candidate solutions. In addition, the learning curves in Figure 5(a) decreased faster compared to those of in Figure 5(b), demonstrating that the MAEA can increase its chances of searching the better solution in each generation using the proposed MAC and MAM.



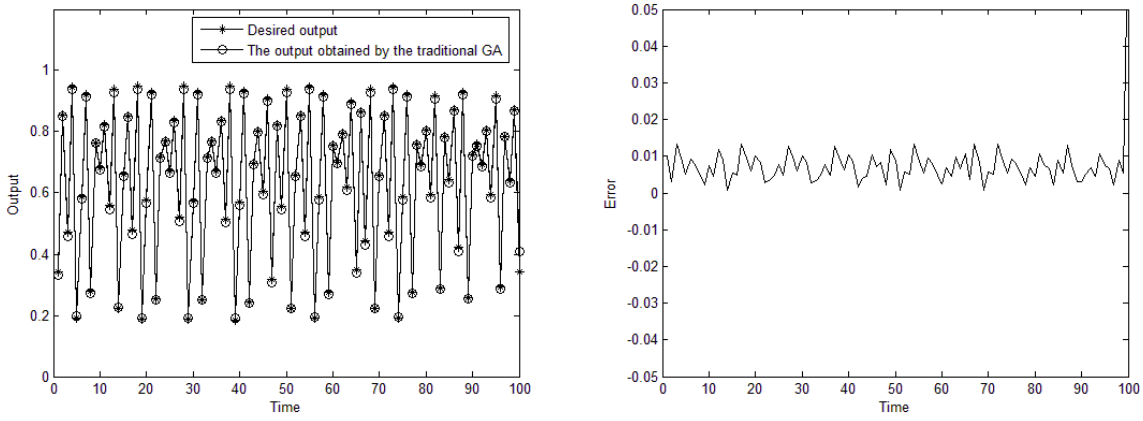
(a) (b)

FIGURE 5. The learning curves of (a) the MAEA and (b) the traditional GA



(a) (b)

FIGURE 6. The testing prediction results (a) and errors (b) of the MAEA



(a) (b)

FIGURE 7. The testing prediction results (a) and errors (b) of the traditional GA

The testing results were investigated to further validate the argument stated above. Figure 6 shows the testing results (see Figure 6(a)) and testing errors (see Figure 6(b)) between the desired outputs of the MAEA and the outputs obtained using the well-trained TNFN. Figure 7 shows the testing results and errors of the traditional GA. As can be seen in Figures 6 and 7, the MAEA outperforms the traditional GA, implying that the proposed MAEA can adjust the parameters of the TNFN more efficiently than the traditional GA.

In addition, the other existing well-known genetic algorithms mentioned in Section 2 [27,34-36,41] were compared with the MAEA to provide reliable evidence on the performance of the MAEA. Similar to the MAEA, the parameters of the other algorithms were defined using parameter exploration. Moreover, each algorithm was used to perform a simulation for 30 runs. The simulation environment is shown in Table 4.

TABLE 4. The simulation environment

Equipment	**Dec.**
CPU	Intel Core i5 2.6GHz
RAM	4GB
OS	Windows 7
Software	Java (SE 6)
Mode	Command mode

The aforementioned computing issues indicated that the proposed MAEA and the other methods run in the standard framework for the simulation. In other words, the computing results were run in a fair computing environment, implying that the proposed MAEA can easily be implemented since it can be run in a common environment.

The performances of the compared genetic algorithms in the training step, including the mean and standard deviation values of the RMS errors and CPU time are shown in Table 5. As can be seen in the table, the MAEA outperformed the other algorithms. Since the performance of the evolutionary algorithms proposed in [36,41] are close to the MAEA, they are too complex to implement. Moreover, such evolutionary process is time consuming. The results shown in Table 5 indicate that the MAEA not only spends smaller CPU time but also obtains lower RMS errors than other algorithms.

TABLE 5. The performance comparison of various existing models

Method	RMS errors (Training)		RMS errors (Testing)		CPU Time (Training)	
	Mean	Deviation	Mean	Deviation	Mean (second)	Deviation (second)
MAEA	**0.00098**	**0.00015**	**0.00103**	**0.00018**	**1.32**	**0.15**
[27]	0.01042	0.003	0.01107	0.007	2.13	0.24
[34]	0.00663	0.0022	0.00684	0.0024	4.18	0.63
[35]	0.00418	0.0018	0.00446	0.0021	4.03	0.49
[36]	0.00174	0.00034	0.00191	0.00047	12.73	2.16
[41]	0.00151	0.00058	0.00163	0.00063	10.17	3.81

In short, the MAEA demonstrated higher performance than the other existing algorithms via the MAC and MAM. More specifically, the MAC and MAM consider multiple angles to perform the crossover and mutation to let the evolutionary process seek in not only a large searching space or range when the solution is far away from the optimal solution but also in a small space or range when the solution is near the optimal solution. Thus, the MAEA performs better than other existing evolutionary algorithms.

Even though the aforementioned results indicate that the MAEA can robustly adjust the parameters of the TNFN and obtain the highest performance among various existing evolutionary algorithms, only the whole MAEA is used to evaluate its performance. In other words, each component of the MAEA, including the MAC and MAM, cannot be evaluated independently. Therefore, the contribution of each component cannot be demonstrated. Thus, the current study investigates each component of the MAEA to demonstrate the performance of the MAEA.

Three types of models, including MAEA, MAEA-MAC, and MAEA-MAM, as shown in Table 6, were used to evaluate the training and testing performances. The MAEA-MAC adopts the traditional two-point crossover [47] instead of the proposed MAC in the MAEA to exchange the genes between offspring. In other words, only the proposed MAM was used to consider the multiple angles for mutating the chromosomes. The MSE-MAM used the traditional unit mutation [58] to mutate the unit gene of a single chromosome. Thus, only the proposed MAC was used in the crossover.

TABLE 6. The training and testing results of different components

Method	**RMS errors (Training)**		**RMS errors (Testing)**		**CPU Time (Training)**	
	Mean	**Deviation**	**Mean**	**Deviation**	**Mean (second)**	**Deviation (second)**
MAEA	**0.00098**	**0.00015**	**0.00103**	**0.00018**	**1.32**	**0.15**
MAEA-MAC	0.00117	0.00020	0.00126	0.00023	1.35	0.17
MAEA-MAM	0.00131	0.00026	0.00143	0.00031	1.41	0.21

As can be seen in the training and testing results in Table 6, the MAEA-MAC outperformed the MAEA-MAM. In other words, the MAC can obtain better performance than the MAM because it uses the crossover as its main evolutionary process for finding the optimal solution [54], and thus the modifications in the crossover step can obtain much more benefits than those in the mutation step. However, as can be seen in Table 6, the MAEA, which combines both the MAC and MAM, can obtain the best performance among the three types of models. Thus, each component of the MAEA was shown to be necessary in improving the performance of meeting optimal solution.

Compared with other existing methods, the MAEA has the following features:

1. The MAEA can increase its chances of meeting the optimal solution because it uses multiple angles for considering different strategies that can find the optimal solution.
2. The MAEA can reduce the evolutionary generations because it can seek the solution efficiently. More specifically, the MAEA can explore in a large searching space when the individuals perform poorly. Moreover, it can also reduce the searching space if the individuals perform well.
3. The MAEA can obtain smooth learning curves because it can perform MAC and MAM, which consider various candidate solutions efficiently. More specifically, the MAC and MAM can explore large ranges of searching space when the candidate solutions are far away from the optimal solution but they investigate in a small searching space when the solutions are near the optimal solution. Such consideration is helpful for evolutionary algorithms when considering various candidate solutions that can find the optimal solution in the early stages. Moreover, the MAC and MAM can also focus on particular potential solutions to meet the optimal solution in the latter stages.
4. The population size of the proposed MAEA can be reduced because the algorithm provides an efficient way of meeting the optimal solution, as shown in the parameter exploration. Therefore, only few chromosomes are necessary to perform evolution.

4.2. **Prediction of the chaotic time series.** Even though the benefits of the MAEA were demonstrated in Section 4.1, the example is too simple to demonstrate the robustness and efficiency of the MAEA. Thus, in this section, another complex chaotic time series problem, named the Mackey-Glass chaotic time series [52], is used to demonstrate the robustness and efficiency of the MAEA. The following equation describes the function related to such kind of chaotic signal.

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{22}$$

As shown in Equation (22), $x(t)$ was considered to generate the chaotic time series using the delay differential equation. The following equations were used to generate and identify the input and output patterns [52].

$$InputPattern = [x(t-18), x(t-12), x(t-6), x(t)] \tag{23}$$

$$OutputPattern = x(t+6) \tag{24}$$

According to Equations (23) and (24), each training pattern consists of an input pattern and an output pattern. The input pattern consists of four past values of $x(t)$, and the output pattern represents the value $x(t+\Delta t)$, where Δt is a time prediction value. In general, the chaotic time series can be generated using $\tau = 17$ and $x(0) = 1.2$ [36]. Moreover, the first 500 pairs (i.e., $x(1)$-$x(500)$) are used as training patterns to model the TNFN, whereas the other 500 pairs (i.e., $x(501)$-$x(1000)$) are used as the testing data set to validate the well-trained TNFN.

In this example, the normalized root mean square error (NRMS error) [55] was used to evaluate the performance of the MAEA and provide reliable simulation results because the NRMS error is a benchmark when evaluating the Mackey-Glass chaotic time series [36,55]. More specifically, the NRMS error can be computed based on the RMS error. The NRMS error can be calculated as follows:

$$NRMSE = \frac{1}{\sigma_t}\left[\frac{1}{N_t}\sum_{l=1}^{N_t}\left(Y_l(t+6) - Y_l^d(t+6)\right)^2\right]^{1/2}, \tag{25}$$

where σ_t is the estimated variance of the data, N_t is the number of the training data, $Y_l^d(t+6) = x(t+6)$ is the desired value, $Y_i(t+6)$ is the predicted value obtained from the model.

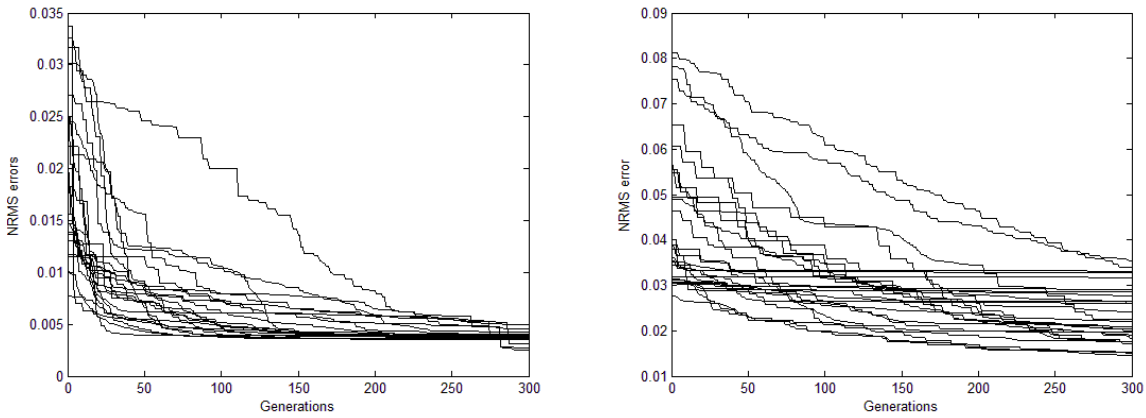
In addition, the parameter exploration shown in Section 4.1 was also applied to decide the predefined parameters of the proposed MAEA (see Table 7). Moreover, the simulation was conducted for 30 runs and each run started with the same initial parameters.

TABLE 7. The predefined parameters of the MAEA

Parameters	**Value**
The number of rules	6
The number of chromosomes in a population	30
The number of generation times	300
MaxMutationNum	9
RangeValue for weight	$[5.000, -5.000]$
RangeValue for membership functions	$[2.500, -2.500]$
Mutation rate	0.2
Crossover rate	0.4

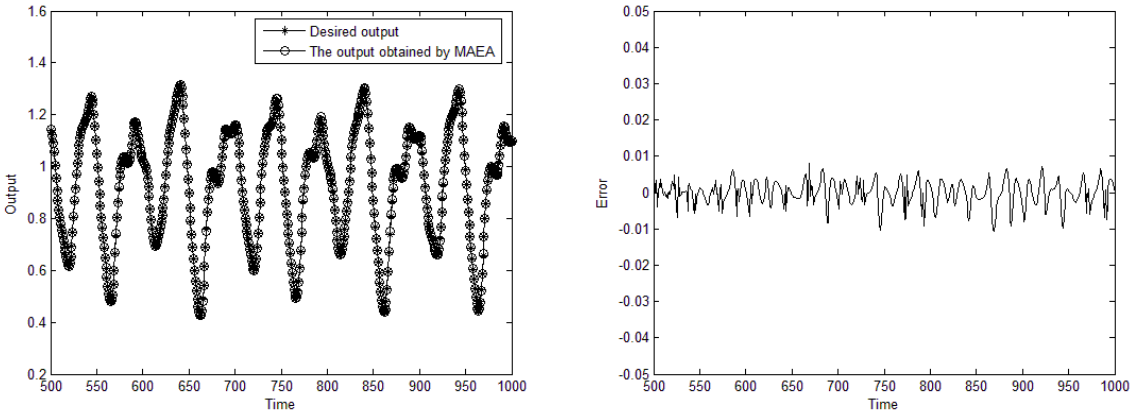
Figure 8(a) shows the learning curves of the MAEA, where each curve can reach the low NRMS error, which was close to 0.005, implying that the proposed MAEA can continue its good performance when solving such complex problem. Moreover, the traditional GA [27] was compared with the MAEA. Figure 8(b) shows the 30 learning curves of the traditional GA, which further demonstrates that the proposed MAEA can outperform the traditional GA dramatically since the MAC and MAM are helpful for improving performance (see Figures 7(a) and 8(b). Moreover, the testing results of the MAEA (see Figure 9) and traditional GA (see Figure 10) were investigated to further investigate the argument stated above. As can be seen in Figures 9 and 10 the testing results of the MAEA outperformed the results of the GA, implying that the MAEA performs robustly and efficiently when solving different complex time series problems.

In addition, other existing well-known genetic algorithms (i.e., [27,34-36,41]) were compared with the MAEA to provide reliable evidence on the aforementioned argument. The performances of the compared genetic algorithms in both training and testing steps, including the mean and standard deviation values of NRMS errors and CPU time are shown



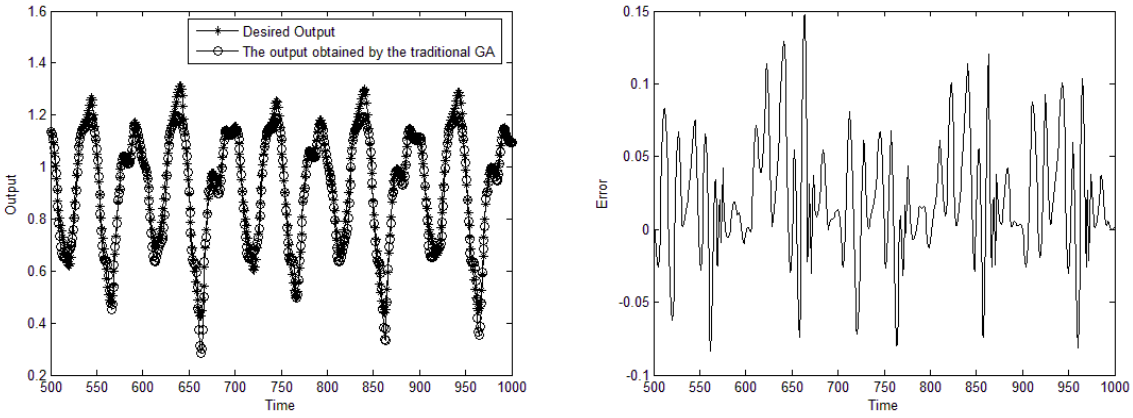
(a) (b)

FIGURE 8. The learning curves of (a) the MAEA and (b) the traditional GA



(a) (b)

FIGURE 9. The testing prediction results (a) and errors (b) of the MAEA



(a) (b)

FIGURE 10. The testing prediction results (a) and errors (b) of the traditional GA

TABLE 8. The performance comparison of various existing models

Method	RMS errors (Training)		RMS errors (Testing)		CPU Time (Training)	
	Mean	Deviation	Mean	Deviation	Mean (second)	Deviation (second)
MAEA	**0.0038**	**0.00065**	**0.0041**	**0.00073**	**41.28**	**2.47**
[27]	0.032	0.00684	0.041	0.00767	62.85	3.50
[34]	0.025	0.0091	0.034	0.0101	78.42	6.34
[35]	0.023	0.0083	0.029	0.0091	75.30	5.83
[36]	0.0089	0.0012	0.0093	0.0015	258.81	97.52
[41]	0.0076	0.0014	0.0082	0.0016	196.26	89.73

in Table 8. The MAEA not only spends smaller CPU time but also obtains lower NRMS errors than other algorithms in both training and testing steps.

In summary, the proposed MAEA has higher performance via the MAC and MAM than other existing evolutionary algorithms. Moreover, the MAEA is suitable for solving

different complex problems. In other words, the MAEA can robustly and efficiently adjust the parameters of the TNFN to face different complex problems, implying that multiple angles are useful for seeking the suitable searching space or range that can increase the chances of meeting the optimal solution. Thus, the MAEA can obtain better performance than other existing evolutionary algorithms.

4.3. **Prediction of the sunspot number.** The sunspot numbers from 1700 to 2004 exhibit nonlinear, non-stationary, and non-Gaussian cycles. In other words, the sunspot numbers are complex and even difficult to predict [13], and thus they can demonstrate if the proposed MAEA is suitable for practical applications. The input patterns used for predicting the sunspot in the proposed MAEA can be described as follows:

$$x_i(t) = y^d(t-1), \text{ where } i = 1, 2, 3. \tag{26}$$

where t represents the year and $y^d(t)$ is the sunspot numbers at the tth year. In this example, the first 180 years (from 1705 to 1884) of the sunspot numbers are used as training data set while the remaining 119 years (from 1885 to 2004) of the sunspot numbers are used as testing data set. The simulation was conducted for 30 runs and each run started with the same initial parameters. Figure 11 shows the learning curves of the MAEA in 30 runs. As can be seen in the figure, each curve can reach the low RMS error, which was close to 7.06, implying that the proposed MAEA can still keep reaching the good performance when solving practical applications. Moreover, Figure 12 shows the prediction results of the MAEA, further implying that the proposed MAEA can still obtain robust and efficient results in such practical application.

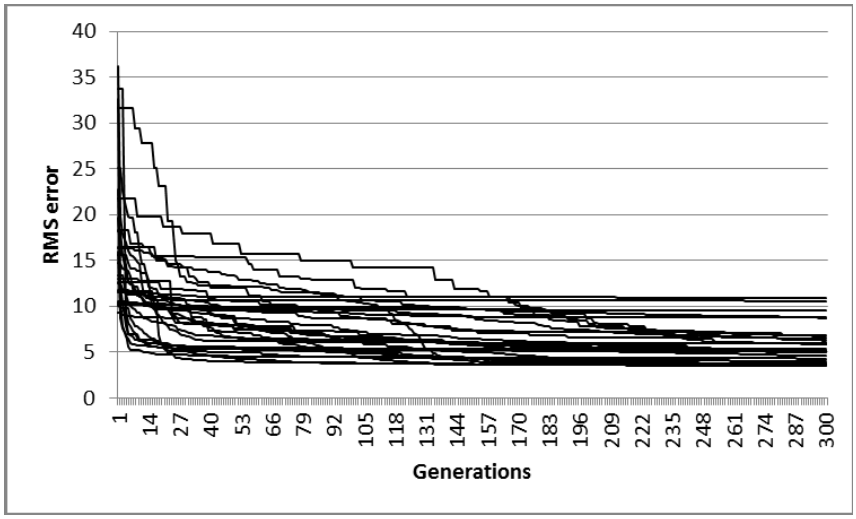
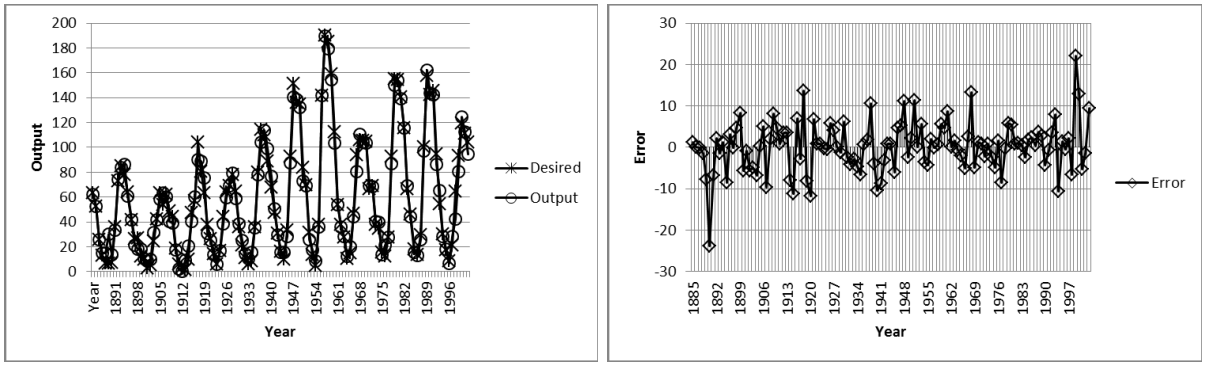


FIGURE 11. The learning curves of the MAEA in the prediction of the sunspot number

In addition, similar to the previous two examples, other existing well-known genetic algorithms (i.e., [27,34-36,41]) were compared with the MAEA to provide reliable evidence on the argument that the MAEA is suitable for practical applications. The performances of these compared genetic algorithms in both training and testing steps, including the mean and standard deviation values of RMS errors and CPU time are shown in Table 9. The MAEA not only spends smaller CPU time but also obtains lower RMS errors than other algorithms in both training and testing steps. Moreover, two performance indices, as shown in Table 10, were used to further investigate the training and forecasting error



(a) (b)

FIGURE 12. The prediction results (a) and errors (b) of the MAEA

TABLE 9. The performance comparison of various existing models in the prediction of sunspot number

Method	RMS errors (Training)		RMS errors (Testing)		CPU Time (Training)	
	Mean	Deviation	Mean	Deviation	Mean (second)	Deviation (second)
MAEA	**5.43**	**0.46**	**5.78**	**0.73**	**93.57**	**3.53**
[27]	28.37	3.35	32.78	4.12	143.54	6.83
[34]	19.51	1.86	20.94	2.23	257.13	8.72
[35]	14.76	1.31	16.78	1.76	216.87	9.69
[36]	11.86	0.95	12.57	1.12	398.59	25.46
[41]	9.12	0.83	10.36	1.03	363.74	19.57

TABLE 10. Training and forecasting error comparison of various existing models in the prediction of sunspot number

Method	Training error		Forecasting error	
	Mean	Deviation	Mean	Deviation
MAEA	**4.13**	**0.31**	**6.74**	**0.48**
[27]	12.92	1.87	18.76	2.05
[34]	10.65	1.10	15.68	1.89
[35]	8.45	0.98	13.68	1.63
[36]	7.83	0.86	12.58	1.05
[41]	7.12	0.81	11.96	0.98

of the aforementioned methods. The two performance indices are:

$$\textit{Training error: } \sum_{t=1705}^{1884} \frac{\left|y^d(t) - y(t)\right|}{180} \tag{27}$$

$$\textit{Forecasting error: } \sum_{t=1885}^{2004} \frac{\left|y^d(t) - y(t)\right|}{119} \tag{28}$$

where $y^d(t)$ represents the desired sunspot numbers at the tth year and $y(t)$ indicates the output of the MAEA at the tth year. Table 10 shows that the proposed MAEA can obtain better performance in both of the training and testing cases.

In summary, this example demonstrated that the MAEA can be successfully applied to practical applications because the performances of the predicted sunspots are better than other existing models. Moreover, the MAEA can also reduce the time spent in seeking the optimal solution, implying that the MAEA is helpful for not only increasing the chances of meeting the optimal solution but also for reducing the time consumed.

5. **The Development of the Framework.** This study shows that the proposed MAEA is a suitable algorithm for improving the performance of neuro-fuzzy networks by considering multiple angles to adjust the searching space and range. Moreover, it also indicates that the MAEA can perform better than existing evolutionary algorithms. In brief, the MAEA can be used to reach the objectives of the current study. Based on the results, a framework was developed to provide a guideline for implementing the evolutionary algorithm, which can apply multiple angles to increasing the chances of meeting the optimal solution when constructing neuro-fuzzy networks. Figure 13 shows the details of the framework.

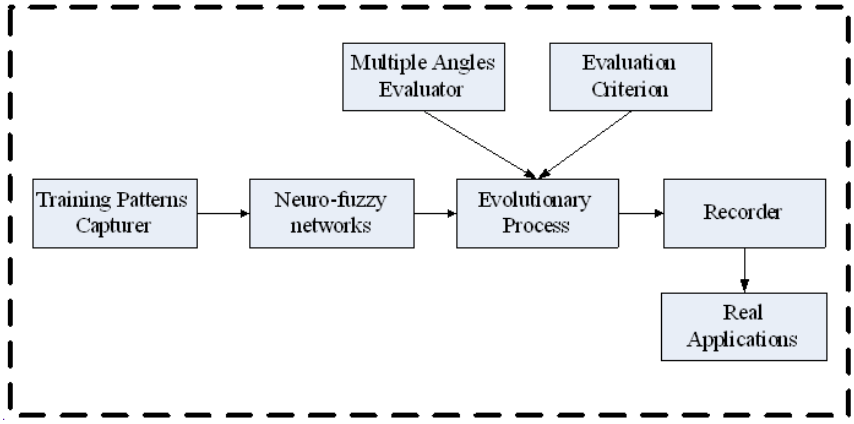


FIGURE 13. The framework of the multi-angles evolutionary algorithm

As can be seen in Figure 13, the training patterns capturer is mainly used to capture the input and output patterns, which are used to train neuro-fuzzy networks. The neuro-fuzzy networks are mainly used to obtain the relationships between the captured input patterns and the outcomes. The evolutionary process is used to help the neuro-fuzzy networks find the optimal relationship between the captured input patterns and the outcomes. More specifically, the multiple angles evaluator is mainly used to judge if the neuro-fuzzy networks are good enough and the evaluation criterion is used to define multiple angles, which are then used to adjust the searching space and range when performing the evolutionary process.

After finishing the evolutionary process, the recorder is used to store the best solution (e.g., the best combination of the parameters of the TNFN). The recorder saves the fruitful results of the evolutionary process. Moreover, it can also help in the constructing the neuro-fuzzy networks that are used to address real applications.

In summary, the framework is helpful for developing an architecture that can help neuro-fuzzy networks meet the optimal solution. More specifically, the framework can use multiple angles to adjust the searching space and range automatically and to let the evolutionary process seek the optimal solution in a suitable searching space and range. In doing so, the performance of the evolutionary process can be improved. Thus, the framework can be treated as a benchmark when developing the feasible and robust evolutionary algorithm for seeking the optimal solution of neuro-fuzzy networks.

6. **Conclusions.** This study aims to develop a feasible and robust architecture to seek the optimal solution when constructing neuro-fuzzy networks. Thus, the MAEA was proposed to consider multiple angles when adjusting the searching space to improve the performance of evolutionary process. Moreover, the MAEA was used to adjust the parameters of the TNFN. As shown in the results, the MAEA can efficiently adjust the parameters of the TNFN to find the suitable relationship between the input patterns and the outcomes by considering multiple angles. Thus, a framework was proposed based on the results to provide a guideline for the development of a novel architecture that can increase the chances of meeting the optimal solution by considering multiple angles.

Even though the MAEA can efficiently seek the optimal solution by considering multiple angles, some limitations should be considered in future works. More specifically, some parameters of the MAEA, including the crossover and mutation rate, population size, generation times, number of fuzzy rules, and value ranges should be predefined. Thus, an automatic and robust way of defining such parameters is necessary. Besides, the MAEA was only used to adjust the parameters of the TNFNs. The other issues of future works need to apply the MAEA to other applications (e.g., neural networks [12], fuzzy controllers [30], and stock predictors [3]) to demonstrate that the proposed framework is suitable for various fields. Moreover, since the proposed MAC and MAM play important roles in the MAEA, the manner by which the performance can be improved in future works is necessary. For instance, the self-adapter architecture [39] may be used to automatically decide the suitable point or updated value based on the performance of each individual. In doing so, the performances of the crossover and mutation processes may be improved.

REFERENCES

[1] E. Frias-Martinez, S. Y. Chen and X. Liu, Survey of data mining approaches to user modeling for adaptive hypermedia, *IEEE Trans. SMC: Part C*, vol.36, no.6, pp.734-749, 2006.

[2] C. J. Lin and Y. J. Xu, The design of TSK-type fuzzy controllers using a new hybrid learning approach, *International Journal of Adaptive Control and Signal Processing*, vol.20, pp.1-25, 2006.

[3] J. Tan and C. Quek, A BCM theory of meta-plasticity for online self-reorganizing fuzzy-associative learning, *IEEE Trans. Neural Networks*, vol.21, no.6, pp.985-103, 2010.

[4] S. K. Ng and H. J. Chizeck, Fuzzy model identification for classification of gait events in paraplegics, *IEEE Trans. Fuzzy Systems*, vol.5, no.4, pp.536-544, 1997.

[5] C. J. Lin, H. C. Chuang and Y. J. Xu, Face detection in color images using efficient genetic algorithms, *Optical Engineering*, vol.45, no.4, 2006.

[6] J. Zhang and J. Morris, Neuro-fuzzy networks for process modeling and model-based control, *IEE Colloquium on Neural and Fuzzy Systems: Design, Hardware and Applications*, 1997.

[7] J. Zhang, Modeling and optimal control of batch processes using recurrent neuro-fuzzy networks, *IEEE Trans. Fuzzy Systems*, vol.13, no.4, pp.417-427, 2005.

[8] G. G. Towell and J. W. Shavlik, Extracting refined rules from knowledge-based neural networks, *Machine Learning*, vol.13, pp.71-101, 1993.

[9] S. Mitra and Y. Hayashi, Neuro-fuzzy rule generation: Survey in soft computing framework, *IEEE Trans. Neural Networks*, vol.11, no.3, pp.748-768, 2000.

[10] Z. Zhang, H. Zha and M. Zhang, Spectral methods for semi-supervised manifold learning, *IEEE Int'l Joint Conf. on Computer Vision and Pattern Recognition*, pp.1-6, 2008.

[11] C. Y. Hsu, Y. C. Hsu and S. F. Lin, Reinforcement evolutionary learning using data mining algorithm with TSK-type fuzzy controllers, *Appl. Soft Comput.*, vol.11, no.3, pp.3247-3259, 2011.

[12] D. B. Fogel, L. J. Fogel and V. W. Porto, Evolutionary methods for training neural networks, *IEEE Int'l Joint Conf. on Neural Networks for Ocean Engineering*, pp.317-327, 1991.

[13] C. J. Lin and Y. J. Xu, A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications, *Fuzzy Sets and Systems*, vol.157, no.8, pp.1036-1056, 2006.

[14] S. H. Liao, H. H. Ho and H. W. Lin, Mining stock category association and cluster on Taiwan stock market, *Expert Systems with Applications*, vol.35, pp.19-29, 2008.

[15] C. Hou, C. Hsieh, Y. C. Hsu and S. Y. Chen, Multiple considerations for identifying disorientation problems within web-based learning: A neural network approach, *Asia-Pacific Conference on Technology Enhanced Learning*, 2011.
[16] K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks*, vol.1, pp.4-27, 1990.
[17] C. F. Juang and C. T. Lin, A recurrent self-organizing neural fuzzy inference network, *IEEE Trans. Neural Networks*, vol.10, no.4, pp.828-845, 1999.
[18] H. Ishibuchi and M. Nii, Generating fuzzy if-then rules from trained neural networks: Linguistic analysis of neural networks, *IEEE Int'l Joint Conf. on Neural Networks*, vol.2, pp.1133-1138, 1996.
[19] H. Seki and M. Mizumoto, Fuzzy functional inference method, *IEEE Int'l Joint Conf. on Fuzzy Systems*, pp.1-6, 2010.
[20] B. Mendil and K. Benmahammed, Generalized adaptive defuzzifier, *IEEE Int'l Joint Conf. on Fuzzy Systems*, vol.2, pp.1680-1683, 1998.
[21] C. F. Juang and C. T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, *IEEE Trans. Fuzzy Systems*, vol.6, no.1, pp.12-31, 1998.
[22] J. S. R. Jang, ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Trans. Systems Man Cybern.*, vol.23, pp.665-685, 1993.
[23] F. J. Lin, C. H. Lin and P. H. Shen, Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive, *IEEE Trans. Fuzzy Systems*, vol.9, no.5, pp.751-759, 2001.
[24] F. D. Zahlay, K. S. R. Rao and T. B. Ibrahim, A new intelligent autoreclosing scheme using artificial neural network and taguchi's methodology, *IEEE Trans. Industry Applications*, vol.47, no.1, pp.306-313, 2011.
[25] C. J. Lin, Y. J. Xu and C. Y. Lee, An efficient genetic algorithm for TSK-type neural fuzzy identifier design, *Int'l Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Lecture Notes in Artificial Intelligence*, vol.3533, pp.551-553, 2005.
[26] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, MA, 1989.
[27] J. K. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, MIT Press, Cambridge, MA, 1992.
[28] L. J. Fogel, Evolutionary programming in perspective: The top-down view, in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II and C. Goldberg (eds.), Piscataway, NJ, IEEE Press, 1994.
[29] I. Rechenberg, Evolution strategy, in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II and C. Goldberg (eds.), Piscataway, NJ, IEEE Press, 1994.
[30] C. L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, *Proc. of the 4th Int. Conf. Genetic Algorithms*, pp.450-457, 1991.
[31] C. J. Lin and Y. J. Xu, A novel genetic reinforcement learning for nonlinear fuzzy control problems, *Neurocomputing*, vol.69, no.16-18, pp.2078-2089, 2006.
[32] C. T. Lin and C. P. Jou, GA-based fuzzy reinforcement learning for control of a magnetic bearing system, *IEEE Trans. Systems Man Cybern. Part B*, vol.30, no.2, pp.276-289, 2000.
[33] C. F. Juang, J. Y. Lin and C. T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design, *IEEE Trans. Systems Man, Cybern. Part B*, vol.30, no.2, pp.290-302, 2000.
[34] F. J. Gomez, *Robust Non-linear Control through Neuroevolution*, Ph.D. Thesis, The University of Texas at Austin, 2003.
[35] C. J. Lin, Y. J. Xu and C. Y. Lee, A self-constructing neural fuzzy network with dynamic-form symbiotic evolution, *Int'l Conf. Machine Learning; Models, Technologies and Applications*, Las Vegas, USA, 2005.
[36] J. H. Holland, *Adaptation in Neural and Artificial System*, MIT Press, Cambridge, MA, USA, 1992.
[37] R. E. Smith, S. Forrest and A. S. Perelson, Searching for diverse, cooperative populations with genetic algorithms, *Evol. Comput.*, vol.1, no.2, pp.127-149, 1993.
[38] F. Gomez and J. Schmidhuber, Co-evolving recurrent neurons learn deep memory POMDPs, *Proc. of Conf. Genetic and Evolutionary Computation*, Washington, DC, USA, pp.491-498, 2005.
[39] C. J. Lin and Y. J. Xu, A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications, *Fuzzy Sets and Systems*, vol.157, no.8, pp.1036-1056, 2006.
[40] M. Bowman, L. C. Briand and Y. Labiche, Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms, *IEEE Trans. Software Engineering*, vol.36, no.6, pp.817-837, 2010.

[41] C. J. Lin and Y. C. Hsu, Reinforcement hybrid evolutionary learning for recurrent wavelet-based neuro-fuzzy systems, *IEEE Trans. on Fuzzy Systems*, vol.15, no.4, pp.729-745, 2007.
[42] C. J. Lin and Y. J. Xu, Efficient reinforcement learning through dynamical symbiotic evolution for TSK-type fuzzy controller design, *International Journal of General Systems*, vol.34, no.5, pp.559-578, 2005.
[43] Y. Hao, Deriving analytical input & output relationship for fuzzy controllers using arbitrary input fuzzy sets and Zadeh fuzzy AND operator, *IEEE Trans. Fuzzy Syst.*, vol.14, no.5, pp.654-662, 2006.
[44] C. Reyes, T. Hilaire, S. Paul and C. F. Mecklenbrauker, Evaluation of the root mean square error performance of the past-consensus algorithm, *International ITG Workshop on Smart Antennas*, pp.156-160, 2010.
[45] Y. P. Zou, Z. K. Mi and M. H. Xu, Dynamic load balancing based on roulette wheel selection, *Proc. of IEEE Int. Conf. Communications, Circuits and Systems*, Guilin, China, vol.3, pp.1732-1734, 2006.
[46] P. M. Godley, D. E. Cairns, J. Cowie and J. McCall, Fitness directed intervention crossover approaches applied to bio-scheduling problems, *Proc. of IEEE Int. Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, Sun Valley, USA, pp.120-127, 2008.
[47] G. R. Raidl, G. Koller and B. A. Julstrom, Biased mutation operators for subgraph-selection problems, *IEEE Trans. Evolutionary Computation*, vol.10, no.2, pp.145-156, 2006.
[48] H. J. Lee, Y. S. Ma and Y. R. Kwon, Empirical evaluation of orthogonality of class mutation operator, *Proc. of IEEE Int. Conf. Software Engineering*, pp.512-518, 2004.
[49] C. J. Lin and C. T. Lin, An ART-based fuzzy adaptive learning control network, *IEEE Trans. Fuzzy Systs.*, vol.5, no.4, pp.477-496, 1997.
[50] X. Xu and H. G. He, Residual-gradient-based neural reinforcement learning for the optimal control of an acrobat, *Proc. of IEEE Int. Conf. Intelligent Control*, pp.27-30, 2002.
[51] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*, Prentice-Hall, Englewood Cliffs, NJ, 1996.
[52] R. S. Cowder III, *Predicting the Mackey-Glass Time Series with Cascade-Correlation Learning*, D. Touretzky, G. Hinton and T. Sejnowski (eds.), 1990.
[53] K. A. De Jong, *Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. Thesis, The University of Michigan, 1975.
[54] C. Pitangui and G. Zaverucha, Improved natural crossover operators in GBIVIL, *Proc. of IEEE Int. Conf. Evolutionary Computation*, pp.2157-2164, 2008.
[55] J. H. Nie and T. H. Lee, Rule-based modeling: Fast construction and optimal manipulation, *IEEE Trans. Systems Man Cybern. Part A*, vol.26, no.6, pp.728-738, 1996.