Expert Systems with Applications 39 (2012) 10749–10763

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

The partial solutions consideration based self-adaptive evolutionary algorithm: A learning structure of neuro-fuzzy networks

Pei-Chia Hung *, Sheng-Fuu Lin

Department of Electrical and Control Engineering, National Chiao-Tung University, Taiwan

ARTICLE INFO

Keywords:
Evolutionary algorithm
Self adaptive
Global solution
Neuro-fuzzy networks

ABSTRACT

Evolutionary computation has become a popular research field due to its global searching ability. Therefore, it raises a challenge to develop an efficient and robustness evolutionary algorithm to not only reduce the evolution process but also increase the chances to meet the global solution. To this end, this study aims to provide a novel evolutionary algorithm named the partial solutions consideration based self-adaptive evolutionary algorithm (PSC-SEA) to address this issue; the proposed algorithm is applied to adjust the parameters of the neuro-fuzzy networks. More specifically, different from the existing evolution algorithms, the partial solutions consideration (PSC) tends to consider both the specializations and complementary relationships of the partial solutions from the complete solution to prevent converging to suboptimal solution. Moreover, a self-adaptive evolutionary algorithm (SEA) is proposed to dynamically adjust the searching space according to the performance. By doing so, the proposed evolutionary algorithm can consider the influence of partial solutions and provide a suitable searching space to increase the chances to meet the global solution. As shown in the results, the proposed evolutionary algorithm obtains better performance and smoother learning curves than other existing evolutionary algorithms. In other words, the proposed evolutionary algorithm can efficient tune the parameters of the neuro-fuzzy networks to meet the global solution. Base on the results, a framework is proposed to build a benchmark for developing the evolutionary algorithms that can not only consider the influence of partial solutions but also provide a suitable searching space.

1. Introduction

In recent decade, the research related to learning structures of the neuro-fuzzy networks have become a popular research field due to its powerful adaptive ability in various applications such as nonlinear control (Lin & Xu, 2007), face detection (Lin, Chuang, & Xu, 2006), stock prediction (Lee, 2004) and etc. More specifically, the learning structure is mainly used to let neuro-fuzzy networks can be suitable for particular application by constructing the relationship between the input and output patterns. In other words, it is mainly used to develop the suitable way to adjust the parameters of the neuro-fuzzy networks (Sadighi & Kim, 2011). Thus, the learning structure plays an important to influence the performance of the neuro-fuzzy networks.

Since the learning structures can influence the performance of the neuro-fuzzy networks, there is an important issue for developing the suitable learning structure when modeling the neuro-fuzzy networks. To address this issue, in this study, we aim to develop the efficiently and robustly learning structure to model the neuro-fuzzy networks that meet the global solution (Lin & Hsu, 2007). Among various global based learning structures, the evolutionary algorithm (Hsu, Lin, & Cheng, 2010) is widely used as the learning structure of the neuro-fuzzy networks due to its powerful global searching ability. More specifically, the evolutionary algorithms are parallel and global based search techniques that can help construct the neuro-fuzzy networks that can meet the optimal solution. Moreover, it can simultaneously evaluate many points in the searching space.

Although the evolutionary algorithm can reach our aim, the traditional evolutionary algorithms only pay more attention to the whole solution when finding the optimal solution (Karr, 1991; Lin & Jou, 2000; Lin & Xu, 2006). In other words, single individual is used to represent the overall performance. However, single solution cannot consider the specializations to ensure diversity that can prevent the single solution converging to suboptimal solutions (Smith, Forrest, & Perelson, 1993). Therefore, in moist serious case, the single solution consideration may easy meet the suboptimal solutions.

In addition, the traditional evolutionary algorithms may also focus on evaluating the fixed searching space for finding the optimal solution (Karr, 1991; Lin & Jou, 2000; Lin & Xu, 2006). In other words, the searching space in such algorithms cannot be adjusted

* Corresponding author.
E-mail addresses: mikehome.ece93g@nctu.edu.tw (P.-C. Hung), sflin@mail.nctu.edu.tw).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.02.190

according to the performance of each candidate solution. For instance, if a candidate solution is near to the optimal solution, the candidate solution may still need to search in the large range of space. Therefore, such algorithms may slowly or even hardly meet the optimal solution by using the single criterion evaluation.

To address this issue, this study proposed a novel evolutionary algorithm named partial solutions consideration based self-adaptive evolutionary algorithm (PSC-SEA). More specifically, the PSC-SEA can be divided into two parts such as the partial solutions consideration (PSC) and self-adaptive evolutionary algorithm (SEA). Regarding the PSC, it is mainly used to consider the influence of the partial solutions (specializations) from the complete solution to prevent the single solution converging to suboptimal solutions. More specifically, the complete solution is partitioned into several individuals. The PSC considers both specialization of the partial solution and complementary relationship among individuals to prevent the evolutionary algorithm to meet the suboptimal solution. Regarding the SEA, it is mainly proposed to adjust the searching space automatically according to the performance of each individual. More specifically, an individual can be further searched in the small range of space when the individual is near to the optimal solution. Conversely, if an individual is far away from the optimal solution, the individual should be further searched in the large range of space. By doing so, the parameters of the neuro-fuzzy network can be tuned efficiently and the accuracy of the outputs obtained from the neuro-fuzzy networks can be improved.

In summary, this study aims to provide a novel evolutionary algorithm to not only consider the influence of partial solution but also provide a self adaptive searching space when adjusting the parameters of the neuro-fuzzy networks. By doing so, the neuro-fuzzy networks can not only efficiently and robustly tune the parameters of the neuro-fuzzy networks but also increase the chances to meet the optimal solution. This paper is organized as follows. The related works are introduced in Section 2. In Section 3, the methodology development is described. In Section 4, the illustrative results are presented. In Section 5, a framework related to the novel evolutionary algorithm is summarized. Finally, the concluding remarks are showed in the last section.

2. Literature review

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

2.1. Neuro-fuzzy networks

Recently, neuro-fuzzy networks have been widely applied in several fields (Lee, 2004; Lin & Xu, 2007; Lin et al., 2006). The main purpose of the neuro-fuzzy networks is to provide robust structure that can be applied to different applications (Lin & Xu, 2006; Tan & Quek, 2010). More specifically, in the classical applications, an accurate mathematical model is required to achieve their purposes. In other words, different applications may need different mathematical model. However, in real world applications, it is difficult or even hardly to generate a suitable mathematical model to represent the applications. Thus, to avoid strongly associated with the mathematical models especially for nonlinear and complex problems (Juang & Lin, 1999; Narendra & Parthasarathy, 1990), the neuro-fuzzy network provide a suitable solution to address diverse purposes of different applications.

Generally, the neuro-fuzzy network can be divided into a forward and backward structure (learning structure). Regarding the forward structure, it is mainly used to produce the output of the network. More specifically, it consists of a set of fuzzy if-then rules (Mitra & Hayashi, 2000), fuzzy inference mechanism (Seki & Mizumoto, 2010) and defuzzifier mechanism (Melgarejo, 2002). Table 1 shows each of these components.

Table 1
Components of the forward structure.

Component	Content
Fuzzy if-then rules	It consists of an antecedent and consequent part. The former represents different degree (membership degrees) that each input patterns belong to each membership function and the latter means the firing strength of each fuzzy rule
Fuzzy inference schema	It is mainly used to conduct the fuzzy output by inferring fuzzy rules
Defuzzifier schema	It is used to generate the crisp output from fuzzy output

Regarding the backward structure (learning structure), it is mainly used to improve the performance of the forward structure by adjusting the parameters of the neuro-fuzzy network (i.e. the parameters of the antecedent part and consequent part). Recently, several studies tend to proposed novel algorithms applied to learning structure to adjust the parameters of fuzzy if-then rules (Jang, 1993; Juang & Lin, 1998; Lin, Lin, & Shen, 2001; Zahlay, Rao, & Ibrahim, 2011). Among them, the most well known learning algorithm is back-propagation (BP) Lin et al., 2001; Zahlay et al., 2011. More specifically, it is used to adjust the parameters of the neuro-fuzzy networks by using the steepest descent technique to minimize the error function (Xu & Zhang, 2010). Since the BP can be successfully applied to adjust the parameters of the neuro-fuzzy networks, it may easily reach the local minima or even never find the global solution (Lin, Xu, & Lee, 2005). Beside, for different structures of neuro-fuzzy networks, new mathematical expressions are needed for each network structure.

To face aforementioned issue, the global based learning structures are proposed to prevent achieving the sub-optimal solution (Goldberg, 1989). More specifically, such structures mainly adopt the principle of the heuristics learning to help the networks meet the optimal solution (Lin et al., 2005). Among various global based learning structures, evolutionary algorithms are widely used to prevent the neuro-fuzzy networks achieving the sub-optimal solution (Karr, 1991; Lin & Jou, 2000; Lin & Xu, 2006). In other words, the evolutionary algorithms can be applied to find an optimal solution when adjusting the parameters of neuro-fuzzy networks. Moreover, they can also be suitable for different neuro-fuzzy network structures. It implies that evolutionary algorithms do not need to develop new mathematical expressions for different structures of the neuro-fuzzy networks. Thus, the evolutionary algorithms are better candidates than the BP algorithms to train the parameters of the neuro-fuzzy networks. Such arguments can also explain why this study aims to improve the learning structure about the evolutionary algorithms.

2.2. Evolutionary algorithm

As mentioned above, the evolutionary algorithms can be successfully applied to improve the global searching ability of traditional learning structure (e.g. BP algorithm). Thus, recently, there are several evolutionary algorithms have been applied to tune the parameters of neuro-fuzzy networks for increasing the chances to meet the optimal solution (Fogel, 1994; Koza, 1992; Rechenberg, 1994). Such algorithms cannot only provide parallel and global searching techniques to seek the solutions but also simultaneously evaluate many points in the search space. Among various evolutionary algorithms, the most well-known model is genetic fuzzy models (Karr, 1991; Lin & Jou, 2000; Lin & Xu, 2006) that used the genetic algorithms (GAs) Holland, 1992, to train the parameters of fuzzy models. For instance, Karr applied GA to design a fuzzy controller (Karr, 1991). As shown in his work, the GA is mainly used to seek the optimal parameters of each membership function. Moreover, Xu and Lin applied reinforcement GA to seek the optimal parameters of the TSK-type neural fuzzy controller (Lin & Xu, 2006). Their work showed that the well-trained TSK-type neural fuzzy controller could obtain better performance than the BP algorithms in several control applications.

Since genetic fuzzy models are useful to improve the global searching ability, it still has some challenges needed to be taken into account. More specifically, the major issue of the evolutionary algorithms is "how to design the criteria to generate the individuals that can meet the optimal solution?". In other words, there is a need to develop a novel architecture to improve the generations of the evolutionary algorithms. To address this issue, several studies tend to develop novel evolutionary algorithms. For instance, Smith et al. proposed a symbiotic evolution to provide a multiple consideration when evaluating the individuals (Smith et al., 1993). More specifically, the symbiotic evolution divides the complete solution into several partial solutions. Each partial solution can be characterized as specializations. The specialization property can ensure the diversity of evolution. In other words, such diversity can prevent a population to meet suboptimal solutions. As shown in their work, the symbiotic evolution can obtain the well performance by evaluating the partial solution. Similar to the search of Smith et al., Gomez and Schmidhuber also proposed the enforced sub-populations (ESP) to use multiple considerations to prevent for meeting the suboptimal solutions (Gomez, 2003). As shown in their works, the sub-populations are used to evaluate the partial solutions. Their results indicated that the ESP can obtain better performance than systems with only single consideration.

More recently, Lin and Xu modified such multiple considerations to efficiently seek the parameters of the particular NFN (i.e. TSK-type NFN) Lin & Xu, 2007. More specifically, the fine adjusting is used to adjust the parameters of the NFNs for improving the evolutionary algorithm. Their work indicated that the multiple considerations can demonstrate high performance than traditional evolutionary algorithms. Besides, Lin and Hsu proposed the hybrid evolutionary learning algorithm (HELA) to apply the multiple considerations to both a structure and parameter learning when training the parameters of wavelet neuro-fuzzy networks (Lin & Hsu, 2007). Their results demonstrated that the HELA obtain better performance than other existing evolutionary algorithm.

Although aforementioned studies verified the performance of their proposed algorithms, they may lack to not only take into account the influence of both specializations and complementary relationship but also consider adjusting searching space when performing the evolutionary process (e.g. reproduction, crossover and mutation). More specifically, regarding the former, even if the multiple considerations can demonstrate high performance (Gomez, 2003; Lin & Hsu, 2007; Lin & Xu, 2007; Smith et al., 1993); they only focus on evaluating the efficiency of a particular solution by considering the performance of all the possible combinations that contains that particular solution. In other words, the performance of each partial solution cannot completely represent the performance of the complete solution. It implies that the aforementioned algorithms can not exactly combine the partial solutions that individually perform well to generate the best performed solution. Regarding the latter, aforementioned algorithms only focuses on fixed searching space when performing the evolution. In other words, such algorithms may not offer a suitable searching space to increase the chances to let the individuals generate better offspring in each generation. In more serious case, these algorithms may be difficult to meet the optimal solution.

To address aforementioned issues, this study proposed a novel evolutionary algorithm to consider both multiple considerations and self adaptive searching space. Regarding the former, the influence of the specializations and complementary relationship in partial solutions is considered to prevent the evolutionary algorithms converging to suboptimal solutions. More specifically, each partial solution can not only be evaluated independently but also consider the complementary relationship among different partial solutions. Regarding the latter, the searching space will be adjusted automatically according to the performance of each individual. More specifically, an individual can be searched in the suitable searching space. By doing so, the parameters of the neuro-fuzzy network can be trained efficiently and the accuracy of the outputs obtained from the neuro-fuzzy networks can be improved.

3. Methodology

The methodology of this study is introduced in this section. More specifically, the neuro-fuzzy network and partial solutions consideration based self-adaptive evolutionary algorithm (PSC-SEA) are described as following sections.

3.1. Neuro-fuzzy network

Neuro-fuzzy networks are mainly used to represent the fuzzy if–then rules in a network structure. By doing so, the well-know learning algorithms of artificial neural networks can be applied to train the fuzzy if–then rules. Generally, the most popular neuro-fuzzy network is Takagi-Sugeno-Kang (TSK) type neuro-fuzzy networks (Lin & Xu, 2007; Lin et al., 2006; Lin & Xu, 2007) since it could achieve superior performance than the various neuro-fuzzy networks in both network size and learning accuracy. To this end, a TSK-type neuro-fuzzy network (TNFN) is used to reach the aims of this study. In other words, the PSC-SEA, which will be described in the following section, was used to train the parameters of the TNFN.

The TNFN provides the minimum fuzzy implication to perform the fuzzy reasoning. Moreover, the consequence of each rule is a function related to the input variables in the network. The general adopted function is the linear combination of input variables plus a constant term. A TSK-type fuzzy rule is shown below

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

As shown in Eq. (1), the parameter $\boldsymbol{w_{0j}}$ represents the constant term that used to plus the linear combination of input variables to generate the consequence of jth rule node. Moreover, the parameter $\boldsymbol{w_{ij}}$ represents the ith parameter, which used to product 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. Instead, the model output is computed as the weighted average of the crisp rule outputs, which is computationally less expensive then calculating the center of gravity.

To further understand the TNFN, the structure of the TNFN is shown in Fig. 1. It is a five-layer network structure. The function of each layer is shown below.

3.1.1. Input layer

This layer is mainly used to transmit the input values to the next layer. The function of each node in this layer is shown below

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

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

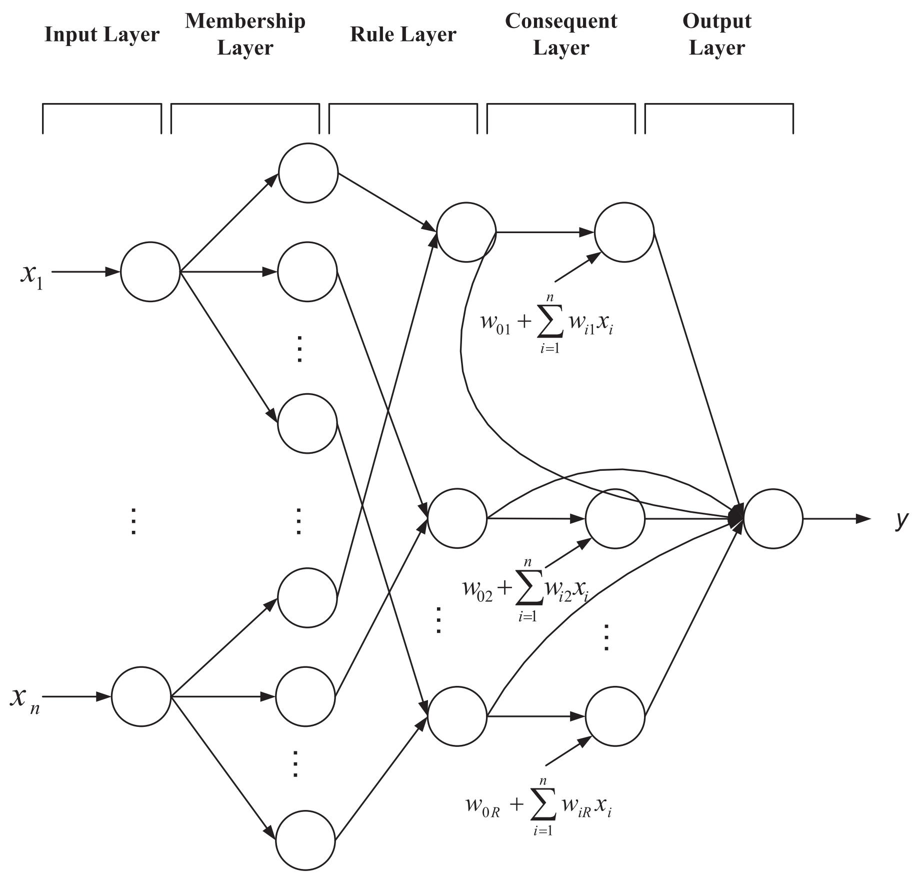

Input Layer
Membership Layer
Rule Layer
Consequent Layer
Output Layer
x_1
x_n
$w_{01}+\sum_{i=1}^{n} w_{i1}x_i$
$w_{02}+\sum_{i=1}^{n} w_{i2}x_i$
$w_{0R}+\sum_{i=1}^{n} w_{iR}x_i$
y

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

3.1.2. Membership function layer

The nodes of this layer are used to compute the membership degree that corresponds to each input node in the *Input Layer*. By doing so, the membership degree of each input node that belongs to a fuzzy set (Wang, Wang, Bai, Chen, & Sun, 2010) is calculated in this layer. Each node in this layer corresponds to one linguistic label of the particular input node in the *Input Layer*. In this study, the most well know Gaussian membership function is adopted in this layer (Baradaran-K, Shekofteh, Toosizadeh, & Akbarzadeh-T, 2010). The function of each node in this layer is shown below

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

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

3.1.3. Rule layer

This layer tends to compute the firing strength of each rule. More specifically, each node in this layer is determined by the fuzzy inference operation. In this study, the "AND" fuzzy inference operation is adopted (Wang, He, Peng, & Liang, 2010). In other words, the product operation is used to compute the firing strength of each rule. The function of each rule is shown below

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

3.1.4. Consequent Layer

The nodes of this layer are mainly used to determine the consequent part of each rule. More specifically, each node in this layer is compute based on the linear combination of input variables. Moreover, the input nodes in this layer consist of the outputs delivered from the previous layer and the input variables from the *Input Layer* (see Fig. 1). The function of each node is shown below

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

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

3.1.5. Output Layer

Each node in this layer is used to compute 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 the crisp output value. The function of this layer is shown below

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i)}{\sum_{j=1}^{R} u_j^{(3)}} \tag{6}$$

where R is the number of fuzz rules.

3.2. Partial solutions consideration based self-adaptive evolutionary algorithm (PSC-SEA)

In this section, the PSC-SEA is introduced in the following subsections, including learning components (Section 3.2.1) and learning procedure (Section 3.2.2).

3.2.1. Learning components

This section describes the main components of the proposed PSC-SEA. More specifically, the required components can be further divided into the components related to partial solutions consideration (PSC) and the components related to Self-Adaptive Evolutionary Algorithm (SEA). The details of these two types of components are shown below.

3.2.1.1. Partial solutions consideration (PSC). The main purpose of the PSC is to consider both specializations and complementary relationship. Regarding the former, the specializations of both individuals and population are considered. More specifically, the structure of the individuals and coding schema should be modified to let the individuals can be evaluated individually. Regarding the latter, the fitness value evaluation should be modified to consider not only the performance of each partial solution but also the complementary performance among different partial solutions. The details of these two types of components are shown below.

3.2.1.1.1. Individuals. The idea of PSC is that each individual in a population represents only a partial solution instead of the complete solution. In other words, complete solution is obtained from combining several individuals. As mention in Section 2, previous studies indicated that partial solutions can consider diversity to prevent the evolutionary algorithms converging to suboptimal solutions. By doing so, the evolutionary algorithms can have more chances to meet the optimal solution than the traditional evolutionary algorithms (Fogel, 1994; Karr, 1991; Koza, 1992; Lin & Jou, 2000; Rechenberg, 1994).

Generally, the applications considered partial solutions only used a single population to evaluate each partial solution (Smith et al., 1993). In other words, there is a lack of considering the specializations of the populations. More specifically, single population may generate similar individuals when performing the evolutionary process. By doing so, the complete solution may generate by combining several individuals that overlap too much. It is due to the fact that the evolutionary steps tend to find out the optimal solution by using selecting the well-perform individuals. In other words, the well-perform individuals may have more chances to survive or even reproduce in each generation. However, the individuals only represent a particular solution. It implies that particular individuals may generate too much to let the complete solution too difficult to meet the optimal solution. To this end, this study proposed partial solutions consideration (PSC) to prevent the complete solution that generates form similar individuals. In other words, the specializations of the population are taken into account when building the population. More specifically, a single population of the PSC consists of several sub-populations. Each sub-population represents the collection of individuals that represent the particular partial solution. The complete solution is generated from selecting individuals from several sub-populations. By doing so, the PSC can prevent to building the complete solution from similar individuals.

To further understand the concept of the PSC, the structure of the population in the PSC is shown in Fig. 2. Fig. 2 showed that the complete TNFN is constructed from selecting several individuals from several sub-populations. Each individual in a sub-population represent only a particular fuzzy rule. In other words, the number of the sub-population is equals to the number of the fuzzy rules of the TNFN.

3.2.1.1.2. Coding. After defining the structure of the individuals, the other issue of the PSC is to encode the adjustable parameters into an individual. In other words, the parameters of the TNFN are encoded into an individual in this operation. More specifically, the adjustable parameters of each fuzzy rule are translated into an individual. According to the Eq. (1), 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 part of a fuzzy rule are encoded in to an individual. The coding schema of an individual is shown in Fig. 3. As shown in this figure, i and j represent the ith input node in the jth rule. The coding type of the chromosomes in this study is a *float point* type.

3.2.1.1.3. Fitness assignment. Although aforementioned steps can consider the specializations of both population and individuals, there still lack of considering the complementary relationship among various individuals. More specifically, in traditional evolutionary algorithm considered the principle of particular solution (Gomez, 2003; Lin & Hsu, 2007; Lin & Xu, 2007; Smith et al., 1993), the performance of an individual is evaluated by summing up the fitness values of all the possible combinations that contains that the individual. In other words, the performance of each individual cannot really be used to represent the performance of the complete solution. For instance, a set of individuals with the highest performance do not mean they can be used to combine to generate the complete solution with the highest performance. Conversely, their combinations may cause the complete solution to meet the bad performance. Thus, even if the specializations of

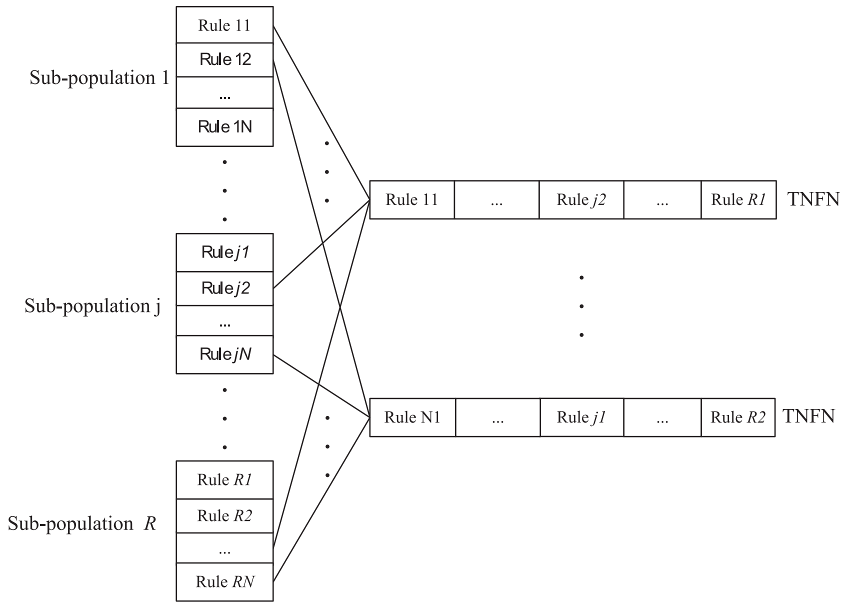


Fig. 2. The structure of the population in the PSC.

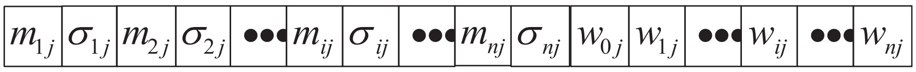


Fig. 3. The coding schema of a chromosome.

the population are considered in this study, it may be still difficult to meet the optimal solution.

To address this issue, there is a need to consider the complementary relationships among various sub-populations. More specifically, the performance of the whole solution should be considered when performing the evolutionary process. To this end, the fitness assignment is proposed for the PSC that can consider the complementary relationships among various individuals. Like the traditional evolutionary algorithm (Gomez, 2003; Lin & Hsu, 2007; Lin & Xu, 2007; Smith et al., 1993), the proposed fitness assignment also compute the fitness value of an individual by summing up the fitness values of all the possible combinations which contains the individual. However, for avoiding aforementioned problem, the proposed fitness assignment considers the combination of performance when summing up the fitness values. More specifically, the performance of the whole solution is also considered when evaluating the performance of each individual. By doing so, the performance of each individual can really be used to represent the performance of the whole solution. Thus, the PSC can increase the chances to meet the optimal solution. The details of the proposed fitness assignment are described as follows.

Step 1: First of all, the fitness assignment randomly chooses *R* fuzzy rules from the *R* sub-populations to construct a TNFN. The aforementioned step is repeated for *SelectionTimes* times, which is predefined by using triad-and-error testing. The following equation presents how to form the TNFNs.

$$\begin{aligned}&TNFN_i = \{Ind_{1Sel_1}, Ind_{2Sel_2}, \ldots Ind_{RSel_R}\},\\ &i = 1, 2, \ldots, SelectionTimes,\\ &Sel_j = Random[1, SP_{size}],\\ &j = 1, 2, \ldots R,\end{aligned} \tag{7}$$

where

$TNFN_i$ reprents the *i*th generated TNFN.
Ind_{1Sel_1} reprents the individual the selects to formthe the TNFN.
Sel_j means the selected index of the individual in the *j*th sub-population.
SP_{size} represent the number of individuals in a sub-population.
*Selection*Times represets the total number of selected TNFNs in each generation.

Step 2: The fitness assignment evaluates each TNFN, which is generated from step1, to obtain a fitness value. The fitness value is mainly used to represent the performance of each TNFN. In other words, it is the main process of the evolution because the fitness value plays an important role to decide whether the optimal solution is found. The well design fitness value can help the individuals be evaluated efficiently and vice versa. In this study, the most well-known root mean square (RMS) errors Veretelnikova, 2004 are used to evaluate the performance of the TNFN since it can robust reflect the performance of the models. The following equation presents the fitness function designed in this study.

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

where x_i and x_i' represent the *i*th output of TNFN and the *i*th desired output respectively. As shown in Eq. (8), the high *FitnessValue* means that the outputs of TNFN are close to the desired outputs and vice versa.

Step 3: After obtaining the fitness value of each selected TNFN, the fitness assignment then computes the fitness value of each individual that contains in the TNFN. More specifically, the fitness value obtained from step 2 is divided by the number of the fuzzy rules (i. e. *R*). After that, the divided fitness value is accumulated to the selected individuals. To consider the complementary relationships among the various individuals, this study considers the performance of the complete solution when accumulating the fitness value to the selected individuals. More specifically, the fitness value of each individual is computed as Eq. (9). It is mainly used to prevent the problem that a set of individuals with the highest performance may cause the complete solution to meet the bad performance. By doing so, the best combination of the individuals will be kept.

```
if $Fitness\ Value_{Ind_{ij}} = Best\ Fitness\ Value/R$
  if $Fitness\ Value/R <= Best\ Fitness\ Value/R$
    $Fitness\ Value_{Ind_{ij}} = Fitness\ Value_{Ind_{ij}}$
  else                                                        (9)
  $Fitness\ Value_{Ind_{ij}} = Fitness\ Value_{Ind_{ij}} + Fitness\ Value/R$
else
    $Fitness\ Value_{Ind_{ij}} = Fitness\ Value_{Ind_{ij}} + Fitness\ Value/R$
```

where

Fitness $Value_{Ind_{ij}}$ is the fitness value of the *j*th individual in *i*th sub-population.
Best Fitness Value represents the best fitness value in current excited generations.

Step 4: In the last step, the accumulated fitness value of each individual will be further divided by the number of times it has been selected. After that, the average fitness value represents the performance of an individual. The following equation presents how to compute the average fitness value.

$$\begin{aligned}&Fitness\ Value_{Ind_{ij}} = Fitness\ Value_{Ind_{ij}}/Select\ Times_{Ind_{ij}}\\ &where\\ &i = 1, 2, \ldots R,\\ &j = 1, 2, \ldots SP_{size}.\\ &Select\ Times_{Ind_{ij}} \text{means the number of times} Ind_{ij} \text{has been selected.}\end{aligned} \tag{10}$$

In summary, the PSC can contribute to consider the specialization and the complementary relationships when performing the aforementioned steps. More specifically, it can take into account specializations of the individual and population. Moreover, the PSC can also consider the complementary relationships among various sub-populations to let the set of individuals with the highest performance avoid meeting the bad performance. By doing so, the PSC can not only keep the performance in each generation but also consider the specializations of both individuals and population to increase the chances to meet the optimal solution.

3.2.1.2. Self-adaptive evolutionary algorithm (SEA). In addition to the PSC, the concept of the other proposed method (i.e. SEA) is also introduced. The main purpose of the SEA is to decide the suitable searching space when performing the evolutionary process. More specifically, the evolutionary steps, which related to seeking the solutions, will need to consider adjusting the search space according to the performance of each individual. In other words, the SEA is applied to a crossover and mutation step. The details of these two components are shown below.

3.2.1.2.1Crossover. The major task of the crossover is to exchange the genes between two parents to generate the offspring (Vereteln, 2004). By doing so, the new combination of the individuals in each sub-population will be generated in the next generation. Generally, the individuals in the top help of each sub-population will be used to perform the crossover to generate the individuals in the other half of each sub-population. It implies that such evolutionary step can seek the different combination of the individuals in each sub-population that perform well to find the optimal solution.

Although the crossover can bring aforementioned benefits, only the single criterion is used to perform the crossover. More specifically, in the traditional evolutionary algorithms, the crossover only tends to exchange the genes that selected randomly (Fogel, 1994; Gomez, 2003; Holland, 1992; Karr, 1991; Koza, 1992; Lin & Hsu, 2007; Lin & Jou, 2000; Lin & Xu, 2007; Rechenberg, 1994; Smith et al., 1993). In other words, there is no idea to consider the fitness value to provide different actions used to generate the different combination of the chromosomes. More specifically, using a random process to generate the searching range of genes may be difficult to find the optimal solution especially when the solution is near to the optimal solution. For instance, if the individuals are near to the optimal solution, it needs to search the optimal solution in a small searching range. Conversely, if the individuals are far away from the optimal solution, it needs to search the optimal solution in a large searching range. To this end, there is a need to provide multiple criteria based on the fitness value when performing the crossover. By doing so, the evolutionary step can increase the chances to find the optimal solution.

To this end, the proposed SEA applies the multiple criteria to perform the crossover. The multiple criteria are mainly used to consider the fitness value when performing the crossover. The main concept of the SEA is to consider the fitness value to provide different actions to perform the crossover. More specifically, the SEA for crossover uses the fitness value of parent individuals to determine the suitable number of genes used to exchanging the genes to generate offspring in the next generation. By doing so, the parents with low fitness value may seek the optimal solution by exchanging many genes. Conversely, the parents with the high fitness value may seek the optimal solution by exchanging a few genes. The details of the SEA for crossover are shown in the following equations.

$$CrossoverSite_1 = Rand(IndLength) \tag{11}$$

$$MaxCroPoints = chrLength * (MaxFitness\,value/R - FitnessValue_{Ind_{ij}}) \tag{12}$$

$$\begin{aligned} &CrossoverSite_2 = CrossoverSite_1 + Rand(\pm MaxCroPoints) \\ &CrossoverSite_2 \in [1, IndLength] \end{aligned} \tag{13}$$

where

$$\begin{aligned} &CrossoverSite_1 \text{ is the first crossover site used to perform the SEA} \\ &CrossoverSite_2 \text{ is the second crossover site used to perform the SEA.} \\ &MaxFitness\,value \text{ is the maximal value of fitness value} \\ &(\text{In this study, the } MaxFitness\,value \text{ is } 1.00). \\ &IndLength \text{ is the length of a individual.} \\ &MaxCroPoints \text{ is the range used to decide the } CrossoverSite_2. \end{aligned} \tag{14}$$

As shown in Eq. (12), the relationships between *MaxCroPoints* and $FitnessValue_{Ind_{ij}}$ are in inverse. It implies that the SEA for crossover can exchange *many genes* of the two parent individuals with low performance. In contrast, it can also exchange *a few genes* of the two parent individuals with high performance (see Eqs. (11)–(14)). Moreover, for exchanging the genes between two parents, the two-point crossover, which is mainly used to exchange the genes between $CrossoverSite_1$ and $CrossoverSite_2$ form the two parents, is adopted in this study (Lin & Yao, 1997). The two-point crossover is shown in Fig. 4. After doing so, each sub-population could robust produce the different combination of the individuals according to their fitness value to seek the optimal solution when performing the SEA for crossover.

3.2.1.2.2. Mutation. Although the crossover can generate the different combinations of the individuals by exchanging the existing genes, no any new value is considered for seeking the optimal solution. In other words, the evolution may only depend on the initial value of the individuals and may difficultly find the optimal solution. To face this issue, there is a need to attend new genes in existing individuals (chromosomes) to extend the searching space for seeking the optimal solution. The mutation is the way to achieve such purpose. More specifically, it is mainly used to generate new gene to replace the existing gene in the individuals (Lee, Bai, & Chen, 2008). By doing so, the individuals can extend their searching space to increase the chance to meet the optimal solution.

Like the crossover, the SEA can also be applied to the mutation, which is used to consider the fitness value when performing the mutation, to provide an efficiency way to perform the mutation. The main concept of the SEA for mutation is to provide the multiple criteria to adjust the searching space. More specifically, it can consider the fitness value of each individual to determine the range values used to update the mutation genes. By doing so, the searching space can have a self-adaptive ability. For example, the individuals with low fitness value may seek the optimal solution by updating a mutation point by using the large value range and vice versa. The details of the SEA for mutation are shown in the following equations.

$$Fitness\ Range = Max\ Fitness\ Value/R - Fitness\ Value_{Ind_{ij}} \tag{15}$$

$$MutSite = Rand(IndLength) \tag{16}$$

$$Ind_{MutSite} = Ind_{MutSite} + MutValue \tag{17}$$

$$MutValue = Rand(\pm Range\ Value \times Fitness\ Range) \tag{18}$$

Where

$$\begin{aligned} &MutSite \text{ is the mutation site of the mutated individual.} \\ &Ind_{MutSite} \text{ is the MutSite th gene of the mutated chromosome.} \\ &RangeValue \text{ is the predefined value related to the input space.} \end{aligned} \tag{19}$$

The aforementioned equations indicate that the relationship between *MutValue* and $Fitness\ Value_{Ind_{ij}}$ are in inverse (see Eqs. (15)–(19)). It implies that the SEA for mutation can rough tune the selected gene of the individual with low performance while it can fine tune the selected gene of the individual with high performance. By doing so, the SEA can robust adjust the parameters of the TNFN when performing the mutation.

In summary, the proposed SEA can contribute to consider the multiple criteria to evaluate the individuals in each sub-population. More specifically, it can take into account such criteria to perform the crossover and mutation. By doing so, the evolutionary steps cannot only seek in a large searching space when the solution is far away from the optimal solution but also seek in a small searching space when the solution is near to the optimal

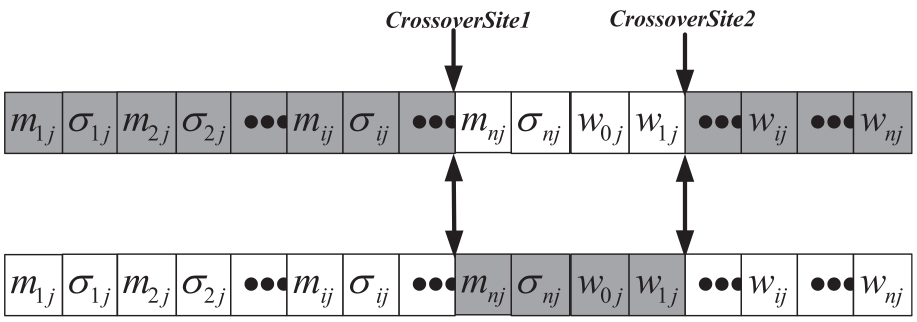

CrossoverSite1
CrossoverSite2
m_{1j} σ_{1j} m_{2j} σ_{2j} ●●● m_{ij} σ_{ij} ●●● m_{nj} σ_{nj} w_{0j} w_{1j} ●●● w_{ij} ●●● w_{nj}

Fig. 4. The two-point crossover.

solution. Thus, the SEA can provide a robust way to evolutes the sub-populations.

3.2.2. Learning procedure

After introducing the components of both the PSC and SEA, the learning procedure of the PSC-SEA is introduced in this section. As shown in Fig. 5, the procedure consists of eight steps. Each step is shown below:

P1. In this step, the PSC is applied to generate the initial sub-populations. Each initial sub-population (see Fig. 2), which consists of several individuals, is generated based on the coding schema (see Fig. 3). Each gene of an individual is generated randomly according the predefined *RangeValue*.

P2. The PSC then perform the *Fitness Assignment* to evaluate the performance of each individual in each sub-population. In other words, Eqs. (7)–(10) are used to compute the fitness value of each individual in each sub-population.

P3. After that, the proposed PSC then judges whether the evolutionary process is finished based on a predefined criterion. Generally, the criterion is defined according to a predefined generation times or the predefined desired fitness value (Fogel, 1994; Gomez, 2003; Holland, 1992; Karr, 1991; Koza, 1992; Lin & Xu, 2007; Rechenberg, 1994; Smith et al., 1993). In this study, the predefined generation times is used to judge whether the evolutionary process is finished.

P4. If the evolutionary process is not finished yet, the proposed PSC then perform the *reproduction* to keep the well-perform individuals in the top half of each sub-population. The concept of the reproduction is to sum up the fitness values of the top half of chromosomes in each sub-population and then compute the fitness ratio of each individual based on the summation fitness values. After doing so, the individuals are reproduced according to the fitness ratio. In this study, the roulette-wheel selection is used to select the reproduced individuals since it provides more robust way than other selection methods (Zou, Mi, & Xu, 2006).

P5. After performing the *reproduction*, the SEA then performs the crossover step. More specifically, a rate will be generated randomly to judge whether the crossover is performed based on the predefined *CrossoverRate*. If the crossover needs to be done, the proposed evolutionary algorithm will continue to perform the P 6. Otherwise, the evolutionary algorithm will go to the P7.

P6. If the random rate is greater than *CrossoverRate*, the SEA for crossover is then performed. The parents, which are used to perform the SEA for crossover, are chosen randomly from the top half of each sub-population. After doing so, the SEA for crossover is performed based on the selected parents (see Eqs. (11)–(14)).

P7. After that, the SEA then performs the mutation step. Likewise, a rate will be generated randomly to judge whether the mutation is performed based on the predefined *MutationRate*. If the mutation needs to be done, the evolutionary process will go to the P 8. Otherwise, the evolutionary process will go to the P 2 until the predefined criterion is reached.

P8. The SEA for mutation is performed in this step. More specifically, the individual is selected from the new generated individuals randomly in each sub-population. After doing so, the SEA is performed based on the selected individual (see Eqs. (15)–(19)). After that, the evolutionary process then goes to the P2.

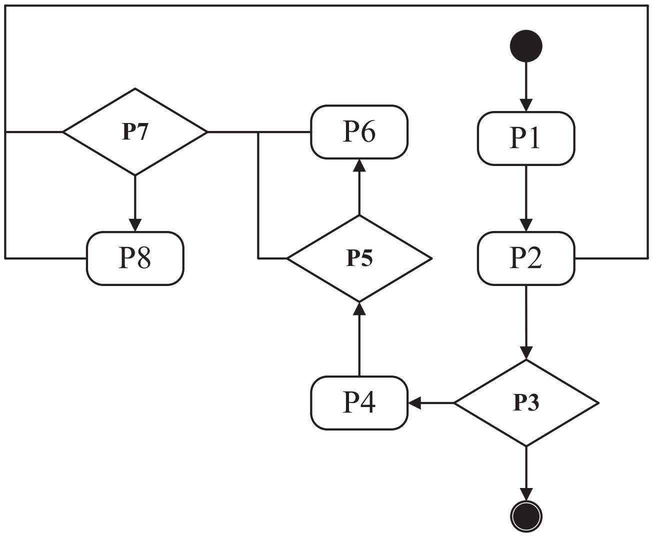


Fig. 5. Procedures of the PSC-SEA.

4. Illustrative examples

To verify the performance of the proposed PSC-SEA, two simulations are discussed in this section. Regarding the first simulation, the complex identification application given by Narendra and Parthasarathy (Juang & Lin, 1998) is applied to not only investigate the benefits of the PSC-SEA but also demonstrate the performance of each component. Regarding the other example, a real world application is applied to investigate the performance of the proposed PSC-SEA. More specifically, the proposed PSC-SEA is applied to forecast the number of sunspots (Lin & Xu, 2006) to verify the robustness and efficiency of the proposed PSC-SEA in real world application.

4.1. The Identification of a nonlinear dynamic system

The nonlinear dynamic system given by Narendra and Parthasarathy (Juang & Lin, 1998) is used to not only investigate the performance of the proposed PSC-SEA but also verify the performance of each component. It is due to the fact that such system is easy to implement and obtain the well performance. Thus, it can be use as a benchmark to verify the benefits of the PSC-SEA. To this end, the nonlinear dynamic system should be identified firstly. The following equation describes the function related to the system.

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

According to the Eq. (20), the output of the nonlinear dynamic system is nonlinearly influenced by the past state of the output and input value. Generally, the training patterns were generated by setting the input values to $\sin(2\pi k/25)$ (i.e. $u(k) = \sin(2\pi k/25)$). Moreover, the first 100 pairs (i.e. $y(1)$ to $y(100)$) are used as the training patterns to model the TNFN that suits for identifying such system while the other 100 pairs (from $y(101)$ to $x(200)$) are used as the testing patterns to validate the performance of the well-trained TNFN.

In addition, there is also an issue to determine the predefined parameters of the PSC-SEA. To address such issue, a parameter exploration proposed by De Jong (De Jong, 1975) is adopted to determine the suitable predefined parameters since it is suitable for the smaller dataset. To this end, the parameter exploration was applied to decide the predefined parameters of the proposed PSC-SEA. More specifically, it uses the different ranges of the values to evaluate the performance of the PSC-SEA to judge whether the values are suitable. For instance, the number of fuzzy rules has the range from 3 to 10 in increments of 1, the number of individuals in a sub-population (i.e. SP_{size} in Eq. (7)) has the range from 10 to 50 in increments of 5, the crossover rate has the range from 0.20 to 0.80 in increments of 0.05 and the mutation rate has the range from 0.0 to 0.4 in exponential increments. The other parameters of the PSC-SEA are defined as the same way. After performing the parameter exploration, the parameters of the proposed PSC-SEA are defined in Table 2.

After deciding the parameters, the proposed PSC-SEA is then used to perform the evolutionary procedure (see Fig. 5). The simulation was carried out for 15 runs and each run starts with the same initial parameters. Such simulation can provide the reliable evidence about the performance of the proposed PSC-SEA.

After performing the simulation, the learning curves of 15 runs are shown in Fig. 6. To easy identify the performance of the proposed PSC-SEA, each learning curve is represented by using the RMS error (Vereteln, 2004) instead of the fitness value. The RMS error (Vereteln, 2004) is showed in the following equation.

$$RMSerror = \sqrt{\frac{\sum_{i=1}^{n} |x_i - x_i'|^2}{n}} \tag{21}$$

As shown in Fig. 6, each learning curve of the proposed PSC-SEA can reach the low RMS error. It implies that the proposed PSC-SEA can obtain good performance in 15 runs. To further verify the performance of the proposed PSC-SEA, a tradition genetic algorithm (GA) Karr, 1991 and evaluated sub-population (ESP) Gomez, 2003, which is related to the application of the particular solution consideration, are used to compare the performance with the PSC-SEA.

Table 2
The predefined parameters of the PSC-SEA.

Parameters	Value
The number of rules	5
The number of individuals in a sub-population	20
The number of generation times	100
Selection times	30
RangeValue for weight	[5.000, −5.000]
RangeValue for membership functions	[2.000, -2.000]
Mutation rate	0.3
Crossover rate	0.5

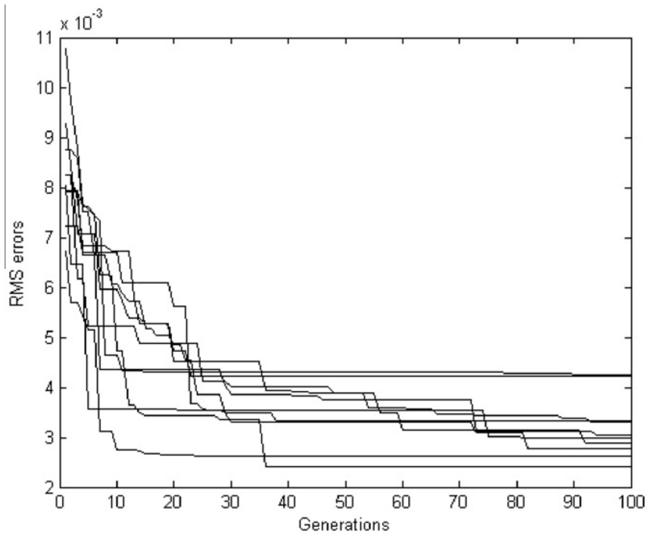


Fig. 6. The learning curves of the proposed method.

Like the PSC-SEA, the parameters of the GA and ESP are defined by using the parameter exploration. Moreover, such two algorithms are used to perform the simulation for 15 runs. The learning curves of the GA and ESP are shown in Figs. 7 and 8. Comparing to Fig. 6, the proposed PSC-SEA can dramatically outperform than the GA and ESP. The reason may be because that the proposed PSC-SEA can take into account both specializations and suitable searching space to increase the chances to meet the optimal solution. By doing so, the PSC-SEA can generate efficiency candidate solutions. Thus, the PSC-SEA can obtain better performance than others (GA and ESP). Moreover, the learning curves in Fig. 6 are smoother than those of Figs. 7 and 8. It also implies that the PSC-SEA can improve to search the better solution in each generation by using the proposed PSC and SEA.

For further demonstrating such argument, the testing patterns are used to evaluate the performance of the proposed PSC-SEA, GA and ESP. More specifically, regarding the PSC-SEA, the testing result (see Fig. 9(a)) and testing errors (see Fig. 9(a)) between the desired output and the outputs obtained by the well-trained TNFN are shown in Fig. 9. Regarding the GA and ESP, the testing results and errors are show in Figs. 10 and 11. As shown in Figs. 9–11, we can further verify that the PSC-SEA can outperform than the

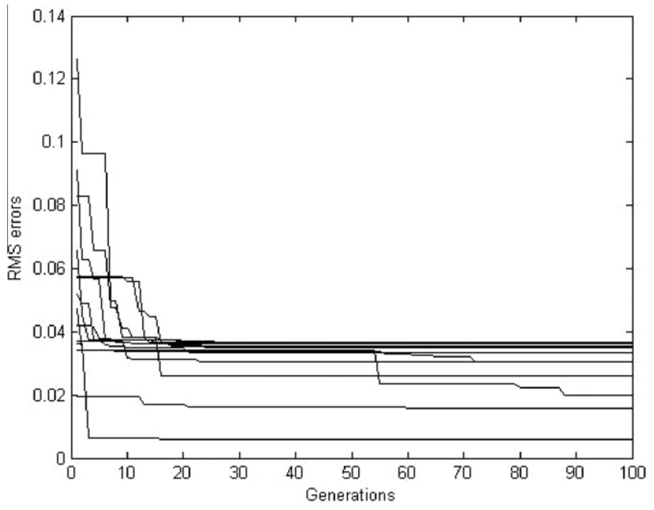


Fig. 7. The learning curves of the traditional GA (Karr, 1991).

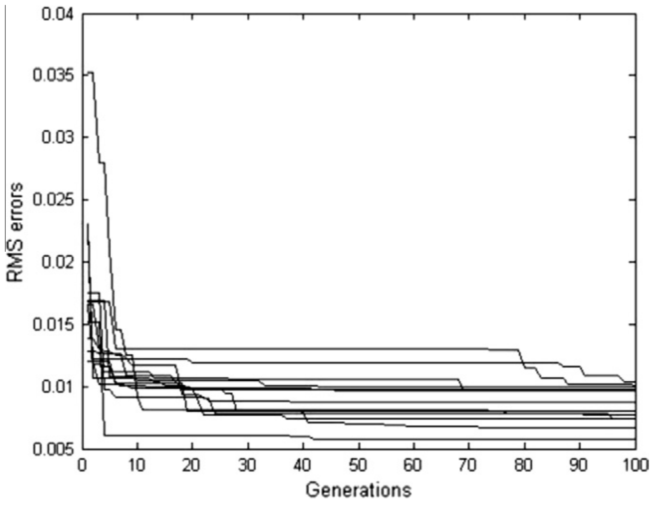


Fig. 8. The learning curves of the ESP (Gomez, 2003).

GA and ESP. It implies that the proposed PSC-SEA can efficiently and robustly adjust the parameters of the TNFN than the traditional GA (Karr, 1991) and ESP (Gomez, 2003).

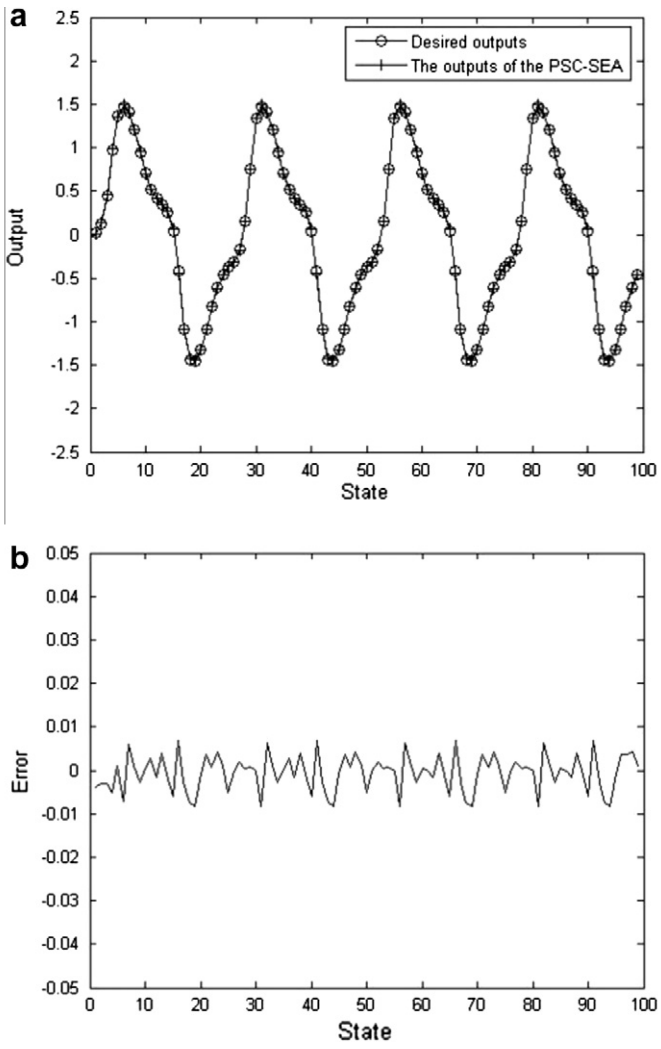


Fig. 9. The testing results (a) and errors (b) of the PSC-SEA.

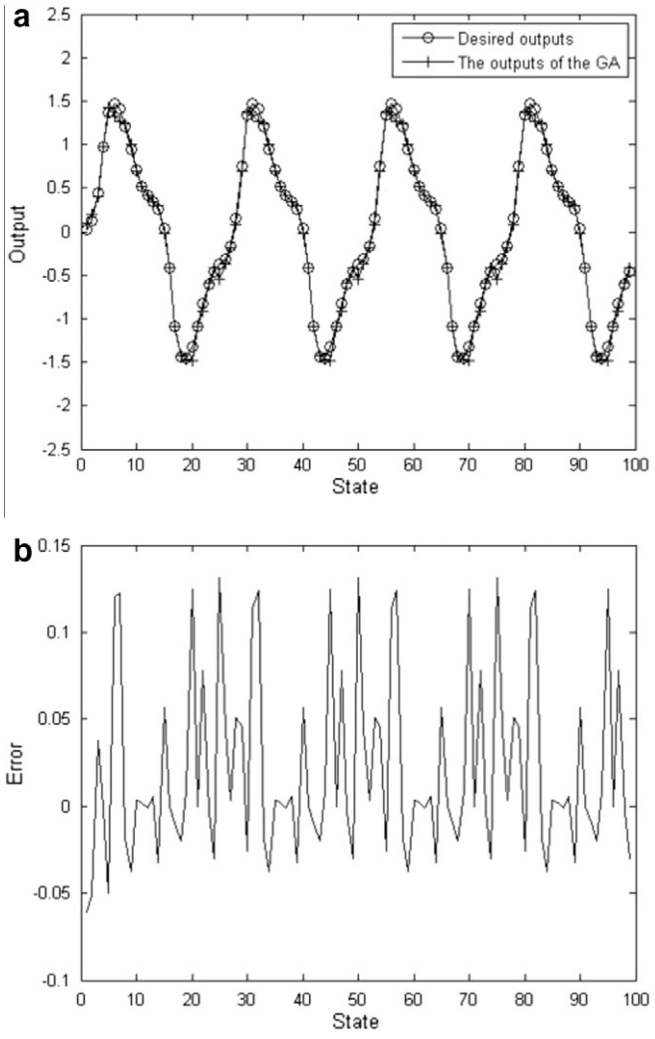
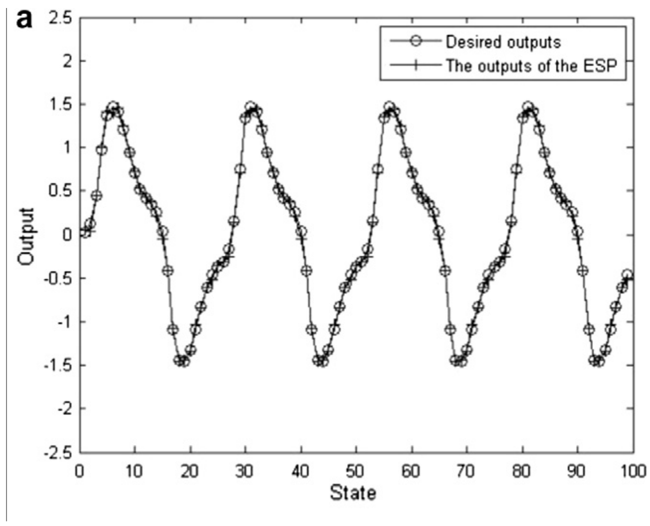


Fig. 10. The testing results (a) and errors (b) of the traditional GA.

In addition, for providing reliable evidence about the performance of the PSC-SEA, various existing well-known genetic algorithm mentioned in Section 2, including (Gomez, 2003; Karr, 1991; Lin & Hsu, 2007; Lin & Xu, 2007; Smith et al., 1993), are also applied to compare the performance with the proposed PSC-SEA. Likewise, the parameters of these algorithms are also defined by using the parameter exploration. Moreover, each algorithm is used to perform the simulation for 15 runs. The performance, which consists of mean and standard deviation of the RMS errors and CPU Time, of aforementioned algorithms in both training and testing results are shown in Tables 3. The PSC-SEA can outperform than other algorithms (Gomez, 2003; Karr, 1991; Lin & Hsu, 2007; Lin & Xu, 2007; Smith et al., 1993). In other words, the PSC-SEA cannot only spend smaller CPU time but also obtain lower RMS errors than other algorithms in both training and testing results.

In brief, the proposed PSC-SEA can demonstrate high performance via the both PSC and SEA. More specifically, the PSC can take into account not only specializations of both individuals and population but also the complementary relationships to prevent the evolutionary algorithm meeting the sub-optimal solution. Moreover, the SEA can used multiple criteria to adjust the searching space to let the evolutionary process can seek the solution in a suitable searching space. Thus, the PSC-SEA can obtain the better performance than other existing evolutionary algorithms.

Although aforementioned results can demonstrate the performance of the proposed PSC-SEA, only the complete PSC-SEA is



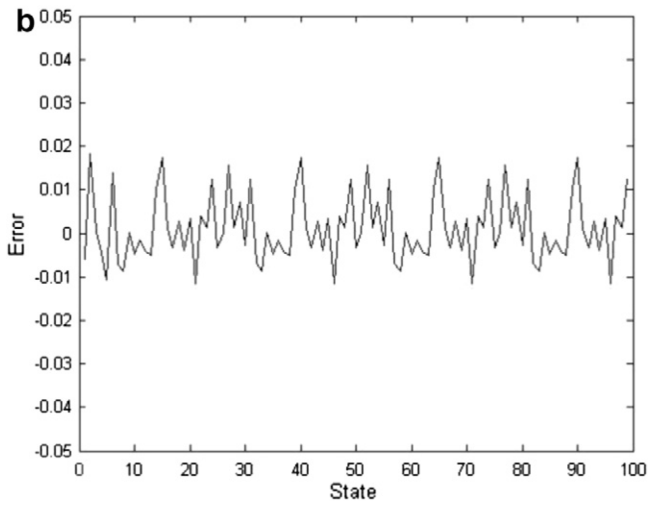


Fig. 11. The testing results (a) and errors (b) of the ESP.

evaluated. In other words, each proposed component of the both PSC and SEA cannot be evaluated independently. Therefore, we cannot demonstrate the contribution of each proposed component. To this end, in this example, we also tend to investigate the performance of each proposed component.

As shown in Table 4, the five types of models, including the PSC-SEA, PSC, SEA, PSC-SEA-I and PSC-SEA-II are used to evaluate the training and testing performance. Regarding the PSC, instead of the proposed SEA in the PSC-SEA, it adopts the traditional two-point crossover (Lin & Yao, 1997) and mutation (Lee et al., 2008) to generate the offspring. Regarding the SEA, instead of the proposed PSC, it adopts the traditional GA to perform the SEA. In other words, the SEA only considers the complete solution when performing the evolutionary process. Regarding the PSC-SEA-I, instead of the SEA for crossover, it adopts the traditional two-point crossover to generate the offspring. In other words, the proposed SEA is only applied to perform the mutation. Regarding the PSC-SEA-II, the traditional unit mutation (Lee et al., 2008) is used to mutate the unit gene of the single individual. Thus, the proposed SEA is only applied to perform the crossover.

Table 4 indicates that the SEA outperform than the PSC. In other words, the SEA can obtain better performance than the PSC. It implies that the suitable searching space is an important issue when developing the evolutionary algorithm. Thus, even if the proposed PSC can take into account not only the specializations of the individuals and population but also the complementary relationships among various particular solutions, it still needs a suitable searching space to perform the evolutionary procedure. It is due to the fact that the searching space may influence the converging efficiency when performing the evolutionary process so the SEA can improve the efficiency of the PSC.

Besides, the PSC-SEA-I outperform than the PSC-SEA-II. In other words, the SEA for crossover can obtain better performance than the SEA for mutation. It may be because that the crossover is the main evolutionary procedure for finding the optimal solution (Pitangui & Zaverucha, 2008) so the modifications in the crossover step can bring much more benefits than those in the mutation step.

However, as shown in Table 4, we can further see that the proposed PSC-SEA, which combines both the PSC and complete SEA, can obtain the best performance among five types of the models. Thus, we can demonstrate that each component of the PSC-SEA is necessary used to improve the performance. In summary, the components of the PSC-SEA are complementary to each other to increase the chances to meet the optimal solution. Thus, the PSC-SAE can be successfully applied to adjust the parameters of the TNFNs.

Table 3
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)
PSC-SEA	0.0037	0.0015	0.0043	0.0018	2.76	0.21
Lin and Xu (2007)	0.0074	0.0034	0.0087	0.0039	13.87	2.10
Gomez (2003)	0.0082	0.0026	0.0096	0.0029	4.18	1.25
Smith et al. (1993)	0.012	0.007	0.015	0.008	13.73	3.19
Lin and Hsu (2007)	0.057	0.009	0.041	0.011	14.61	2.92
Karr (1991)	0.064	0.012	0.054	0.017	11.52	3.74

Table 4
The training and testing performance of different components.

Method	RMS errors (training)		RMS errors (testing)		CPU time (training)	
	Mean	Deviation	Mean	Deviation (second)	Mean	Deviation (second)
PSC-SEA	0.0037	0.0015	0.0043	0.0018	2.76	0.21
PSC	0.0063	0.0024	0.0071	0.0027	2.42	0.22
SEA	0.0053	0.0018	0.0059	0.0020	1.87	0.18
PSC-SEA-I	0.0045	0.0020	0.0049	0.0022	2.69	0.23
PSC-SEA-II	0.0049	0.0021	0.0056	0.0024	2.78	0.26

4.2. Forecasting the number of sunspot

Since the contributions of the PSC-SEA can be demonstrated in the Section 4.1, the example is too simple to demonstrate the robustness and efficiency of the PSC-SEA. Moreover, Section 4.1 only applied the PSC-SEA to the application related to function identification. In other words, how the PSC-SEA work in real application cannot be evaluated. To this end, in this section, the real world application is adopted to evaluate the efficiency and robustness of the proposed PSC-SEA. More specifically, the PSC-SEA is applied to forecast the number of sunspots (Lin & Xu, 2006).

As showed in previous studies, the sunspot prediction is a difficult task since it displays the distribution of nonlinear, nonstationary and non-Gaussian cycles (from 1700 to 2004) Lin & Xu, 2006. Thus, it is suitable for applying to evaluate the performance of the PSC-SEA in real application. To this end, we apply the proposed PSC-SEA to forecast the number of sunspots. The following equation describes the function related to the input and out pairs of the sunspots forecasting.

$$\begin{aligned}
&\textit{Input Pattern} = [x_1(t), x_2(t), x_3(t)],\\
&\textit{Output Pattern} = y(t)\\
&\text{where}\\
&x_i(t) = y(t-i),\\
&t \text{ represents the year and } y(t) \text{ is the number of sunspots at the } t \text{ th year.}
\end{aligned} \tag{22}$$

According to the Eq. (22), each training pattern consists of input and output patterns. Regarding the input pattern, it consists of three past values of the number of sunspots. Regarding the output pattern (i.e. $y(t)$), it represents the number of sunspots at tth year. Generally, the first 180 years (from 1705 to 1884) of the sunspots data are used as the training patterns while the remaining 120 years (from 1885 to 2004) of the sunspot numbers were used as testing patterns to evaluate the performance of proposed PSC-SEA.

Likewise, the RMS error is also used to evaluate the performance of the PSC-SEA. Moreover, the parameter exploration was also applied to decide the predefined parameters of the proposed PSC-SEA (see Table 5).The simulation was carried out for 15 runs and each run starts with the same initial parameters.

The learning curves of the PSC-SEA are shown in Fig. 12. It verifies that the proposed PSC-SEA can still keep reaching the good performance when solving such complex real application. Beside, for further investigating such argument, the testing results of the PSC-SEA (Fig. 13) are also investigated. The testing results indicate that the outputs of the PSC-SEA can accurately match the desired outputs of the sunspots (see Fig. 13(b)). It implies that the PSC-SEA can robustly and efficiently obtain well performance when solving the different or even complex application.

In addition, for providing reliable evidence about the aforementioned argument, other existing well-known evolutionary algorithms (i.e. Gomez, 2003; Karr, 1991; Lin & Hsu, 2007; Lin & Xu, 2007; Smith et al., 1993 are also applied to compare the performance with the proposed PSC-SEA. The performances, which include mean and standard deviation value of RMS errors and CPU Time, the training error of the training patterns and the forecasting

Table 5
The predefined parameters of the PSC-SEA.

Parameters	Value
The number of rules	6
The number of individuals in a sub-population	30
The number of generation times	300
SelectionTimes	50
RangeValue for weight	[5.000,0.000]
RangeValue for membership functions	[1.000,0.000]
Mutation rate	0.2
Crossover rate	0.4

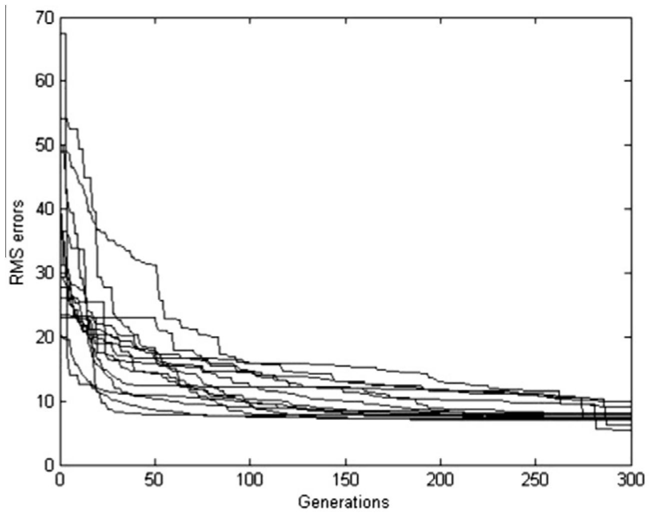


Fig. 12. The learning curves of the proposed PSC-SEA.

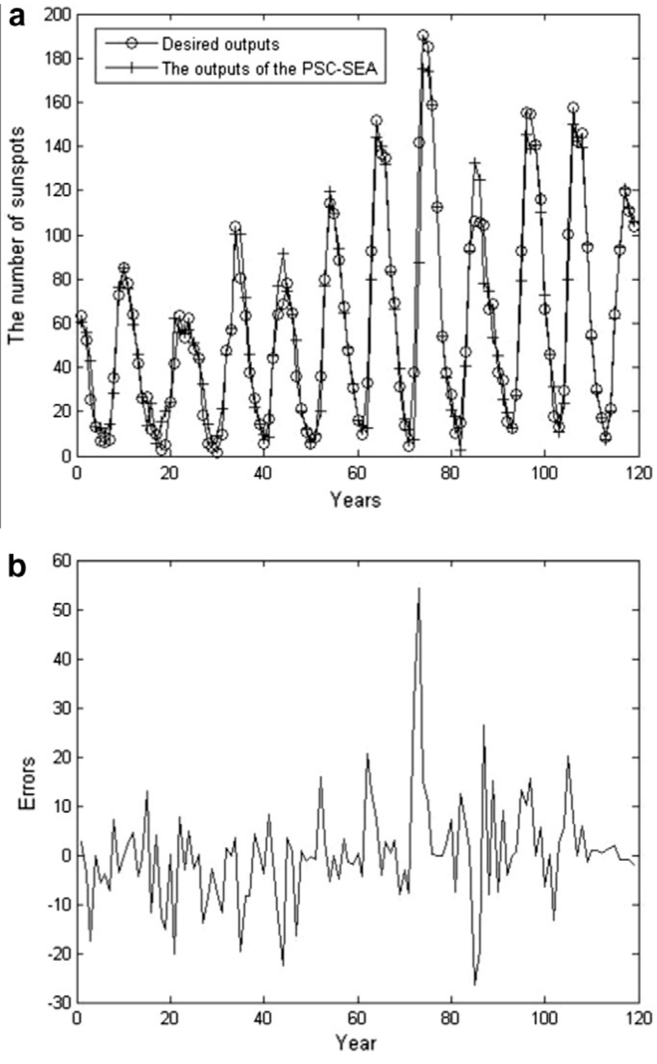


Fig. 13. The testing prediction results (a) and errors (b) of the PSC-SEA.

Table 6
The performance comparison of various existing models.

Method	Training						Testing			
	RMS errors		Training error		CPU Time		RMS errors		Testing error	
	Mean	Dev.	Mean	Dev.	Mean	Dev.	Mean	Dev.	Mean	Dev.
PSC-SEA	8.78	1.24	6.25	1.24	38.28	3.15	12.45	1.79	8.58	1.64
Lin and Xu (2007)	15.67	3.72	13.54	2.98	196.15	69.21	18.31	3.53	16.67	3.41
Gomez (2003)	21.71	3.14	18.43	2.57	81.45	7.18	24.54	2.84	21.45	2.69
Smith et al. (1993)	31.29	3.96	26.26	3.14	181.27	45.91	34.25	3.67	29.64	3.52
Lin and Hsu (2007)	35.67	4.16	31.91	3.72	208.91	91.16	39.72	4.91	36.59	4.87
Karr (1991)	43.54	5.94	38.61	4.54	176.82	68.61	48.64	6.53	42.13	6.12

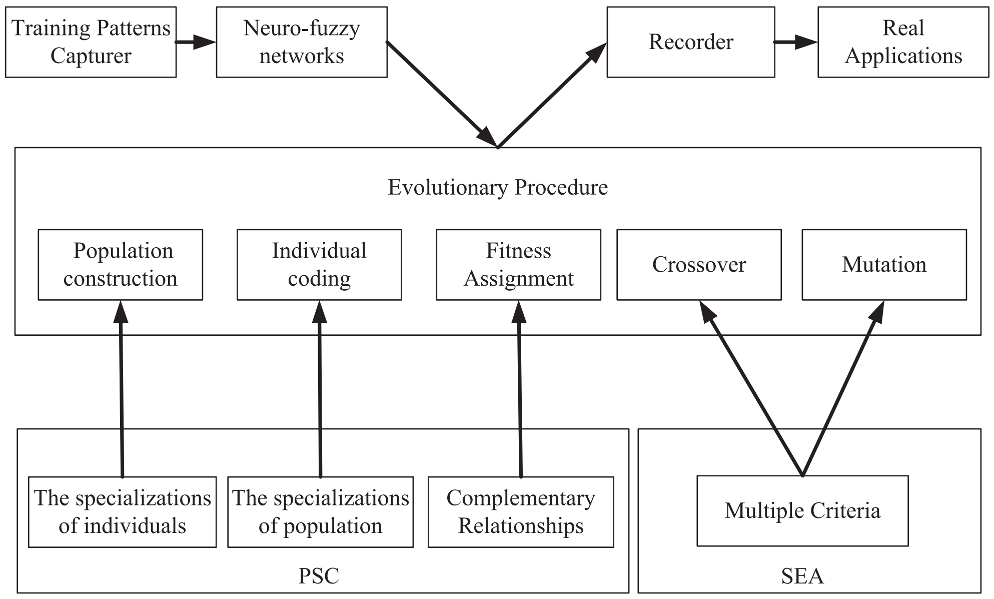


Fig. 14. The framework of the PSC-SEA.

Table 7
Components of the framework.

Part	Components	Purposes
Training patterns capturer		It is mainly used to capture the input and output patterns used to train the neuro-fuzzy networks
Neuro-fuzzy networks		It is used to discuss the relationship between the captured input patterns and outcomes
Evolutionary procedure	Population construction	It adopts the specializations of populations to construct the population
	Individual Coding	It uses the specializations of individuals to code the individuals
	Fitness Assignment	It uses the complementary considerations to design the fitness value of each individual
	Crossover	It is mainly used to apply the SEA to perform the crossover
	Mutation	It is mainly used to apply the SEA to perform the mutation
PSC	Specializations of Individuals	Each individual represents only particular solution. It can increase the chances to meet the optimal solution
	Specializations of population	Each sub-population is used to evaluate each particular solution independently. It can prevent building the complete solution from similar individuals
	Complementary relationships	It is used to keep the best combination of the individuals
SEA	Multiple criteria	It considers multiple criteria to adjust the searching space when performing the crossover and mutation
Recorder		It is used to store the best solution (e.g. the best combination of the parameters of the TNFN)
Real applications		It is mainly used to apply the recorder to address the real applications

error of the testing patterns, of these compared evolutionary algorithms in both training and testing steps are shown in Tables 6. Again, the PSC-SEA can not only spend smaller CPU time but also obtain better performance (i.e. lower RMS errors, training error and testing error) than other evolutionary algorithms.

In summary, like the results obtained from the Section 4.1, the proposed PSC-SEA can demonstrate high performance than other existing evolutionary algorithms in the real application. It implies that the PSC-SEA can be suitable for solving complex and nonlinear real problems. In other words, the PSC-SEA can robustly and efficiently adjust the parameters of the TNFN to meet the optimal solution of different complex and nonlinear applications. Thus, we can demonstrate that considering both the specializations (i.e. PSC) and suitable searching space (i.e. SEA) can help the evolutionary algorithm obtain the better performance than others.

5. The development of the framework

This paper aims to develop a novel evolutionary algorithm for helping the neuro-fuzzy networks that can meet the optimal solution. As shown in the results, the proposed PSC-SEA is a suitable algorithm for achieving such purpose by considering not only specializations and complementary relationship to prevent the algorithm meet the suboptimal solution but also multiple criteria to decide the suitable searching space. In addition, the results also indicate that the PSC-SEA can contribute to provide robust way to obtain the better performance than existing evolutionary algorithms. Moreover, the results also demonstrate that the PSC-SEA can successfully apply to complex real application. In brief, the PSC-SEA can be used to reach the aims of this study. Based on the results, the framework is developed for implementing the evolutionary algorithm that can apply the concept of partial solutions and multiple criteria to increase the chance to meet the optimal solution when constructing the neuro-fuzzy networks (Fig. 14). The details of the framework are showed below.

As shown in Fig. 14, the framework consists of seven parts such as a training patterns capturer, neuro-fuzzy networks, evolutionary procedure, partial solution consideration (PSC), self-adaptive evolutionary algorithm (SEA), recorder and real applications. Each part is described in Table 7.

In summary, the framework is helpful to develop an evolutionary algorithm that can help the neuro-fuzzy networks to meet the optimal solution. More specifically, it can consider both specialization and complementary relationships to prevent the evolutionary algorithm converging to the sub-optimal solution. Moreover, it can use multiple criteria to automatically adjust the searching space to let the evolutionary algorithm seek the optimal solution in a suitable searching space. By doing so, the performance of the evolutionary algorithm can be improved. Thus, the framework can be tread as a benchmark when developing the evolutionary algorithm that can increase the chances to meet the optimal solution.

6. Conclusion

This study aims to develop the novel evolutionary algorithm (PSC-SEA) to adjust the parameters of the TSK-type neuro-fuzzy network (TNFN). More specifically, regarding the PSC, it cannot only consider both specializations of the individuals and population but also take into account the complementary relationships among various particular solutions. Regarding the SEA, it can consider multiple criteria to adjust the searching space (i.e. the number crossover points and the range of mutated value) for improving the performance of evolutionary algorithm. As shown in the results, the PSC-SEA can efficiently and robustly adjust the parameters of the TNFN to meet the good performance. Moreover, the performance of each component of the PSC-SEA can also be verified in our results. In brief, the components of the PSC-SEA are complementary to each to increase the chances to meet the optimal solution. Based on the results, the framework is proposed for developing a novel evolutionary algorithm that can increase the chance to meet the optimal solution by considering the both specializations of complete solution and suitable searching space.

Although the PSC-SEA can bring several benefits, there are still some limitations that need to be addressed in the future study. More specifically, some parameters of the proposed SPC-SEA such as crossover rate, mutation rate, population size, generation times, number of fuzzy rules and value ranges, need to be defined before performing the PSC-SEA. Thus, there is a need to provide an automatic and robust way to define these parameters. In addition, in this study, the PSC-SEA is only used to demonstrate the performance in the applications related to identification and prediction. Thus, the future works should pay more attention to applying the proposed PSC-SEA to other real applications (e.g. stock prediction (Lee, 2004), image processing (Lin et al., 2006) and controller (Lin & Xu, 2007) to demonstrate that the proposed framework can be suitable for seeking the optimal solution in various real applications.

References

Baradaran-K, M., Shekofteh, S. K., Toosizadeh, S., Akbarzadeh-T, M. R. (2010). A fuzzy approximator with Gaussian membership functions to estimate a human's head pose. In *Proceedings of the international joint conference on computer and automation, engineering,* (pp. 1154–1158).

De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems. Ph. D. Disseration, The University of Michigan, Ann Arbor, MI.

Fogel, L. J. (1994). Evolutionary programming in perspective: The Top-down View. In J. M. Zurada, R. J. Marks, II, & C. Goldberg (Eds.), *Computational intelligence. Imitating life*. NJ, Piscataway: IEEE Press.

Goldberg, D. E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Gomez, F. J. (2003). Robust Non-linear Control through Neuroevolution, Ph. D. Disseration, The University of Texas at Austin.

Holland, J. H. (1992). *Adaptation in neural and artificial system*. Cambridge, MA, USA: MIT Press.

Hsu, Y. C., Lin, S. F., & Cheng, Y. C (2010). Multi groups cooperation based symbiotic evolution for TSK-type neuro-fuzzy systems design. *Expert Systems with Applications, 37*(7), 5320–5330.

Jang, J. S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems Man and Cybernetics, 23*, 665–685.

Juang, C. F., & Lin, C. T. (1998). An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems, 6*(1), 12–31.

Juang, C. F., & Lin, C. T. (1998). An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions Fuzzy Systems, 6*(1), 12–31.

Juang, C. F., & Lin, C. T. (1999). A recurrent self-organizing neural fuzzy inference network. *IEEE Transactions on Neural Networks, 10*(4), 828–845.

Karr, C.L. (1991). Design of an adaptive fuzzy logic controller using a genetic algorithm. In *Proceeding of the fourth international conference genetic algorithms,* (pp. 450–457).

Koza, J. K. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge: MIT Press, MA.

Lee, R. S. T. (2004). iJADE stock advisor: An intelligent agent based stock prediction system using hybrid RBF recurrent network. *IEEE Transactions on SMC, Part A, 34*(3), 421–428.

Lee, S.F., Bai, X. Y., & Chen, Y.O. 2008. "Automatic Mutation Testing and Simulation on OWL-S Specified Web Services", Joint Conf. on Simulation, Symposium, pp. 149–156.

Lin, G. M., & Yao, X. (1997). Analysing crossover operators by search step size. In *Proceedings of the IEEE joint conference on, evolutionary computation,* (pp. 107–110).

Lin, C. J., Chuang, H. C., & Xu, Y. J. (2006). Face Detection in Color Images Using Efficient Genetic Algorithms. *Optical Engineering, 45*(4). April 2006.

Lin, C. J., Xu, Y. J., & Lee, C. Y. (2005). An efficient genetic algorithm for TSK-type neural fuzzy identifier design. In *Proceedings of the International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems,* June 22–25, Bari, Italy. Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, June, vol. 3533, (pp. 551–553).

Lin, C. J., & Hsu, Y. C. (2007). Reinforcement hybrid evolutionary learning for recurrent wavelet-based neuro-fuzzy systems. *IEEE Transactions on Fuzzy Systems, 15*(4), 729–745.

Lin, C. T., & Jou, C. P. (2000). GA-based fuzzy reinforcement learning for control of a magnetic bearing system. *IEEE Transactions on Systems Man and Cybernetics Part B, 30*(2), 276–289.

Lin, F. J., Lin, C. H., & Shen, P. H. (2001). Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Transactions on Fuzzy Systems, 9*(5), 751–759.

Lin, C. J., & Xu, Y. J. (2006). The design of TSK-type fuzzy controllers using a new hybrid learning approach. *International Journal of Adaptive Control and Signal Processing, 20*, 1–25.

Lin, C. J., & Xu, Y. J. (2006). A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications. *Fuzzy Sets and Systems, 157*(8), 1036–1056.

Lin, C. J., & Xu, Y. J. (2006). A novel genetic reinforcement learning for nonlinear fuzzy control problems. *Neurocomputing, 69*(16–18), 2078–2089.

Lin, C. J., & Xu, Y. J. (2007). Design of neuro-fuzzy systems using a hybrid evolutionary learning algorithm. *Journal of Information Science and Engineering, 23*(2), 463–477.

Lin, C. J., & Xu, Y. J. (2007). A self-constructing neural fuzzy network with dynamic-form symbiotic evolution. *AutoSoft Journal-Intelligent Automation and Soft Computing, 13*(2), 123–137.

Melgarejo, M. A. (2002). Modified center average defuzzifier for improving the inverted pendulum dynamics. *Proc. IEEE Int. Conf. Fuzzy Systems, 1*, 460–463.

Mitra, S., & Hayashi, Y. (2000). Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Transactions on Neural Networks, 11*(3), 748–768.

Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks, 1*, 4–27.

Pitangui, C., & Zaverucha, G. (2008). Improved natural crossover operators in GBIVIL. In *Proceedings of the IEEE International Conference on Evolutionary Computation,* (pp. 2157–2164).

Rechenberg, I. (1994). Evolution strategy. In J. M. Zurada, R. J. Marks, II, & C. Goldberg (Eds.), *Computational intelligence. Imitating life*. NJ, Piscataway: IEEE Press.

Sadighi, A., & Kim, W. J. (2011). Adaptive-Neuro-Fuzzy-Based Sensorless Control of a Smart-Material Actuator. *IEEE Trans. On Mechatronics, 16*(2), 371–379.

Seki, H., Mizumoto, M. (2010). Fuzzy functional inference method. In *IEEE International Joint Conference on Fuzzy Systems,* (pp. 1–6).

Smith, R. E., Forrest, S., & Perelson, A. S. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation, 1*(2), 127–149.

Tan, J., & Quek, C. (2010). A BCM theory of meta-plasticity for online self-reorganizing fuzzy-associative learning. *IEEE Transactions on Neural Networks, 21*(6). 985-103.

Veretelnikova, E. L. (2004). Choice of coefficient of criterion of a minimum of the resulting root-mean-square error for linear automatic control systems. In *Proceedings of the international joint conference on electronic instrument engineering proceedings,* pp. 203–204.

Wang, Z. X., Wang, Q., Bai, M., Chen, Z. Q., & Sun, Z. (2010). Further exploration on relationship between crisp sets and fuzzy sets. In *Proceedings of the international joint conf. on computer and automation engineering,* vol. 2, (pp. 609–613).

Wang, G., He, B., Peng, Y., & Liang, G. (2010). A fuzzy-inference based neural network model and its application in real-time reservoir flood operation decision-making. In *Proceedings of the international joint conference on fuzzy systems and knowledge discovery (FSKD),* vol. 1, (pp. 37–40).

Xu, H., & Zhang, Q. (2010). Comparison of convergence of the modified and relaxed hybrid steepest-descent methods for variational inequalities under different conditions. In *Proceedings of the international conference on computational science and optimization (CSO),* vol. 2, (pp. 527–530).

Zahlay, F. D., Rao, K. S. R., & Ibrahim, T. B. (2011). A new intelligent autoreclosing scheme using artificial neural network and Taguchi's methodology. *IEEE Transactions on Industry Applications, 47*(1), 306–313.

Zou, Y.P., Mi, Z.K., & Xu, M.H. (2006). Dynamic load balancing based on roulette wheel selection. In *Proceedings of the joint conference on communications, circuits and systems proceedings,* (pp. 149–156).