Adaptive Learning Networks in Evolutionary Framework

[*1] Dong Won Kim

[1]*Department of Digital Electronics, Inha Technical College, South Korea*
dwnkim@inhatc.ac.kr

Abstract

We discuss a new design methodology of the self-organizing approximator technique (self-organizing polynomial neural networks (SOPNN)) using the evolutionary algorithm (EA). The SOPNN is based on the ideas of group method of data handling. The performances of SOPNN depend strongly on the number of input variables available to the model, and the number of input variables and polynomial type (order) to each node. These variables and polynomial types must be fixed by the designer in advance before the architecture is constructed, and thus, the trial and error method is burdened with heavy computation and low efficiency. Moreover, even after such procedures, the SOPNN may not be the best one. In this paper, we propose an EA-based SOPNN to alleviate these problems. The order of the polynomial, the number of input variables, and the optimum input variables are encoded as a chromosome and the fitness of each chromosome are computed. The appropriate information of each node is evolved accordingly and tuned gradually throughout the EA iterations. We can show that the EA-based SOPNN is a sophisticated and versatile architecture, which can construct models from a limited set of data as well as for poorly defined complex problems. Comprehensive comparisons showed that with a very simple structure, the EA-based SOPNN gave significantly improved performance than the conventional SOPNN model as well as previous identification methods with respect to approximation and prediction abilities.

Keywords: *SOPNN, evolutionary algorithm, approximation and prediction abilities, identification methods, self-organizing approximator technique*

1. Introduction

System modeling and identification is important for system analysis, control, and automation as well as for scientific research. Therefore, advanced system modeling techniques have been developed. Neural networks (NNs) and fuzzy systems have been widely used for modeling nonlinear systems. The approximation capabilities of neural networks, such as multilayer perceptrons, radial basis function (RBF) networks, or dynamic recurrent neural networks have been investigated [1-3]. Hybrid neural network is presented to extract fuzzy rules from the training data. Neurons in the model can be added or pruned in learning process. Similar membership functions (MF) are combined into a new MF to reduce the complexity of the model. [35]

On the other hand, fuzzy systems are able to approximate nonlinear functions with arbitrary accuracy [4-5]. But the resultant neural network representation is very complex and difficult to understand, and fuzzy systems require too many fuzzy rules for accurate function approximation, particularly in the case of multi-dimensional inputs.

Another method is the GMDH-type algorithm. Group Method of Data Handling (GMDH) was introduced by Ivakhnenko in the early 1970's [6-10]. GMDH-type algorithms have been extensively used since the mid-1970's for prediction and modeling complex nonlinear processes. GMDH is a self-organizing and can automatically select essential input variables without using prior information on the relationship among input-output variables [11].

Received: Jul. 21, 2014, Revised Aug.13, Accepted: Sep. 27, 2014

Self-organizing Polynomial Neural Networks (SOPNN) [12-14] is a GMDH-type algorithm and one of the useful approximate techniques. SOPNN has an architecture similar to feedforward neural networks, whose neurons are replaced by polynomial nodes. The output of the each node in an SOPNN structure is obtained by using several types of high-order polynomials, such as linear, quadratic, and modified quadratic polynomials, of input variables. These polynomials are called partial descriptions (PDs). SOPNNs have fewer nodes than NNs, but the nodes are more flexible. The SOPNN shows a superb performance in comparison to the previous fuzzy, neural network modeling methods [36-37]. Although the SOPNN is structured by a systematic design procedure, it has some drawbacks that need to be solved. If there are sufficiently large numbers of input variables and data points, SOPNN algorithm tends to produce overly complex networks. On the other hand, if sufficient numbers of input variables are unavailable, SOPNN cannot provide a model with good performance. Moreover, the performances of SOPNN depend strongly on the number of input variables available to the model as well as the number of input variables and polynomial types or order in each PD. These parameters of the SOPNN must be chosen in advance before the architecture of SOPNN is constructed. In most cases, they are determined by the trial and error method, which is burdened by heavy computation and low efficiency. Moreover, the SOPNN algorithm is a heuristic method so it does not guarantee the best SOPNN for nonlinear system modeling. Therefore, the above-mentioned drawbacks must be solved.

To handle these difficulties of the SOPNN, genetic algorithm is used in [38]. But the network size of the optimized SOPNN is not reduced at all compared with the original version of the SOPNN in [13]. And maximum number of inputs to be selected for a node in a layer is 10. As a result, genetic operators would consume much time due to complexity of the network. In addition, the design parameters (maximum generations and population size) of the model for each layer are all fixed so they can not reflect well characteristic of the system to be targeted. Consequently, it will be pointless for employing GA to optimize the SOPNN.

In this paper, we present a new design methodology of SOPNN. This methodology uses an evolutionary algorithm (EA) to alleviate the above-mentioned drawbacks of the SOPNN. This new network is called the EA-based SOPNN. Evolutionary Algorithm (EA) has been widely used as a parallel global search method for optimization problems [15-17]. The EA is used to determine the number of input variables for each node, which are optimally chosen among many input variables, and to determine the appropriate type of polynomials in each PD.

This paper is organized as follows. The conventional SOPNN [13] and its drawbacks are briefly explained to illustrate the proposed modeling algorithm in Section 2. The new algorithm used for the design of the EA-based SOPNN is described, and the coding of the key factors of the SOPNN, chromosome representation, and fitness function are also discussed in Section 2. The proposed EA-based SOPNN is applied to nonlinear system modeling to determine its performance, and its simulation results are compared with those of other methods, including the conventional SOPNN in Section 3. Conclusions are given in Section 4.

2. Design of EA-based SOPNN

The conventional SOPNN algorithm is based on the GMDH method and utilizes various classes of polynomials such as linear, quadratic, and modified quadratic types. By choosing the most significant input variables and polynomial types among the various types or forms available, the PDs in each layer can be obtained. The framework of the design procedure of the SOPNN and further discussion on the conventional SOPNN can be obtained in [13, 14].

When the final layer has been constructed, the node with the best predictive capability is selected as the output node. All remaining nodes except the output node in the final layer are discarded. Furthermore, all the nodes in the previous layers that do not influence the output node are also removed by tracing the data flow path of each layer.

The SOPNN is a flexible neural architecture whose structure is developed through a modeling process because each PD can have a different number of input variables and can exploit a different order of the polynomial. As a result, SOPNN provides a systematic design procedure. But the number of input variables and polynomial order type must be fixed by the designer in advance before the architecture is constructed. Thus, the trial and error method is burdened by heavy computation and low efficiency. However, this does not guarantee the best model, but provides only the assurance that a

good model has been produced for a certain system. Therefore, the performances of SOPNN depend strongly on a few factors, stated in the section 1. In this section, we propose the new design procedure using EA for the systemic design of an optimum SOPNN .

In the SOPNN algorithm, the key problems are how to determine the optimal number of input variables, which input variables are to be chosen, and how to select the order of the polynomial forming the PD in each node.

In [13], these factors are determined in advance by the trial and error method. But in this paper, these problems are solved by using EA automatically. The EA is implemented by using crossover and mutation probability rates for better exploitation of the optimal inputs and order of polynomial in each node of SOPNN. All of the initial EA populations are randomized, which implies that minimum heuristic knowledge is used. The appropriate inputs and order are evolved accordingly and are tuned gradually throughout the EA iterations.

In the evolutionary design procedure, key issues are the encodings of the polynomial order, the number of input variables, and the optimum input variables as a chromosome and the defining of a criterion to compute the fitness of each chromosome. In the following sections, the detailed representation of the coding strategy and choice of the fitness function are given.

2.1 Representation of chromosome for appropriate information of each PD

When we design the SOPNN using EA, the most important consideration is the representation strategy, that is, the encoding of the key factors of the SOPNN into a chromosome. We employ binary coding for the available design specifications. We code the polynomial order and the inputs of each node in the SOPNN as a finite-length string. Our chromosomes are made of three sub-chromosomes. The first one consists of 2 bits for the order of polynomial (PD), the second one of 3 bits for the number of inputs of PD, and the last one of N bits, which are equal to the number of all input candidates in the current layer. These input candidates are the node outputs of the previous layer. The representation of binary chromosomes is illustrated in Figure 1.

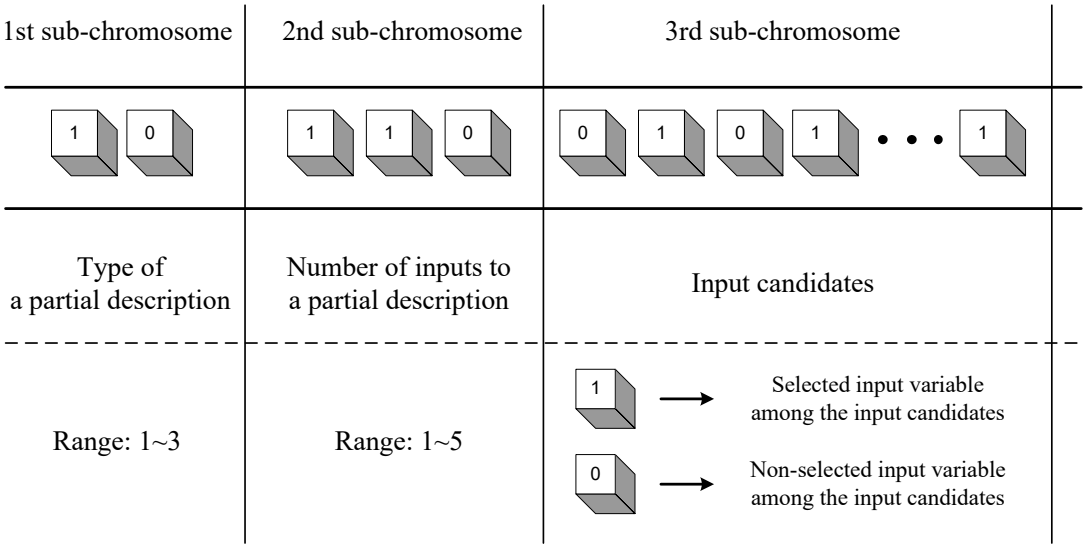


Figure 1. Structure of binary chromosome for a PD

The 1st sub-chromosome is made of 2 bits. It represents several types of order of PD. The relationship between the bits in the 1st sub-chromosome and the order of PD is shown in Table 1. Thus, each node can exploit different orders of the polynomial.

Table 1. Relationship between bits in the 1st sub-chromosome and order of PD

Bits in the 1st sub-chromosome	Order of polynomial(PD)
00	Type 1 – Linear
01	Type 2 – Quadratic
10	
11	Type 3 – Modified quadratic

The 3rd sub-chromosome has N bits, which are concatenated bits of 0s and 1s coding. The input candidate is represented by a 1 bit if it is chosen as an input variable to the PD and by a 0 bit if it is not chosen. This approach solves the problem of which input variables is to be chosen.

If many input candidates are chosen for a model design, the modeling becomes computationally complex, and normally, would require much time to achieve good results. In addition, improper results can be obtained, and generalization ability can degrade. Good approximation performance does not necessarily guarantee good generalization capability [19]. To overcome this drawback, we introduce the 2nd sub-chromosome into the chromosome. The 2nd sub-chromosome consists of 3 bits and represents the number of input variables to be selected. The number based on the 2nd sub-chromosome is shown in the Table 2. Input variables as many as the number represented in the 2nd sub-chromosome are selected for each node among all input candidates. Designer must determine the maximum number in consideration of the characteristic of the system, design specification, and some prior knowledge of the model. With this method, we can solve the problems such as the conflict between overfitting and generalization and the required long computation time.

Table 2. Relationship between bits in the 2nd sub-chromosome and number of inputs to PD

Bits in the 2nd sub-chromosome	Number of inputs to PD
000	1
001	2
010	2
011	3
100	3
101	4
110	4
111	5

The relationship between chromosome and information on PD is shown in Figure 2. The PD corresponding to the chromosome in Figure 2 is described briefly as Figure 3.

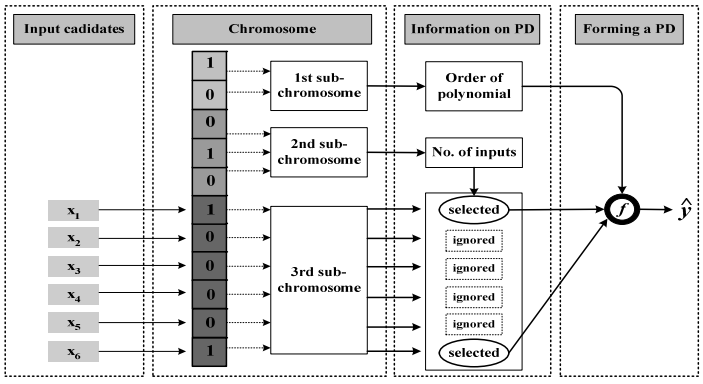


Figure 2. Example of PD whose various pieces of required information are obtained from its chromosome

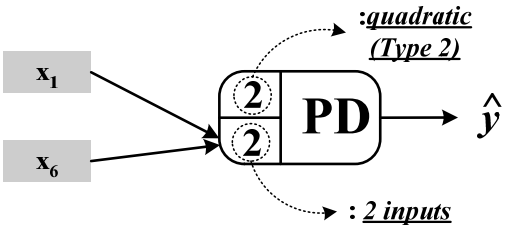


Figure 3. Node with PD corresponding to chromosome in Figure 2

Figure 2 shows an example of PD. The various pieces of the required information are obtained from chromosome. The 1st sub-chromosome shows that the polynomial order is Type 2 (quadratic form). The 2nd sub-chromosome shows two input variables to this node. The 3rd sub-chromosome tells that the selection of x_1 and x_6 as input variables. The node with PD corresponding to Figure 2 is shown in Figure 3. Thus, the output of this PD, $\hat{y}$, can be expressed as (1).

$$\hat{y} = f(x_1, x_6) = c_0 + c_1 x_1 + c_2 x_6 + c_3 x_1^2 + c_4 x_6^2 + c_5 x_1 x_6 \tag{1}$$

where coefficients c_0, c_1, …, c_5 are evaluated by using the training data set for the standard least square estimation (LSE).

Therefore, the polynomial function of PD is formed automatically according to the information of the sub-chromosomes.

The design procedure of EA-based SOPNN is shown in Figure 4.

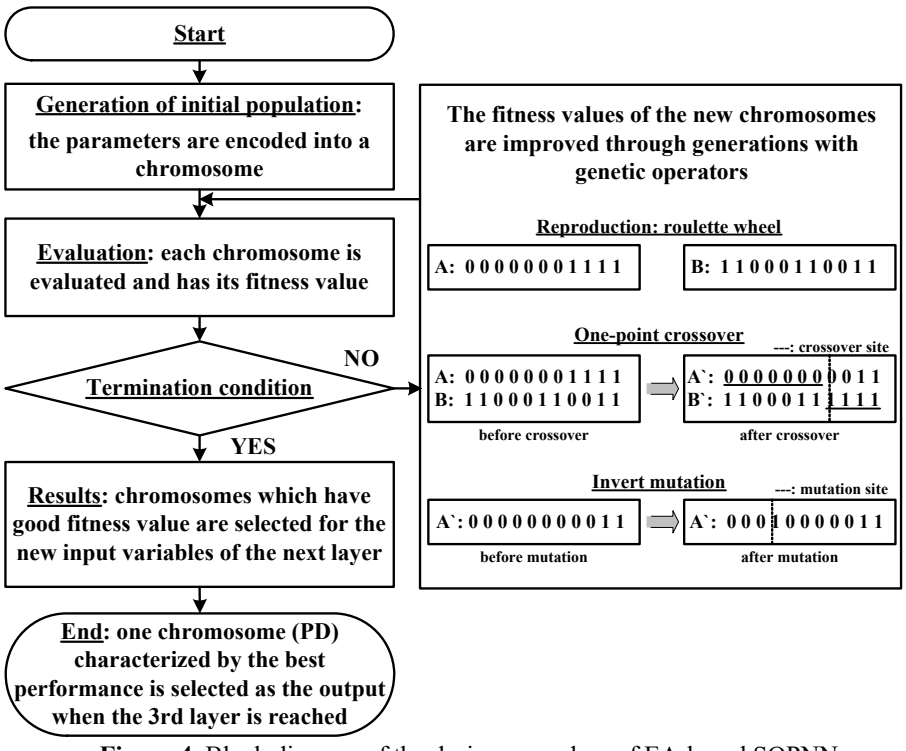


Figure 4. Block diagram of the design procedure of EA-based SOPNN

In the beginning of the process, the initial populations comprise a set of chromosomes that are scattered all over the search space. The populations are all randomly initialized. Thus, the use of

heuristic knowledge is minimized. The assignment of fitness in EA guides the search toward the optimal solution. The fitness function for several specific cases of modeling will be explained later. After each chromosome is evaluated and associated with a fitness, the current population undergoes the reproduction process to create the next generation of population. The roulette-wheel selection scheme is used to determine the members of the new generation of population. After the new group of population is built, the mating pool is formed, and the crossover is carried out. The crossover proceeds in three steps. First, two, newly reproduced strings are selected from the mating pool produced by reproduction. Second, a position (one point) along the two strings is selected uniformly at random. The third step is to exchange all characters following the crossing site. We use one-point crossover operator with a crossover probability of P_c (0.85). This exchange is then followed by the mutation operation. The mutation is the occasional alteration of a value at a particular bit position (we flip the states of a bit from 0 to 1 or vice versa). The mutation serves as an insurance policy for recovery of a loss of a particular piece of information (any simple bit). The mutation rate used is fixed at 0.05 (P_m). Generally, after these three operations, the overall fitness of the population improves. Each population generated then goes through a series of evaluation, reproduction, crossover, and mutation, and the procedure is repeated until a termination condition is reached. After the evolution process, the final population generated would consist of highly fit bits that can provide optimal solutions. After the termination condition is satisfied, one chromosome (PD) with the best performance in the final population is selected as the output PD. All other remaining chromosomes are discarded, and all the nodes that do not influence this output PD in the previous layers are also removed. Finally, the EA-based SOPNN model is obtained. Practical usage of the EA-based SOPNN for gas furnace process and three-input nonlinear function and evolutionary operators are depicted in simulation results. In the simulation, design parameters and finally structured model are shown in Table 3, Table 7, Figure 8, and Figure 13, respectively.

2.2 Fitness function for modeling

In EA, it is important to determine the fitness function. The genotype representation encodes the problem into a string, while the fitness function measures the performance of the model. It is quite important for evolving systems to find a good fitness measurement. To construct models with significant approximation and generalization ability, we introduce the error function [26] such as

$$E = \theta \times PI + (1-\theta) \times EPI \tag{2}$$

where $\theta \in [0,1]$ is a weighting factor for PI and EPI, which denote the values of the performance indices of the training data and testing data, respectively, as expressed in (4). Then, the fitness value [12] is determined as follows:

$$F = \frac{1}{1+E} \tag{3}$$

Maximizing F is identical to minimizing *E*. The choice of θ establishes a certain tradeoff between the approximation and generalization abilities of the EA-based SOPNN.

3. Simulation Results

In this section, we show the performance of our new EA-based SOPNN for two, well known, nonlinear system models. One is a time series of a gas furnace (Box-Jenkins data)[20], which was studied previously in [20-29]. The other is a nonlinear system already exploited in fuzzy modeling [30-34].

3.1 Gas furnace process

The delayed terms of methane gas flow rate u(t) and carbon dioxide density y(t) such as u(t-3), u(t-2), u(t-1), y(t-3), y(t-2), and y(t-1)are used as input variables to the EA-based SOPNN. The actual system output y(t) is used as the target output variable for this model. We choose the input variables of the

nodes in the 1st layer from these input variables. The total data set consisting of 296 input-output pairs is split into two parts. The first part of the data set (consisting of 148 pairs) is used for training. The remaining part of the data set serves as the testing set. Using the training data set, the coefficients of the polynomial are estimated by using the standard LSE. The performance index (PI, EPI) is defined as the mean squared error

$$\mathrm{PI} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{4}$$

where y_i is the actual system output, $\hat{y}_i$ is the estimated output of each node, and m is the number of data.

The design parameters of EA-based SOPNN for modeling are shown in Table 3. In the 1st layer, 20 chromosomes are generated and evolved during 40 generations, where each chromosome in the population is defined as a corresponding node. Therefore, 20 nodes (PDs) are produced in the 1st layer based on the EA operators. All PDs are estimated and evaluated by using the training and testing data sets, respectively. They are also evaluated by the fitness function of (3) and ranked according to their fitness value. We choose nodes as many as the predetermined number *w* from the highest ranking node, and use their outputs as the new input variables to the nodes in the next layer. In other words, the chosen PDs (*w* nodes) must be preserved for the design of the next layer and the outputs of the preserved PDs serve as inputs to the next layer. The value of *w* varies from layer to layer; this variation is also shown in Table 3. This procedure is repeated for the 2nd layer and the 3rd layer.

Table 3. Design parameters of EA-based SOPNN for modeling

Parameters	1st layer	2nd layer	3rd layer
Maximum generations	40	60	80
Population size:(*w)*	20:(15)	60:(50)	80
String length	11	20	55
Crossover rate (P_c)	0.85		
Mutation rate (P_m)	0.05		
Weighting factor: θ	0.1~0.9		
Type (order)	1~3		

w: the number of chosen nodes whose outputs are used as inputs to the next layer

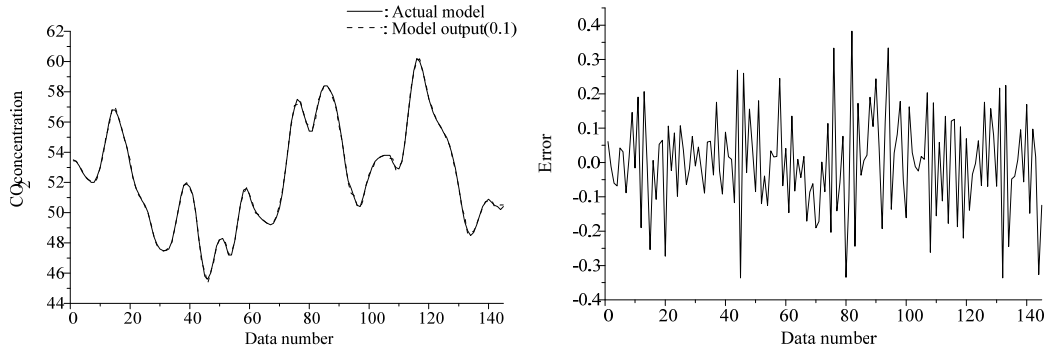
Table 4 summarizes the values of the performance indices, PI and EPI, of the proposed EA-based SOPNN according to weighting factor. These values are the lowest values in each layer. The overall lowest value of the performance index is obtained at the third layer for the weighting factor is 0.5. If this model is designed to have a fourth or higher layer, the performance values will be much lower, and computation time will be long and the model network will be much too complex. To show the best output of the networks compared with the actual output according to the weighting factor, Figure 5 is depicted. In the figure, we can clearly get the results from training and testing in (a)-(e) and (f)-(j), respectively.

Figure 6 depicts the trend of the performance index values produced in successive generations of the EA when the weighting factor θ is 0.5. Figure 7 illustrates the values of the error function and fitness function in successive EA generations when θ =0.5.

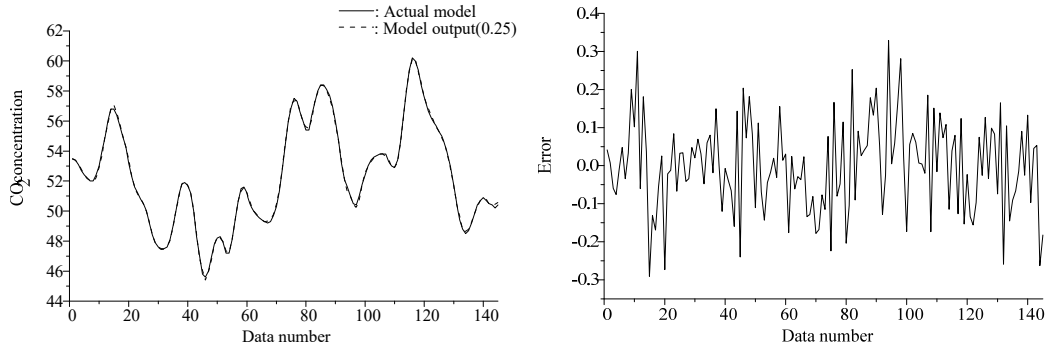
Table 4. Values of performance index of the proposed EA-based SOPNN

Weighting factor (θ)	1st layer		2nd layer		3rd layer	
	PI	EPI	PI	EPI	PI	EPI
0.1	0.0214	0.1260	0.0200	0.1231	0.0199	0.1228
0.25	0.0214	0.1260	0.0149	0.1228	0.0145	0.1191
0.5	0.0214	0.1260	0.0139	0.1212	0.0129	0.1086
0.75	0.0214	0.1260	0.0139	0.1293	0.0138	0.1235

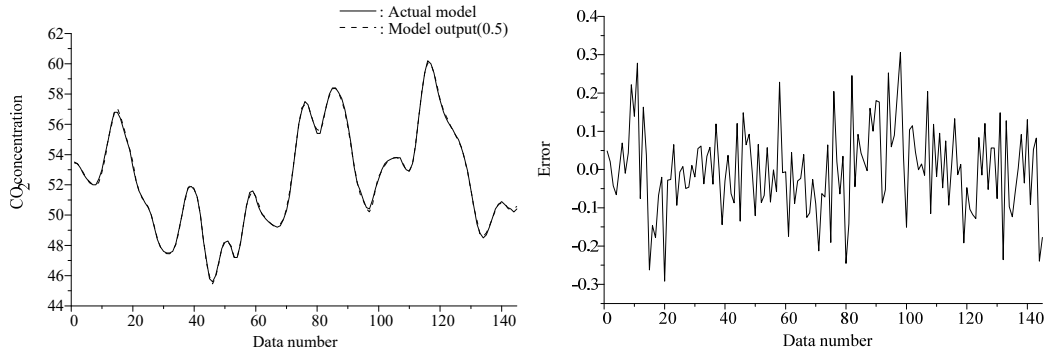
0.9	0.0173	0.1411	0.0137	0.1315	0.0129	0.1278



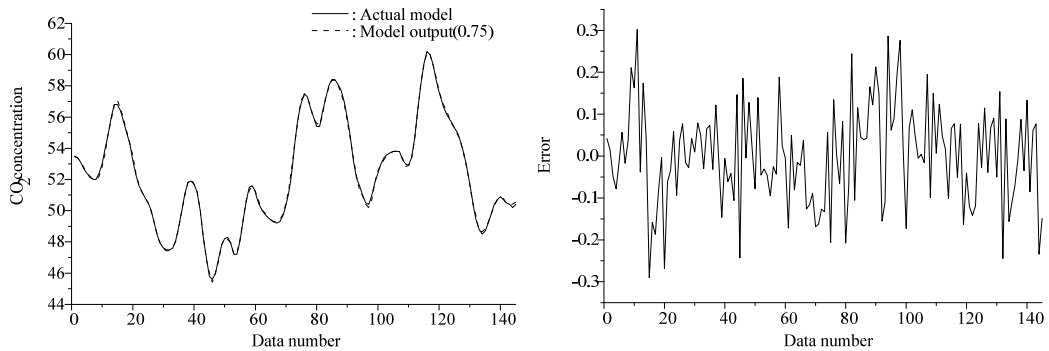
(a) actual output vs. model output (θ=0.1) of training data set and its errors



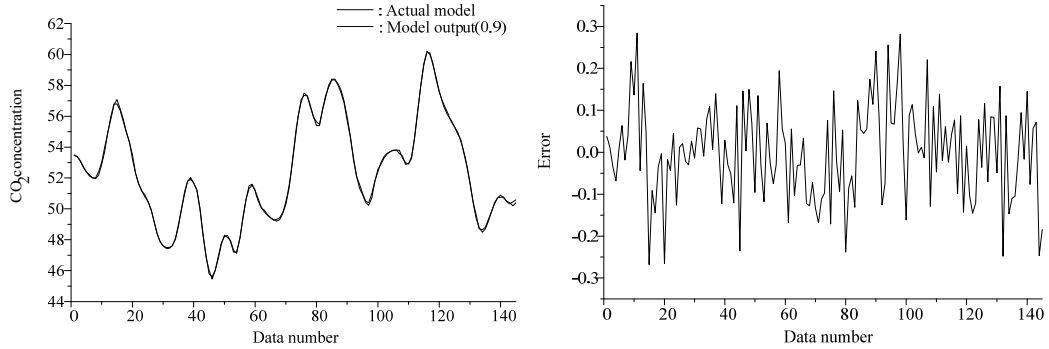
(b) actual output vs. model output (θ=0.25) of training data set and its errors



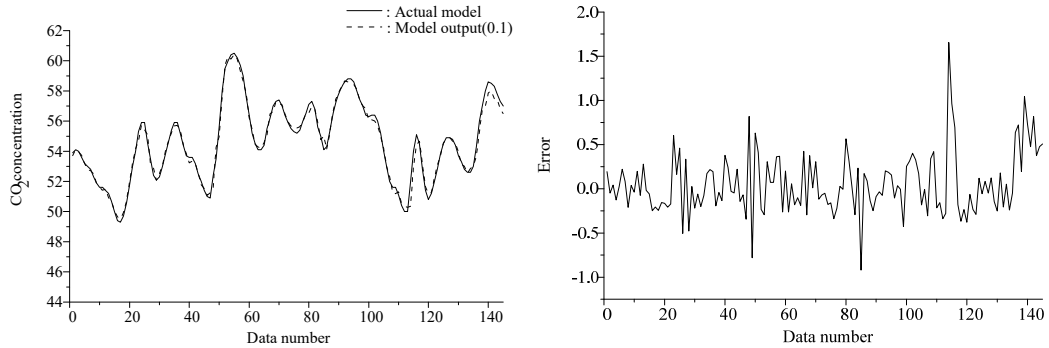
(c) actual output vs. model output (θ=0.5) of training data set and its errors



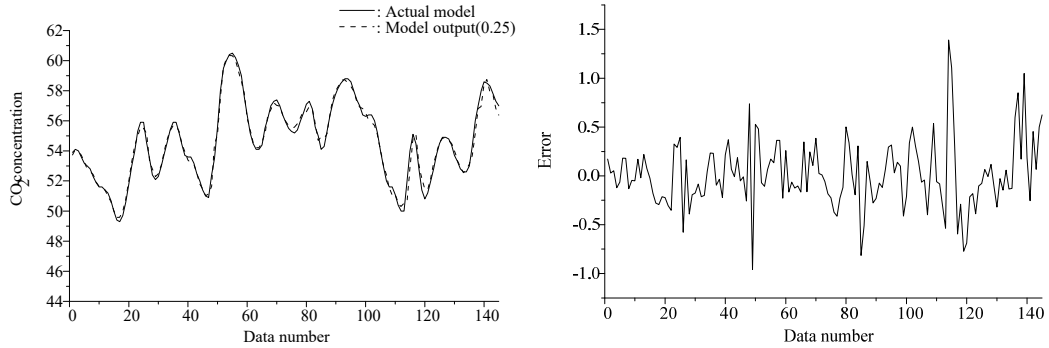
(d) actual output vs. model output (θ=0.75) of training data set and its errors



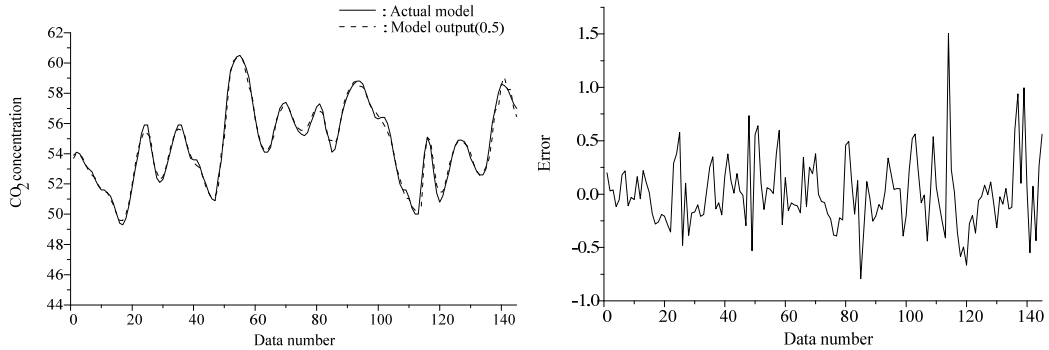
(e) actual output vs. model output (θ=0.9) of training data set and its errors



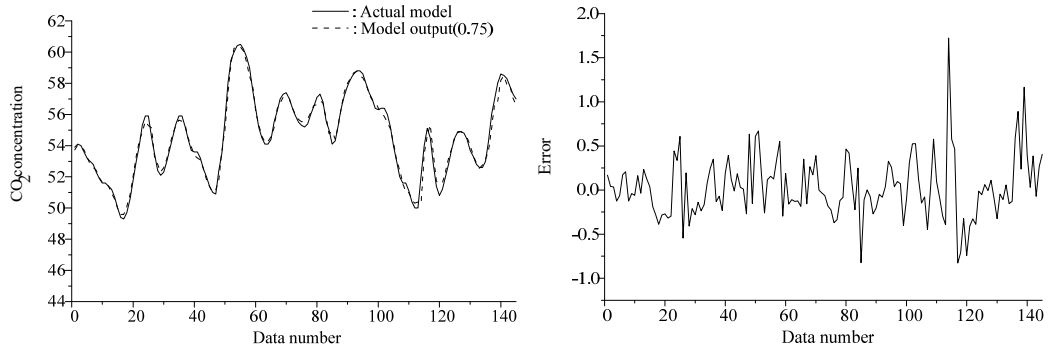
(f) actual output vs. model output (θ=0.1) of testing data set and its errors



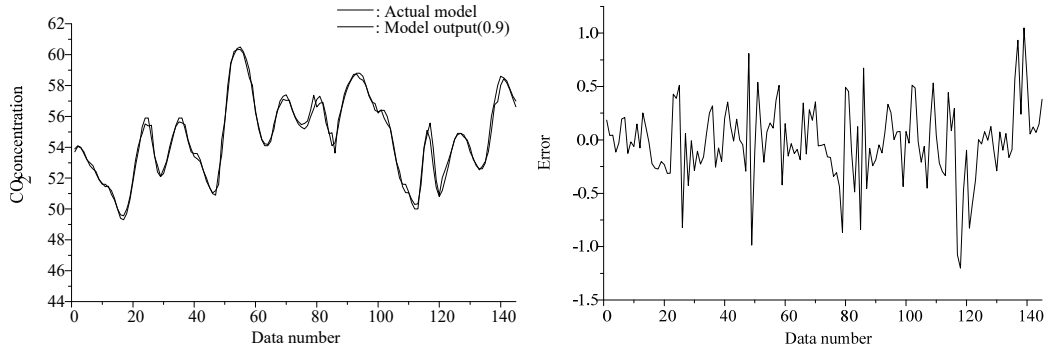
(g) actual output vs. model output (θ=0.25) of testing data set and its errors



(h) actual output vs. model output (θ=0.5) of testing data set and its errors

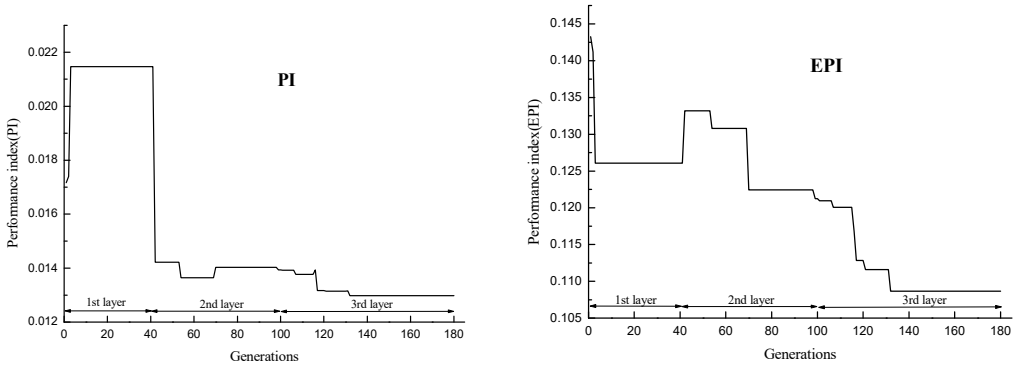


(i) actual output vs. model output (θ=0.75) of testing data set and its errors



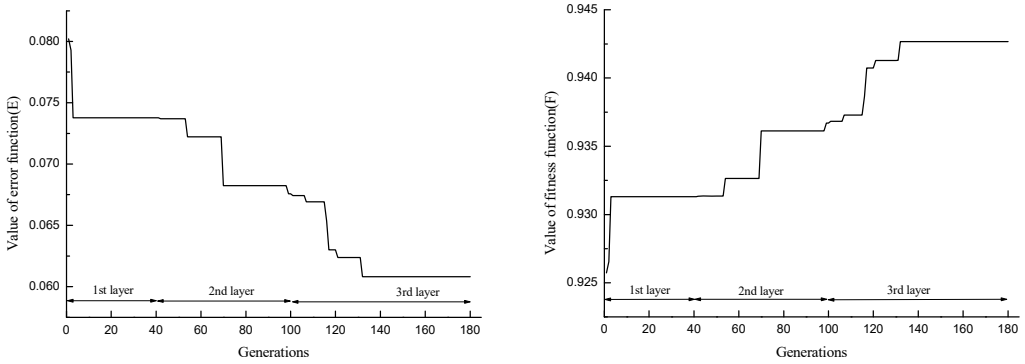
(j) actual output vs. model output (θ=0.9) of testing data set and its errors

Figure 5. Best output of the networks compared with the actual output according to the weighting factor



(a) performance index for the training data set (b) performance index for the testing data set

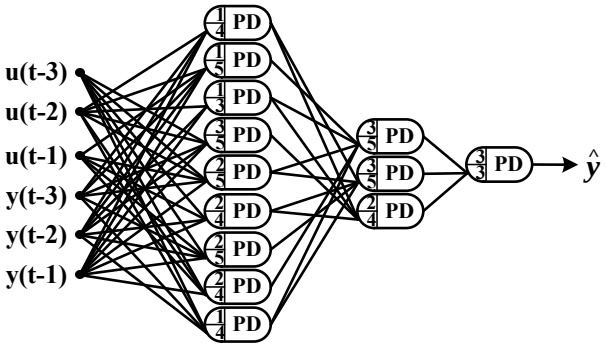
Figure 6. Trend of performance index values with respect to generations through layers (θ=0.5)



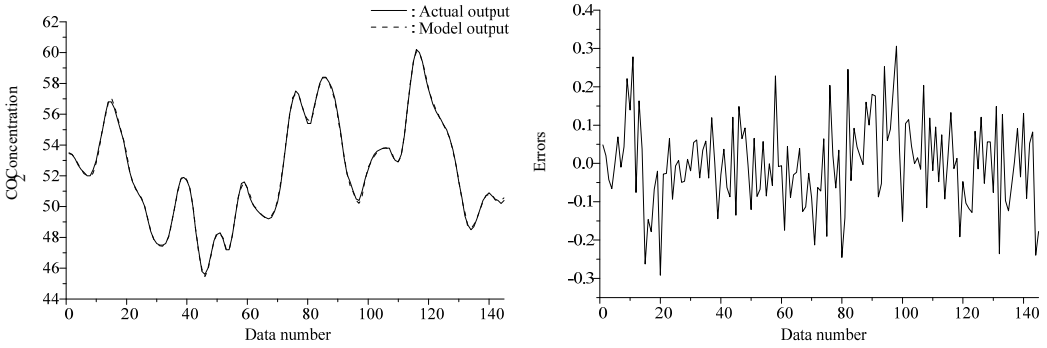
(a) error function (E) (b) fitness function (F)

Figure 7. Values of the error function and fitness function with respect to the successive generations (θ=0.5)

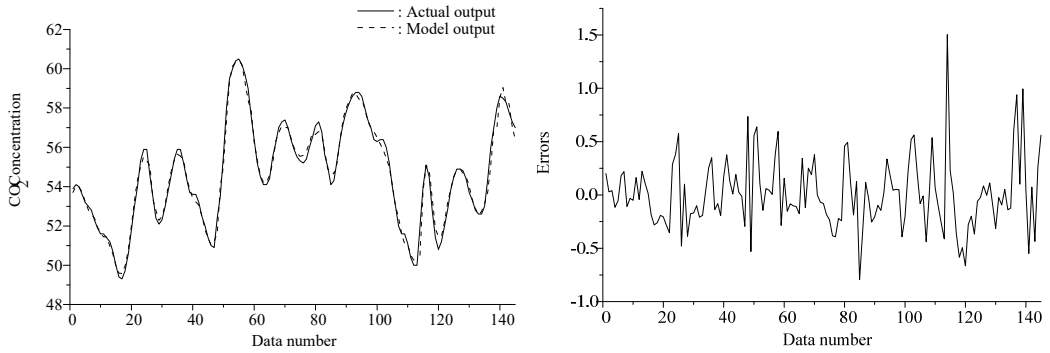
Figure 8 shows the proposed EA-based SOPNN model with 3 layers and its identification performance when the θ =0.5. The model output follows the actual output very well, the values of the performance indices of the proposed method are equal to PI=0.012 and EPI=0.108, respectively.



(a) Proposed EA-based SOPNN model with 3 layers



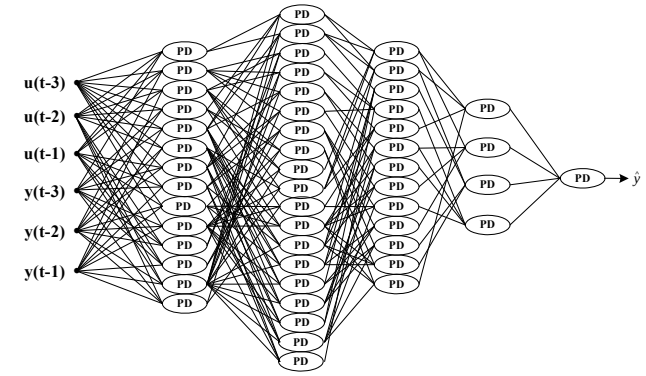
(b) actual output vs. model output of training data set (c) errors of (b)



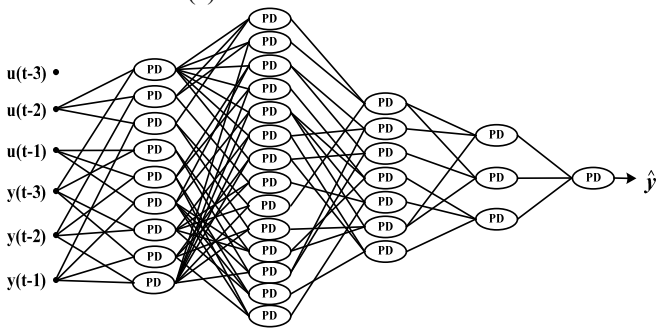
(d) actual output vs. model output of testing data set (e) errors of (d)

Figure 8. Proposed EA-based SOPNN model with 3 layers and its identification performance (θ=0.5)

The network size of the proposed EA-based SOPNN is compared with that of conventional SOPNN whose conventional SOPNN models are visualized in Figure 9. The structure of the basic SOPNN & Case 1 in Figure 9 (a) is obtained by the use of 4 input variables and Type 3 polynomial for every node in all layers to the fifth layer. Its performance is as follows. PI=0.012, EPI=0.084. On the other hand, the structure of the modified SOPNN and Case 2 in Figure 9 (b) is obtained by the use of 2 input variables and Type 1 polynomial for every node in the 1st layer, and 3 input variables and Type 2 polynomial for every node from the 2nd layer to the 5th layer. In this model, PI is 0.016, and EPI is 0.101. Figures 8 and 9 show that the structure of the EA-based SOPNN is much simpler than that of the conventional SOPNN in terms of the number of nodes and layers, although both types of SOPNN show almost the same performance.



(a) Basic SOPNN & Case 1



(b) Modified SOPNN & Case 2

Figure 9. Conventional SOPNN models with 5 layers

Table 5. Values of performance indices of some identification models

Model		Performance index		
		P	PI	EPI
Tong's model[21]		0.469		
Sugeno's model[22]		0.190		
Xu's model[23]		0.328		
Pedrycz's model[24]		0.320		
Leski's model[25]		0.047		
Oh's model[26]		0.123		
Kang's model[27]		0.161		
Kim's model[28]			0.034	0.244
Lin's model[29]			0.071	0.261
Kim's model [12]			0.013	0.126
SOPNN (5 layers) [13]	Basic &Case 1		0.012	0.084
	Modified&Case2		0.016	0.101
EA-based SOPNN (3 layers)			0.012	0.108

Table 5 provides a comparison of the proposed model with other techniques already proposed in the literature. The same performance indices are compared between the training and testing data sets. Additionally, P denotes a performance index of the models for the entire data set (not being split into a training and a testing set). PI denotes a performance index of the model for the training data set, while EPI for the testing data. The proposed architecture, EA-based SOPNN model, outperforms the other models both in terms of accuracy and higher generalization capabilities.

Considering time consuming of the model, 328.04 seconds are taken for the results of gas furnace process. It is about 6 minutes using Pentium 4 CPU 2.8GHz with 760MB RAM. More specifically, computation time took in the 1st layer, 2nd layer, and 3rd layer in Table 5 are 5.5 seconds, 118.8 seconds, and 203.7 seconds, respectively.

3.2 Three-input nonlinear function

In this example, we will demonstrate how the proposed EA-based SOPNN model can be employed to identify a highly nonlinear function. The performance of this model will be compared with that of earlier works. The function to be identified is a three-input nonlinear function given by (5). It is nonlinear static system with 3 input variables confined to the range 1-5. We can show three-dimensional input-output relationship, x_1-x_2-y and x_2-x_3-y, in Figure 10, respectively.

$$y = (1 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5})^2, \tag{5}$$

which was widely used by Takagi and Hayashi[30], Sugeno and Kang[31], and Kondo[32] to test their modeling approaches.

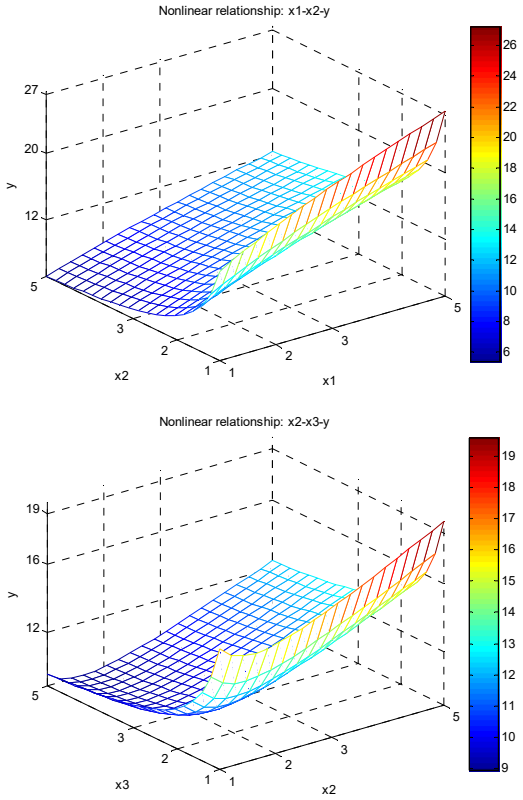


Figure 10. Input-output relation of nonlinear system

Table 6. Input-output data of three-input nonlinear function

Training data (1-20)						Testing data (21-40)					
No.	x_1	x_2	x_3	x_4	y	No.	x_1	x_2	x_3	x_4	y
1	1	3	1	1	11.11	21	1	1	5	1	9.545
2	1	5	2	1	6.521	22	1	3	4	1	6.043
3	1	1	3	5	10.19	23	1	5	3	5	5.724
4	1	3	4	5	6.043	24	1	1	2	5	11.25
5	1	5	5	1	5.242	25	1	3	1	1	11.11
6	5	1	4	1	19.02	26	5	5	2	1	14.36
7	5	3	3	5	14.15	27	5	1	3	5	19.61
8	5	5	2	5	14.36	28	5	3	4	5	13.65
9	5	1	1	1	27.42	29	5	5	5	1	12.43
10	5	3	2	1	15.39	30	5	1	4	1	19.02
11	1	5	3	5	5.724	31	1	3	3	5	6.38
12	1	1	4	5	9.766	32	1	5	2	5	6.521
13	1	3	5	1	5.87	33	1	1	1	1	16
14	1	5	4	1	5.406	34	1	3	2	1	7.219
15	1	1	3	5	10.19	35	1	5	3	5	5.724
16	5	3	2	5	15.39	36	5	1	4	5	19.02
17	5	5	1	1	19.68	37	5	3	5	1	13.39
18	5	1	2	1	21.06	38	5	5	4	1	12.68
19	5	3	3	5	14.15	39	5	1	3	5	19.61
20	5	5	4	5	12.68	40	5	3	2	5	15.39

Table 6 shows 40 pairs of the input-output data obtained from (5) [34]. The input x_4 is a dummy variable that has no relation to (5). The data in Table 6 are divided into the training data set (Nos. 1-20) and the testing data set (Nos. 21-40). To compare the performance, the same performance index, that is, the average percentage error (APE), adopted in [30-34], is used.

$$APE = \frac{1}{m}\sum_{i=1}^{m}\frac{|\, y_i - \hat{y}_i \,|}{y_i} \times 100 \qquad (\%) \tag{6}$$

where m is the number of data pairs and y_i and $\hat{y}_i$ are the i-th actual output and model output, respectively.

Again, a series of comprehensive experiments was conducted and the results are summarized. The design parameters of EA-based SOPNN in each layer are shown in Table 7.

Table 7. Design parameters of EA-based SOPNN for modeling

Parameters	1st layer	2nd layer	3rd layer
Maximum generations	40	60	80
Population size:(*w*)	20:(15)	60:(50)	80
String length	8	20	55
Crossover rate (P_c)	0.85		
Mutation rate (P_m)	0.05		
Weighting factor: θ	0.1~0.9		
Type (order)	1~3		

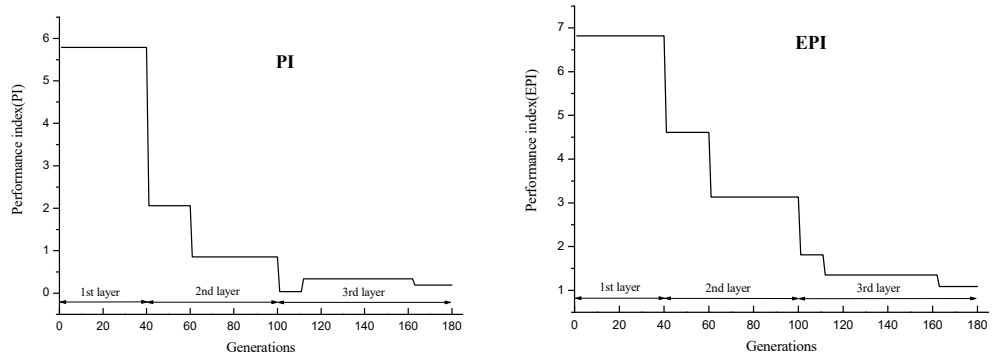
w: the number of chosen nodes whose outputs are used as inputs to the next layer

The simulation results of the EA-based SOPNN are summarized in Table 8. The overall lowest values of the performance indicies, PI=0.188 EPI=1.087, are obtained at the third layer when the weighting factor (θ) is 0.25.

Table 8. Values of performance index of the proposed EA-based SOPNN model

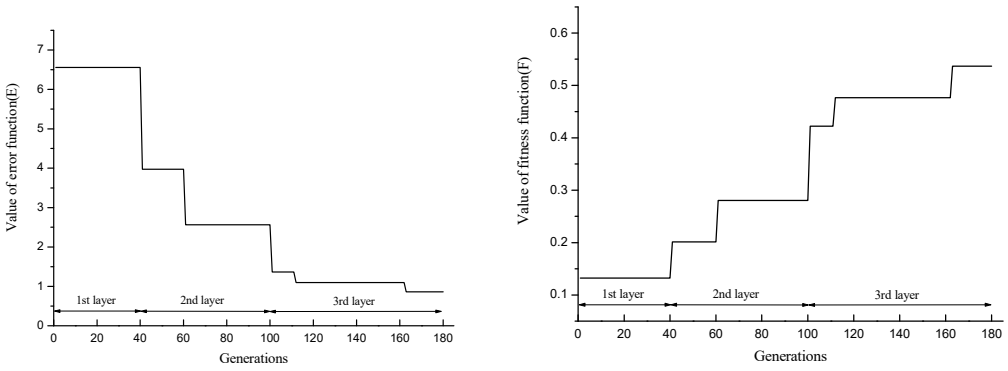
Weighting factor	1st layer		2nd layer		3rd layer	
	PI	EPI	PI	EPI	PI	EPI
0.1	5.7845	6.8199	2.3895	3.3400	2.2837	3.1418
0.25	5.7845	6.8199	0.8535	3.1356	0.1881	1.0879
0.5	5.7845	6.8199	1.6324	5.5291	1.2268	3.5526
0.75	5.7845	6.8199	1.9092	4.0896	0.5634	2.2097
0.9	5.7845	6.8199	2.5083	5.1444	0.0002	4.8804

Figure 11 illustrates the trend of the performance index values produced in successive generations of the EA when the weighting factor θ is 0.25. Figure 12 shows the values of error function and fitness function in successive EA generations when the θ is 0.25.



(a) performance index for the training data set (b) performance index for the testing data set

Figure 11. Trend of performance index values with respect to generations through layers (θ =0.25)



(a) error function (E) (b) fitness function (F)

Figure 12. Values of the error function and fitness function with respect to the successive generations (θ =0.25)

Figure 13 depicts the proposed EA-based SOPNN model with 3 layers when the θ is 0.25. The structure of EA-based SOPNN is very simple and has a good performance. But the conventional SOPNN has difficulty structuring a model for this nonlinear function. Therefore, only few input candidates are considered.

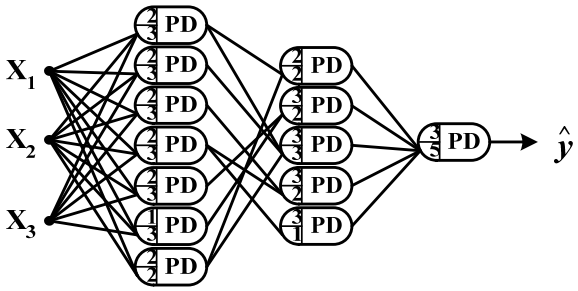
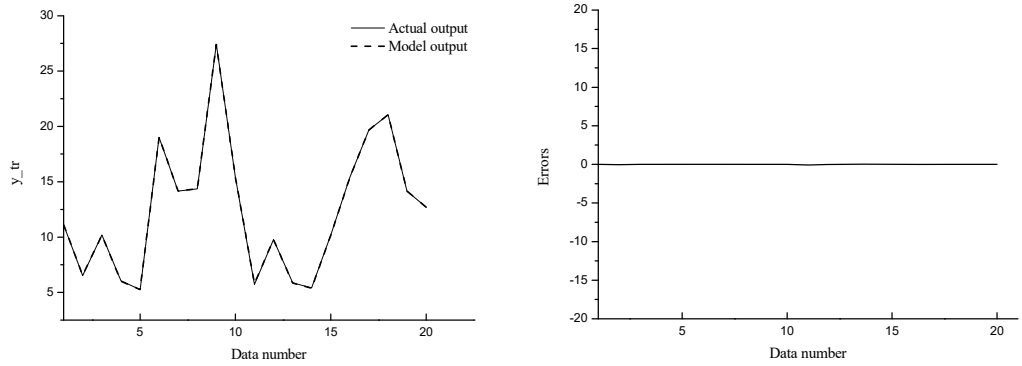
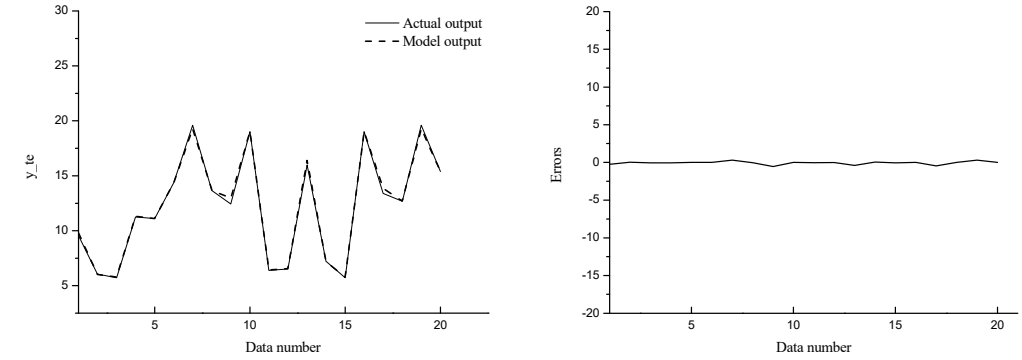


Figure 13. Structure of the EA-based SOPNN model with 3 layers (θ =0.25)

Figure 14 shows the identification performance of the proposed EA-based SOPNN and its errors when the θ is 0.25. The output of the EA-based SOPNN follows the actual output very well.



(a) actual output versus model output of training data set (b) errors of (a)



(c) actual output versus model output of testing data set (d) errors of (c)

Figure 14. Identification performance of EA-based SOPNN model with 3 layers and its errors

Table 9 shows the performances of the proposed EA-based SOPNN model and other models studied in the literature. The experimental results clearly show that the proposed model outperforms the existing models both in terms of better approximation capabilities (PI) as well as superb generalization abilities (EPI). But the conventional SOPNN cannot be applied properly to this example. With regard to the time consuming of the model, 576.9 seconds are took for modeling. It is about 10 minutes using Pentium 4 CPU 2.8GHz with 760MB RAM. More specifically, computation time in Table 9 took in the 1st layer, 2nd layer, and 3rd layer are 12.8 seconds, 219.5 seconds, and 344.6 seconds, respectively.

Table 9. Performance comparison of various identification models

Model		APE	
		PI (%)	EPI (%)
GMDH model[32]		4.7	5.7
Fuzzy model [31]	Model 1	1.5	2.1
	Model 2	0.59	3.4
FNN [34]	Type 1	0.84	1.22
	Type 2	0.73	1.28
	Type 3	0.63	1.25
GD-FNN [33]		2.11	1.54
SOPNN	Basic&Case1	2.59	8.52

(5 layers) [13]	Modified	Impossible	
EA-based SOPNN		0.188	1.087

4. Conclusions

In this paper, by using an evolutionary algorithm, we propose a new design methodology of SOPNN , which is called the EA-based SOPNN, and study the properties of the EA-based SOPNN. The EA-based SOPNN is a sophisticated and versatile architecture which can construct models from a limited set of data and for poorly defined complex problems. Moreover, the architecture of the model is not predetermined, but zcan be self-organized automatically during the design process. The conflict between overfitting and generalization can be avoided by using the fitness function with weighting factor. The experimental results show that the proposed EA-based SOPNN is superior to the conventional SOPNN models as well as other previous models in terms of modeling performance.

Examples employed in the paper, gas furnace process and three-input nonlinear system, are well modeled by proposed method and we can get a reasonable approximation capability. Especially, the network size is smaller and performance is better than classical SOPNN. This could be possible by two techniques combining optimization of evolutionary algorithm and polynomial approximation of SOPNN. Occasionally, these good results can be varied according to the peculiarity of the system to be identified so we need further research about this. The EA-based SOPNN model can be an effective means to construct a predictive model for poorly defined complex systems. According to the sophisticated and versatile architecture, the significance of these works is not restricted to gas furnace and nonlinear static function. There are a lot of applications in very different fields such as economic systems, ecological systems analysis and prediction, environment systems, medical diagnostics, weather modeling, econometric modeling and marketing, manufacturing and materials, physical experiments, and military systems. Thus the work is significant for any complex process with nonlinearity.

5. References

[1] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *IEEE Trans. Neural Netw.*, Vol. 2, pp. 359–366, Mar. 1989.

[2] T. Chen and H. Chen, “Approximation capability to functions of several variables, nonlinear functions, and operators by radial basis function neural networks,” *IEEE Trans. Neural Netw.*, Vol. 6, pp. 904–910, July 1995.

[3] L. K. Li, “Approximation theory and recurrent networks,” in Proc. of the International Joint Conference on (IJCNN), Baltimore, MD, Vol. 2, June 1992, pp. 266–271.

[4] L. X. Wang and J. M. Mendel, “Generating fuzzy rules by learning from examples,” *IEEE Transaction on,* Vol. 22, pp. 1414–1427, June 1992.

[5] L. X. Wang and J. M. Mendel, “Fuzzy basis function, universal approximation, and orthogonal least-squares learning,” *IEEE Trans. Neural Netw.*, Vol. 3, pp. 807–814, Sept. 1992.

[6] A. G. Ivakhnenko, “Polynomial theory of complex systems,” *IEEE Trans. Syst., Man, Cybern.,* Vol. SMC-1, No. 1, pp. 364–378, Oct. 1971.

[7] A. G. Ivakhnenko and N. A. Ivakhnenko, “Long-term prediction by GMDH algorithms using the unbiased criterion and the balance-of-variables criterion,” *Sov. Automat. Contr.*, Vol. 7, pp. 40–45, 1974.

[8] A. G. Ivakhnenko and N. A. Ivakhnenko, “Long-term prediction by GMDH algorithms using the unbiased criterion and the balance-of-variables criterion, part 2,” *Sov. Automat. Contr.*, Vol. 8, pp. 24–38, 1975.

[9] A. G. Ivakhnenko, V. N. Vysotskiy, and N. A. Ivakhnenko, “Principal version of the minimum bias criterion for a model and an investigation of their noise immunity,” *Sov. Automat. Contr.*, Vol. 11, pp. 27–45, 1978.

[10] A. G. Ivakhnenko, G. I. Krotov, and N. A. Ivakhnenko, "Identification of the mathematical model of a complex system by the self-organization method," in *Theoretical Systems Ecology: Advances and Case Studies*, E. Halfon, Ed. New York: Academic, ch. 13, 1970.

[11] S. J. Farlow, "Self-Organizing Methods in Modeling, GMDH Type-Algorithms," *New York: Marcel Dekker*, 1984.

[12] D. W. Kim, "Evolutionary Design of Self-Organizing Polynomial Neural Networks," in Proc. of the Control & Instrum., Wonkwang Univ., Chonbuk, 2002.

[13] S. K. Oh and W. Pedrycz, "The design of self-organizing Polynomial Neural Networks," *Inf. Sci.*, Vol. 141, pp. 237–258, Apr. 2002.

[14] D. W. Kim, and G. T. Park, "A Novel Design of Self-Organizing Approximator Technique: An Evolutionary Approach," in Proc. of the IEEE International Conference on Systems, Man and Cybernetics (SMC), Oct. 2003, pp. 4643–4648.

[15] Y. Shi, R. Eberhart, and Y. Chen, "Implementation of Evolutionary Fuzzy Systems," *IEEE Trans. on Fuzzy Systems,* Vol. 7, No. 2, pp. 109–119, Apr. 1999.

[16] K. Kristinnson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Trans. Syst., Man, Cybern.,* Vol. 22, No. 5, pp. 1033–1046, Sep–Oct. 1992.

[17] S. Uckun, S. Bagchi, Y. Miyabe, and K. Kawamura, "Managing genetic search in job shop scheduling," *IEEE Expert*, Vol. 8, No. 5, pp. 15–24, Oct. 1993.

[18] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning. Reading," *MA: Addison-Wesley*, 1989.

[19] S. Y. Kung and J. S. Taur, "Decision-based neural networks with signal/image classification applications," *IEEE Trans. Neural Netw.*, Vol. 6, pp. 170–181, Jan. 1995.

[20] G. E. P. Box and F. M. Jenkins, "Time Series Analysis: Forecasting and Control," *Holden-day*, 1976.

[21] R. M. Tong, "The evaluation of fuzzy models derived from experimental data," *Fuzzy Sets Syst.*, Vol. 13, No. 1, pp. 1–12, July 1980.

[22] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, Vol. 1, No. 1, pp. 7–31, 1993.

[23] C. W. Xu, and Y. Zailu, "Fuzzy model identification self-learning for dynamic system," *IEEE Trans. Syst., Man, Cybern.,* Vol. 17, No. 4, pp .683–689, Aug. 1987.

[24] W. Pedrycz, "An identification algorithm in fuzzy relational system," *Fuzzy Sets Syst.*, Vol. 13, No. 2, pp.153–167, July 1984.

[25] J. Leski, and E. Czogala, "A new artificial neural networks based fuzzy inference system with moving consequents in if-then rules and selected applications," *Fuzzy Sets Syst.*, Vol. 108, No. 3, pp. 289–297, Dec. 1999.

[26] S. Oh, and W. Pedrycz, "Identification of fuzzy systems by means of an auto-tuning algorithm and its application to nonlinear systems," *Fuzzy Sets Syst.*, Vol. 115, No. 2, pp. 205–230, Oct. 2000.

[27] S. J. Kang, C. H. Woo, H. S. Hwang, and K. B. Woo, "Evolutionary Design of Fuzzy Rule Base for Nonlinear System Modeling and Control," *IEEE Trans. Fuzzy Syst.*, Vol. 8, No. 1, pp. 37–45, Feb. 2000.

[28] E. Kim, H. Lee, M. Park, and M. Park, "A simply identified Sugeno-type fuzzy model via double clustering," *Inf. Sci.*, Vol. 110, No. 1–2, pp. 25–39, Sept. 1998.

[29] Y. Lin, G. A. Cunningham III, "A new approach to fuzzy-neural modeling," *IEEE Trans. Fuzzy Syst.*, Vol. 3, No. 2, pp. 190–197, May 1995.

[30] H. Takagi and I. Hayashi, "NN-driven fuzzy reasoning," *Int. J. Approx. Reasoning*, Vol. 5, No. 3, pp. 191–212, May 1991.

[31] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets Syst.*, Vol. 28, No. 1, pp. 15–33, Oct. 1988.

[32] T. Kondo, "Revised GMDH algorithm estimating degree of the complete polynomial," *Tran. Soc. Instrum. Control Eng.*, Vol. 22, No. 9, pp. 928–934, 1986.

[33] S. Wu, M. J. Er, and Y. Gao, "A Fast Approach for Automatic Generation of Fuzzy Rules by Generalized Dynamic Fuzzy Neural Networks," *IEEE Trans. Fuzzy Syst.*, Vol. 9, No. 4, pp. 578–594, Aug. 2001.

[34] S. I. Horikawa, T. Furuhashi, and Y. Uchikawa, "On Fuzzy modeling Using Fuzzy Neural

Networks with the Back-Propagation Algorithm," *IEEE Trans. Neural Netw.*, Vol. 3, No. 5, pp. 801–806, Sept. 1992.

[35] G. Leng, T. M. McGinnity, and G. Prasad, "An Approach for On-Line Extraction of Fuzzy Rules using a Self-organising Fuzzy Neural Networks," *Fuzzy Sets and Systems*, Vol. 150, No. 2, pp 211–243, Mar. 2005.

[36] D. Kim, B. Kim, and G. -T. Park, "A Plasma Etching Process Modeling Via a Polynomial Neural Network," *ETRI Journal*, Vol. 26, No. 4, pp. 297–306, Aug. 2004.

[37] B. Kim, D. Kim, and G. -T. Park, "Prediction of Plasma Etching Using Polynomial Neural Network," *IEEE Trans. Plasma Science*, Vol. 31, No. 6, pp. 1330–1336, Dec. 2003.

[38] S. -K. Oh and W. Pedrycz, "Multi-layer self-organizing neural networks and their development with the use of genetic algorithms", *Journal of the Franklin Institute*, Vol. 343, No. 2, pp 125–136, 2006.