Chapter 6
GA-Based RBF Neural Network for Nonlinear SISO System

Radial basis function (RBF) neural network is efficient to model nonlinear systems with its simpler network structure and faster learning capability. The temperature and pressure modeling of the coke furnace in an industrial coke equipment is not very easy due to disturbances, nonlinearity, and switches of coke towers. To construct the temperature and pressure models in a coke furnace, RBF neural network is utilized to improve the modeling precision. Moreover, the shortcoming of RBF neural network, such as over-fitting is overcome. Moreover, the improved RNA-GA, MOGA, and PCA-based NSGA-II are utilized to optimize both the structure and parameters of the RBF network. Encoding/decoding, genetic operations, and fitness functions are designed to obtain satisfying modeling performances. The industrial data sets in the industrial coke furnace are utilized to construct the RBF neural network model by using three modeling optimization strategies.

6.1 Introduction

Since Broomhead and Lowe, proposed0 Radial basis function (RBF) neural networks in 1988 [1], RBF networks have attracted a lot of interests to application research in various fields because of the partial response character of the neuron, better approximation capability, simpler network structure, and faster learning capability than other artificial neural networks (ANNs) [2–4]. However, how to design radial basis functions remains a critical issue for RBF networks. The number and parameters of radial basis functions control the structure complexity and the generalization capability of RBF networks. A RBF network with too few radial basis functions gives poor generalization on new data because of the limited flexibility, while a RBF network with too many radial basis functions yields poor generalization since it is too flexible and may fit the noise in the training data. The best generalization performance can be obtained via a compromise between the conflicting requirements of reducing prediction error while simultaneously decreasing model complexity [5, 6]. This trade-off

J. Tao et al., *DNA Computing Based Genetic Algorithm*,
https://doi.org/10.1007/978-981-15-5403-2_6

highlights the importance of optimizing the structure complexity of the RBF network to improve its generalization capability.

More specifically, the network structure of the RBF network needs to be given before training other parameters in the neural network. The procedures usually proceed in two steps: First, the centers of radial basis functions are determined by a clustering method; second, the final-layer weights are calculated by the least square method. Usually, an unsupervised method that is separated from the actual objective of minimizing the modeling error will be executed in the first stage. The structure optimization in the construction of the network is desirable, however, it is a rather difficult problem and cannot be easily solved by the standard optimization method [7].

An interesting alternative for solving this complicated problem can be offered by the recently developed evolutionary algorithms. Perhaps the most popular and successful strategies are the genetic algorithms (GAs), which have succeeded in the structure selection of several kinds of neural networks, such as, Back propagation (BP) neural networks [8, 9] and recurrent neural networks [10, 11], etc. As for RBF neural networks, Vesin et al. used a GA to solve the whole optimization problem of the RBF network, but the centers of the potential nodes were restricted among the training data set [12]. Esposito et al. employed a GA-based technique for the determination of the widths of Gaussian radial basis functions [13], while Sarimveis et al., utilized a GA approach for optimizing RBF network only based on prediction errors [14].

When RBF networks are used to model the nonlinear system, the learning algorithm of the RBF network to determine its structure and parameters is critical, because different learning algorithms have a great influence on the performances of the derived RBF-based models.

Studies on parameter learning algorithm and the network structure optimization have been developed in-depth. Huang et al., proposed a simple sequential learning algorithm for RBF neural networks, which is referred to as the RBF growing and pruning algorithm [15]. Du et al., proposed a multi-output fast recursive algorithm (MFRA) that formulates the construction of an RBF network as a linear parameter optimization problem [16]. Han et al., presented a flexible structural radial basis function (FS-RBF) neural network, which changed its structure dynamically in order to maintain the prediction accuracy [17]. Most previous algorithms will become inefficient with too large search space and trap into the local minimum.

Although GA is a global searching algorithm, it is challenged by its weak local-search capability and premature convergence. As such, some biological operations at the gene level are effectively adopted in SGA, and the global searching speed can be largely improved [18, 19]. Moreover, the pruning operation is introduced to simplify the structure of the RBF neural network. In addition, the fewest process variables for accurate modeling are often of great interest by means of the most relevant variables selection, thus, the modeling, control, optimization, and monitoring issues for quality improvement of industrial production will be much easier [20–22]. Hence, it is anticipated that prediction accuracy can be improved by variable selection techniques, which will reduce the model complexity and capture the nature of industrial processes better [23–26].

Research on various variable selection methods for ANNs has been developed continuously. Huang et al., utilized the least absolute shrinkage operator for the input variables selection of a multilayer perceptron neural network in nonlinear industrial processes [27]. A sequential backward multiplayer perceptron (SBS-MLP) was proposed to perform feature selection [28]. Souza et al., have considerably reduced the computational cost and improved the model accuracy by variable selection comparing with SBS-MLP [29]. Estévez et al., proposed an improved variable selection method by introducing the average normalized mutual information for the measurement of redundancy [30].

The variable selection using principal component analysis (PCA) has also been studied in recent years [31–33]. However, the principal components are obtained by the linear combination of all variables, which makes the interpretation of principal component variable quite difficult. Therefore, a variety of criterion functions, such as Similarity indices, RM criterion, RV criterion, Generalized Coefficient of Determination (GCD) criterion have been proposed for subset selection [34]. In addition, the heuristics algorithm [35], simulated annealing [36], stochastic approximation iteration [37], genetic algorithm [38], etc., have also been applied to select the variables. Though some variable selection methods are efficient in the literatures [31–35, 37–39], most of them are not included in system modelling, while the variable selection in neural network only considered modeling accuracy [27].

Coking is an important process to improve economic benefits and has been widely used for refineries [40, 41]. A coke unit usually consists of coke furnaces, fractionating towers and coke towers. The temperature control of the coke furnaces is one of the operation goals in the unit, due to the coke furnaces, fractionating towers and coke towers in a completed process stream with their dynamic characteristics interacting with one another, the tasks are complicated. For example, the temperature affects the coking rate in the tubes of coke towers, which in turn has an impact on the temperature in the furnace [42]. Modeling is very important for advanced controller design but is even more difficult in term of the nonlinear characteristics, time delay and other various disturbances, such as feeding quantity, feeding temperature, fuel amount, etc. One of the most serious disturbances is the switches of coke towers, which disturb the temperature periodically and cause severe temperature fluctuations.

In this chapter, the structure optimization is included, and the fitness value of each chromosome is calculated based on the prediction error and the structure complexity criterion. In order to simplify the optimization of the RBF network, thin-plate-spline function can be chosen as the radial basis function, which is not required to determine its widths. However, the Gaussian function may obtain better performance with suitable centers. Generally, the RBF centers are determined based on a self-organizing clustering process, such as k-means clustering, the nearest neighbor clustering. The application of the above algorithms requires the network structure to be selected through trial and error, and only the input data is considered. Herein, several RBF neural network optimization methods are given as follows:

First, a pruning operator is designed to simplify the RBFNN structure, and a RNA-GA is first developed to optimize the RBF neural network structure and its corresponding parameters of radial basis functions to improve the approximation and generalization performance of RBFNN for temperature modeling in a coke furnace [7].

Second, the structure of the input and hidden layers, the parameters of the Gaussian basis functions are encoded in a chromosome. The local search operator and the prolong operator are proposed to obtain multiple RBF neural network structure. And an improved MOEA is then designed for the RBFNN modeling of the chamber pressure [43].

Finally, PCA variable selection is combined with ANN for nonlinear system modeling, and an RV criterion function of PCA is used to select the effective variables. Since both RV criterion and modeling accuracy are considered, the multi-objective evolution algorithm (MOEA) is adopted. Among MOEAs, NSGA-II is adopted due to its popularity and efficiency in solving ANN optimization and modeling problems [44, 45]. Here, it is also used to solve the variable selection and ANN modeling problem.

6.2 The Coke Unit

The whole process flow is shown in Fig. 6.1. It consists of such equipment as one fractionating tower (T102), three coke furnaces (F101/1, 2, 3), and six coke towers (T101/1, 2, 3, 4, 5, 6). The detailed flow of each part of the unit is shown in Fig. 6.2, its main job is to coke residual oil. Take furnace (F101/3) as an example, the process flow is as follows: The flow of residual oil is divided into two branches (FRC8103, FRC8105) and sent into the convection chamber of the furnace (F101/3) to be heated to about 330 °C, then the two branches are combined and flow out of the radiation chamber of the furnace and go to the fractionating tower (T102) for heat exchange with gas oil from the coke towers (T101/5, 6). After heat exchange, the heavy part of both residual oil and the gas oil join together, which is called circulating oil. The circulating oil is then divided into two branches (FRC8107, FRC8108) by pumps (102/1, 2, 3) and returned to enter the radiation chamber of the furnace (F101/3) to be heated to about 495 °C. Finally, the two branches join together and go to the coke towers (T101/5, 6) to remove coke. This process is called the coking of residues. The flows of the other two furnaces are the same as that of the furnace (F101/3), but the corresponding coke towers are different. The coke towers (T101/1, 2) are for the furnace (F101/1) and (T101/3, 4) for the furnace (F101/2). Each time, only one of each pair of coke towers works for its corresponding furnace, and when it is full, the other one replaces it. This replacement is called the switch of coke towers and the procedure recycles. The switch time of three pairs of coke towers is different. The heat exchange with gas oil from the coke towers poses a continuous disturbance on the outlet temperature because of the volume of the gas oil from the coke towers. During the switch of the coke towers, the outlet temperature of the furnace often

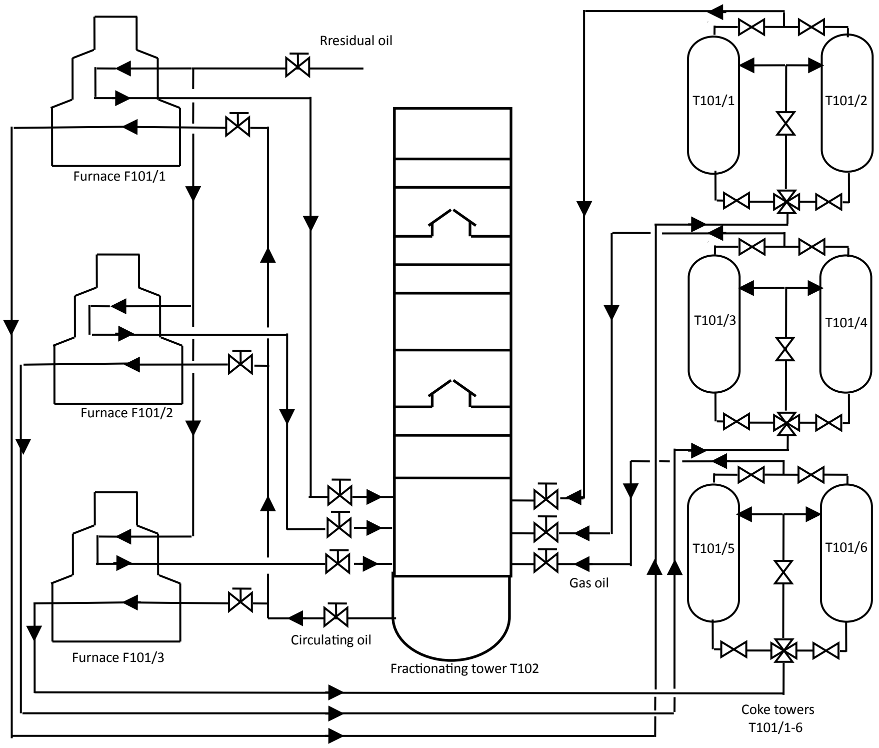


Fig. 6.1 Overall flow of coke unit

drops and rises sharply because some of the oil in it will become gas oil, and part of the inlet gas oil flowing into it will be used for the heating of coke towers. What's more, the random switch time of three pairs of coke towers adds to this serious problem.

The outlet pressure, temperature, and relevant variables are sampled using the experimental equipment CENTUM CS3000 Distributed Control System (DCS), as shown in Fig. 6.3. The DCS has a database, namely PAI database, for process data acquisition. To ensure the modeling precision and make the administrator conveniently analyze the process data, the sampling period 0.5 s is set in the PAI database, and 2 digits after the decimal point are retained in the sampling dataset.

6.3 RBF Neural Network

A schematic of the RBF network with n inputs and a scalar output is shown in Fig. 6.4.

In the RBF neural network (RBFNN), the function form $\varphi(\cdot)$ and the centers $\mathbf{c}_i$ are assumed fixed. Here we denote a set of the inputs $\mathbf{x}(k)$ as $\mathbf{x}(k) =$

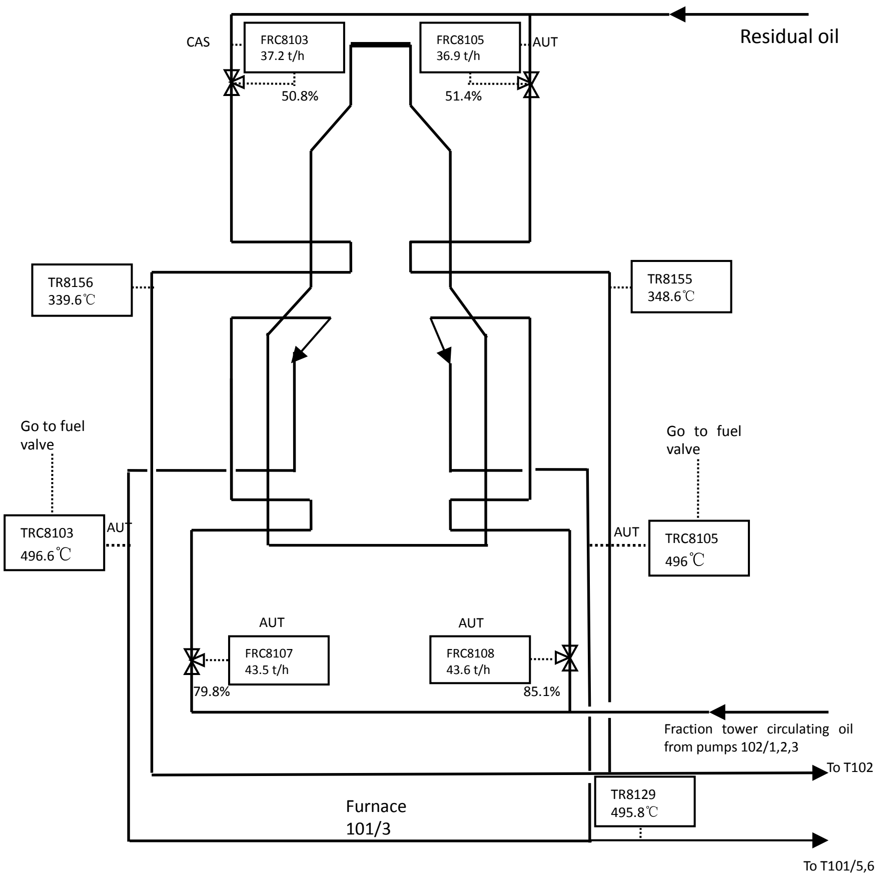


Fig. 6.2 Overall flow of coke furnace

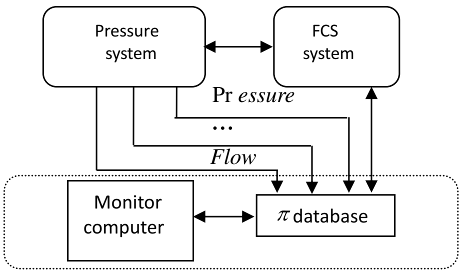
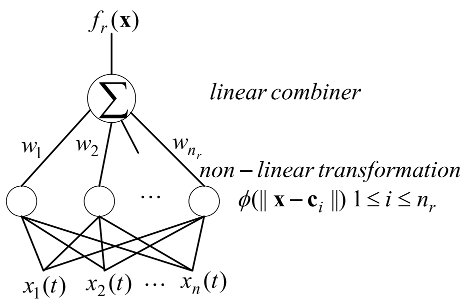


Fig. 6.3 Data acquisition configuration for system outputs



$f_r(\mathbf{x})$
$\sum$
linear combiner
w_1
w_2
w_{n_r}
non − linear transformation
$\phi(\| \mathbf{x} - \mathbf{c}_i \|)\ 1 \le i \le n_r$
$x_1(t)$ $x_2(t)$ $\cdots$ $x_n(t)$

Fig. 6.4 Schematic of RBF network

$[y(k-1), \ldots, y(k-n), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-m)]$, $\mathbf{u}(k)$ as the selected manipulated variables evaluated by RV criterion of PCA, $\hat{y}$ as the output of RBFNN, and $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_{n_r}]$ as the weights between the hidden layer and the output layer, where n_r is the number of the nodes in the hidden layer. The choices of $\varphi(\cdot)$ and $\mathbf{c}_i$ must be carefully considered for the RBF neural network to obtain both the approximation capability and generalization performance. The thin-plate-spline function and the Gaussian function are two typical choices, both of them have obtained good approximation capabilities according to the fitting result of RBF networks [44].

Here, $\phi_i(\mathbf{x})$ is the ith neuron output in the hidden layer, which is selected as the Gaussian function or thin-plate-spline function:

$$\phi_i(\|x\|) = \exp\left(-\frac{\|x - c_i\|}{\sigma_i^2}\right) \text{ or } \phi_i(\|x\|) = \|x - c_i\|^2 \log\|x - c_i\|,\ i = 1,\ 2, \ldots,\ n_r \tag{6.1}$$

where $\|\mathbf{x} - \mathbf{c}_i\|$ is the Euclidean distance between $\mathbf{x}$ and $\mathbf{c}_i$, $c_i \in \Re^{n+m}$ is the center vector and $\sigma_i \in \Re$ represents the spread of radial basis function, respectively.

The prediction of RBFNN, $\hat{y}(k)$, can be expressed as a linear weighted sum of n_r hidden functions

$$y(\mathbf{x}(k)) = \sum_{i=1}^{n_r} \omega_i \phi_i(\|\mathbf{x}(k)\|) = \omega \Phi(k) \tag{6.2}$$

where $\boldsymbol{\Phi} = [\phi_1, \cdots, \phi_{n_r}]^T$. Given N_1 samples of training data, $\mathbf{Y}_1 = [y_1(1), \cdots, y_1(N_1)]$ and $\mathbf{U} = [\mathbf{u}(1), \cdots, \mathbf{u}(N_1)]$, the weight coefficients can be calculated by recursive least squares (RLS) method [20]

$$\begin{cases} \boldsymbol{\omega}(k) = \boldsymbol{\omega}(k-1) + \mathbf{K}(k)[f_i(k) - \boldsymbol{\Phi}^T(k)\boldsymbol{\omega}_i(k-1)] \\ \mathbf{K}(k) = \mathbf{P}(k-1)\boldsymbol{\Phi}(k)[\boldsymbol{\Phi}^T(k)\mathbf{P}(k-1)\boldsymbol{\Phi}(k) + \mu]^{-1} \\ \mathbf{P}(k) = 1\big/\mu[\mathbf{I} - \mathbf{K}(k)\boldsymbol{\Phi}^T(k)]\mathbf{P}(k-1) \end{cases} \tag{6.3}$$

where $0 < \mu < 1$ is the forgetting factor, $\boldsymbol{P}(k)$ is a positive definite covariance matrix, $\boldsymbol{P}(0) = \alpha^2\boldsymbol{I}$, and $\boldsymbol{I}$ is an $(n+m) \times (n+m)$ identity matrix, α is a sufficiently large real number set to 10^5 and $\boldsymbol{\omega}(0) = \boldsymbol{\varepsilon}$, and $\boldsymbol{\varepsilon}$ is a sufficiently small $n + m$ real vector as set to 10^{-3}, $\mathbf{K}(k)$ is a weight matrix.

6.4 RNA-GA Based RBFNN for Temperature Modeling

By giving a set of the inputs $\mathbf{x}(t)$ and the corresponding output $y(t)$ for $t = 1$ to N_1, the weights of RBFNN can be derived using RLS method in 0.3. However, the number of neuron nodes in the input and hidden layer will determine the structure complexity of RBFNN, and the parameter selection of radial basis function is quite important in order to obtain a good approximation capability. The better modeling capability with a simpler structure was tried to be obtained by an improved RNA-GA. Since the encoding/decoding method and the genetic operations will affect the efficiency of GA, this section is focused on the optimization of the RBF network by the RNA-GA.

6.4.1 Encoding and Decoding

Select the Gaussian function as the radial basis function, σ_i, $\mathbf{c}_i$, and the number of hidden nodes of RBFNN of the lth chromosome is shown as follows:

$$\mathbf{C}_l = \begin{bmatrix} c^l_{1,1} & c^l_{1,2} & \cdots & c^l_{1,n} & \sigma_1 \\ c^l_{2,1} & c^l_{2,2} & \cdots & c^l_{2,n} & \sigma_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c^l_{n_r,1} & c^l_{n_r,2} & \cdots & c^l_{n_r,n} & \sigma_{n_r} \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.4}$$

where $l = 1, 2, \ldots N$, N is the population size, n_r is generated randomly between 1 and D, D is the maximal number of hidden neurons. The rows between $[n_r + 1, D]$ are set to zeros and do not correspond to a center. The number of neurons in the input layer (n) is generated randomly between 2 and 5. Since the choice of the input layer is limited, it then uses the enumerated method during the optimization process. There are entirely $D \times (n + 1)$ real parameters to be optimized in the RBFNN, which means one chromosome should represent $D \times (n + 1)$ real number. The elements of $\mathbf{C}_l$ are then encoded by 0123/CUAG as shown in Fig. 6.5.

The parameters of the chromosome can be decoded by using the following equations:

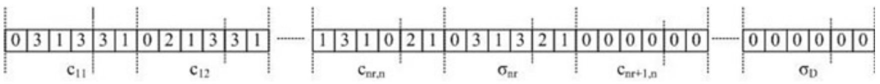


Fig. 6.5 Quanternary encoding for $\mathbf{C}_l$

$$c_{ij} = x_{j,\min} + \frac{x}{4^L - 1} \cdot \left(x_{j,\max} - x_{j,\min}\right), \quad 1 \le i \le n_r, \quad 1 \le j \le n \tag{6.5}$$

$$\sigma_j = \frac{x}{4^L - 1} w_{\max} \tag{6.6}$$

where x is the integer decoded by quaternary encoding with the encode length L, $x_{j,\min}$ and $x_{j,\max}$ are the minimum and maximum values of input variables given in the problem, $w_{\max}$ is the maximum width of Gaussian basis function.

6.4.2 Fitness Function

The training procedures using the improved RNA-GA (IRNA-GA) are processed in two steps. First, the network structure and the parameters of radial basis functions are determined by the chromosomes in an individual. Second, the final-layer weights are calculated by the RLS method, because IRNA-GA is a random search algorithm, and the constructed RBFNN system maybe ill-conditioned. Hence, the ordinary least square method cannot be applied in the optimization procedure. At each generation of IRNA-GA, the calculation of the weights in the output layer completes the formulation of N RBFNN, which can be expressed by the pairs $(\mathbf{C}_1,\ \mathbf{w}_1),\ \ (\mathbf{C}_2,\ \mathbf{w}_2), \ldots (\mathbf{C}_N,\ \mathbf{w}_N)$.

To obtain good generalization capability of RBF network, the sampled data set is divided into 3 groups, where one group of data subset $(\mathbf{X}_1, \mathbf{Y}_1)$ are used to calculate the weights of the final layer, the second group $(\mathbf{C}_1, \boldsymbol{\omega}_1)$ are utilized to evaluate the modeling performance of RBFNN at each generation, and the third group $\cdots (\mathbf{C}_N,\ \boldsymbol{\omega}_N)$ is used to verify the modeling performance of the optimal RBF network. This scheme incorporates a testing procedure into the training process and ensures good generalization performance of RBFNN. However, to obtain a better approximation capability with a simpler structure and avoid neural network overfitting, the objective function considering both the approximation capability and structure complexity is shown as follows:

$$J(\mathbf{C}_i, \mathbf{w}_i) = \sum_{t=1}^{N_1} |Y_1(t) - \hat{Y}_1(t)|^2 + \sum_{t=1}^{N_2} |Y_2(t) - \hat{Y}_2(t)|^2 + \lambda(n_r + n) \tag{6.7}$$

It can be seen that a compromise has been made between the modeling errors and the complexity of network structure. Here, λ is a coefficient between 0 and 1, and the bigger λ is, the more complicated the structure of RBFNN.

6.4.3 Operators of RBFNN Optimization

Li et al. has summarized various operations of DNA computing, such as elongation operation, deletion operation, absent operation, insertion operation, translocation operation, transformation operation, and permutation operation, etc. [46]. In addition to selection, crossover, and mutation operators, other appropriate operations of DNA computing can also be adopted to improve the performances of RBFNN modeling.

(1) Selection operator

A set of individuals from the previous population must be selected for reproduction depending on their fitness values. Individuals with bigger fitness value have more probability to survive. There exist several types of selection operators, and Roulette wheel method is applied to produce the parents of crossover and mutation operators. The probability of an individual being selected, $P(\mathbf{C}_i)$, is given by

$$P(\mathbf{C}_i) = \frac{f(\mathbf{C}_i)}{\sum_{l=1}^{N} f(\mathbf{C}_i)} \tag{6.8}$$

where $f\mathbf{C}_i$ is the fitness value of an individual $\mathbf{C}_i$ by using reciprocals of Eq. (6.7), i.e., $1/j(\mathbf{C}_i\mathbf{W}_i)$. The roulette wheel is placed with N equally spaced pointers. A single spin of the roulette wheel will simultaneously pick N individuals of the next population.

(2) Crossover operator

The crossover operator is executed with the crossover probability p_c among the selected individuals, and generates new structure and the parameters of RBFNN. If the randomly generated number is less than p_c, crossover operation is carried out between the current chosen individual $\mathbf{C}_l$ and the next individual $\mathbf{C}_{l+1}$, and yields the offspring chromosomes $\mathbf{C}'_l$, $\mathbf{C}'_{l+1}$. Since the number of input neurons n is fixed during an optimization process, the procedure is illustrated with an example presented in Fig. 6.6, which includes a scheme of the multi-point crossover operation, where the crossover points are generated randomly between 1 and L. The operator is prone to generate more hidden neurons, e.g., after the crossover of $c_{\mathrm{nr}+1,\mathrm{n}}$ of $\mathbf{C}_l$ and $c_{\mathrm{nr}+1,\mathrm{n}}$ of $\mathbf{C}_{l+1}$, the new nonzero chromosomes are generated, and the number of the hidden nodes in $\mathbf{C}'_l$ becomes $n_r + 1$.

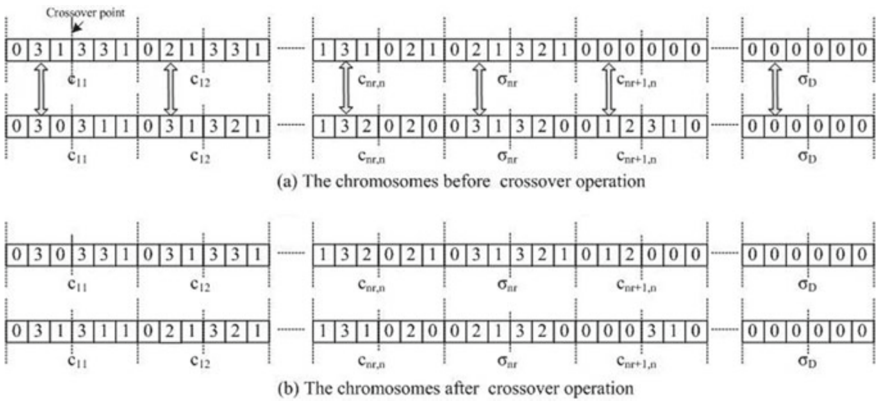

Crossover point
(a) The chromosomes before crossover operation
(b) The chromosomes after crossover operation

Fig. 6.6 Example of the crossover operation

(3) Mutation operator

To have a better exploration of the search space, the mutation operator is implemented. Because there exist four elements (0123/CUAG) in RNA sequence, the mutation of the nucleotide base is relatively complex. Three mutation operations on a single RNA sequence, i.e., reversal, transition, and exchange operations are adopted. The reversal operator makes $0 \leftrightarrow 2$, $1 \leftrightarrow 3$, transition operator makes $0 \leftrightarrow 1, 2 \leftrightarrow 3$, and exchange operator makes $2 \leftrightarrow 1, 0 \leftrightarrow 3$. When the element of an individual is mutated with a probability p_m, three mutation operators are executed simultaneously. This will generate more than N individuals after mutation operators, but the population size still remains invariant after selection operator.

The mutation probability is critical and generally small since too large mutation probability makes RNA-GA become a random search algorithm. At the beginning stage of the evolution process, larger probability of mutation is assigned to explore the larger feasible region. When the region of the global optimum is found, the mutation probabilities are reduced to prevent better solutions from disruption. Therefore, the dynamic mutation probability p_m is described as follows:

$$p_m = a_0 + \frac{b_0}{1 + e^{aa(g-g_0)}} \tag{6.9}$$

where a_0 denotes the initial mutation probability of p_m, b_0 is the variation range of mutation probability, g is the evolution generation, g_0 decides the generation where a great change of mutation probability occurs, and aa denotes the speed of change. The coefficients of Eq.(6.9) are selected as follows: $a_0 = 0.02$,$b_0 = 0.2$, $g_0 = G/2$, $aa = 20/G$. Let G be 1000, the probability curve $\mathbf{C}_l$ changing with evolution generation is shown in Fig. 6.7.

After calculating the mutation probability in terms of 0.9, $L \times N$ decimal fractions between 0 and 1 are produced compared with the above dynamic mutation probability.

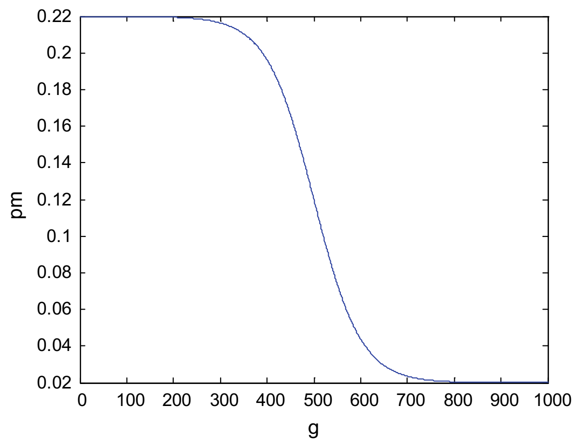


Fig. 6.7 Mutation probability decreasing with g increasing

If the decimal fraction is less than the corresponding probability in Fig. 6.7, 3 RNA mutation operators are executed meanwhile, and 3 new individuals will be produced.

(4) Pruning operator

Since the chromosomes are generated randomly, the effectiveness of every hidden neuron is evaluated in terms of the active firing (AF) of the hidden neurons, which is described as follows [17]:

$$Af_i = \rho e^{-\|\mathbf{x}-\mathbf{c}_i\|} \frac{\phi_i(\mathbf{x})}{\sum_{i=1}^{n_r} \phi_i(\mathbf{x})}, i = 1, \ldots, n_r \tag{6.10}$$

where Af_i is the active firing of the ith hidden neuron, $\phi_i(\mathbf{x})$ is the output of the ith hidden neuron, $\rho > 1$ is a positive constant, which is set as 100. When Af_i is less than the activity threshold Af_o ($0.05 < Af_o < 0.3$), the hidden neuron i is regarded as an inactive neuron. The number of the hidden neurons (n_r) will be decreased and the corresponding $\boldsymbol{c}_i$ is moved to the last location of $\boldsymbol{c}_{nr,}$ its values of chromosomes are then set to zeros.

6.4.4 Procedure of the Algorithm

The fitness function evaluation, selection, crossover, mutation, and pruning operators are described for RNA-GA to be appropriate to optimize RBFNN, the running procedure is given in the following steps.

Step 1: Generate input layer with n inputs, $\boldsymbol{x}$(t), which consists of $\langle n/2\rangle$ system input (u) and n–$\langle n/2\rangle$ previous values of system output (y). Here $\langle\cdot\rangle$ is to round the elements to the nearest integer. As an example of 3 inputs, 2 inputs $u(k)$, $u(k–1)$ and 1 previous system output $y(k–1)$ are produced, that is, $\boldsymbol{x}(t) = [u(k), u(k\text{-}1), y(k\text{-}1)]$.

Step 2: Generate randomly N quaternary encoding chromosomes with a length of $D \times L$ in the search space, where N is the population size.

Step 3: Decode and compute the performance J of each individual.

Step 4: Select the chromosomes to generate N new chromosomes as the parents of the next generation by tournament selection operator. Before selection operator, the best $\langle 3\,N/4 \rangle$ individuals and the worst $\langle N/4 \rangle$ individuals are derived to make up of N individuals to keep population diversity.

Step 5: Judge if the crossover probability is satisfied, if yes, select one point randomly in l quaternary genes, and totally Dn points are generated as shown in Fig. 6.6, and exchange the codes of $\mathbf{C}_l$ and the next individual $\mathbf{C}_{l+1}$. Repeat this for all the $p_c \times N/2$ pairs of parents produced at step 4.

Step 6: For effective mutation, execute 3 RNA mutation operators once the random number is less than the dynamic mutation probability in 0.7, and this step may generate the individuals more than N.

Step 7: If the number of individuals is greater than N, the pruning operator is performed to improve the quality of RBFNN, else the pruning operator is not carried out.

Step 8: Repeat steps 3–7 until a termination criterion is met, that is, the maximal evolution generation (G). Moreover, elitism, the inclusion of the best individual in the next population is used throughout the optimization procedure.

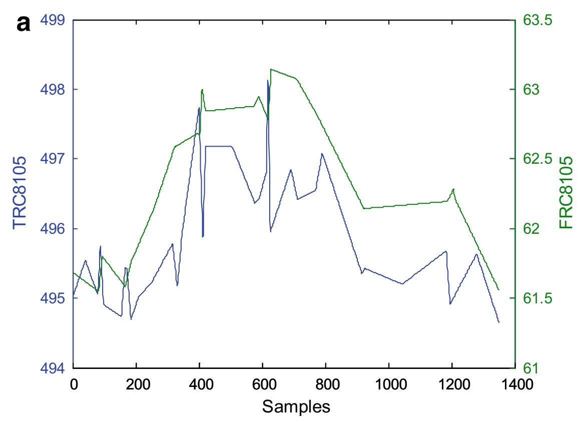
Step 9: Increase the number of the input nodes and repeat steps 2–8. Choose the best RBFNN in terms of the value of an objective function using the test data set ($\mathbf{X}_3$, $\mathbf{Y}_3$).

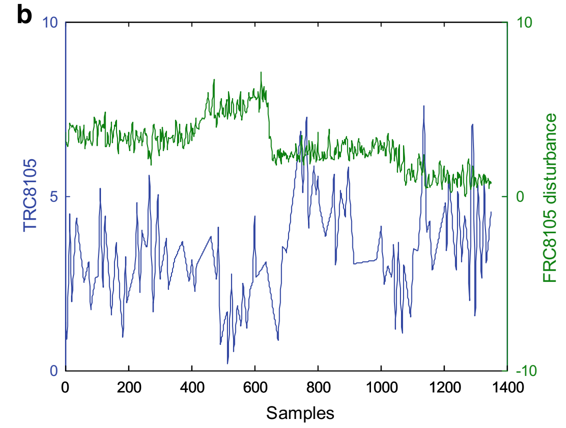
6.4.5 Temperature Modeling in a Coke Furnace

Advanced temperature control is critical for the coke unit and the first important issue to advanced controller design is system modeling. In this section, RBFNN optimized by the IRNA-GA is used to construct the north and south sides of the temperature models and the main disturbances in the coking furnace.

The experimental data are collected from the industrial coking unit of a refinery controlled by CENTUM CS3000, which is described in Sect. 6.2. The temperature is measured by thermocouple with the measuring precision ±1.5°C. The flow rate is measured by the mass flowmeter. There are totally 1350 data sampled from PAI database of the control system. Each measurement sample includes four inputs and four outputs, that is, the north side primary channel model of the outlet temperature (TRC8105) and the input fuel flow (FRC8105), its disturbance channel model of the perturbation of FRC8105 and its corresponding temperature perturbation of TRC8105, the south side primary channel model of the outlet temperature (TRC8103) and the input fuel flow (FRC8103), and its disturbance channel model of the perturbation of FRC8103. Four groups of input and output data are plotted from Fig. 6.8a–d,

Fig. 6.8 **a** Input FRC8105 and output TRC8105 for north side primary channel modeling. **b** Input FRC8105 perturbation and output TRC8105 for north side disturbances modeling. **c** Input FRC8103 and output TRC8103 for south side primary channel modeling. **d** Input FRC8103 perturbation and output TRC8103 for south side disturbances modeling





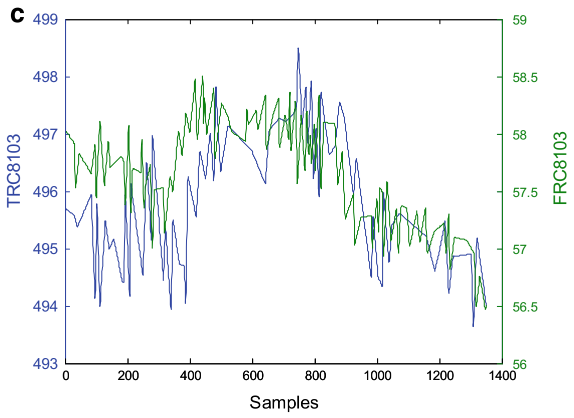
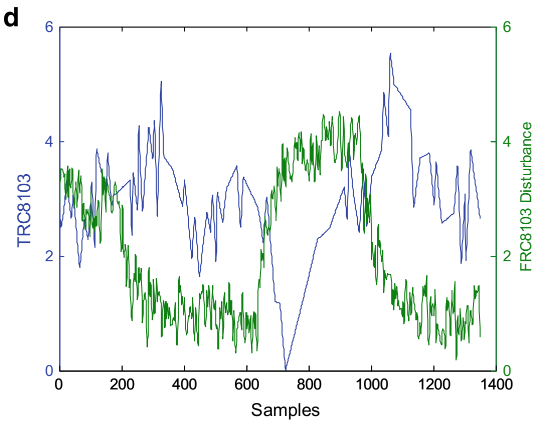


Fig. 6.8 (continued)



where the x-axis is the number of samples, y-axis labeling on the left is the system output, and the right one is the system input.

All collected 1350 samples are divided into three groups. The first group of 450 samples is selected as the training set, and the intermediate 450 samples are used to verify the generalization capability of RBFNN, the remaining 450 samples are used as the final testing set. Based on the three sets of data, the IRNA-GA is employed to optimize the structure and parameters of RBFNN by minimizing 0.7. Here, the parameters of the IRNA-GA are set as follows: the population size N is 60, the maximal evolution generation G is 1000, the individual length L is $3 \times D$, the probability of crossover operator p_c is 0.6, the mutation probability p_m is dynamically changed according to Eq. (6.9), the activity threshold Af_o is 0.1, and λ is 0.3. To examine the generalization capability of the constructed model, the trained RBFNN is used to predict the coke temperature yield of the testing samples, which are not included in the training data. In addition, for validation of the effectiveness of the random optimization algorithm, RBFNN is trained for 10 times. At each time, the parameters of IRNA-GA and data set are kept invariant. The best results are listed in Table 6.1, where e_1 is Root Mean Squared Error (RMSE) of the testing data.

The IRNA-GA is compared with the k-means method, which is used to train the centers of the RBF network. The pruning operator is also applied and final-layer weights are derived using the RLS method, and the number of the input nodes is the same as the optimized RBFNN. The maximal number of hidden neuron nodes is set to 38, which is also obtained based on the maximal number of hidden nodes

Table 6.1 The simulation results comparison with 2 methods

Methods	TRC8105			TRC8105 disturbances			TRC8103			TRC8103 disturbances		
	n_1	n_2	e_1	n_1	n_2	e_1	n_1	n_2	e_1	n_1	n_2	e_1
IRNA-GA	4	32	0.0094	3	38	0.0439	4	31	0.0305	3	28	0.0813
k-means	4	38	0.0584	3	38	0.3245	4	38	0.2707	3	38	0.0866

optimized by IRNA-GA. From Table 6.1, it can be seen that the best results of IRNA-GA can obtain better prediction precision than using the k-means method in terms of $e_{1.}$ Moreover, RBFNN using IRNA-GA can obtain smaller errors with fewer hidden nodes for the four groups of the testing dataset. Though the RMSE of the TRC8103 disturbance model using IRNA-GA is similar to that of the k-means method, the number of the hidden nodes using IRNA-GA is reduced greatly. All the results in 10 runs are superior to those of the k-means method, because the RBFNN with fewer hidden nodes gains better generalization capability. The simpler structure of RBFNN with higher modeling precision is obtained after running IRNA-GA.

To reflect the prediction accuracy of the established RBFNN model, the predicted temperature is compared with the measured temperature on the testing set for the main channels of the north side and south side (TRC8105, TRC8103) and their disturbance channels, the comparison results are given in Figs. 6.9, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, and 6.24, respectively.

Figure 6.9 shows the predicted yields comparing with measured outputs on the testing set by IRNA-GA, while the corresponding prediction error is plotted in Fig. 6.10. Figure 6.12 shows the fitting curve of the prediction outputs and the measured outputs using the k-means method, and the estimation errors are given in Fig. 6.11. Comparing Figs. 6.9 with 6.11, it can be seen that the maximal modeling error obtained by the k-means method is several times larger than that obtained by IRNA-GA. Similar results can be observed by comparing with the modeling error of TRC8103 and their disturbance models, which are shown Figs. 6.14, 6.16, 6.18, 6.20, 6.22, and 6.24, respectively, it can be seen from Figs. 6.9, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, and 6.24 that the IRNA-GA optimal RBFNN modeling approach has obtained considerably smaller modeling error with simpler network structure.

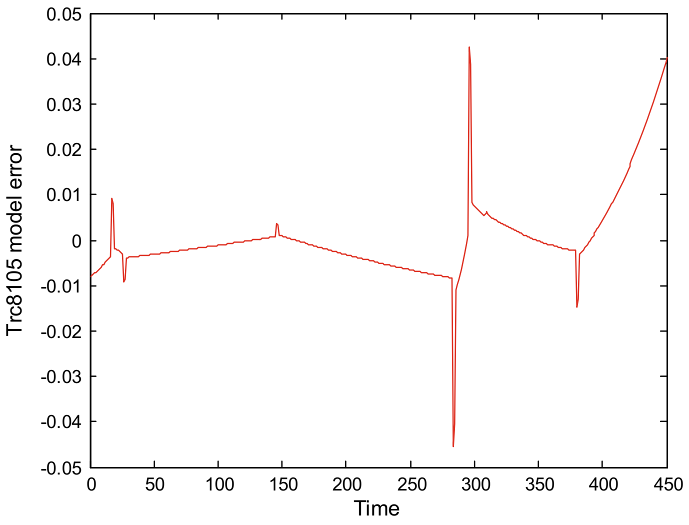


Fig. 6.9 Modeling error of TRC8105 using IRNA-GA

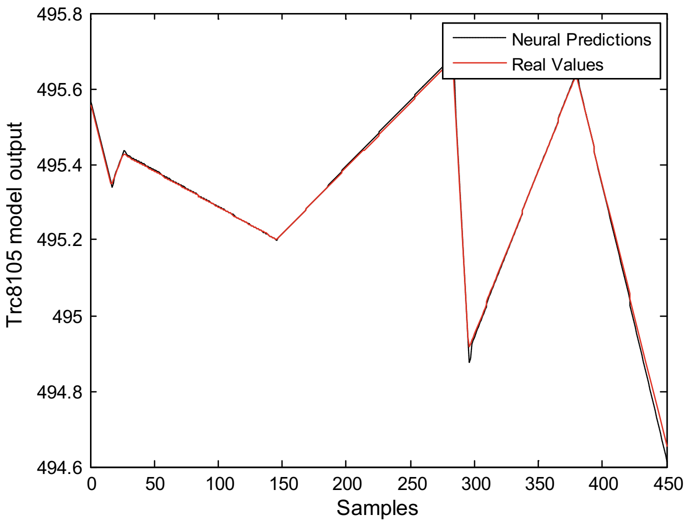


Fig. 6.10 RBFNN model output for TRC8105 using IRNA-GA

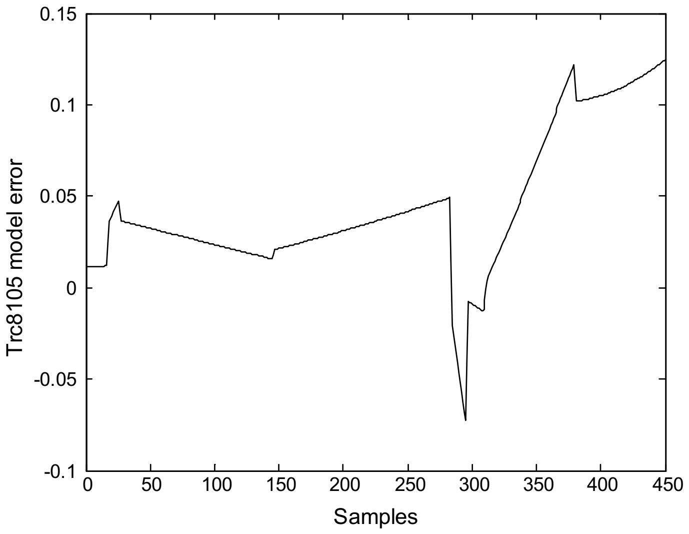


Fig. 6.11 Modeling error of TRC8105 using k-means method

6.5 Improved MOEA Based RBF Neural Network for Chamber Pressure

In Sect. 6.4, RBFNN is optimized by the weighted-sum method in (6.7). In this section, RBFNN is to be optimized by an improved MOEA (IMOEA) considering

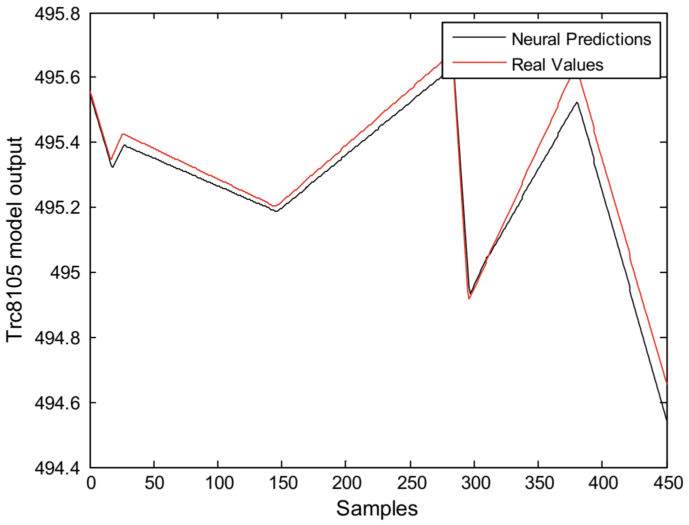


Fig. 6.12 RBFNN model output for TRC8105 using k-means method

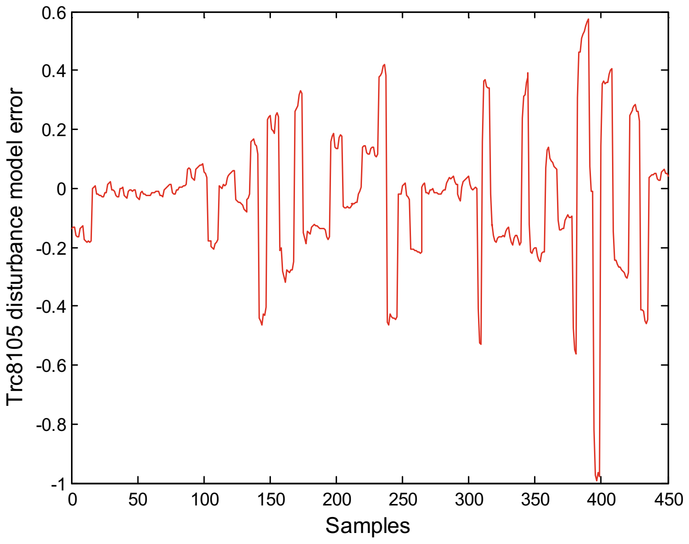


Fig. 6.13 Modeling error for TRC8105 disturbances using IRNA-GA

two objectives: the smallest modeling error and the simplest structure. The encoding method and various operators for the RBFNN structure and parameter optimization are also designed to solve the bi-objective optimization problem.

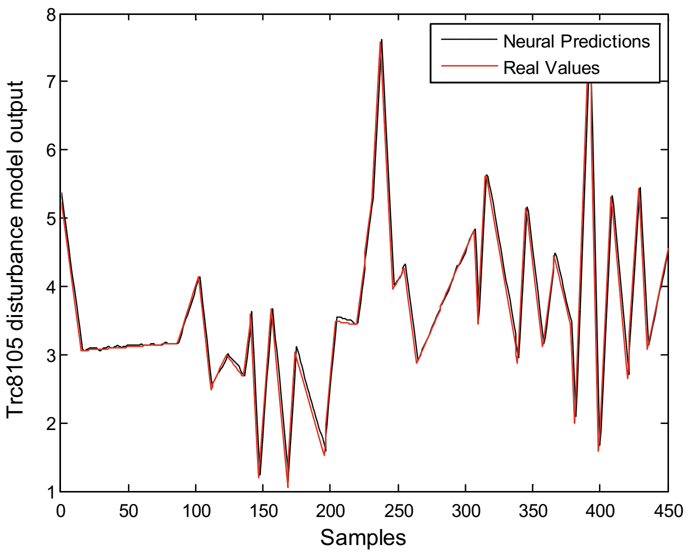


Fig. 6.14 RBFNN model output for TRC8105 disturbances using IRNA-GA

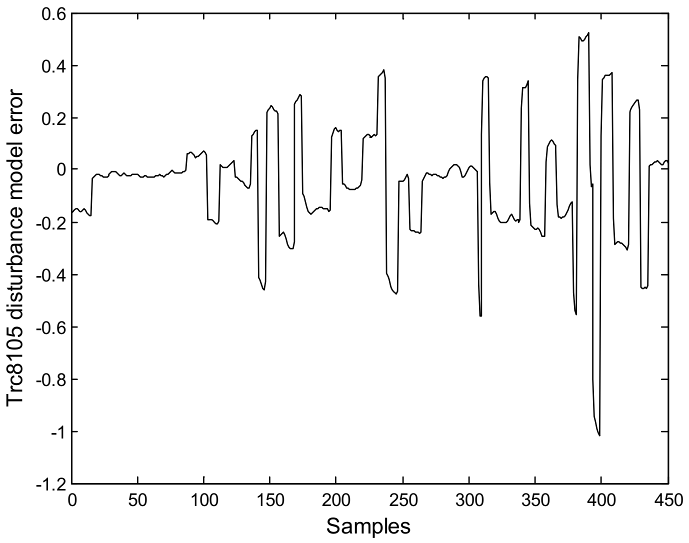


Fig. 6.15 Modeling error of TRC8105 disturbances using k-means method

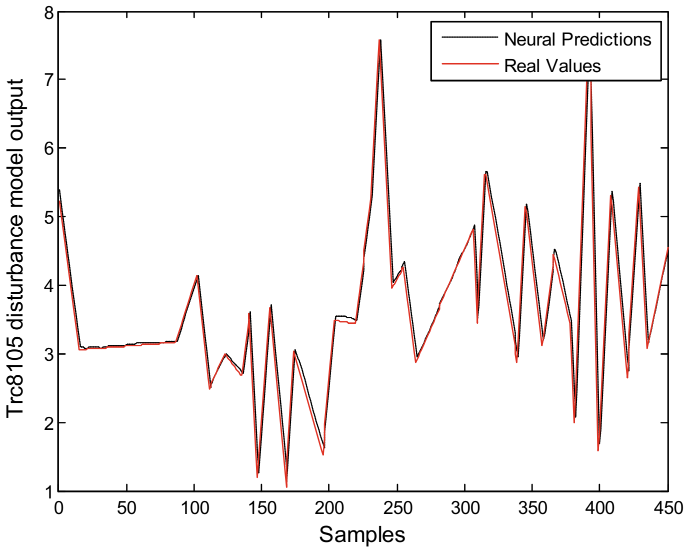


Fig. 6.16 RBFNN model output for TRC8105 disturbances using k-means method

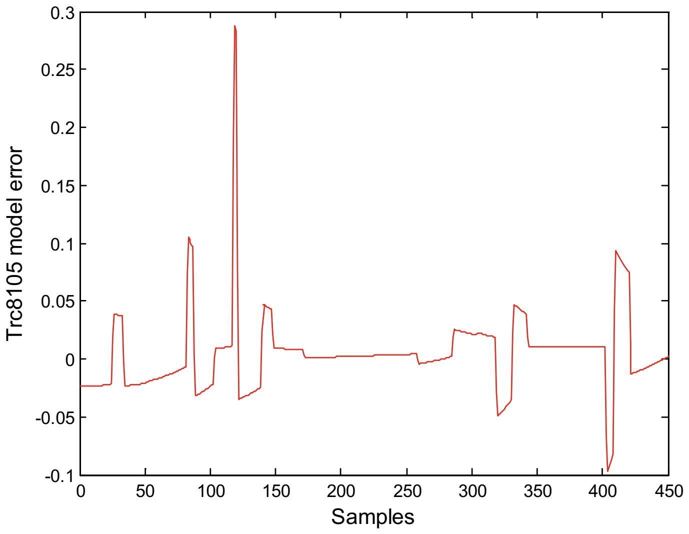


Fig. 6.17 Modeling error of TRC8103 using IRNA-GA

6.5.1 Encoding of IMOEA

Herein, m and n in the input layer, the number of the neurons in the hidden layer n_r and the parameters of the Gaussian functions $\mathbf{c}_i, \sigma_i, i = 1, \ldots, n_r$ are optimized

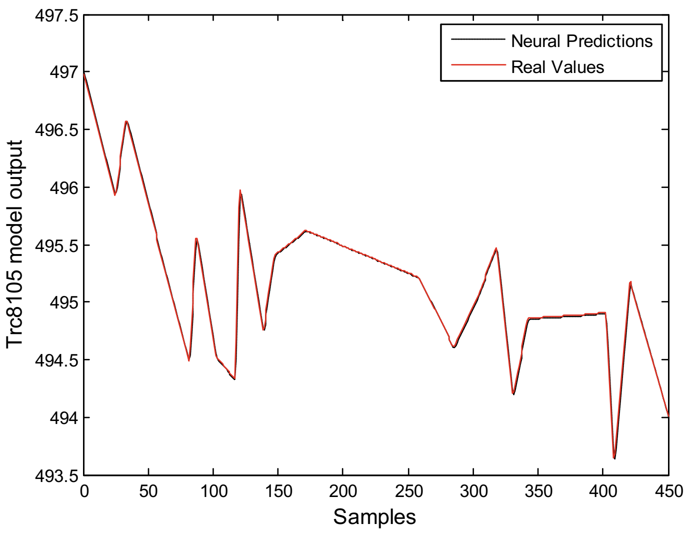


Fig. 6.18 RBFNN model output for TRC8103 using IRNA-GA

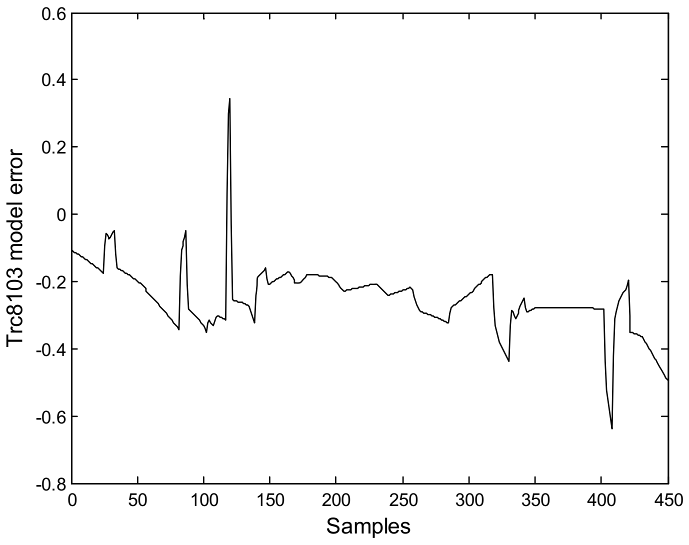


Fig. 6.19 Modeling error of TRC8103 using k-means method

simultaneously. The encoding for all the parameters is designed similarly to Eq. (6.4), and the lth chromosome is given as follows:

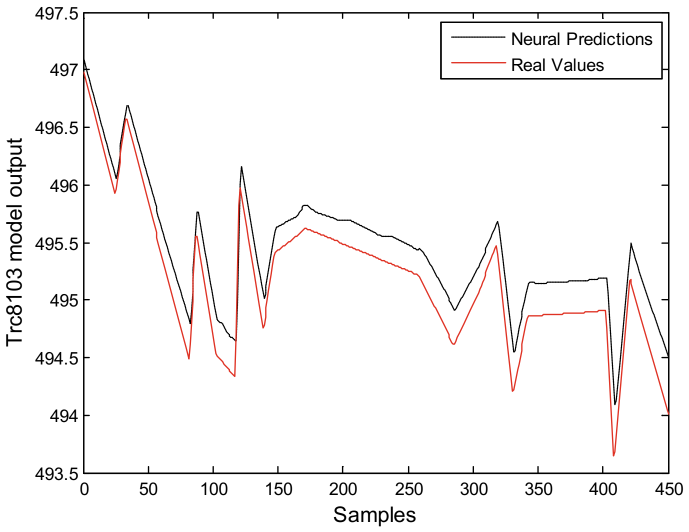


Fig. 6.20 RBFNN model output for TRC8103 using k-means method

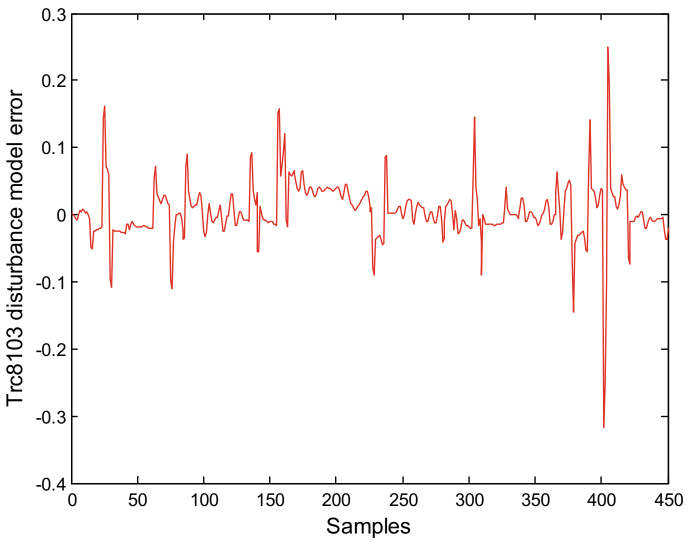


Fig. 6.21 Modeling error of TRC8103 disturbances using IRNA-GA

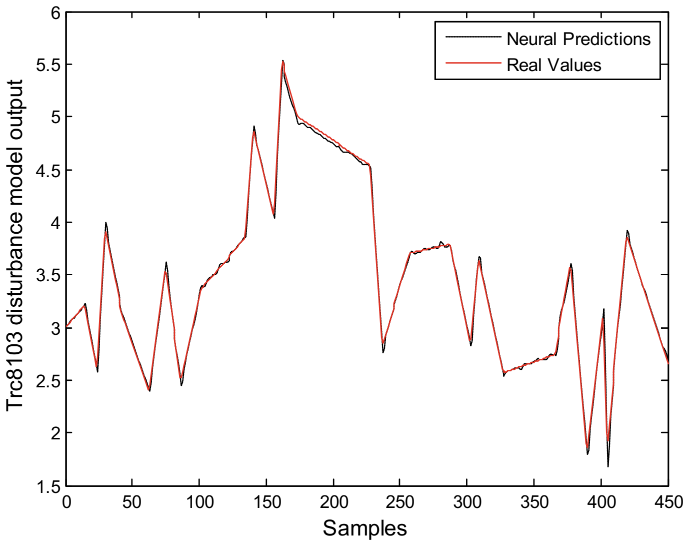


Fig. 6.22 RBFNN model output for TRC8103 disturbances using IRNA-GA

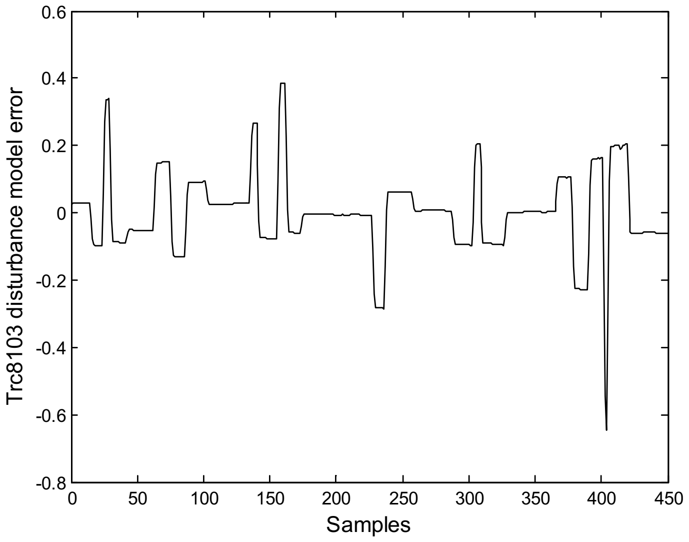


Fig. 6.23 Modeling error of TRC8103 disturbances using k-means method

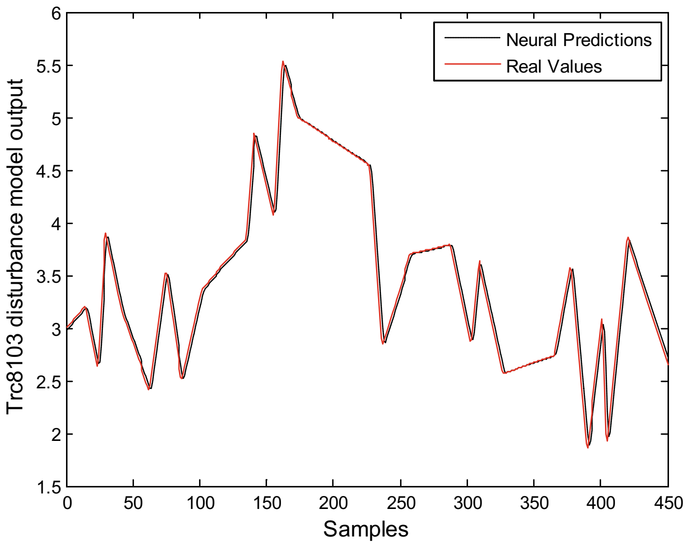


Fig. 6.24 RBFNN model output for TRC8103 disturbances using k-means method

$$\mathbf{C}_l = \begin{bmatrix} c_{1,1} & \cdots & c_{1,n} & 0 & c_{1,N_n+1} & \cdots & c_{1,N_n+m+1} & 0 & \sigma_1 \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{n_h,1} & \cdots & c_{n_h,n} & 0 & c_{n_h,N_n+1} & \cdots & c_{n_h,N_n+m+1} & 0 & \sigma_{n_h} \\ & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \; 0 \end{bmatrix} \tag{6.11}$$

where $l = 1, 2, \ldots N$, m, n, and n_r are limited to $1 \le m \le N_m$, $1 \le n \le N_n$,$1 \le n_r \le D$, respectively. $\boldsymbol{C}_l$ is a $D \times (N_m+N_n+1)$ matrix and the rows below n_r are set to zeros. The columns of $(n, N_n]$ and $(m + N_n, N_m + N_n]$ are also set to zeros. Hence, there are actually $n_r \cdot (m + n + 1)$ parameters to be optimized. m, n, and n_r are first generated randomly among the given range. In Sect. 6.4.1, the number of the input nodes is obtained by using enumeration method, here, it is encoded in Eq. (6.11) and optimized by the evolution algorithm. The elements in $\boldsymbol{C}_l$ can be obtained as follows:

$$c_{ij} = \begin{cases} y_{\min} + r(y_{\max} - y_{\min}) & 1 \le i \le n_r, \quad 1 \le j \le n \\ u_{\min} + r(u_{\max} - u_{\min}) & 1 \le i \le n_r\,, \quad N_n < j \le N_n + m \end{cases} \tag{6.12}$$

$$\sigma_i = r w_{\max} \quad 1 \le i \le n_h \tag{6.13}$$

where r is randomly generated between [0.01, 1], $u_{\min}$ and $u_{\max}$ are the minimal and maximal values of the system inputs, and $y_{\min}$ and $y_{\max}$ are the minimal and maximal values of the system outputs. $w_{\max}$ is the maximal width of the Gaussian basis function that is set to $\max(u_{\max}, y_{\max})$.

Once $\boldsymbol{C}_l$ is generated randomly, the structure and parameters of the RBFNN are determined and the connecting weight vector can be derived by the RLS algorithm by using the training data. N RBFNNs can then be obtained, denoted as $(\mathbf{C}_1, \boldsymbol{\omega}_1), \ldots (\mathbf{C}_N, \ \boldsymbol{\omega}_N)$.

6.5.2 Optimization Objectives of RBFNN Model

Two objectives considering the structure complexity and modeling accuracy of RBFNN are expressed as follows:

$$\text{Min}\begin{cases} f_1 = \sqrt{\sum_{k=1}^{N_1} \left| y_1(k) - \hat{y}_1(k) \right|^2} + \sqrt{\sum_{k=1}^{N_2} \left| y_2(k) - \hat{y}_2(k) \right|^2} \\ f_2 = (m+n)n_r \end{cases} \tag{6.14}$$

We denote $\mathbf{Y}_1 = [y_1(1), \ldots y_1(N_1)]$ as the training data set used to calculate the weight vector $\boldsymbol{\omega}$, $\mathbf{Y}_2 = [y_2(1), \ldots y_2(N_2)]$ as the testing data set, $\hat{\mathbf{Y}}_1 = [\hat{y}_1(1), \ldots \hat{y}_1(N_1)]$ and $\hat{\mathbf{Y}}_2 = [\hat{y}_2(1), \ldots, \hat{y}_2(N_2)]$ as the prediction outputs of the RBFNN. Here f_1 is the modeling accuracy by using the sum of the root of square errors (RSE) for $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$, in which the generalization capability of the RBFNN is involved. f_2 is the structure complexity of the RBFNN by using the product of the number of neurons in the input layer and the hidden layer. $\boldsymbol{Y}_3$ is used to choose the best RBFNN among the Pareto frontier.

6.5.3 Operators of IMOEA for RBFNN

After the Roulette wheel selection of the parents from individuals in terms of the top $N/2 f_1$ and the top $N/2 f_2$, respectively, the crossover and mutation operators are then implemented to generate the offspring.

(1) Crossover and mutation operators

The crossover operation is performed with probability p_c between individuals $\mathbf{C}_l$ and $\mathbf{C}_{l+1}$, and the offspring $\mathbf{C}_l^{'}$ and $\mathbf{C}_{l+1}^{'}$ are produced. The crossover position is generated between $[1, n_r]$ randomly. The number of the input nodes $m+n$ and the corresponding parameters of the radial basis functions are changed dynamically with the evolution processes going on. However, the number of the hidden nodes cannot be changed by crossover operation. In Fig. 6.25, an example of the crossover operation is given in the genes surrounded by a dotted line, all genes in the dotted line are exchanged, and obviously, this is a multi-point crossover operator in nature.

For a better exploration, a mutation operator is also designed with the probability p_{m}. When the mutation operator is implemented, m, n, and n_r are first produced in

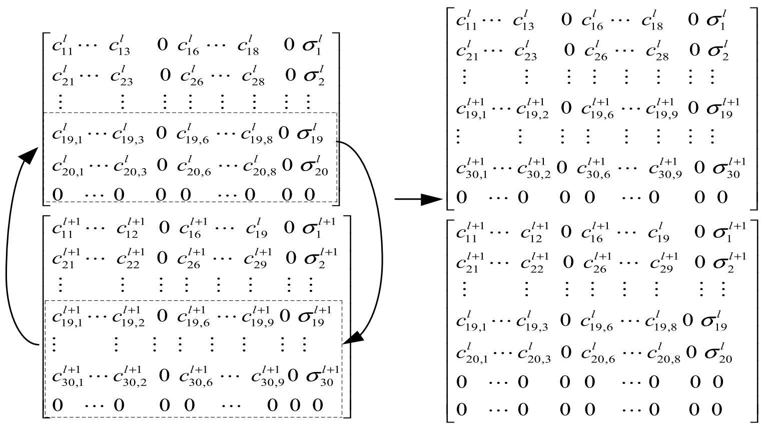


Fig. 6.25 Example of the crossover operator

random among the given ranges in Sect. 6.5.1. and the elements of the mutation individual are replicated according to the Eqs. (6.12)–(6.13). The new structure of the RBFNN is thus generated.

In addition to the crossover and mutation operators, the local search operator, prolong and pruning operators are designed to improve the search capability of MOEA and guarantee the rationality of the RBFNN.

(2) Local search operator

The local search operator is given as follows:

$$\mathbf{C} = \alpha_1 \mathbf{C}_l + (1 - \alpha_1)\mathbf{C}_l^{'} \tag{6.15}$$

$$\mathbf{C} = \mathbf{C}_l + \alpha_2 \mathbf{C}_l \tag{6.16}$$

where C_l is selected randomly from the former $N/2$ parents, $\mathbf{C}'_l$ is chosen randomly from the latter $N/2$ parents, and α_1 is randomly generated between (0,1) that is to generate excellent offspring inheriting the gene information of C_l and $\mathbf{C}'_l$. If C_l is equal to $\mathbf{C}'_l$, 0.146 is utilized to generate the offspring and α_2 is generated randomly between $(-1, 1)$. In order to keep the population diversity and avoid running into the local optima as the evolution goes on, a similar dynamical probability of 0.9 is adopted for local search operator. The difference is that the probability is increased from p_{l0} to p_{lG} with the generation increasing from 1 to G by using minus aa.

(3) Prolong and pruning operators

Since the crossover operator cannot generate new structures of the hidden layer and the probability of mutation is low, the prolong operator is designed. That is, the number of the hidden nodes is reproduced randomly between n_r and D with probability p_p. The elements of the newly added node can be calculated according to Eqs. (6.12)–(6.13).

Because the chromosomes are randomly generated, the crossover and mutation operations are also of randomness and there may be inactive structures in the population. The neuron with $\boldsymbol{c}_i = 0$, $i < n_r$ will first be deleted, and each hidden neuron is evaluated in terms of the active firing (AF) in Sect. 6.4.3, using the same value of ρ. Here the upper threshold Af_o is also selected from [0.05, 0.3]. When the hidden neuron is judged as inactive, the corresponding hidden neuron is deleted.

(4) Elitism maintaining scheme

The fast non-dominated sorting scheme is adopted and all non-dominated individuals in the population are regarded as the elitists, which will be found and stored to an archive. Because the size of the archive will increase with the evolution going on, the maximum size of the elitist archive is set as N_e. If the current size of the elitist archive is larger than N_e, the maintaining scheme will be performed to keep the evenness of the elitist population. The fast non-dominated sorting algorithm is implemented and the dominated individuals will be removed from the archive. If the archive size becomes less than N_e, the maintaining procedure will not be carried out, otherwise, a modified adaptive cell density maintaining scheme will be implemented by dividing the objective spaces into $\prod_{i=1}^{2} k_1$ cells, and at most one individual can be kept at each cell [45]. Matlab code of the maintaining scheme has been given in Chap. 4. When the maximal number of the individuals distributed at the Pareto frontier is set as, and $\sum_{i=1}^{2} k_i - 1$, and $N_e < \sum_{i-1}^{2} K_i$ must be satisfied to keep the evenness of the population distribution which can refer to the analysis of MOEA in Chap. 4.

6.5.4 The Procedure of IMOEA

The whole procedure of IMOEA can be run by using the following steps.

Step 1: Initialize the population size N, the maximum generations G, the operator probabilities $p_c, p_m, p_{l0}, p_{lG}\ p_p$, and Af_o, the RBFNN N_m, N_n, D, the number of cells for the ith objective function K_i, and the maximal archive size N_e, then generate randomly N chromosomes using 0.11.

Step 2: Calculate the two fitness functions based on 0.14.

Step 3: Implement the non-dominated sorting algorithm in NSGA-II and keep the elitists in the archive. Execute the elitist maintaining scheme when the size of the elitist archive is larger than N_e.

Step 4: Select the parent individuals using the Roulette wheel method in terms of f_1 and f_2, and Pareto Elite individuals are also selected as the parents to produce the offspring by genetic operators. To keep the individual diversity, N_e is set as not larger than $N/2$.

Step 5: Execute the crossover and mutation operators with probability p_c and $p_{m,}$ respectively, then implement the local search operator with dynamic probability, prolong operator with probability p_p, and pruning operator with probability 1 to produce the offspring.

Step 6: Repeat steps 2–5 until the maximum generation G is met.

Step 7: Calculate RMSE of an unused data set for determining the final solution, and the RBFNN model with a minimal value of f_1 is selected as the final optimal one.

6.5.5 The Chamber Pressure Modeling in a Coke Furnace

This section describes the application of the IMOEA to optimize the RBFNN model for the chamber pressure in the industrial coke furnace in Sect. 6.2. Herein, the parameters of the IMOEA are set as $N = 60, G = 1000, p_c = 0.9, p_m = 0.1, p_{l0} = 0.02$, $p_{lG} = 0.22$, and $Af_o = 0.1$. Since the prolong operator is used to increase the number of the hidden nodes, the probability p_p is set relatively small as 0.1. The pruning operator is designed to keep the rationality of the RBFNN structure and its probability is set as 1. $K_i, i = 1, 2$, is set to 20 and the archive size N_e is set to 30 to satisfy $N_e \le N/2$ and $N_e < \sum_{i-1}^{2} K_i \cdot N_m$, N_n and D are directly related to the model complexity and can be selected among the following ranges:N_m, $N_n \in [3, 10]$, $D \in [10, 60]$, where a little of prior knowledge is required to set suitable values for these parameters. Note that the simpler the modeled system, the smaller the value is to be set, which may speed up the convergence of the algorithm. N_m, $N_{n,}$ and D in this section are set as 5, 5, 60, respectively.

Two pressure branches, i.e., the main channel and its coupling disturbance should be modeled here. Then, several sets of step tests are performed for system analysis and modeling. The input step signal is the set point of the originally designed PID controller and this signal also poses disturbances on the other side of the chamber pressure. The experimental data are collected from the same industrial coke unit equipped with a distributed control system CENTUM DCS3000. And all data are filtered to reduce the impact of measurement noise. There are totally 4 groups of 1200 input/output samples as plotted in Figs. 6.26 and 6.27. For the main channel of the chamber pressure PRC8112A/PRC8112B, the input is the valve opening given by the PID controller and the output is the chamber pressure PRC8112A/PRC8112B. The output responses of PRC8112A and its coupling disturbance on PRC8112B are shown in Fig. 6.26 when the set point is set as −0.029 kPa, −0.024 kPa, −0.019 kPa, and −0.024 kPa, respectively. Figure 6.27 shows the output responses of PRC8112B and its coupling disturbance on PRC8112A when the set point is set as −0.022 kPa, −0.016 kPa, −0.02 kPa, −0.026 kPa, and −0.02 kPa, respectively. The sampled dataset is equally divided into three groups, where the former 1/3 data are selected as the training data $\mathbf{Y}_1$, the intermediate 1/3 data as $\mathbf{Y}_2$, and the latter 1/3 data as $\mathbf{Y}_3$. In addition, the RBFNN is optimized by running 10 times and the parameters of IMOEA remain unchanged at each run.

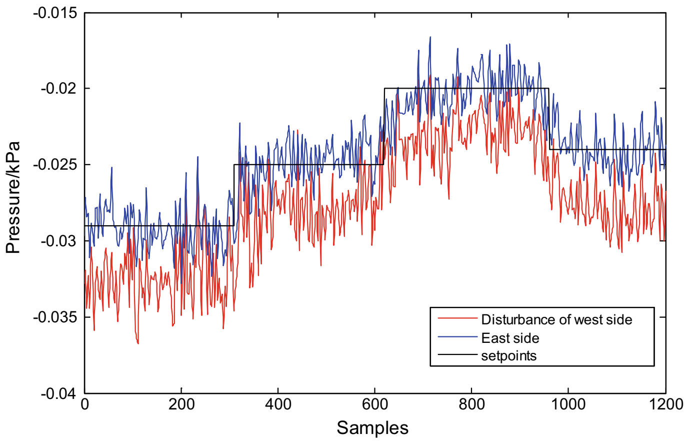


Fig. 6.26 Outputs of PRC8112A (main channel) and its disturbance on pressure PRC8112B

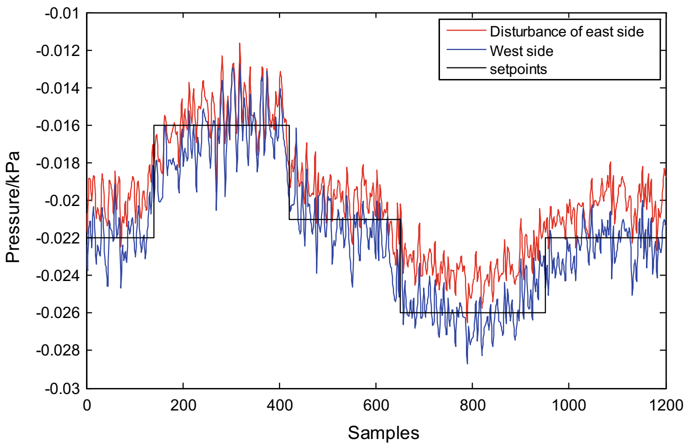


Fig. 6.27 Outputs of PRC8112B (main channel) and its disturbance on pressure PRC8112A

The IMOEA method is compared with NSGA-II, IMOEA with fixed input layer (IMOEA-Fix), IRNA-GA with WSO method [19], and artificial neural network with LM algorithm (LMANN). The details of the compared algorithms are given as follows:

(1) The encoding/decoding method, crossover, mutation, and zero $\boldsymbol{c}_i$ deletion of IMOEA are adopted in NSGA-II, while the individuals in the mating pool are selected in terms of the non-dominated sorting rank and its spread evenness information.

(2) For IMOEA-Fix, all the parameters and operators are the same as those of IMOEA except for the structure of the input layer, where m is set as 3, n as 2, and they are fixed during the whole optimization procedure.
(3) A weighted sum of objectives based on IRNA-GA that has obtained good results of temperature modeling in Sect. 6.4 is chosen to be compared. The objective function in Eq. (6.17) for IRNA-GA is similar to Eq. (6.7) and that in Ref. [19], however, the weight coefficient is decreased to 0.01 by trial and error focusing on the modeling accuracy.

$$J = f_1 + \lambda f_2 \tag{6.17}$$

(4) For LMANN, an enumeration method is applied to select the NN structure, i.e., the number of the neurons in the input layer $(m + n)$ is enumerated from 2 to 10 and the number of the neurons in the hidden layer (n_h) is enumerated from 1 to 60, and there are totally 25×60 combinations of ANN.

The selecting criterion in step 7 is used to choose the ultimate RBFNN among Pareto individuals and also for selecting the best one in 10 runs of IRNA-GA and LMANN.

To illustrate the population diversity and distribution evenness, the Pareto frontier with the maximal number of individuals in 10 runs using 3 MOEAs are shown in Figs. 6.28, 6.29, 6.30,and 6.31, where the modeling error f_1 is in the horizontal coordinate and the structure complexity f_2 is in y-coordinate. Obviously, the objectives are conflicting with each other; the RBFNN with simpler structure, that is smaller f_2, has weaker approximation capability, that means larger f_1, and vice versa. When f_2 is less than 10, f_1 grows quickly because of too simple structure of RBFNN. Since only zero c_i deletion is used to keep the rationality of the RBFNN in NSGA-II, the value of f_2 in NSGA-II is larger than that of IMOEA as shown in Figs. 6.28, 6.29, 6.30, and 6.31. Moreover, in Figs. 6.29 and 6.30, it is obvious that IMOEA is nearer to the Pareto frontier compared with NSGA-II because the local search operator and pruning operator are beneficial to produce more individuals and decrease the structure complexity. In Fig. 6.31, though IMOEA-fix has more individuals, only three

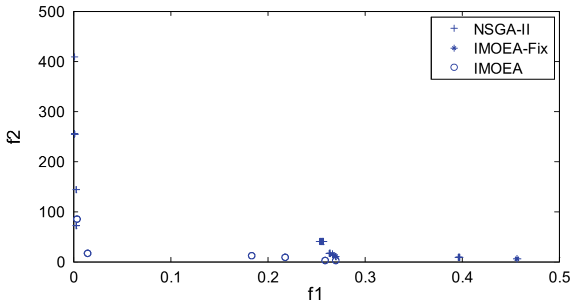


Fig. 6.28 Pareto frontier for PRC8112A

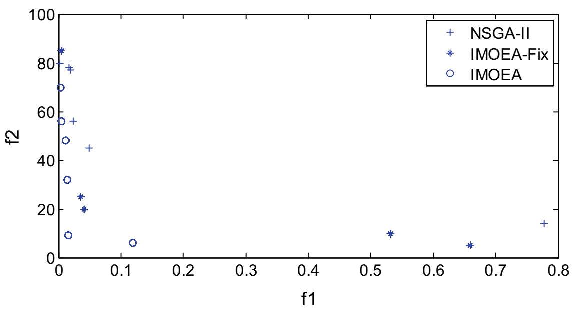


Fig. 6.29 Pareto frontier for the disturbance of PRC8112B

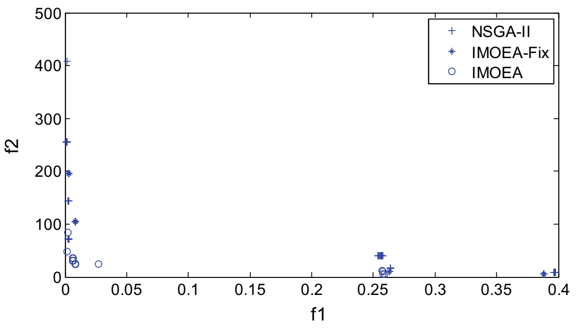


Fig. 6.30 Pareto frontier for PRC8112B

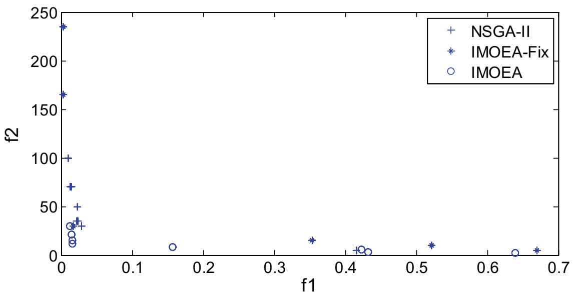


Fig. 6.31 Pareto frontier for the disturbance of PRC8112A

individuals have good modeling accuracy where f_1 is less than 0.1. Since the structure of the input layer is fixed, the performance of the RBFNN is restricted greatly. The maintaining scheme of IMOEA has kept the elitist archive size in a rational range during the optimization process. However, the number of individuals in the Pareto frontier in 3 MOEAs is relatively small, i.e., the size of the elitist archive is less than N_e and the evenness distribution problem is not serious, thus, the maintaining scheme will not be implemented in most evolution generations.

The best RBFNNs of five methods are obtained after selecting in terms of modeling accuracy and their modeling errors are plotted in Figs. 6.32, 6.33, 6.34, and 6.35. It is obvious that the errors of IRNA-GA are much larger than those of MOEA

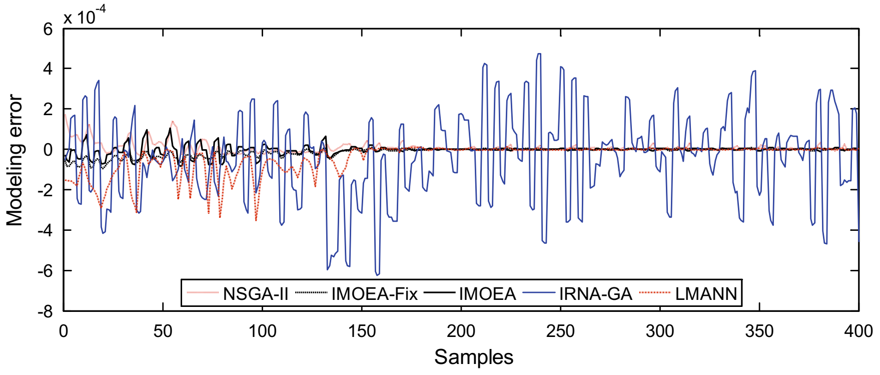


Fig. 6.32 Errors of best RBFNN for PRC8112A

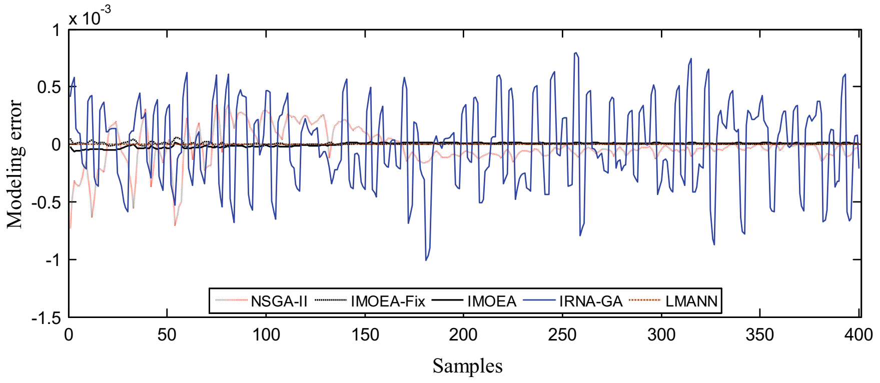


Fig. 6.33 Errors of best RBFNN for PRC8112B

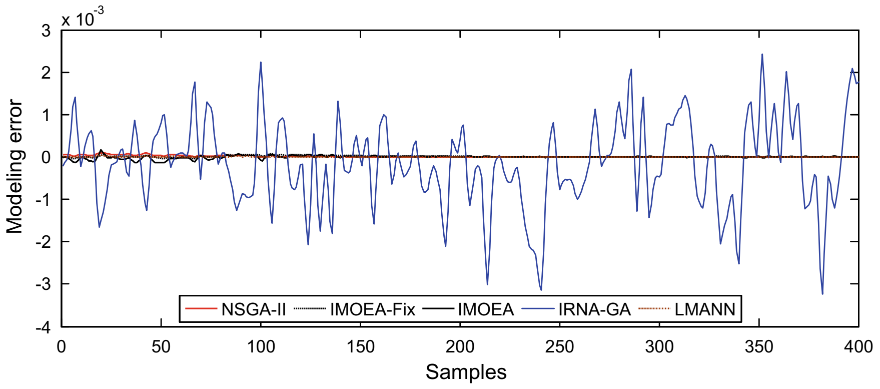


Fig. 6.34 Errors of best RBFNN for the disturbance of PRC8112B

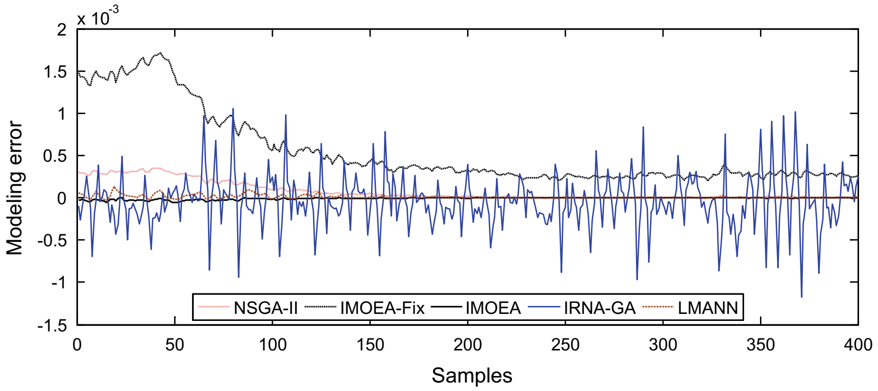


Fig. 6.35 Errors of best RBFNN for the disturbance of PRC8112A

partly because the structure simplification in Eq. (6.17) is considered during the optimization processes. As a typical and widely used MOEA, NSGA-II can obtain satisfactory modeling accuracy as shown in Fig. 6.34. However, by introducing local search, prolong and pruning operators, IMOEA has obtained smaller errors than other GA-based methods. As for IMOEA-fix, if the input layer is not set appropriately beforehand as shown in Fig. 6.35, its modeling error will be larger than those of IRNA-GA and NSGA-II. Otherwise, its modeling error of IMOEA-fix is similar to that of IMOEA.

The statistical results about the structure of the input layer, the number of hidden nodes of the best RBFNN, and their RMSEs of six methods among 10 runs are listed in Tables 6.2 and 6.3, respectively. In Table 6.2, LMANN has obtained the best results of PRC8112B and its disturbance channel using the enumeration method, however, the average running time is quite long, which depends on the range of m, n, and n_r, and the enumeration method may be impracticable with large variable range. LMANN-fix used the same structure of the optimal RBFNN's, however, its results are worse than those of the enumerated LMANN because the structure, weight learning algorithm, and radial basis function are quite different between ANN and RBFNN, and the optimal structure of the RBFNN is not suitable for ANN. Moreover, ANN will be greatly affected by the initialization of the weight vector that is implemented automatically by the toolbox, and its average RMSE ($\overline{\text{RMSE}}$) in Table 6.3 is much larger than their RMSEs in Table 6.2. The RMSE of IRNA-GA in Table 6.2 is several times larger than those in three MOEAs except for the disturbance model of PRC8112A by IMOEA-fix, which is consistent with Fig. 6.34. The variation range of n_r by IRNA-GA in Table 6.3 is relatively smaller than that of MOEA because the WSO method takes into account the structure simplification in the optimization process and the average running time ($\bar{T}$) of the single-objective optimization is relatively shorter. Compared with the implementation of NSGA-II, IMOEA can obtain better accuracy with simpler structure, because several new operators are carried out, however, its $\bar{T}$ is the longest in the MOEAs. As for IMOEA-Fix, the

Table 6.2 The comparison of best simulation results by 6 methods

Methods	PRC8112A				Disturbance of PRC8112B				PRC8112B				PRC8112B			
	m	n	n_h	RMSE	m	n	n_h	RMSE	m	n	n_h	RMSE	m	n	n_h	RMSE
LM ANN	1	1	9	3.5e − 5	2	2	6	5.73e − 6	3	2	10	5.96e − 6	1	2	6	2.25e − 5
LMANN-Fix	1	2	9	7.65e − 5	1	1	11	1.41e − 4	3	2	7	3.53e − 5	1	3	18	8.18e − 5
IRNA-GA	2	1	7	2.28e − 4	2	2	6	3.69e − 4	2	3	10	2.89e − 4	3	2	11	3.05e − 4
NSGA-II	2	1	12	7.42e − 5	1	2	10	1.94e − 5	4	1	23	2.41e − 5	2	4	19	1.32e − 4
IMOEA-Fix	3	2	13	2.53e − 5	3	2	13	8.77e − 5	3	2	24	2.22e − 5	3	2	12	3.76e − 4
IMOEA	1	2	9	2.47e − 5	1	1	11	6.01e − 6	3	2	7	3.62e − 5	1	3	18	1.51e − 5

Table 6.3 The performance comparison of 6 methods in 10 runs

Methods	PRC8112A					Disturbance of PRC8112B					PRC8112B					PRC8112B				
	m	n	n_h	$\bar{T}$	$\overline{RMSE}$	m	n	n_h	$\bar{T}$	$\overline{RMSE}$	m	n	n_h	$\bar{T}$	$\overline{RMSE}$	m	n	n_h	$\bar{T}$	$\overline{RMSE}$
LM ANN	2 ± 1	2 ± 1	6 ± 3	3150	8.15e − 5	2 ± 1	2 ± 1	4 ± 2	2850	1.18e − 4	2 ± 1	2 ± 1	6 ± 4	2550	3.47e − 4	2 ± 1	2 ± 1	6 ± 4	2820	2.51e − 4
LMANN-Fix	1 ± 0	2 ± 0	9 ± 0	21	3.96e − 4	1 ± 0	1 ± 0	11 ± 0	19	7.13e − 4	3 ± 0	2 ± 0	7 ± 0	17	2.5e − 4	1 ± 0	3 ± 0	18 ± 0	19	8.74e − 4
IRNA-GA	3 ± 1	3 ± 2	10 ± 8	651	9.81e − 4	3 ± 1	3 ± 1	16 ± 12	853	1.52e − 3	3 ± 1	3 ± 1	15 ± 9	607	1.12e − 3	3 ± 1	3 ± 1	13 ± 8	569	1.05e − 3
NSGA-II	3 ± 1	3 ± 1	28 ± 19	1029	4.46e − 4	3 ± 2	3 ± 2	34 ± 24	1152	5.19e − 4	3 ± 2	3 ± 2	26 ± 20	1027	2.19e − 4	3 ± 1	4 ± 1	37 ± 22	925	1.75e − 4
IMOEA-Fix	3 ± 0	2 ± 0	19 ± 10	1043	5.03e − 3	3 ± 0	2 ± 0	26 ± 13	1022	2.76e − 3	3 ± 0	2 ± 0	30 ± 24	1012	2.84e − 3	3 ± 0	2 ± 1	19 ± 16	982	1.41e − 3
IMOEA	3 ± 2	2 ± 1	15 ± 10	1829	7.32e − 5	3 ± 2	3 ± 2	26 ± 15	1774	4.25e − 5	3 ± 1	2 ± 1	15 ± 10	1783	1.23e − 4	2 ± 1	4 ± 1	29 ± 12	1624	4.91e − 4

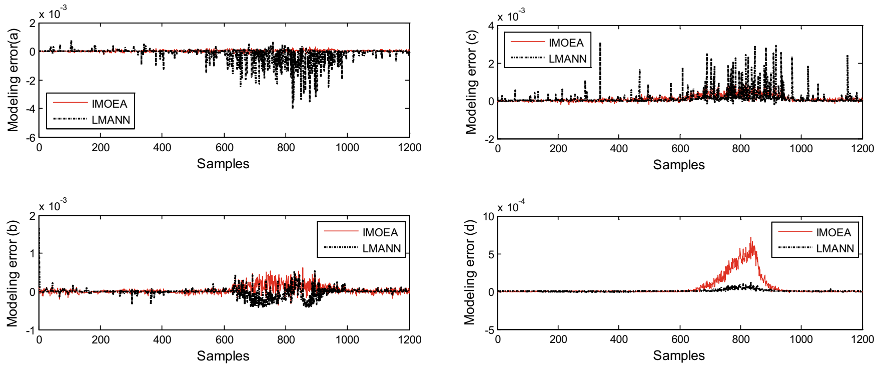


Fig. 6.36 Comparison IMOEA RBFNN with LMANN in the case of noisy data **a** PRC8112A **b** PRC8112B **c** Disturbance of PRC8112B **d** Disturbance of PRC8112A

complicated input layer structure seems to require more hidden nodes to obtain the similar modeling precision compared with IMOEA. Moreover, one order higher of $\overline{\text{RMSE}}$ in Table 6.3 illustrates that $m = 3$, $n = 2$ is not appropriate to construct the chamber pressure model. Hence, the input layer will affect both the model structure and modeling accuracy greatly.

In order to illustrate the generalization capability of the optimization methods, 10% of noise levels are added to the sampled data, and other parameters are kept unchanged. The modeling errors of IMOEA RBFNN are compared with the enumerated LMANN and plotted in Fig. 6.36. The RMSEs of enumerated LMANN are 5.10e − 4, 3.83e − 4, 1.26e − 4, 1.67e − 5, while the IMOEA RBFNN's are 6.70e − 5, 2.32e − 4, 1.178e − 4 and 1.39e − 4, respectively. Though small RMSEs of PRC8112B and its disturbance of PRC8112B have been derived by the enumerated LMANN in Table 6.2, much larger error has been observed from the noisy data as shown in Fig. 6.36 and reflected in their RMSEs, which may be caused by overfitting. IMOEA based RBFNN is more robust than LMANN in the aspect of generalization capability. It can be concluded that IMOEA is superior to the other four methods with respect to the smaller errors in model accuracy and simpler structure of RBFNN, however, the algorithm is more complicated and its running time is longest.

6.6 PCA and INSGA-II Based RBFNN Disturbance Modeling of Chamber Pressure

6.6.1 RV Criterion in PCA Variable Selection

The key variables are important to be found for multiple variables system modeling, the RV criterion in principal component analysis has been utilized to measure the similarity of the selected subset. If the value of RV criterion is the largest among all possible subsets, the optimal subset will be obtained. In order to calculate the RV criterion, the following augmentation of notation has been listed in Table 6.4.

The optimal solution for a given subset, P, is equivalent to maximize the RV criterion as follows [22]:

$$f_1 = \sqrt{\frac{tr((S_P^{-1}[S^2]_P)^2)}{tr(S^2)}} \tag{6.18}$$

If all variables are selected, the maximum value of f_1 reaches 1. Since f_1 is to be maximized, the objective is then changed into the minimization problem by reciprocal operation of Eq. (6.18), shown as follows:

$$J_1 = 1 \big/ f_1. \tag{6.19}$$

Once the selected variables are determined in terms of J_1, they are used as the inputs of the system model.

When the RBFNN model is constructed and its parameters are trained, its modeling accuracy can be evaluated according to the sum of RMSEs of the training and testing data

$$J_2 = \sqrt{\sum_{k=1}^{N_1} |y_1(k) - \hat{y}_1(k)|^2 \Big/ N_1} + \sqrt{\sum_{k=1}^{N_2} |y_2(k) - \hat{y}_2(k)|^2 \Big/ N_2} \tag{6.20}$$

The two objectives J_1, J_2 can be optimized simultaneously. The encoding method and various operators in an improved NSGA-II for variable selection, RBFNN structure, and parameter optimization are then designed to solve this multi-objective optimization problem.

Table 6.4 Notation for RV criterion

Parameters	Description
X	an $N \times M$ data matrix for N objects measured on M variables
$X(P)$	the $N \times p$ vector of X for the selected variables in P
S	the covariance matrix for the full data matrix, X
S^2	the product of the covariance matrix and itself, $S^2 = SS$
S_P	the $p \times p$ submatrix of S corresponding to the selected variables in P
$[S^2]_P$	the $p \times p$ submatrix of S^2 corresponding to the selected variables in P

6.6.2 Encoding of RBFNN

For simplicity, n in the input layer is set as 2, while m for one input variable is set as 1, here, there are at most six disturbance variables according to prior knowledge of coke furnace, m is then limited to [1, 6]. Once the key variables are selected, the input nodes are then determined. The number of the hidden neurons (n_r) and the parameters of Gaussian functions $\mathbf{c}_i$, σ_i, $i = 1, \ldots, n_r$, $1 \le n_r \le D$, with D being the maximal number of the hidden nodes, are to be optimized. The encoding for different variables selection and RBFNN is then designed, and the ith chromosome is shown as follows:

$$\mathbf{C}_i = \begin{bmatrix} c_{11} & c_{21} & c_{31} & \cdots & c_{81} & \sigma_1 \\ \vdots & \vdots & \vdots & \vdots\ \vdots & \vdots & \vdots \\ c_{1n_r} & c_{2n_r} & c_{3n_r} & \cdots & c_{8n_r} & \sigma_{n_r} \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 1 & 0 \end{bmatrix} \tag{6.21}$$

Here $1 \le i \le N$, $(m + n) \in [3, 8]$ and the elements in rows $[1, n_r]$ can be obtained as follows:

$$\begin{cases} c_{ij} = y_{\min} + r(y_{\max} - y_{\min}) & 1 \le i \le 2\ 1 \le j \le n_r \\ u_{\min} + r(u_{\max} - u_{\min}) & 3 \le i \le 8\ 1 < j \le n_r \\ \sigma_j = rw_{\max} & 1 \le j \le n_r \end{cases} \tag{6.22}$$

where r is randomly generated in [0.01,1], $u_{\min,}$ $u_{\max}$ $y_{\min,}$ $y_{\max,}$ and $w_{\max}$ are the same as those in Eqs. (6.12)–(6.13).

The last row in $\mathbf{C}_i$ delegates which variable of columns 3–8 will be selected, thus it is encoded by binary encode, and the valid bits are located at [3 8], for example

$$c_{D+1} = [0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0] \tag{6.23}$$

It means that u_3, u_4, u_6 are selected, and columns $\mathbf{c}_5, \mathbf{c}_6, \mathbf{c}_8$ are valid centers of the Gaussian functions.

Once $\boldsymbol{C}_i$ is obtained, both the structure and the parameters of RBFNN can be determined, and the weight $\boldsymbol{w}$ can be further calculated by RLS based on the training data.

6.6.3 Operators of INSGA-II

(1) Selection operator

When the NSGA-II is implemented, the rank and crowding distance of individuals can be obtained. The individuals with rank 1 are regarded as the elitists and chosen as the parents. The individuals at each frontier from rank 1 are selected into the parent population one by one until exceeding the population size N. Then, the crowing distance in the current frontier is compared by sorting in descend and the individuals with larger crowing distance are selected into the parent population. If the size is still less than the population size, the Roulette wheel selection operator is implemented to select half of the rest of the population in terms of J_1 and the other half of the rest of the population in terms of J_2.

(2) Crossover and mutation operators

The crossover and mutation operators are carried out among the selected population to produce the offspring.

In Fig. 6.37, the crossover operator with probability p_c is executed between the individuals $\mathbf{C}_i$ and $\mathbf{C}_i'$, and the location is randomly generated between [1 9]. The parameters of the radial basis function are changed and the selected variables are also changed in the offspring. Note that the number of the hidden nodes cannot be changed by using this crossover operator.

The element in Eq. (6.21) is mutated with the probability $p_{\rm m}$. When the mutation operator is implemented, the elements are produced in terms of Eq. (6.22) and the elements in Eq. (6.23) performs a logic NOT operation, that is, 1 to 0 and 0 to 1. The new structure of RBFNN and different key variables can be obtained.

In addition to crossover and mutation, the prolong and pruning are also designed for improving the searching capability of NSGA-II and keeping the rationality of RBFNN.

(3) Prolong and pruning operators

Due to the fact that the number of the hidden nodes is not changed and some irrational structure may be produced by random operators, the prolong and pruning operators are then designed. If the number of the hidden neuron is less than 2, the prolong

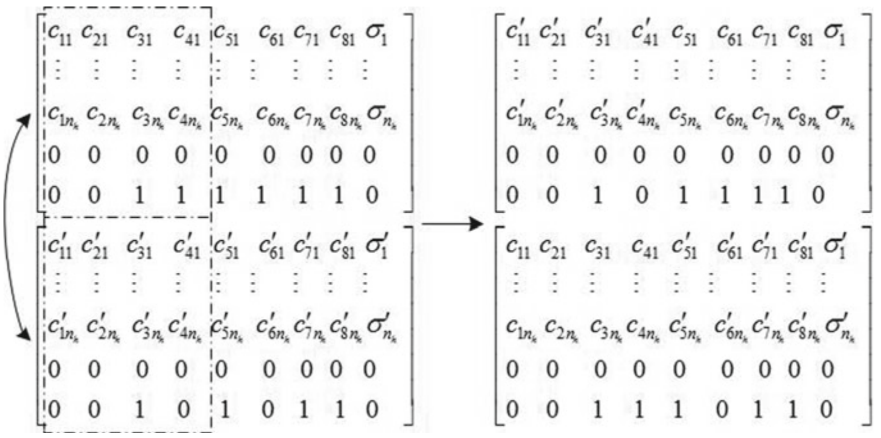


Fig. 6.37 Example of the crossover operator

operator is executed, i.e., a random number between [1, D-2] is produced randomly as the newly added neuron and the elements of the new neuron are calculated in terms of Eq. (6.22). If the neuron has only one nonzero element in $\boldsymbol{c}_i$, the pruning operator is implemented. Here, the neuron will be deleted and the number of the hidden neuron is reduced.

6.6.4 The Procedure of Improved NSGA-II

The whole procedure of the INSGA-II applied for RBFNN optimization is shown as follows:

Step 1: Initialize the population size N, the maximum generations G, the operators' probabilities p_c, p_m, the system parameters $u_{\min}$, $u_{\max}$, $y_{\min}$ and $y_{\max}$, then generate N chromosomes randomly.

Step 2: Select variables in terms of Eq. (6.18) and calculate J_1.

Step 3: Construct the RBFNN and calculate the value of J_2.

Step 4: Implement INSGA-II and select the parent population in terms of the front rank, crowing distance, and Roulette wheel selection operator.

Step 5: Implement the crossover and mutation operators with p_c and p_m, then the prolong and pruning operators are carried out for the offspring.

Step 6: Repeat steps 2–5 until G is met.

6.6.5 Main Disturbance Modeling of Chamber Pressure

The chamber pressure operation in the coke furnace is critical to guarantee burning security. However, its system model for advanced control is a highly complex task because nonlinear characteristics, time delay, and a lot of disturbances such as fuel volume, the coupling of pressures, etc., coexist in the unit. The input variable of the main channel is known, but the main disturbance model is especially difficult to be obtained because of the above various disturbances. How to select the key disturbance variables and construct its disturbance model is still challenging.

The pressure PRC8112A coupled with the pressure PRC8112B, the temperature in the chamber TR8109A, TR8109B, the oxygen content AR8102, ARC8101, and the external flow XLF103, are sampled. Meanwhile, the other side pressure PRC8112B with similar disturbance variables is also collected and stored in the PAI database. Since the values of different variables in Figs. 6.38a and 6.39a vary considerably large, they are normalized to [0, 1] and shown in Figs. 6.38b and 6.39b. It is obvious that the dynamic response is complex, and accurate disturbance modeling is difficult.

The INSGA-II is used to select the main disturbances and optimize both the structure and the parameters of RBFNN such that the nonlinear dynamic behavior of the disturbance of chamber pressure can be captured. The population size N is set to 60, the maximal evolution generation G is 1000, and the operator probabilities p_c

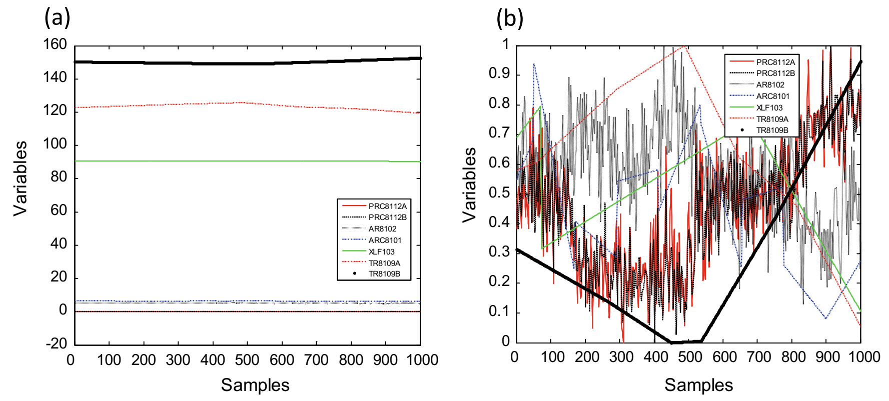


Fig. 6.38 **a** Original variables sampled in the PAI database for PRC8112A, **b** normalized variables

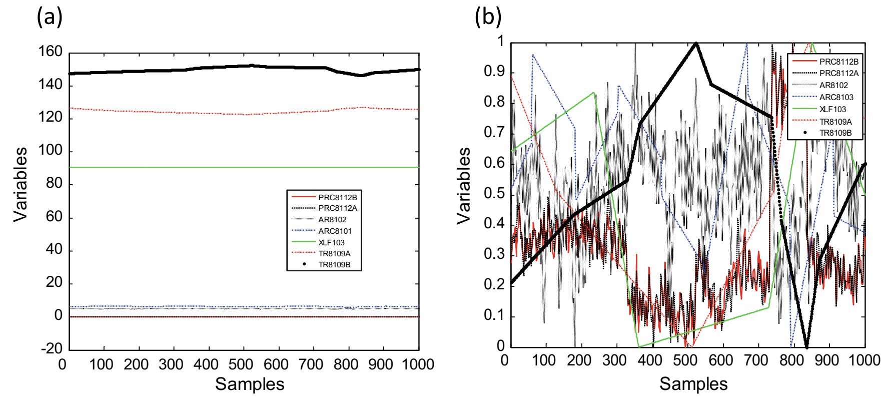


Fig. 6.39 **a** Original variables sampled in the PAI database for PRC8112B, **b** normalized variables

and p_m are set to 0.9 and 0.1, N_1, N_2 of the training data ($\mathbf{Y}_1$) and the testing data ($\mathbf{Y}_2$) are set as 400, 400, N, M in $\boldsymbol{X}$ are 6 and 800, respectively. The maximal number of the hidden nodes (D) is set to 30, [$u_{\min}$, $u_{\max}$] is [0, 1] and [$y_{\min}$, $y_{\max}$] is [0, 1]. The optimization result for PRC8112A and PRC8112B is a group of Pareto optimal solutions and shown in Figs. 6.40 and 6.41, respectively.

It can be seen in Figs. 6.40 and 6.41 that the RMSE of RBFNN model J_2 becomes larger with the value of RV criterion J_1 close to 1, where the number of the selected variables has been changed from 1 to 6. Though the RMSE becomes larger, the difference between the maximal and minimal values of J_1 in the Pareto frontier is not large, e.g., PRC8112A's is 0.18 and PRC8112B's is 0.35. J_2 is then used to select the final solution. The individual with the minimal value of J_2 is chosen as the final solution, *i.e.*, the individuals where (J_1, J_2) is (1.18, 0.071) in Fig. 6.40, and (J_1, J_2) is (1.18, 0.071) in Fig. 6.41, are chosen. In the selected individual, c_{D+1} is

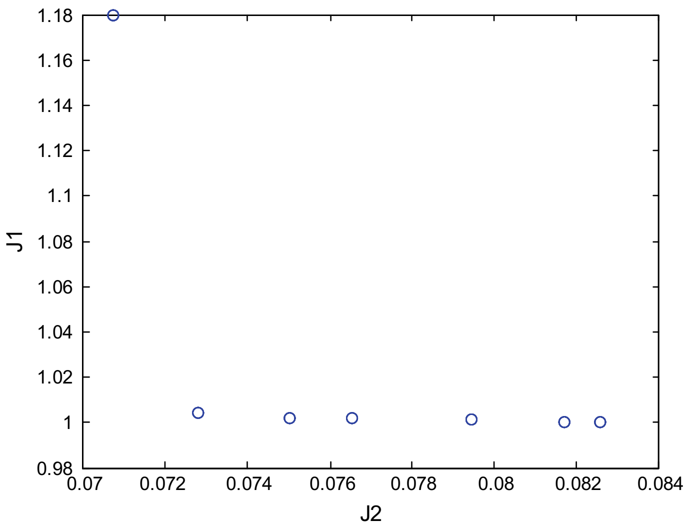


Fig. 6.40 Pareto front for PRC8112A

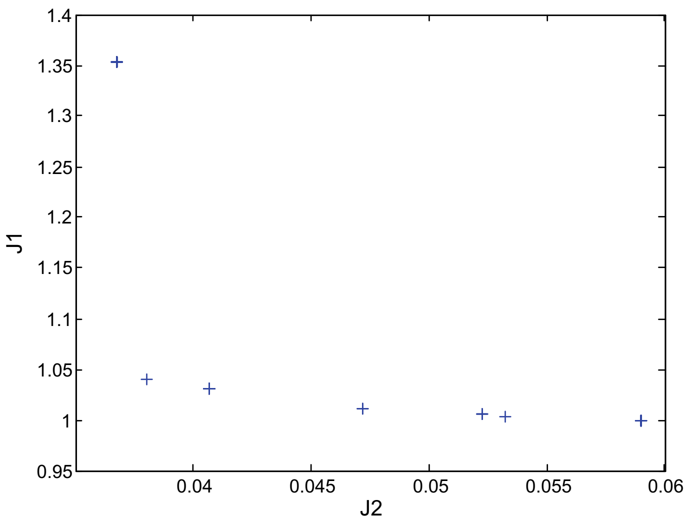


Fig. 6.41 Pareto front for PRC8112B

[00100000], which means PRC8112B is the main disturbance for PRC8112A and PRC8112A is the main disturbance for PRC8112B. Therefore, the input vector of RBFNN is $[y(k-1), y(k-2), u_1(k)]$. The model output and its modeling errors for PRC8112A and PRC8112B are plotted in Figs. 6.42 and 6.43.

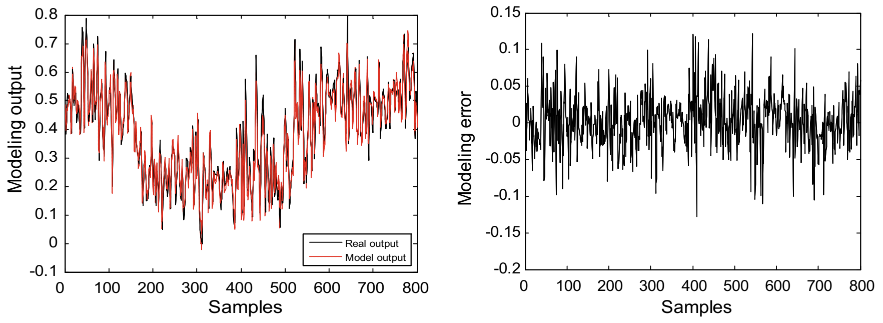


Fig. 6.42 Outputs and errors of RBF disturbance model for PRC8112A by proposed method

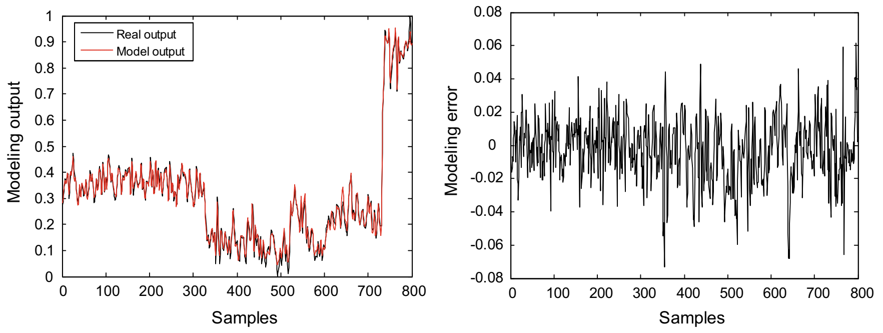


Fig. 6.43 Outputs and errors of RBF disturbance model for PRC8112B by proposed method

We also selected several methods to select the key variables and construct the disturbance model, which are PCA variable selection method [47] with RBFNN optimized by an improved GA [7] aimed at minimizing J_2 (PCAGA RBF), simulated annealing for variable selection in terms of RV criterion [23] and RBFNN modeling (SAPCA RBF), and the multilayer perceptron (MLP) neural network with least absolute shrinkage and selection operator to select the input variables (LASSO NN) [27].

In the PCAGA RBF method, the component with small eigenvalue, usually less than 0.7, is of less importance. Consequently, the variable that dominates it should be superfluous. Here, the eigenvector is [19.6760, 9.3106, 6.1678, 3.8346, 3.3994, 1.2042] for PRC8112A and [17.7668, 10.5847, 8.3911, 6.6600, 4.7317, 3.2693] for PRC8112B, obviously, all values in the eigenvector are larger than 0.7, thus, the disturbances need to be kept. The parameters of the radial basis function and the number of the hidden nodes are derived after optimization. The errors of the constructed RBFNN disturbance models for PRC8112A and PRC8112B are shown in Fig. 6.44.

When using SAPCA RBF method, the third disturbance and the fifth disturbance are selected as the main disturbance for PRC8112A and PRC8112B, respectively.

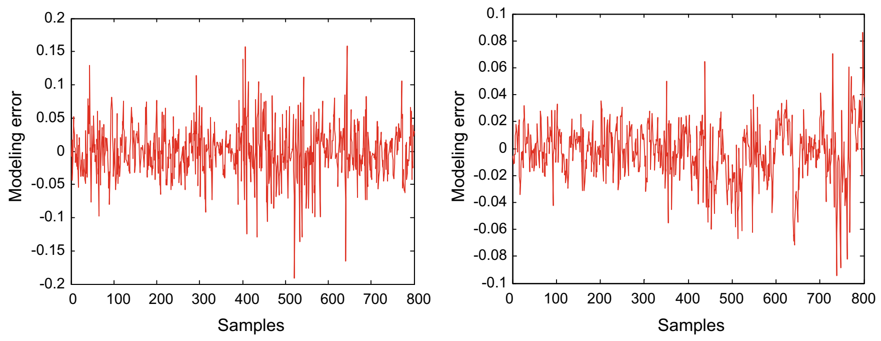


Fig. 6.44 The modeling errors of PRC8112A and PRC8112B by PCARBF method

The input vector of the RBFNN for the two-side disturbance model is $[y(k-1), y(k-2), u_3(k)]$ and $[y(k-1), y(k-2), u_5(k)]$, respectively. The modeling error of the RBFNN for PRC8112A and PRC8112B are plotted in Fig. 6.45.

When applying the LASSO NN method, the MLP neural network is first obtained by the Levenberg-Marquard learning algorithm, then the LASSO is applied to select the variables. Three main disturbances $[u_1(k), u_2(k), u_5(k)]$ for PRC8112A and four variables $[u_1(k), u_2(k), u_4(k), u_5(k)]$ for PRC8112B are obtained, and their modeling errors are given in Fig. 6.46.

It can be seen from Figs. 6.42, 6.43, 6.44, 6.45, and 6.46, that the INSGA-II method has obtained the best modeling accuracy. In order to be convenient to show the different performances of the above methods, RMSEs of the training data and test data, which are denoted as RMSE_1 and RMSE_2, the parameters of the RBFNN, the running time, and the RV criterion are listed in Table 6.5.

In Table 6.5, the number of disturbances is only one with the less similarity RV value for the INSGA-II method, while the PCA eigenvalue analysis method selected all disturbance. For the SAPCA RBF method, larger RV value is obtained than the proposed method because the RV criterion is optimized independently by SA.

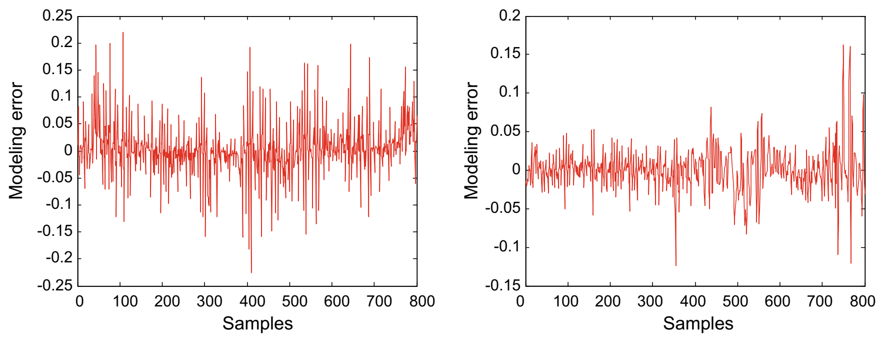


Fig. 6.45 The modeling errors of PRC8112A and PRC8112B by SAPCA RBF method

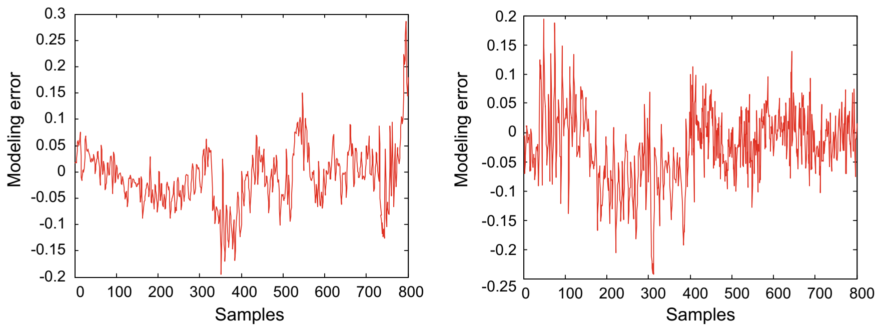


Fig. 6.46 The modeling errors of PRC8112A and PRC8112B by LASSO NN method

Table 6.5 The comparison results of 4 methods

Methods	Plant	No. of disturbances	No. of input nodes	No. of hidden nodes	$RMSE_1$	$RMSE_2$	Running time (s)	RV
Proposed	PRC8112A	1	3	7	0.019	0.020	5035	0.847
	PRC8112B	1	3	6	0.0154	0.0213	5106	0.739
PCAGA RBF	PRC8112A	6	8	20	0.032	0.043	2096	1
	PRC8112B	6	8	22	0.0145	0.0258	2198	1
SAPCA RBF	PRC8112A	1	3	6	0.0235	0.0356	606	0.953
	PRC8112B	1	3	4	0.0427	0.0477	609	0.767
LASSO NN	PRC8112A	3	5	15	0.0542	0.0599	3.92	0.991
	PRC8112B	4	6	15	0.0829	0.0422	3.89	0.996

However, the final $RMSE_1$ and $RMSE_2$ of the disturbance models are inferior to the INSGA-II's. Besides, the number of the hidden nodes of PCAGA RBF is the largest because of the complicated input structure by using eight inputs. Since the computation of INSGA-II is complicated, the running time of INSGA-II is the longest among the four methods, while the running time of the LASSO NN method without using the random search algorithm is the shortest. In summary, the multi-objective optimization method considering the RV criterion coordinated with modeling accuracy can gain a group of solutions to be selected for specific purposes. The values of $RMSE_1$ and $RMSE_2$ using the INAGA-II method are the smallest with a simple structure of using less input and hidden nodes. The modeling method is efficient in key variable selection, such as the disturbances selection, and complex system model construction.

6.7 Summary

In this chapter, temperature, pressure, and its disturbance models of coke furnace are constructed using RBF Neural Network and improved GA. All data sets were gathered from the same industrial equipment.

(1) To improve the approximation and generalization performance, an improved RNAGA is first developed to optimize the RBF neural network structure and its corresponding parameters of radial basis functions. A pruning operator is also designed to simplify the RBFNN structure. The simulation results show that the constructed RBFNN model optimized by IRNA-GA can obtain good prediction accuracy with a relatively simple network structure.
(2) An IMOEA is used for the RBFNN modeling of the chamber pressure. The encoding method is designed for the structure of the input layer, the hidden layer, and the parameters of the Gaussian basis function. The local search operators are helpful to improve the search capability, and the prolong and pruning operators are beneficial to change the hidden layer structure. The adaptive archive maintaining is applied to retain the elitists and maintain their evenness. It is efficient and easy to be implemented in industrial processes with a little prior knowledge.
(3) The disturbance selection using the RV criterion of principle analysis and RBFNN modeling for nonlinear processes are optimized by using an INSGA-II, where the RV criterion and modeling error are optimized simultaneously. In addition, the encoding, prolong, and pruning operators are adopted to make NSGA-II suitable for RBFNN optimization. Among a set of Pareto solutions, RMSE value is used to choose the final result in the Pareto frontier. The main disturbance can be selected successfully and the RBFNN has established the disturbance model with satisfactory accuracy.

References

1. Broomhead, D.S., and D. Lowe. 1988. Multivariable functional interpolation and adaptive networks. *Complex Systems* 2 (3): 321–355.
2. Wei, C., and J. Qiao. 2014. Passive robust fault detection using RBF neural modeling based on set membership identification. *Engineering Applications of Artificial Intelligence* 28 (1): 1–12.
3. Wilamowski, B.M., et al. 2015. A novel RBF training algorithm for short-term electric load forecasting and comparative studies. *IEEE Transactions on Industrial Electronics* 62 (10): 6519–6529.
4. Reiner, P., and B.M. Wilamowski. 2015. *Efficient incremental construction of RBF networks using quasi-gradient method.* 150: 349–356.
5. Ahmadizar, F., et al. 2015. Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm. *Engineering Applications of Artificial Intelligence* 39: 1–13.
6. Wu, J., L. Jin, and M. Liu. 2015. Evolving RBF neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm. *Neurocomputing* 148 (2): 136–142.

7. Zhang, R., J. Tao, and F. Gao. 2014. Temperature modeling in a coke furnace with an improved RNA-GA based RBF network. *Industrial Engineering Chemistry Research* 53 (8): 3236–3245.
8. Wang, Y., and P. Yao. 2003. Simulation and optimization for thermally coupled distillation using artificial neural network and genetic algorithm. *Chinese Journal of Chemical Engineering* 11 (3): 307–311.
9. Yang, T., H.C. Lin, and M.L. Chen. 2006. Metamodeling approach in solving the machine parameters optimization problem using neural network and genetic algorithms: A case study. *Robotics Computer Integrated Manufacturing* 22 (4): 322–331.
10. Blanco, A., M. Delgado, and M.C. Pegalajar. 2001. A real-coded genetic algorithm for training recurrent neural networks. *Journal of the International Neural Network Society* 14 (1): 93–105.
11. Delgado, M., and M.C. Pegalajar. 2005. A multiobjective genetic algorithm for obtaining the optimal size of a recurrent neural network for grammatical inference. *Pattern Recognition* 38 (9): 1444–1456.
12. Yang, L. and J. Yen. 2010. An Adaptive Simplex Genetic algorithm. in *Genetic & Evolutionary Computation Conference.*
13. Esposito, A., et al. 2000. Approximation of continuous and discontinuous mappings by a growing neural RBF-based algorithm. *Neural Networks the Official Journal of the International Neural Network Society* 13 (6): 651–665.
14. Sarimveis, H., et al. 2004. A new algorithm for developing dynamic radial basis function neural network models based on genetic algorithms. *Computers Chemical Engineering Journal* 28 (1–2): 209–217.
15. Guang-Bin, H., P. Saratchandran, and S. Narasimhan. 2005. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Transactions on Neural Networks* 16 (1): 57–67.
16. Du, D., L. Kang, and M. Fei. 2010. A fast multi-output RBF neural network construction method. *Neurocomputing* 73 (10): 2196–2202.
17. Han, H.G., Q.L. Chen, and J.F. Qiao. 2011. An efficient self-organizing RBF neural network for water quality prediction. *Neural Networks the Official Journal of the International Neural Network Society* 24 (7): 717–725.
18. Chen, X., and N. Wang. 2009. A DNA based genetic algorithm for parameter estimation in the hydrogenation reaction. *Chemical Engineering Journal* 150 (2): 527–535.
19. Tao, J., and N. Wang. 2007. DNA computing based RNA genetic algorithm with applications in parameter estimation of chemical engineering processes. *Computers Chemical Engineering Journal* 31 (12): 1602–1618.
20. Dayal, B.S., and J.F. Macgregor. 1997. Improved PLS algorithms. *Journal of Chemometrics* 11 (1): 73–85.
21. Zhang, R., J. Tao, and F. Gao. 2016. A new approach of takagi-sugeno fuzzy modeling using an improved genetic algorithm optimization for oxygen content in a coke furnace. *Industrial Engineering Chemistry Research* 55 (22): 6465–6474.
22. Zhang, R., et al. 2016. New minmax linear quadratic fault-tolerant tracking control for batch processes. *IEEE Transactions on Automatic Control* 61 (10): 3045–3051.
23. Brusco, M.J. 2014. A comparison of simulated annealing algorithms for variable selection in principal component analysis and discriminant analysis. *Computational Statistics & Data Analysis* 77: 38–53.
24. Li, J., C. Duan, and Z. Fei. 2016. A Novel Variable Selection Approach for Redundant Information Elimination Purpose of Process Control. *IEEE Transactions on Industrial Electronics* 63 (3): 1737–1744.
25. Zhang, R. and J. Tao,. 2017. A nonlinear fuzzy neural network modeling approach using an improved genetic algorithm. *IEEE Transactions on Industrial Electronics 65*(7): 5882–5892.
26. Andersen, C.M., and R. Bro. 2010. Variable selection in regression—a tutorial. *Journal of Chemometrics* 24 (11–12): 728–737.
27. Sun, K., et al. 2016. Design and application of a variable selection method for multilayer perceptron neural network with LASSO. *IEEE Transactions on Neural Networks and Learning Systems 28*(6): 1386–1396.

28. Enrique, R. and S. Josep María, Romero. 2008. Performing feature selection with multilayer perceptrons. *IEEE Transactions on Neural Networks 19*(3): 431–441.
29. Souza, F.A.A., et al. 2013. A multilayer-perceptron based method for variable selection in soft sensor design. *Journal of Process Control* 23 (10): 1371–1378.
30. Estévez, P.A., et al. 2009. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks* 20 (2): 189–201.
31. Zhang, R., et al. 2016. Decoupled ARX and RBF neural network modeling using PCA and GA optimization for nonlinear distributed parameter systems. *IEEE Transactions on Neural Networks Learning Systems* 29 (2): 457–469.
32. Pacheco, J., S. Casado, and S. Porras. 2013. Exact methods for variable selection in principal component analysis: Guide functions and pre-selection. *Computational Statistics Data Analysis* 57 (1): 95–111.
33. Puggini, L., and S. Mcloone. 2017. Forward selection component analysis: Algorithms and applications. *IEEE Transactions on Pattern Analysis Machine Intelligence* 39 (12): 2395–2408.
34. Chen, J. 2004. Computational aspects of algorithms for variable selection in the context of principal components. *Computational Statistics Data Analysis* 47 (2): 225–236.
35. Cadima, J.F.C.L., and I.T. Jolliffe. 2001. Variable selection and the interpretation of principal subspaces. *Journal of Agricultural Biological Environmental Statistics* 6 (1): 62–79.
36. Brusco, M.J. 2014. A comparison of simulated annealing algorithms for variable selection in principal component analysis and discriminant analysis. *Computational Statistics & Data Analysis* 77 (9): 38–53.
37. Li, C.J., et al. 2018. Near-optimal stochastic approximation for online principal component estimation. *Mathematical Programming* 167 (1): 75–97.
38. Wei-Shi, Z., L. Jian-Huang, and P.C. Yuen. 2005. GA-fisher: A new LDA-based face recognition algorithm with selection of principal components. *IEEE Transactions on Systems, Man and Cybernetics Part B* 35 (5): 1065–1078.
39. Brusco, Michael J. 2017. A comparison of simulated annealing algorithms for variable selection in principal component analysis and discriminant analysis. *Computational Statistics & Data Analysis* 77: 38–53.
40. Nchare, M., B. Shen, and S.G. Anagho. 2012. Co-processing vacuum residue with waste plastics in a delayed coking process: Kinetics and modeling. *China Petroleum Processing Petrochemical Technology* 14 (3): 44–49.
41. Bello, O.O., et al. 2006. Effects of operating conditions on compositional characteristics and reaction kinetics of liquid derived by delayed coking of Nigerian petroleum residue. *Brazilian Journal of Chemical Engineering* 23 (3): 331–339.
42. Zhang, R., and S. Wang. 2008. Support vector machine based predictive functional control design for output temperature of coking furnace. *Journal of Process Control* 18 (5): 439–448.
43. Zhang, R., and J. Tao. 2017. Data-driven modeling using improved multi-objective optimization based neural network for coke furnace system. *IEEE Transactions on Industrial Electronics* 64 (4): 3147–3155.
44. Zhang, R., Q. Lv, J. Tao, et al. 2018. Data driven modeling using an optimal principle component analysis based neural network and its application to a nonlinear coke furnace. *Industrial and Engineering Chemistry Research* 57 (18): 6344–6352.
45. Tao, J., X. Chen, and Z. Yong. 2012. Constraint multi-objective automated synthesis for CMOS operational amplifier. *Neurocomputing* 98 (18): 108–113.
46. Li, Y., C. Fang, and Q. Ouyang. 2004. Genetic algorithm in DNA computing: A solution to the maximal clique problem. *Chinese Science Bulletin* 49 (9): 967–971.
47. Martin, N. and H. Maes. 1979 *Multivariate analysis.* London: Academic Press.