10

Multi-Objective Evolutionary Algorithm for Radial Basis Function Neural Network Design

Gary G. Yen

School of Electrical and Computer Engineering
Oklahoma State University
Stillwater, OK 74078-5032, USA
gyen@okstate.edu

Summary. In this chapter, we present a multiobjective evolutionary algorithm based design procedure for radial-basis function neural networks. A Hierarchical Rank Density Genetic Algorithm (HRDGA) is proposed to evolve the neural network's topology and parameters simultaneously. Compared with traditional genetic algorithm based designs for neural networks, the hierarchical approach addresses several deficiencies highlighted in literature. In addition, the rank-density based fitness assignment technique is used to optimize the performance and topology of the evolved neural network to tradeoff between the training performance and network complexity. Instead of producing a single *optimal* solution, HRDGA provides a set of near-optimal neural networks to the designers so that they can have more flexibility for the final decision-making based on certain preferences. In terms of searching for a near-complete set of candidate networks with high performances, the networks designed by the proposed algorithm prove to be competitive, or even superior, to three state-of-the-art designs for radial-basis function neural networks to predict Mackey-Glass chaotic time series.

10.1 Introduction

Neural Networks (NN's) and Genetic Algorithms (GA's) represent two emerging technologies inspired by biologically motivated computational paradigms. NN's are derived from the information-processing framework of a human brain to emulate the learning behavior of *an individual*, while GA's are motivated by the theory of evolution to evolve *a whole population* toward better fitness. Although these two technologies seem quite different in the time period of action, number of involved individuals, and the process scheme, their similar dynamic behaviors stimulate research on whether a synergistic combination of these two technologies may provide more problem solving power than either alone [22].

There has been an extensive analysis of different classes of neural networks possessing various architectures and training algorithms. Without a proven

G.G. Yen: *Multi-Objective Evolutionary Algorithm for Radial Basis Function Neural Network Design*, Studies in Computational Intelligence (SCI) **16**, 221–239 (2006)
www.springerlink.com

guideline, the design of an optimal neural network for a given problem is an *ad hoc* process. Given a sufficient number of neurons, more than one neural network structure (i.e., with different weighting coefficients and numbers of neurons) can be trained to solve a given problem within an error bound if given sufficient training time. The decision of "which network is the best" is often decided by which network will better meet the user's needs for a given problem. It is known that the performance of neural networks is sensitive to the number of neurons. Too few neurons can result in underfitting problems (poor approximation), while too many neurons may contribute to overfitting problems. Obviously, achieving a better network performance and simplifying the network topology are two competing objectives. This has promoted research on how to identify an optimal and efficient neural network structure. AIC (Akaike Information Criterion) [19] and PMDL (Predictive Minimum Description Length) [8] are two well-adopted approaches. However, AIC can be inconsistent and has a tendency to overfit a model, while PMDL only succeeded in relatively simple neural network structures and seemed very difficult to extend to a complex NN structure optimization problem. Moreover, all of these approaches tend to produce a single neural network for each run, offering the designers no alternative choices.

Since the 1990's, evolutionary algorithms have been successfully applied to the design of network topologies and the choice of learning parameters [1, 2, 18, 17, 15]. They reported some encouraging results that are comparable with conventional neural network design approaches. However, multiobjective trade-off characteristic of the neural network design has not been well studied and applied in the real world applications. In this chapter, we propose a Hierarchical Rank Density Genetic Algorithm (HRDGA) for neural network design in order to evolve a set of near-optimal neural networks. Without loss of generality, we will restrict our discussions to the radial basis function neural network. In HRDGA, each chromosome is a candidate neural network and is coded by three different gene segments– high level segments have control genes that can determine the status (activated or deactivated) of genes in lower level segments. Hidden layers and neurons are added or deleted by this "on/off" scheme to achieve an optimal structure through a survival of the fittest evolution. Meanwhile, weights and biases are evolved along with the neural network topology. Treating the neural network design as a bi-objective optimization problem, a new rank-density based fitness assignment technique is developed to evaluate the structure complexity and the performance of the evolved neural network. More importantly, instead of a single network, HRDGA produces a set of near-optimal candidate networks with different trade-off traits from which the designers or decision makers can make flexible choices based on their preferences.

The remainder of this chapter is organized as follows. Section 10.2 discusses the neural network design dilemma and the difficulty of finding a single *optimal* neural network. Section 10.3 reviews various approaches to applying genetic algorithms for neural network design and introduces the proposed hierarchi-

cal structure, genetic operators, and multi-fitness measures of the proposed genetic algorithm. Section 10.4 applies hierarchical genotype representation to a radial-basis function neural network design. Section 10.5 introduces the proposed rank-density fitness assignment technique for multiobjective genetic algorithms and describes HRDGA parameters and design flowchart. Section 10.6 presents a feasible study on the Mackey-Glass chaotic time series prediction using HRDGA evolved neural networks. A time series with chaotic character is trained and the performance is compared with those of the k-nearest neighbors, generalized regression, and orthogonal least square training algorithms. Finally, Section 10.7 provides some concluding remarks along with pertinent observations.

10.2 Neural Network Design Dilemma

To generate a neural network that possesses the practical applicability, several essential conditions need to be considered.

1. A training algorithm that can search for the optimal parameters (i.e., weights and biases) for the specified network structure and training task.
2. A rule or algorithm that can determine the network complexity and ensure it to be sufficient for solving the given training problem.
3. A metric or measure to evaluate the reliability and generalization of the produced neural network.

The design of an optimal neural network involves all of these three problems. As given in [6], the ultimate goal of the construction of a neural network with the input-output relation $\mathbf{y} = f_{NS}(\mathbf{x}, \omega)$ is the minimization of the expectation of a cost function $g_T(f_{NS}(\mathbf{X}, \omega), \mathbf{Y})$ as:

$$E[g_T(f_{NS}(\mathbf{X}, \omega, \mathbf{Y})] = \int \int g_T(f_{NS}(\mathbf{x}, \omega), \mathbf{y}) f_{x,y}(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y}, \tag{10.1}$$

where $f_{\mathbf{x},\mathbf{y}}(\mathbf{x}, \mathbf{y})$ denotes the joint *pdf* that depends on the input vector $\mathbf{x}$ and the target output vector $\mathbf{y}$. $\mathbf{X}$ and $\mathbf{Y}$ are spaces spanned by all individual training samples, $\mathbf{x}$ and $\mathbf{y}$. Given a network structure NS, a family of input-output relations $F_{NS} = \{f_{NS}(\mathbf{x}, \omega)\}$, parameterizd by ω, consisting of all network functions that may be formed with different choices of the weights can be assigned. The structure NS' is said to be *dominated* by NS" if $F_{NS'} \subset F_{NS"}$. In order to choose the optimal neural network, two problems have to be solved.

1. Determination of the network function $f^*_{NS}(\mathbf{x})$ (i.e., the determination of the respective weights) that gives the minimal cost value within the family F_{NS}:

$$f^*_{NS}(\mathbf{x}) = f_{NS}(\mathbf{x}, \omega^*) = \arg\ \min_{\omega} \mathbf{E}[g_L(f_{NS}(\mathbf{X}, \omega), \mathbf{Y})], \tag{10.2}$$

where $g_L(\cdot, \cdot)$ denotes the cost function measuring the performance over the training set.

2. Determination of the network structure NS^* that realizes the minima cost value within a set of structures $\{NS\}$:

$$NS^* = \arg\ \min_{NS \in F_{NS}} \mathbf{E}[g_T(f^*_{NS}(\mathbf{X}), \mathbf{Y})]. \tag{10.3}$$

Obviously, the solutions of both tasks need not result into a unique network. In [6], if several structures $NS^*_1, NS^*_2, \cdots$ meet the criterion as shown in Equation (10.3), the one with the minimal number of hidden neurons is defined as an *optimal*. However, as a neural network can only tune the weights by the given training data sets, and these data sets are always finite, there will be a trade-off between NN learning capability and the number of the hidden neurons. A network with insufficient neurons might not be able to approximate well enough the functional relationship between input and target output. On the other hand, if the number of neurons is excessive, the realized network function will depend greatly on the resulting realization of the given limited training set. This trade-off characteristic implies that a single *optimal* neural network is very difficult to find as extracting $f^*_{NS}(\mathbf{x})$ from F_{NS} by using a finite training data set is a difficult task, if not impossible [9]. Therefore, instead of trying to obtain a single *optimal* neural network, finding a set of *near-optimal* networks with different network structures seems more feasible. Each individual in this neural network set may provide different training and test performances for different training and test data sets. Moreover, the idea of providing "a set of" candidate networks to the decision makers can offer more flexibilities in selecting an appropriate network judged by their own preferences. For this reason, genetic algorithms and multiobjective optimization techniques can be introduced in neural network design problems to evolve network topology along with parameters and present a set of alternative network candidates.

10.3 Neural Network Design with Genetic Algorithm

In the literature of applying genetic algorithms to assist neural networks design, several approaches have been proposed for different objectives. These approaches can be categorized into four different areas.

10.3.1 Data Preparation

GA's were primarily used to help NN design by pre-processing data. Kelly and Davis used a genetic algorithm to find the rotation of a data set and scaling factors for each attribute to improve the performance of a KNN classifier [13]. Chang and Lippmann used a genetic algorithm to reduce the dimensionality of a feature set for a KNN classifier in a speech recognition task [3].

10.3.2 Evolving Network Parameters

Belew and his colleagues used a genetic algorithm to identify a good initial weighting configuration for a back-propagation network [1]. On the other hand, Bruce and Timothy used a genetic algorithm to evolve the centers and widths for a radial basis function neural network [2]. Genetic algorithms are used to evolve the weights or biases in a fixed topology neural network. The structure, number of layers and number of neurons, is pre-determined based upon some heuristic judgments.

10.3.3 Evolving Network Topology

This is the most targeted area with which genetic algorithm can be used in neural network design. Miller *et. al.*, used a genetic algorithm to evolve an optimally connected matrix to form a neural network [16], but this method can only be used for simple problems. Whiltley *et. al.*, used a genetic algorithm to find which links could be eliminated in order to achieve a specific learning objective [21]. Lucas proposed a GA based adaptive neural architecture selection method to evolve a back-propagation neural network [15], and Davila applied GA's schema theory to aid the design of genetic coding for NN topology optimization. Three main problems exist in current topology design research, namely network feasibility, one genotype mapping multiple phenotypes and one phenotype mapping different genotypes [23].

10.3.4 Evolving NN Structures together with Weights and Biases

Dasgupta and McGregor proposed an sGA (Structure Genetic Algorithm) to evolve neural networks [5]. But in their work, only a XOR problem and a 4×4 encoder/decoder problem were tested, which is relatively simple. Since an N-neuron neural network must be expressed as a chromosome with a bit string of length N^2, a complex phenotype will map to a much more complex genotype. As a result, using sGA to evolve a large neural network is computationally expensive, if not impossible. Zhang and Cho proposed Bayesian evolutionary algorithms to evolve the structures and parameters of neural trees, which are then used to predict a time series [24]. However, both of these algorithms use the connection matrix and from-to units, which had been shown to easily produce "one phenotype mapping different genotypes" problem.

To avoid this problem, a hierarchical genotype representation is adopted in this study. HGA (Hierarchical Genetic Algorithm) was first proposed by Ke for fuzzy controller design [12]. They used two layer genes to evolve membership functions for a fuzzy logic design. Based on this idea, Yen and Lu designed an HGA Neural Network (HGA-NN) [23]. In the HGA-NN, a three-layer HGA is used to evolve a Multi-layer Perceptron (MLP) neural network. The chromosome structure (genotype) is shown in Figure 10.1(a). As shown

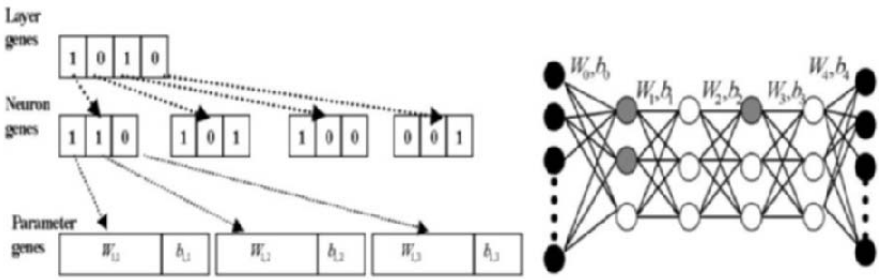


Fig. 10.1. Genotype structure of an individual MLP neural network (left) and the corresponding phenotype with layered neural network topology (right).

in Figure 10.1(a), each candidate chromosome corresponding to a neural network implementation is assumed to have at most four hidden layers (shown in the high-level *layer genes*), where the first and the third hidden layers are activated (as indicated with binary bits 1) and the second and the fourth hidden layers are deactivated (with binary bits 0). Additionally, we assume at most three neurons in each hidden layer as shown in the space available in neuron gene corresponding to each element in layer gene.

The mid-level *neuron genes* indicate that two out of three neurons in the first hidden layer are activated, while only one neuron in the third hidden layer is activated. Since the second and the fourth layers are deactivated, their neurons are not used. The low-level *parameter genes* are then used to represent the weighting and bias parameters of each corresponding neuron activated. The active status of one control gene determines whether the parameters of the next level controlled by this gene will be activated or not. As an example, a genetic chromosome (genotype) shown in Figure 10.1(a) corresponds to an individual neural network (phenotype) with two hidden layers and two and one neuron in each layer in Figure 10.1(b). By using this hierarchical genotype, the problem of "one phenotype mapping different genotypes" can be prevented.

10.4 HGA Evolved Radial-Basis Function NN

In a similar spirit, HGA is tailored in this chapter to evolve an RBF (Radial-Basis Function) neural network. A radial-basis function can be formed as:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \omega_i \exp(-||\mathbf{x} - \mathbf{c}_i||^2), \tag{10.4}$$

where $\mathbf{c}_i$denotes the center of the ith localized function, ω_i is the weighting coefficient connecting the ith Gaussian neuron to the output neuron, and m is the number of Gaussian neurons in the hidden layer. Without loss of generality, we choose the variance as unity for each Gaussian neuron.

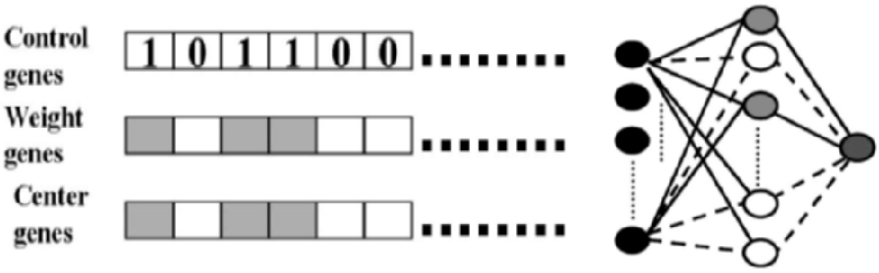


Fig. 10.2. Genotype (left) and phenotype (right) of HGA based RBF neural network.

In HGA based RBF neural network design, genes in the genotype are hierarchically structured into three layers: control genes, weight genes, and center genes. The lengths of these three layers of genes are the same and specified by the user. The value of each control gene (0 or 1) determines the activation status (off or on) of the corresponding weight gene and center gene. On the other hand, the weight genes and center genes are represented by real values. Control genes and weight genes are randomly initialized and the center genes are randomly selected from given training data samples. Figure 10.2 shows the genotype and phenotype of a HGA based RBF neural network, where the first, third and fifth hidden neurons are activated. As a result, their corresponding weight and center parameters are used.

10.5 Multiobjective Genetic Algorithm

As discussed in Section 10.3, neural network design problems have a multiobjective trade-off characteristic in terms of optimizing network topology and performance. Therefore, multiobjective genetic algorithm is applied in NN design procedure.

10.5.1 Multiobjective Optimization Problems

Multiobjective Optimization (MO) is a very important research topic, because most real world problems have not only a multiobjective nature, but also many open issues to be answered qualitatively and quantitatively. In many optimization problems, there is not even a universally accepted definition of "optimum" as in single-objective optimization [10], because the solution to a MO problem is generally not a single point. It consists of a family of non-dominated points, a so-called Pareto front [7], which describes the trade-off among contradicted objectives. The Pareto front yields many candidate solutions— non-dominated points, from which we can choose the desired one under different trade-off conditions. In most cases, the Pareto front is on the

boundary of the feasible range as shown in Figure 10.3. Considering the NN design dilemma outlined in Section 10.2, a neural network design problem can be regarded as a class of MO problems as minimizing network structure and improving network performance, which are two conflicting objectives. Therefore, searching for a near-complete set of non-dominated and near-optimal candidate networks as the design solutions (i.e., Pareto front) is our goal.

10.5.2 Rank-density Based Fitness Assignment

Since the 1980's, several Multiobjective Genetic Algorithms (MOGAs) have been proposed and applied in multiobjective optimization problems [25]. These algorithms all have almost the same purpose— searching for a uniformly distributed and near-optimal Pareto front for a given MO problem. However, this ultimate goal is far from been accomplished by the existing MOGAs described in literature due to trade-off decisions between homogenously distributing the computational resources and GA's strong tendencies to restrict searching efforts (i.e., genetic drift).

In this chapter, we propose a new rank-density based fitness assignment technique in a multiobjective genetic algorithm to assist neural network design. Compared to traditional fitness assignment methods, the proposed rank-density based technique possesses the following characteristics of a) simplifying the problem domain by converting high-dimensional multiple objectives into two objectives to minimize the individual rank value and population density value, b) searching for and keeping better-approximated Pareto points by diffusion and elitism schemes, and c) preventing harmful individuals by introducing a "forbidden region" concept. Three essential techniques were applied in this technique.

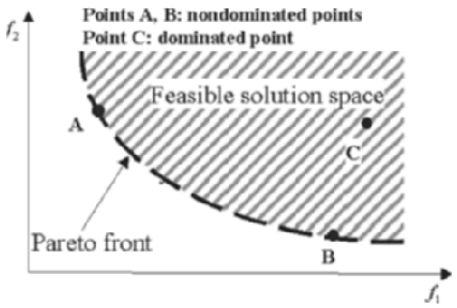


Fig. 10.3. Graphical illustration of the Pareto optimality.

Automatic Accumulated Ranking Strategy (AARS)

In HRDGA, an Automatic Accumulated Ranking Strategy (AARS) is applied to calculate the Pareto rank value, which represents the dominated

relationship among individuals. In AARS, an individual's rank value is defined as the summation of the rank values of the individuals that dominate it. Assume at generation t, individual y is dominated by $p^{(t)}$ individuals $y_1, y_2, \cdots, y_{p^{(t)}}$, whose rank values are already known as $\text{rank}(y_1, t)$, $\text{rank}(y_2, t)$, $\cdots$, $\text{rank}(y_{p^{(t)}}, t)$. Its rank value can be computed by

$$\text{rank}(y, t) = 1 + \sum_{j=1}^{p^{(t)}} \text{rank}(y_j, t). \tag{10.5}$$

Therefore, by AARS, all the non-dominated individuals are still assigned rank value 1, while dominated ones are penalized to reduce the population density and redundancy.

Adaptive Density Value Calculation

To maintain the diversity of the obtained Pareto front, HRDGA adopts an adaptive cell density evaluation scheme as shown in Figure 10.4. The cell width in each objective dimension can be computed as :

$$d_i = \frac{\max_{x \in X} f_i(\mathbf{x}) - \min_{x \in X} f_i(\mathbf{x})}{K_i}, \ i = 1, \cdots, n, \tag{10.6}$$

where d_i is the width of the cell in the ith dimension, K_i denotes the number of cells designated for the ith dimension (i.e., in Figure 10.4, $K_1 = 12$ and $K_2 = 8$), and X denotes the decision vector space. As the maximum and minimum fitness values will change with different generations, the cell size will vary from generation to generation to maintain the accuracy of the density calculation. The density value of an individual is defined as the number of the individuals located in the same cell.

Rank-density Based Fitness Assignment

Because rank and density values represent fitness and population diversity, respectively, the new rank-density fitness formulation can convert any multiobjective optimization problem into a bi-objective optimization problem. Here, population rank and density values are designated as the two fitness values for GA to minimize. Before fitness evaluation, the entire population is divided into two subpopulations with equal sizes; each subpopulation is filled with individuals that are randomly chosen from the current population according to rank and density value, respectively. Afterwards, the entire population is shuffled, and crossover and mutation are then performed.

For crossover, the parent selection and replacement schemes are borrowed from Cellular GA [14] to explore the new search area by "diffusion." For each subpopulation, a fixed number of parents are randomly selected for crossover. Then, each selected parent performs crossover with the best individual (the

one with the lowest rank value) within the same cell and the nearest neighboring cells that contain individuals. If one offspring produces better fitness (a lower rank value or a lower population density value) than its corresponding parent, it replaces its parent. The replacement scheme of the mutation operation is analogous.

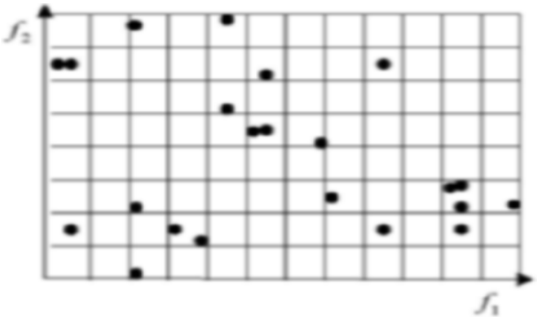


Fig. 10.4. Density map and density grid.

Meanwhile, we take the minimization of the population density value as one of the objectives. It is expected that the entire population will move toward an opposite direction to the Pareto front where the population density value is being minimized. Although moving away from the true Pareto front can reduce population density value, obviously, these individuals are harmful to the population to converge to the Pareto front. To prevent "harmful" offspring surviving and affecting the evolutionary direction and speed, a *forbidden region* concept is proposed in the replacement scheme for the density subpopulation, thereby preventing the "backward" effect. The *forbidden region* includes all the cells dominated by the selected parent. The offspring located in the forbidden region will not survive in the next generation, and thus the selected parent will not be replaced. As shown in Figure 10.5, suppose our goal is to minimize objectives f_1 and f_2, and a resulting offspring of the selected parent $p$is located in the forbidden region. This offspring will be eliminated even if it reduces the population density, because this kind of offspring has the tendency to push the entire population away from the desired evolutionary direction.

Finally, the simple elitism scheme [7] is also applied as the bookkeeping for storing the Pareto individuals obtained in each generation. These individuals are compared to achieve the final Pareto front after the evolution process has stopped.

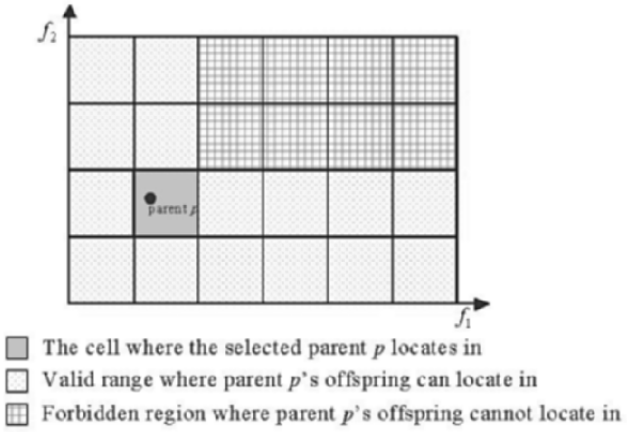


Fig. 10.5. Illustration of the valid range and the forbidden region.

10.5.3 HRDGA for NN Design

To assist RBF network design, HRDGA is applied to carry out the fitness evaluation and mating selection schemes. The HRDGA operators are designed as followed.

Chromosome Representation

In HRDGA, each individual (chromosome) represents a candidate neural network. The control genes are binary bits (0 or 1). For the weight and center genes, real values are adopted as the gene representation to reduce the length of the chromosome. The population size is fixed and chosen *ad hoc* by the difficulty of the problem to be solved.

Crossover and Mutation

We used one-point crossover in the control gene segments and two-point crossover in the other two gene segments. The crossover points were randomly selected and the crossover rates were chosen to be 0.8, 0.7, and 0.7 for the control, weight, and center genes, respectively. One-point mutation was applied in each segment. In the control gene segment, common binary value mutation was adopted. In the weight and center gene segments, real value mutation was performed by adding aGaussian$(0, 1)$, which denotes a Gaussian function with zero mean and unit variance. The mutation rates were set to be 0.1, 0.05, and 0.05 for the control, weight, and center genes, respectively.

Fitness Evaluations and Mating Selection

Since we are trying to use HRDGA to optimize the neural network topology along with its performance, we need to convert them into the rank-density domain. Therefore, the original fitness— network performance and number of neurons—of each individual in a generation is evaluated and ranked, and the density value is calculated. Then the new rank and density fitness values of each individual will be evaluated and the individuals with higher fitness measures will reproduce and crossover with other high fitness individuals with a certain probability. Their offspring replaces the low fitness parents forming a new generation. Mating is then iteratively processed.

Stopping Criteria

When the desired number of generations is met, the evolutionary process stops.

10.6 Mackey-Glass Chaotic Time Series Prediction

Since the proposed HRDGA is designed to evolve the neural network topology together with its best performance, it proves useful in solving complex problems such as time series prediction or pattern classification. For a feasibility check, we use the HRDGA assisted NN design to predict the Mackey-Glass chaotic time series.

10.6.1 Mackey-Glass Time Series

The Mackey-Glass time series is a continuous time-delay data differential equation:

$$\frac{d(x(t))}{d(t)} = \frac{a\ x(t-\tau)}{(1+x^c(t-\tau))} - b\ x(t). \tag{10.7}$$

The chaotic behavior of the Mackey-Glass time series is determined by the delay parameter τ. Some examples are listed in Table 10.1. Larger values of τ produce more chaotic dynamics which are much more difficult to predict. Here we assign $a = 0.2$, $b = 0.1$ and $c = 10$ for Equation (10.7). In this study, we used HRDGA evolved neural networks to predict a chaotic Mackey-Glass time series with $\tau = 150$. The network is set to predict $x(t+6)$ based on $x(t)$, $x(t-6)$, $x(t-12)$, and $x(t-18)$.

In the proposed HRDGA, 150 initial center genes are selected, 150 control genes and 150 weight genes are initially generated as well. Population size was set to be 400. For comparison, we applied three other center selection methods—KNN (K-Nearest Neighbor) [11], GRNN (Generalized Regression

Table 10.1. Characteristics of Mackey-Glass time series

delay parameter τ	Chaotic characteristics
$\tau < 4.53$	A stable fixed point attractor
$4.53 < \tau < 13.3$	A stable limit cycle attractor
$13.3 < \tau < 16.8$	Period limit cycle doubles
$\tau > 16.8$	Chaotic attractor characterized by τ

Neural Network) [20], and OLS (Orthogonal Least Square Error) [4] methods on the same time series prediction problem. For KNN and GRNN types of networks, 70 networks are generated with the neuron numbers increasing from 11 to 80 with the step equals to one. Each of these networks will be trained by KNN and GRNN methods, respectively, and the stopping criterion is the same with the one used in HRDGA. For the OLS method, the selection of the tolerance parameter ρ determines the trade-off between the performance and complexity of the network. Ideally, ρ should be larger than, but very close to, the ratio $\sigma_\varepsilon^2/\sigma_d^2$, where σ_ε^2 is the variance of the residuals, and σ_d^2 is the variance of the desired output. A smaller ρ value will produce a neural network with more neuron number, whereas a larger ρ value generally results in a network with less number of neurons. Therefore, by using different ρ values, we generated a group of neural networks with various training performances and numbers of hidden neurons. For the given Mackey-Glass time series prediction problem, we selected 100 different ρvalues, which are from 0.01 to 0.4 with the step size of 0.01. The stopping criteria for KNN, GRNN, and OLS algorithms is either the epochs exceed 5,000, or the training Sum Square Error (SSE) between two sequential generations is smaller than 0.01. For HRDGA, the stopping generation is set to be 5,000. We used the first 250 seconds of the data as the training data set, and then the data from 250 – 499, 500 – 749, 750 – 999, and 1,000 – 1,249 seconds were used as the corresponding test data sets to be predicted by four different approaches. Each approach runs 30 times with different parameter initializations to obtain the average results. Figure 10.6(a) shows the resulting average training SSEs of neural networks with different number of hidden neurons by four training approaches. Figure 10.6(b) shows the approximated Pareto fronts (i.e., non-dominated sets) by the selected four approaches. Figure 10.7(a) shows the average test SSEs of the resulting networks by using the first test data set for each approach, and Figure 10.7(b) shows their corresponding Pareto fronts. Furthermore, Figures 10.8, 10.9 and 10.10 show the same types of results by using the second, third, and fourth test data, respectively.

Table 10.2 shows the best training and test performances and their corresponding numbers of hidden neurons. From Figures 10.6- 10.10, we can see, comparing to KNN and GRNN, HRDGA and OLS algorithms have much smaller training and test errors for the same network structures. KNN trained networks produce the worst performances, because the RBF centers of the

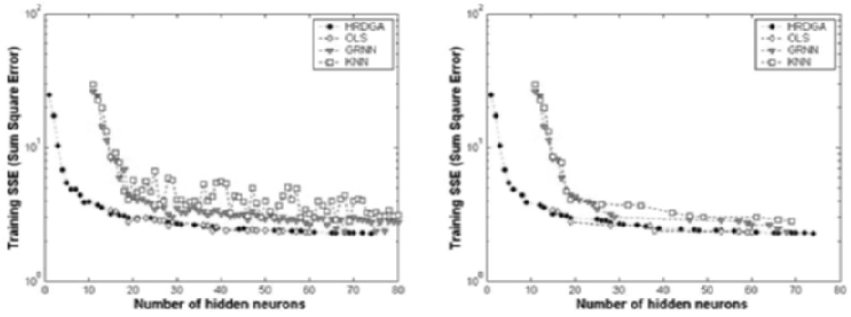


Fig. 10.6. (a) Training performances for the resulting neural networks with different number of hidden neurons and (b) The corresponding Pareto fronts (non-dominated sets).

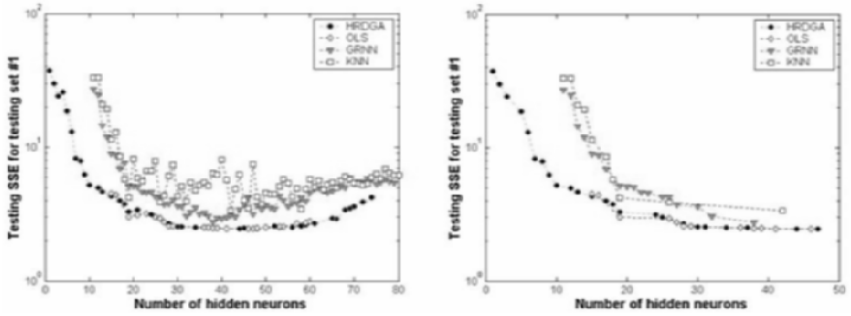
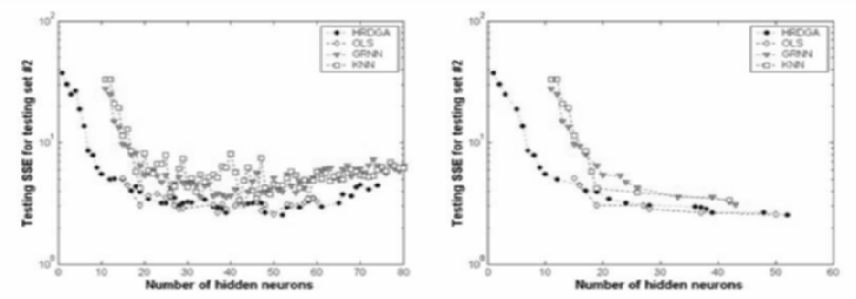


Fig. 10.7. (a) Test performances for the resulting neural networks with different number of hidden neurons and by using test set #1 (b) The corresponding Pareto fronts (non-dominated sets).

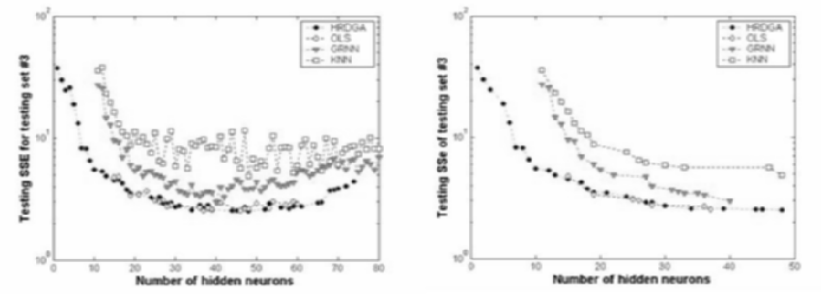
Table 10.2. Comparison of best performance (SSE) and structure (number of neurons) between KNN, OLS, GRNN and HRDGA

	Training set		**Test set #1**		**Test set #2**		**Test set #3**		**Test set #4**	
	SSE	no.	SSE	no.	SSE	no.	SSE	no.	SSE	no.
KNN	2.8339	69	3.3693	42	3.4520	42	4.8586	48	4.8074	19
GRNN	2.3382	68	2.7720	38	3.0711	43	2.9644	40	3.2348	37
OLS	2.3329	60	2.4601	46	2.5856	50	2.5369	37	2.7199	54
HRDGA	2.2901	74	2.4633	47	2.5534	52	2.5226	48	2.7216	58



(a) (b)

Fig. 10.8. (a) Test performances for the resulting neural networks with different number of hidden neurons and by using test set #2 (b) The corresponding Pareto fronts (non-dominated sets).



(a) (b)

Fig. 10.9. (a) Test performances for the resulting neural networks with different number of hidden neurons and by using test set #3 (b) The corresponding Pareto fronts (non-dominated sets).

KNN algorithm are randomly selected, which make KNN to achieve only a "local optimum" solution. Since GA always seeks "global optimum", and the orthogonal result is near optimal, the performances of OLS are comparable to HRDGA.

Moreover, from Figure 10.6, we can see that when the network complexity increases, the training error decreases. This phenomenon can be observed from the results by all of the selected training approaches. However, this phenomenon is only partially maintained for the relationship between the test performances and the network complexity. Before the number of hidden neurons reaches a certain threshold, the test error still decreases as the network complexity increases. After that, the test error has the tendency to fluctuate even when the number of hidden neurons increases. This occurrence can be considered as that the resulting networks are overfitted. The network with the

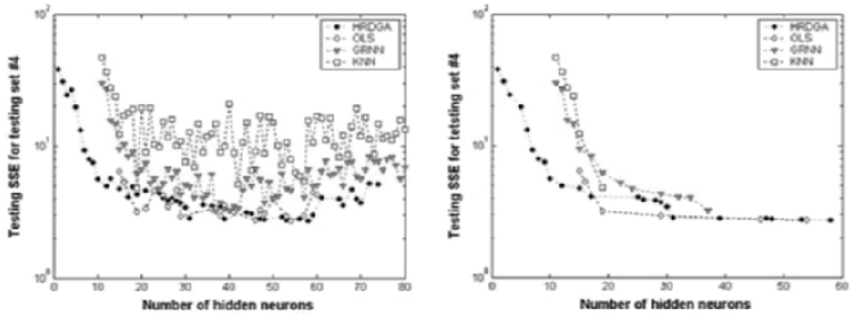


Fig. 10.10. (a) Test performances for the resulting neural networks with different number of hidden neurons and by using test set #4 (b) The corresponding Pareto fronts (non-dominated sets).

best test performance before overfitting occurs is called the *optimal* network and is judged as the final single solution by conventional NN design algorithms. However, from Figures 10.6– 10.10 and Table 10.1, it is very difficult to find a single *optimal* network that can offer the best performances for all the test data sets, since these data sets possess different traits. Therefore, instead of searching for a single *optimal* neural network, an algorithm that can result in a near-complete set of near-optimal networks can be a more reasonable and applicable option. This is the essential reason that multiobjective genetic algorithms can be justified for this type of neural network design problems.

From the simulation results, althoughK KNN and GRNN approaches did not provide better training and test results comparing to the other two approaches, they have the advantage that the designer can control the network complexity by increasing or decreasing the neuron numbers at will. On the other hand, although the OLS algorithm always provides near-optimal network solutions with good training and test performance, it also has serious problem to generate a set of network solutions in that the designers cannot manage the network structure directly. The trade-off characteristic between network performance and complexity totally depends on the value of tolerance parameterρ. Same ρ value means completely different trade-off features for different NN design problems. In addition, as shown in Figure 10.11, the relationship between ρ value and network topology is a nonlinear, many-to-one mapping, which may cause a redundant computation effort in order to generate a near-complete neural network solution set. Compared with the other three training approaches, HRDGA does not have problems in designing trade-off parameters, because it treats each objective equally and independently, and its population diversity preserving techniques help it build a near-uniformly distributed non-dominated solution set.

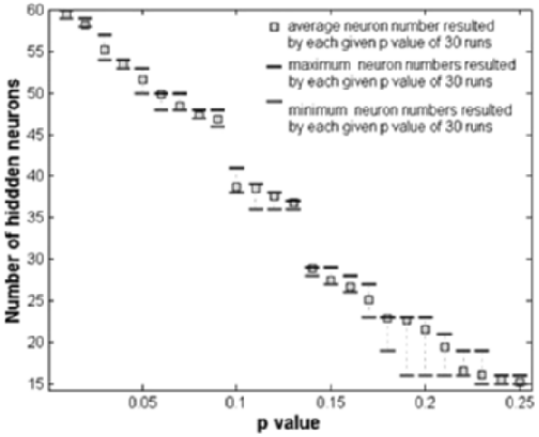


Fig. 10.11. Relationship between ρ values and network complexity.

Therefore, comparing to the other three traditional training approaches, the proposed HRDGA algorithm offers several benefits for the neural network design problems in terms of:

1. providing a set of candidate solutions, which is caused by GA's population-based optimization capability and the definition of Pareto optimality;
2. presenting competitive or even superior individuals with high training and test performances. This is resulted from GA's feature of seeking "global optimum" and HRDGAs' Pareto ranking technique; and
3. offering a near-complete, non-dominated set, and long-extended Pareto front, which is originated from HRDGA's population diversity keeping design that can be found in AARS, density preserving technique, and the concept of "forbidden region".

10.7 Conclusions

In this study, we propose a multiobjective genetic algorithm based design procedure for the radial-basis function neural network. A Hierarchical Rank Density Genetic Algorithm (HRDGA) is developed to evolve both the neural network's topology and parameters simultaneously. Instead of producing a single solution, HRDGA provides a set of near-optimal neural networks from the perspective of Pareto optimality to the designers so that they can have more flexibility for the final decision-making based on certain preferences. From the results presented above, HRDGA shows potential in estimating neural network topology and weighting parameters for complex problems when a heuristic estimation of the neural network structure is not readily available. For the given Mackey–Glass chaotic time series prediction, HRDGA shows competitive, or

even superior performances comparing with the other three selected training algorithms in terms of searching for a set of non-dominated, near-complete neural network solutions with high training and test performances. While we considered radial-basis function neural networks, the proposed hierarchical genetic algorithm may be easily extended to the designs of other neural networks (i.e., feed-forward, feedback, or self-organized). In addition, as some of the traditional neural network training approaches (i.e., OLS algorithm) also provide competitive results, a hybrid algorithm that synergistically integrates traditional training method with the proposed algorithm can be a promising future work.

References

[1] R.K. Belew, J. McInerney, N.N. Schraudolph. Evolving networks: Using genetic algorithms with connectionist learning, *CSE Technical Report*, University of California at Dan Diego, CS90-174, 1990

[2] A.W. Bruce, D.C. Timothy. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions Neural Networks*, 7:869-880, 1996

[3] E.J. Chang, R.P. Lippmann. Using genetic algorithms to improve pattern classification performance. *Neural Information Processing Systems*, 797–803, 1991

[4] S. Chen, C.F. Cowan, P.M. Grant. Orthogonal least square learning algorithm for radial basis function networks. *IEEE Trans. Neural Networks*, 2:302–309, 1991

[5] D. Dasgupta, D.R. McGregor. Designing application-specific neural networks using the structured genetic algorithm. In: *Proc. Int. Workshop on Combinations of Genetic Algorithms and Neural Networks*, pp.87–96, 1992

[6] A. Doering, M. Galicki, H. Witte. Structure optimization of neural networks with the A*-Algorithm. *IEEE Trans. Neural Networks*, 8:1434–1445, 1997

[7] C.M. Fonseca, P.J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3:1–16, 1995

[8] X.M. Gao, S.J. Ovaska, Z.O. Hartimo. Speech signal restoration using an optimal neural network structure. In: *Proc. of the IEEE Int. Conf. Neural Networks*, pp.1841–1846, 1996

[9] S. Geman, E. Bienenstock , R. Dousat. Neural networks and the bias/variance dilemma. *Neural Computation*, 2:303–314, 1989

[10] C.L. Hwang, A.S.M. Masud. Multiple objective decision making—methods and applications. Springer, Berlin

[11] T. Kaylani, S. Dasgupta. A new method for initializing radial basis function classifiers. In: *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, pp.2584–2587, 1994

[12] T.Y. Ke, K.S. Tang, K.F. Man, P.C. Luk. Hierarchical genetic fuzzy controller for a solar power plant. In: *Proc. IEEE Int. Symp. Industrial Electronics*, pp.584–588, 1998

[13] J.D. Kelly, L. Davis. Hybridizing the genetic algorithm and the K-nearest neighbors classification algorithm. In: *Proc. Intl. Conf. Genetic Algorithms*, pp.377–383, 1991

[14] T. Krink, R.K. Ursem. Parameter control using agent based patchwork model. In: *Proc. IEEE Congress on Evolutionary Computation*, pp.77–83, 2000

[15] S. Lucas. Specifying intrinsically adaptive architectures. In: *Proc. 1st IEEE Symp. Combination of Evolutionary Computation and Neural Networks*, pp.224–231, 2000

[16] G.F. Miller, P.M. Todd, S.U. Hedge. Designing neural networks using genetic algorithms. In: *Proc. 3rd Int. Conf. Genetic Algorithms*, pp.379–384, 1989

[17] S.W. Moon, S.G. Kong. Block-based neural network. *IEEE Transactions on Neural Networks*, 12:307–317, 2001

[18] A.K. Morales. Non-standard norms in genetically trained neural networks. In: *Proc. IEEE Symp. Combination of Evolutionary Computation and Neural Networks*, pp.43–51, 2000

[19] N. Murata, S. Yoshizawa, S. Amari. Network information criterion – determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5:865–872, 1994

[20] P.D. Wasserman. Advanced Method in Neural Computing. VNR, New York, 1993

[21] D. Whitley, T. Starkweather, C. Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computting*, 14:347–361, 1990

[22] X. Yao. Evolving artificial neural network. *International Journal of Neural Systems*. 4:203–222, 1993

[23] G.G. Yen, H. Lu. Hierarchical genetic algorithm based neural network design. In: *Proc. IEEE Symp. Combination of Evolutionary Computation and Neural Networks*, pp.168–175, 2000

[24] B. Zhang, D. Cho. Evolving neural trees for time series prediction using Bayesian evolutionary algorithms. In: *Proc. IEEE Symp. Combination of Evolutionary Computation and Neural Networks*, 17–23, 2000

[25] E. Zitzler, L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation*, 3:257–271, 1999