HIERARCHICAL RANK DENSITY GENETIC ALGORITHM FOR RADIAL-BASIS FUNCTION NEURAL NETWORK DESIGN

Gary G. Yen     Haiming Lu  
Intelligent Systems and Control Laboratory 
School of Electrical and Computer Engineering 
Oklahoma State University  
Stillwater, OK 74078-5032

Abstract—In this paper, we propose a genetic algorithm based design procedure for a radial-basis function neural network. A Hierarchical Rank Density Genetic Algorithm (HRDGA) is used to evolve both the neural network’s topology and parameters. In addition, the rank-density based fitness assignment technique is used to optimize the performance and topology of the evolved neural network to deal with the confliction between the training performance and network complexity. Instead of producing a single optimal network, HRDGA provides a set of near-optimal neural networks to the designers or the decision makers so that they can have more flexibility for the final decision-making based on their 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 selected traditional radial-basis function networks for predicting Mackey-Glass chaotic time series.

I. INTRODUCTION

Since the original emergence of Artificial Neural Networks in 1940’s, there has been an extensive qualitative and quantitative analysis on different classes of neural networks possessing various architectures and training algorithms. Without a proven guideline, the design of an optimal neural network for a given problem is often regarded as 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 enough 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 hidden 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 conflicting objectives. This has promoted research on how to identify an optimal and efficient neural network structure. AIC (Akaike Information Criterion) [1] and PMDL (Predictive Minimum Description Length) [2] 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 by each run, which does not offer the designers with alternative choices.

Over the past decade, evolutionary algorithms have been successfully applied to the design of network topologies and the choice of learning parameters [3]. They reported some encouraging results that are comparable with conventional neural network design approaches. However, the multiobjective trade-off characteristic of the neural network design has not been well studied and applied in the real world applications. In this paper, 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.

The remainder of this paper is organized as follows. Section II discusses the neural network design dilemma and the difficulty of finding a single optimal neural network. Section III reviews a Hierarchical Genetic Algorithm based Neural Network (HGA-NN) design approach and applies hierarchical genotype representation to a Radial-Basis Function (RBF) neural network design. Section IV introduces the proposed rank-density fitness assignment technique for multiobjective genetic algorithms and describes HRDGA parameters and design procedure. Section V presents a feasible application on the Mackey-Glass chaotic time series prediction using HRDGA evolved neural networks. A time series with a specific 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 VI provides some concluding remarks along with pertinent observations.

II. 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 [4], the ultimate goal of the
construction of a neural network with the input-output relation \( y = f_{sS}(x, \omega) \) is the minimization of the expectation of a cost function \( g_r(f_{sS}(X, \omega), Y) \) as
\[
E[g_r(f_{sS}(X, \omega), Y)] = \int \int g_r(f_{sS}(x, \omega), y)f_{sS}(x, y) dx dy
\]  
where \( f_{sS}(x, y) \) denotes the joint pdf that depends on the input vector \( x \) and the target output vector \( y \). Given a network structure \( N_S \), a family of input-output relations \( F_{sS} = \{ f_{sS}(x, \omega) \} \), parameterized by \( \omega \), consisting of all network functions that may be formed with different choices of the weights can be assigned. The structure \( N_S^* \) is said to be dominated by \( N_S^\prime \) if \( F_{sS} \subset F_{sS^\prime} \). In order to choose the optimal neural network, we need to find the determination of the network function \( f_{sS}^*(x) \) (i.e., the determination of the respective weights \( \omega^* \)) that gives the minimal cost value within the family \( F_{sS} \)
\[
f_{sS}^*(x) = f_{sS}(x, \omega^*) = \arg \min_{\omega} E[g_r(f_{sS}(X, \omega), Y)],
\]
and the determination of the network structure \( N_S^* \) that realizes the minimal cost value within a set of structures \( \{N_S\} \)
\[
N_S^* = \arg \min_{N_S \in \{N_S\}} E[g_r(f_{sS}^*(X, Y))].
\]  
Obviously, the solutions of this task need not result into a unique network. In [5], if several structures \( N_S^*, N_S^*, \ldots \) meet the criterion as shown in Equation (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 variation of the hidden neuron numbers. 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 \( f_{sS}^*(x) \) from \( F_{sS} \) by using a finite training data set is a difficult task, if not impossible [5]. 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 testing performances for different training and testing 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 candidates network.

### III. RBF NEURAL NETWORK DESIGN

In the literature of using genetic algorithms to assist neural networks design, several approaches have been proposed for evolving NN structure together with weights and biases [3,6-7]. Among all these methods, we incorporate a hierarchical genotype representation into an RBF neural network design.

Hierarchical Genetic Algorithm (HGA) was first proposed by Ke, et. al., [8] for fuzzy controller design using two layer genes to evolve membership. Based on this idea, Yen and Lu [7] designed an HGA Neural Network (HGA-NN). In the HGA-NN, a three-level HGA is used to evolve a Multi-layer Perceptron neural network. Each candidate chromosome corresponds to a neural network and the first, second and third level genes represent the network layer number, the neuron number in each layer and the parameter values of each neuron, respectively. By using this hierarchical genotype coding, the problem of “one phenotype mapping different genotypes” can be prevented [7].

In a similar spirit, HGA is tailored in this paper to evolve an RBF (Radial-Basis Function) neural network. A radial-basic function can be formed as
\[
f(x) = \sum_{i=1}^{m} \omega_i \exp(-\|x - c_i\|^2)
\]
where \( \omega_i \) denotes the center of the \( i \)th localized function, \( \omega_i \), is the weighting coefficient connecting the \( i \)th 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.

In HGA based RBF neural network design, genes in the genotype are classified into three categories: control genes, weight genes and center genes. The lengths of these three kinds of genes are the same. The value of each control gene (0 or 1) determines the activation status (off or on) of the corresponding weight gene and center gene. 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 1 shows the genotype and phenotype of HGA based RBF neural network.

![Figure 1: Genotype and Phenotype of HGA based RBF neural network](Image)

### IV. MULTIOBJECTIVE GENETIC ALGORITHMS

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

A. Multiobjective Genetic Algorithms (MOGAs)

Since the 1980’s, several Multiobjective Genetic Algorithms (MOGAs) have been proposed and applied in Multiobjective Optimization Problems (MOPs) [9]. These algorithms share the same purpose—searching for a uniformly distributed, near-optimal and near-complete family of non-dominated individuals, a so-called Pareto front [9], which describes the trade-off among contradict objectives as shown in Figure 2. For example, considering the NN design dilemma introduced in Section II, a neural network design problem can be regarded as a class of MOPs 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.

![Figure 2 Graphical illustration of the Pareto optimality](image)

B. Rank-density based fitness assignment MOGA

As MOGAs are designated to find a near-optimal and near complete set of Pareto solutions, the fitness assignment scheme is quite different from generic GAs, which are designed to search for a single optimal solution. In this paper, based on the hierarchical phenotype formulation, we propose a new rank-density based fitness assignment scheme in a multiobjective genetic algorithm to assist neural network design. Three essential steps were applied in this technique.

First, an Automatic Accumulated Ranking Strategy (AARS) is applied to calculate the Pareto rank value, which represents the dominated relationship among individuals. In AARS, assume at generation $t$, individual $y$ is dominated by $p^{(i)}$ individuals $y_i,\cdots,y_{p^{(i)}}$, whose rank values are already known as $r(y_i,t),\cdots,r(y_{p^{(i)}},t)$. Its rank value can be computed by

$$r(y,t) = 1 + \sum_{i=1}^{p^{(i)}} r(y_i,t).$$

Therefore, by AARS, individual’s density information is included in its rank value, thus the population diversity will be kept by penalizing those dominated individuals.

Second, to maintain the diversity of the obtained Pareto front, we adopt an adaptive cell density evaluation scheme as shown in Figure 3. The cell width in each objective dimension can be formed as

$$d_i = \frac{\max f_i(x) - \min f_i(x)}{K_i}, i = 1,\ldots,n,$$

where $d_i$ is the width of the cell in the $i$th dimension, $K_i$ denotes the number of cells designated for the $i$th dimension (i.e., in Figure 4, $K_1 = 12$ and $K_2 = 8$), and $X$ denotes the decision vector space. As the maximum and minimum fitness values in objective space will change with different generations, the cell size will vary from generation to generation to maintain the resolution of the density calculation. The density value of an individual is defined as the number of the individuals located in the same cell.

![Figure 3 Density map and density grid](image)

Third, 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. Meanwhile, since 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 when 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 4, assuming 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, thus 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.

![Illustration of the valid range and the forbidden region](image)

Finally, The simple elitism scheme [9] is also applied for bookkeeping the Pareto individuals obtained in each generation. These individuals are compared to achieve the final Pareto front after the evolution process has stopped.

C. HRDGA for NN design

To assist RBF network design, the proposed Hierarchical Rank Density based Genetic Algorithm (HRDGA) is applied to carry out the fitness evaluation and mating selection schemes. The HRDGA operators are designed as follows.

1) 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.

2) 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 a Gaussian(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.

3) 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. After crossover, the offspring replaces the low fitness parents and a new generation is formed. Mating is then iteratively processed.

4) Stopping criteria

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

V. TIME SERIES PREDICTION

Since the proposed HRDGA was 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 a Mackey-Glass chaotic time series:

\[
\frac{dx(t)}{dt} = \frac{a \times x(t - \rho) - b \times x(t)}{1 + x^2(t - \tau)}
\]

where \(\tau = 150\), \(a = 0.2\), \(b = 0.1\) and \(c = 10\). 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) [10], GRNN (Generalized Regression Neural Network) [11] and OLS (Orthogonal Least Square Error) [12] 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 size equals to one. Each of these networks will be trained by KNN and GRNN methods, and the stop criteria is the same with the one we applied in HRDGA. For the OLS method, the selection of the tolerance parameter \(\rho\) determines the trade-off between the performance and complexity of the network. A smaller \(\rho\) value will produce a neural network with more neuron numbers, whereas a larger \(\rho\) generally results in a network with less number of neurons. Therefore, by using different \(\rho\) 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 40 different \(\rho\) values, which are from 0.01 to 0.4 with the step size of 0.01. Optimal \(k\) value in KNN is determined according to reference [10]. The stop criteria for KNN, GRNN and OLS algorithms is either the epochs exceeds 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, and 750 – 999 seconds were used as the testing data sets (labeled #1, #2 and #3) to be predicted by four different approaches. Each
approach runs 30 times with different parameter initializations to obtain the average results. Figure 5(a)-(d) shows the average SSEs of training data set and three testing data sets by the resulting neural networks with different number of hidden neurons. Figure 6(a)-(d) shows the corresponding approximated Pareto fronts (i.e., non-dominated sets) by the selected four approaches. Table 1 shows the best training and testing performances and their corresponding numbers of hidden neurons over 30 runs.

From Figures 5 – 6, we can see, comparing to KNN and GRNN, HRDGA and OLS algorithms have much smaller training and testing errors for the same network structures. KNN trained networks produce the worst performances, because the RBF centers of the KNN algorithm are randomly selected, which make KNN to achieve 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 5, we can see that for all training algorithms, when the network complexity increases, the training error decreases. However, this phenomenon is only partially maintained for the relationship between the testing performances and the network complexity. Before the number of hidden neurons reaches a certain threshold, the testing error decreases as the network complexity increases. After that, the testing 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 best testing performance before overfitting occurs is called the optimal network and judged as the final single solution by conventional NN design algorithms [4]. However, from Figures 5 – 6 and Table 1, it is very difficult to find a single optimal network that can offer the best performances for all the testing data sets, since these data sets possess different traits. Therefore, instead of searching for a single optimal neural network, HRDGA can be a more reasonable and applicable option since it results in a near-complete set of near-optimal networks.

Moreover, from Figure 5, we can see that for all training algorithms, when the network complexity increases, the training error decreases. However, this phenomenon is only partially maintained for the relationship between the testing performances and the network complexity. Before the number of hidden neurons reaches a certain threshold, the testing error decreases as the network complexity increases. After that, the testing 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 best testing performance before overfitting occurs is called the optimal network and judged as the final single solution by conventional NN design algorithms [4]. However, from Figures 5 – 6 and Table 1, it is very difficult to find a single optimal network that can offer the best performances for all the testing data sets, since these data sets possess different traits. Therefore, instead of searching for a single optimal neural network, HRDGA can be a more reasonable and applicable option since it results in a near-complete set of near-optimal networks.

VI. CONCLUSION

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-quality training and testing 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).

REFERENCES


Table 1 - Structure and performance comparison among four algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best performance for Training set</th>
<th>Best performance for Testing set #1</th>
<th>Best performance for Testing set #2</th>
<th>Best performance for Testing set #3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SSE</td>
<td>Neuron number</td>
<td>SSE</td>
<td>Neuron number</td>
</tr>
<tr>
<td>KNN</td>
<td>2.8339</td>
<td>69</td>
<td>3.3693</td>
<td>42</td>
</tr>
<tr>
<td>GRNN</td>
<td>2.3382</td>
<td>68</td>
<td>2.7720</td>
<td>38</td>
</tr>
<tr>
<td>OLS</td>
<td>2.3329</td>
<td>60</td>
<td>2.4601</td>
<td>46</td>
</tr>
<tr>
<td>HRDGA</td>
<td>2.2901</td>
<td>74</td>
<td>2.4633</td>
<td>47</td>
</tr>
</tbody>
</table>

Figure 7 Relationship between \( \rho \) values and network complexity

From the simulation results, although KNN and GRNN provide worse training and testing 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 performances, the designers cannot manage the network structure directly. The trade-off characteristic between network performance and complexity totally depends on the value of tolerance parameter \( \rho \). Same \( \rho \) value means completely different trade-off features for different NN design problems. In addition, as shown in