Monograph:
Billings, S.A. and Zheng, G.L. (1994) Radial Basis Function Network Configuration Using Genetic Algorithms. Research Report. ACSE Research Report 521 . Department of Automatic Control and Systems Engineering

---

Radial Basis Function Network Configuration Using Genetic Algorithms

S. A. Billings and G. L. Zheng
Department of Automatic Control and Systems Engineering,
University of Sheffield, Mappin Street, Sheffield S1 4DU

Abstract — Most training algorithms for radial basis function (**RBF**) neural networks start with a predetermined network structure which is chosen either by using *a priori* knowledge or based on previous experience. The resulting network is often insufficient or unnecessarily complicated and an appropriate network structrue can only be obtained by trial and error. Training algorithms which incorporate structure selection mechanisms are usually based on local search methods and often suffer from a high probability of being trapped at a structural local minima. In the present study, genetic algorithms are proposed to automatically configure **RBF** networks. The network configuration is formed as a subset selection problem. The task is then to find an optimal subset of n_c terms from the N_t training data samples. Each network is coded as a variable length string with distinct integers and genetic operators are proposed to evolve a population of individuals. Criteria including single objective and multiobjective functions are proposed to evaluate the fitness of individual networks. Training based on practical data set is used to demonstrate the performance of the new algorithms.

Keywords — Radial Basis Function, Genetic Algorithms, Network Structure, System Identification, Pattern Recognition.

Research Report No. 521

June 1994

Radial Basis Function Network Configuration Using Genetic Algorithms

S. A. Billings and G. L. Zheng
Department of Automatic Control and Systems Engineering,
University of Sheffield, Mappin Street, Sheffield S1 4DU

April 1994

Abstract — Most training algorithms for radial basis function (**RBF**) neural networks start with a predetermined network structure which is chosen either by using *a priori* knowledge or based on previous experience. The resulting network is often insufficient or unnecessarily complicated and an appropriate network structrue can only be obtained by trial and error. Training algorithms which incorporate structure selection mechanisms are usually based on local search methods and often suffer from a high probability of being trapped at a structural local minima. In the present study, genetic algorithms are proposed to automatically configure **RBF** networks. The network configuration is formed as a subset selection problem. The task is then to find an optimal subset of n_c terms from the N_t training data samples. Each network is coded as a variable length string with distinct integers and genetic operators are proposed to evolve a population of individuals. Criteria including single objective and multiobjective functions are proposed to evaluate the fitness of individual networks. Training based on practical data set is used to demonstrate the performance of the new algorithms.

Keywords — Radial Basis Function, Genetic Algorithms, Network Structure, System Identification, Pattern Recognition.

1 Introduction

Radial basis function (**RBF**) neural networks have been studied by theorists and practitioners in many diverse disciplines in recent years. Theorists have studied the basic aspects of radial basis function approximation, including denseness, uniqueness of interplation and convergence rate [1], [2], [3]. It has been proved [1] [3] that a radial basis function network can approximate arbitraryly well any multivariate continuous function on a compact domain if a sufficient number of radial basis function units are given. Poggio *et al.* [4] developed regularization networks from approximation theory with radial basis function networks as a special case. A network with a finite basis was also developed as a natural approximation to the regularization network. Other extensions such as moving centres, weighted norm and

networks with different types of basis functions and multiple scales were also considered. These works provide a useful theoretical framework for investigating radial basis function networks and learning algorithms. A variety of approaches for training radial basis function networks have been developed. Most of them can be divided into two stages [5], [6], [7], [8], [9], [10]: *i).* learning the centres and widths in the hidden layer; *ii).* learning the connection weights from the hidden layer to the output layer. In these learning algorithms, the network structure or the number of hidden layer nodes are predetermined. An appropriate network structure can only be determined by trial and error. Learning algorithms which incorporate structure selection mechanisms were developed in references [11], [12], [13] and [14]. In reference [11], the network was trained using an orthogonal least squares (**OLS**) algorithm. Akaike's information criterion (**AIC**) was used to determine the number of hidden layer nodes and an error reduction ratio was used to select the centres. The algorithm provides a compromise between network performance and network complexity and automatically determines the number of hidden layer nodes. However, the algorithm is essentially a descent method in the sence that the hidden layer nodes are located in a way such that the approximation errors of the network are most effectively reduced at each step. Such an error reduction is clearly local and the resulting network may become trapped at a local minima. The learning algorithm developed in reference [12] treats the radial basis functions associated with the hidden layer nodes as approximately orthogonal to each other. The network starts with one hidden layer node and additional nodes are added to the network when they are necessary. The locations of the hidden layer nodes are optimized using an optimization package. This offers the opportunity to use advanced optimization methods to train the network. However, the termination criterion of the algorithm is not clear and may be problem dependent. Training algorithms presented in references [13] and [14] are based on supervised hierarchical clustering methods. In reference [13], the learning starts with one hidden layer node with larger width and creates additional nodes when they are desired. The associated widths and the locations are also changed. In reference [14], the learning begins with a larger number of nodes and merges them when possible. The associated widths and locations of the nodes are updated accordingly. These two algorithms are primarily developed for pattern recognition problems and offer alternative ways to determine network structures. But since the number of nodes and node locations are determined by clustering the pattern vectors, they may not be necessarily optimial for the network. This may be especially true for system identification problems. In the present study genetic algorithms are developed to optimize **RBF** network structures and to configure the network in terms of the number of hidden layer nodes and the centre locations. The algorithms evolve to optimize single objective as well as multiobjective functions and automatically determine appropriate network structures accordingly.

The layout of the paper is organized as follows. Section two briefly describes the radial basis function network and the network structure. Section three presents a canonical genetic algorithm and introduces the basic concepts of genetic algorithms. Genetic algorithms for **RBF** network configuration are described in section four together with details of genetic operators and objective functions. Experimental results are given in section five, which indicate that the genetic algorithms are capable of creating appropriate **RBF** networks for several objectives of interest. Finally, section six is devoted to conclusions and directions for future research.

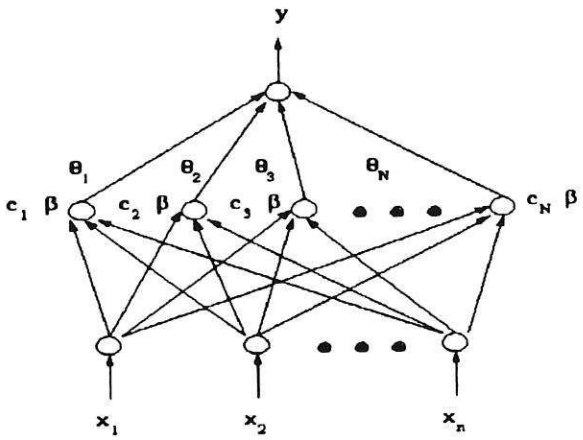


Figure 1: Radial Basis Function Network Architecture

2 Radial Basis Function Networks

A basic radial basis function (**RBF**) network may be depicted as shown in **Fig 1**. Without loss of generality, the number of outputs in the network will be assumed to be one, but the architecture can be readily extended to cope with multi-output problems. The architecture consists of an input layer, a hidden layer and an output layer. The input vector to the network is passed to the hidden layer nodes via unit connection weights. The hidden layer consists of a set of radial basis functions. Associated with each hidden layer node is a parameter vector $\mathbf{c_i}$ called a centre. The hidden layer node calculates the Euclidean distance between the centre and the network input vector and then passes the result to a radial basis function. All the radial basis functions in the hidden layer nodes are usually of the same type. Typical choices of the radial basis functions are

i). the thin-plate-spline function:

$$\phi(\mathbf{v}) = \mathbf{v}^2 \times \log(\mathbf{v}) \tag{1}$$

ii). the Gaussian function:

$$\phi(\mathbf{v}) = e^{-\left(\frac{\mathbf{v}^2}{\beta^2}\right)} \tag{2}$$

iii). the multiquadric function:

$$\phi(\mathbf{v}) = (\mathbf{v}^2 + \beta^2)^{\frac{1}{2}} \tag{3}$$

vi). the inverse multiquadric function:

$$\phi(\mathbf{v}) = \frac{1}{(\mathbf{v}^2 + \beta^2)^{\frac{1}{2}}} \tag{4}$$

where **v** is a non-negative number and is the distance from the input vector **x** to the radial basis function centre **c**, and β is the width of the radial basis functions. In radial basis function networks, the thin-plate-spline function has been used by Chen *et al.* [6], [11], and the Gaussian and multiquadric functions have been used by Moody [5], Broomhead [15] and Poggio [4]. In the present work, the thin-plate-spline function will be implemented in the network. However, other functions listed above can be readily included by using a constant β parameter. It is the authors' intention to develop algorithms to evolve the β parameter as well in further work.

The response of the output layer node may be considered as a map f: $\mathbf{R}^n \rightarrow \mathbf{R}$, that is

$$\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{N} \theta_i \phi(\| \mathbf{x} - \mathbf{c}_i \|) \tag{5}$$

where N is the number of training data and $\| \bullet \|$ denotes the Euclidean norm. $\mathbf{c_i}$ (i=1, 2, ..., N) is the i^{th} centre and is the i^{th} data sample in this particular network structure. $\mathbf{x}, \mathbf{c}_i \in \mathbf{R}^n$, $\theta_\mathbf{i}$ $(i = 1, 2, ..., N)$ are the weights associated with the i^{th} radial basis function centre. It may be seen that the training of the network is an interpolation problem and the solution may be obtained by solving a set of constrained linear equations. The complexity increases with the number of training data, which may make the implementation of the network above unrealistic. In practical applications, it is often desirable to use a network with a finite number of basis functions. A natural approximated solution would be

$$\mathbf{f}^\star(\mathbf{x}) = \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x} - \mathbf{c}_j \|) \tag{6}$$

where n_c is the number of radial basis function centres, $\mathbf{c}_j$ is the j^{th} centre which can be selected from the data samples. Given a set of data $(\mathbf{x}_i, \mathbf{y}_i)$, $(i = 1, 2, ..., N)$ $\mathbf{x}_i \in \mathbf{R}^n$, $y_i \in \mathbf{R}$, $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{in})^T$, the connection weights, centres and widths may be obtained by minimizing the following objective function

$$\mathbf{J}_1(\theta, \mathbf{c}) = \sum_{i=1}^{N} (y_i - \mathbf{f}^\star)^T (y_i - \mathbf{f}^\star) \tag{7}$$

where $\theta = (\theta_1, \theta_2, ..., \theta_{n_c})^T$, $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_{n_c})^T$. The above minimization problem may be solved using a nonlinear optimization or gradient decent algorithm. Note that the number of hidden layer nodes or the network structure is predetermined in this network. The structure may be selected using *a priori* knowledge. However, an appropriate network structure can only be determined by trial and error. It is therefore desirable to optimize the network structure, the centres and the connection weights simutaneously. The objective function could be chosen as

$$\mathbf{J}_2(n_c, \theta, \mathbf{c}) = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x}_i - \mathbf{c}_i \|) \right)^T \left(y_i - \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x}_i - \mathbf{c}_i \|) \right) \tag{8}$$

Note that the best structure which minimizes the objective function $\mathbf{J}_2$ has N hidden layer nodes and the centres tend to the data samples such that the network reverts to the one

given in formula (5). The resulting network can only interplate the particular data set and will fail to capture the underlying functional relation in the data, which will certainly lead to overfitting. To provide a compromise between network performance and network complexity, Akaike's information criterion (**AIC**) may be used and the objective function to be minimized can be amended to

$$\mathbf{J}_3(n_c, \theta, \mathbf{c}) = N \times log \left(\frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x}_i - \mathbf{c}_i \|) \right)^T \left(y_i - \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x}_i - \mathbf{c}_i \|) \right) \right) + 4 \times n_c \tag{9}$$

More discussion on objective functions will be given in section four. In the present work, it is assumed that the number of input nodes is known. Therefore, the minimization of the objective function would determine both the network structure and the parameters simutaneously. In the following sections it will be shown how this objective can be achieved using genetic algorithms.

3 Basics of Genetic Algorithms

Genetic algorithms are inspired by the evolution of populations. In a particular environment, individuals which better fit the environment will be able to survive and hand down their chromosomes to their descendants, while less fit individuals will become extinct. The aim of genetic algorithms is to use simple representations to encode complex structures and simple operations to improve these structures. Genetic algorithms therefore are characterised by their representations and operators. In the canonical genetic algorithm [16] an individual chromosome is represented by a binary string. The bits of each string are called genes and their varying values as alleles. A group of individual chromosomes are called a population. Basic genetic operators include reproduction, crossover and mutation. Genetic algorithms are typically implemented as follows.

i. Define an objective function which indicates the fitness of any potential solution.

ii. Select an appropriate representation or codification and choose genetic operators accordingly.

iii. Randomly choose an initial population of P individual chromosomes $b_i, (i = 1, 2, ..., P)$.

iv. Determine the fitness $f(b_i)$ $(i = 1, 2, ..., P)$ of each individual chromosome.

v. Assign a probability of reproduction p_i $(i = 1, 2, ..., P)$ to each individual.

vi. Generate a new population by selecting individuals from the current population according to assigned probabilities of reproduction p_i $(i = 1, 2, ..., P)$. The selected individuals generate offsprings via the use of genetic operators such as crossover and mutation with respective probabilities.

vii. If a stopping criterion is reached, stop. Otherwise repeat from step *iv*.

In the canonical genetic algorithm, crossover is applied to two chromosomes (parents). A random position is selected along the coded string. The parent chromosomes are split into two sections at this position. The two offspring chromosomes are formed by exchanging the second sections of the two strings. The probability of reproduction of each individual is $p_i = P(b_i \text{ is selected}) = f(b_i)/\sum_j^P f(b_i)$. The mutation is applied to each bit of the selected string with a given probability. The crossover constitutes the main search operator of genetic algorithms. It does not add any new genetic materials to the population. An important role of crossover however, was identifiedin [16] as recombining good "building blocks" in the population. This idea has been known as the building block hypothesis [16]. The mutation serves as a background operator to ensure that all possible alleles can enter the population. The particular characteristic of mutation is to enable the genetic algorithm to overcome local minima.

Genetic algorithms are often used to solve difficult optimization problems in many fields of study. They are especially capable of handling optimization problems in which the objective functions are discontinuous or non-differentiable, non-convex, multi-modal or noisy. Since the algorithms operate on a population instead of a single point in the search space, they can climb many peaks in parallel and therefore reduce the probability of finding local minima.

4 Genetic Algorithms for RBF Network Configuration

In section two, the **RBF** network configuration was formulated as a minimization problem with respect to the number of hidden layer nodes n_c, the centre locations $\mathbf{c}_i$ $(i = 1, 2, ..., n_c)$ and the connection weights θ. Although the objective functions given in formula (8) and (9) are continuous and differentiable with respect to the centre locations $\mathbf{c}_i$ $(i = 1, 2, ..., n_c)$ and the connection weights θ, they are discontinuous and non-differentialable with respect to the number of hidden layer nodes n_c. This presents difficulties for most conventional optimization methods which need derivative information of the objective function. Since genetic algorithms operate directly on the objective functions, this provides an alternative method for constructing **RBF** networks. In addition, the objective function is multi-modal and may have many local minima. Conventional optimization methods are likely to become trapped at these local minima. Moreover, when considering both the network structure and parameters, the search space is virtually infinite. This may become computionally prohibitive for many conventional optimization methods. Genetic algorithms however, can often find good solutions efficiently and quickly for such difficult problems. In this section, genetic algorithms will be proposed for configuring **RBF** neural networks.

Since the objective functions are quadratic in the connection weights when the network structure n_c and the centres $\mathbf{c}_i$ $(i = 1, 2, ..., n_c)$ are known, the connection weights can easily be computed using the least squares algorithm (**LS**). Instead of searching for the centres in real space $\mathbf{R}^n$, the centre locations are restricted to be the data samples $\mathbf{x}_i$ $(i = 1, 2, ..., N)$. This has the advantage that the search space can be significantly reduced. Former research results with the **RBF** networks [11] indicated that the resulting networks are sufficient to

capture the underlying dynamics within the data. Under these simplifications, the network configuration can be transformed into a subset selection problem in which n_c distinct terms are selected from the N data samples as the centres. In the present work, the data sample $\mathbf{x}_i$ is labelled with index i ($i = 1, 2, ..., N$). A **RBF** network can be coded as a chromosome with variable length and allele values ranging from 1 to N. The allele values of genes are referred to as identities. With this representation, the phenotypic value of the following chromosome is a **RBF** network with four hidden layer nodes and four centres located at

100	7	411	286

$\mathbf{c}_1 = \mathbf{x}_{100}$, $\mathbf{c}_2 = \mathbf{x}_7$, $\mathbf{c}_3 = \mathbf{x}_{411}$, $\mathbf{c}_4 = \mathbf{x}_{286}$ respectively. Since repeated usage of the same centre in a network does not improve the approximation capability of the network, each chromosome is restricted to be formed by distinct integers only. This representation for subset selection and the corresponding genetic operators were proposed by Lucasius *et al.* [17] and was used by Fonseca *et al.* [18] for non-linear term selection in NARMAX models. The subset selection problem was defined by Lucasias *et al.* as to select an optimal subset $\mathbf{s}$ of fixed size n_s from a source set $\mathbf{S}$ which contains N_s serially labelled terms. The set containing the remaining $N_s - n_s$ terms in the source set $\mathbf{S}$ is called the complementary subset of $\mathbf{s}$. In the present work, the sizes of the subsets are variable and the representations have variable lengths within a predetermined range. In the following, the genetic operators and the objective functions used in the algorithms will be described in detail.

Crossover — Any crossover operator should create offsprings which satisfy the constrains given above, *i.e.* the length of the string should be in the given range and the string should only contain distinct terms. Two crossover operators are proposed in this work. For presentation purposes they are referred to as the fixed length crossover and the variable length crossover respectively. The lengths of the parent chromosomes are preserved in the fixed length crossover, while they can be changed in the variable length crossover.

The fixed length crossover works as follows. The common terms in both parents are first searched and two binary template strings are created to mask the common terms in both parents. The binary bit in the template string is set to 1 if the corresponding term is a common term and 0 otherwise. The operator selects a random number of distinct terms from the end of the first parent and exchanges the same number of distinct terms with the second parent. The identities and positions of the common terms in both parents are preserved. For example, assume two parent strings are given as

$$P_1 = 1\ 8\ 10\ 3\ 7\ 4$$
$$P_2 = 6\ 1\ 9\ 5\ 4\ 2\ 3$$

To exchange three distinct terms between two parents, the procedure is

1. Create two template strings to mask the parents.

$$T_1 = 1\ 0\ 0\ 1\ 0\ 1$$
$$T_2 = 0\ 1\ 0\ 0\ 1\ 0\ 1$$

2. Exchange three distinct terms from the end of the string and keep the common terms unchanged.

$$C_1 = 1\ 9\ 5\ 3\ 2\ 4$$
$$C_2 = 6\ 1\ 8\ 10\ 4\ 7\ 3$$

It may be seen that the building block defined by the common terms in both parents are preserved in their offsprings. The operator does not change the lengths of the chromosomes.

To increase the diversity of string lengths, a variable length crossover may be used. The operator works as follows. The following procedure exchanges three terms from parent one with two terms from parent two.

1. Create two template strings to mask the parents.

$$T_1 = 1\ 0\ 0\ 1\ 0\ 1$$
$$T_2 = 0\ 1\ 0\ 0\ 1\ 0\ 1$$

2. Exchange three distinct terms from the end of parent one with two distinct terms from the end of parent two and keep the common terms unchanged.

$$C_1 = 1\ 5\ 3\ 2\ 4$$
$$C_2 = 6\ 1\ 9\ 8\ 10\ 4\ 7\ 3$$

Note that the length of child one is one term shorter than parent one and the length of child two is one term longer than parent two. The building block defined by the common terms in both parents are again transferred to their children.

Mutation — Mutation is usually used as a background operator in genetic algorithms. It helps to prevent premature loss of alleles and enables the algorithm to explore the areas containing potentially better fit chromosomes. In the present work, the trade mutation proposed by Lucasius *et al.* [17] is implemented in the algorithms. The operator exchanges with a given probability each term in a selected string with a randomly selected term in the corresponding complementary subset of the string.

Deletion and Addition — The canonical genetic algorithm works by recombining the best building blocks in the population. However, recombination may result in the loss of important building blocks through the premature loss of allele diversity. In a variable length representation, it may result in the premature loss of length diversity as well. The variable length crossover proposed above may alleviate this to a certain extend. But operators which work directly on the string length are still required. The operators deletion and addition are proposed for this purpose.

The deletion and addition provide background operators over the string lengths. A random change in the string length is imposed by either deleting a random number of terms from the string or adding a random number of terms to the string. The deletion and addition operators are applied to each selected string with equal probability. The deletion

operator removes a random number of terms from the string beginning from a randomly selected position. The addition operator always cascades a random number of terms to the end of the string. The newly added terms are randomly chosen from the complementary subset of the selected string.

Objective Functions — To provide a compromise between network performance and network complexity, an appropriate objective function would be Akaike's information criterion (**AIC**). In network training, the data samples are usually divided into two sets, the training set and the test set. The objective is then to minimize **AIC** on the training set and the objective function becomes

$$\mathbf{J}_{3t}(n_c, \theta, \mathbf{c}) = N_t \times log \left(\frac{1}{N_t} \sum_{i=1}^{N_t} \left(y_i - \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x}_i - \mathbf{c}_i \|) \right)^T \left(y_i - \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x}_i - \mathbf{c}_i \|) \right) \right) + 4 \times n_c \tag{10}$$

where N_t is the number of data samples in the training set, y_i, $\mathbf{x}_i$ are the output and input data samples in the training set. The number of centres n_c is initially chosen between a lower boundary n_{cmin} and a upper boundary n_{cmax}. These boundaries can then be changed during evolution if the lengths of all individuals in the population are n_{cmin} or n_{cmax}.

The above objective function is minimized on the training set only. This intrinsicaly involves the risk of overfitting on the training set such that the resulting network has smaller training errors but larger generalization errors. This may be appreciated from the experimental results given later. To alleviate this, it may be advantageous to minimize **AIC** on both the training set and the test set simultaneously. The network configuration may then be formed as a multiobjective minimization problem. The objective function becomes a vector valued quantity (J_{3t} J_{3v}) where J_{3v} is defined as

$$\mathbf{J}_{3v}(n_c, \theta, \mathbf{c}) = N_v \times log \left(\frac{1}{N_v} \sum_{i=1}^{N_v} \left(y_i - \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x}_i - \mathbf{c}_i \|) \right)^T \left(y_i - \sum_{j=1}^{n_c} \theta_j \phi(\| \mathbf{x}_i - \mathbf{c}_i \|) \right) \right) + 4 \times n_c \tag{11}$$

Note that the centres $\mathbf{c}_i$ ($i = 1, 2, ..., n_c$) are selected from the training set, N_v is the number of data samples in the test set and y_i, $\mathbf{x}_i$ are output and data samples in the test set.

Scaling and Ranking — Scaling is often used to prevent a super individual dominating the population in the early stage of evolution and to increase the resolution of selection in the later stage of evolution when many individuals in the population have high fitness values. Note that the computation of **AIC** is equivalent to applying a non-linear scaling to the **MSE** value. At the later stage of evolution, the **AIC** computation can increase the resolution of selection because it stretchs the range of the **MSE** values. However, at the

early stage of evolution the **AIC** computation will tend to emphasis the super individual in the population. In addition, the **AIC** value may be negative. To compute the fitness of the individual it is necessary to transfer the **AIC** values to positive numbers. A linear scaling proposed in reference [16] is implemented in the algorithm. This is given as

$$f(b_i) = a \times AIC_i + b \tag{12}$$

where a is a negative scaling factor, b is a constant offset and AIC_i is the AIC value of individual i. The coefficients a and b are chosen to ensure that average individuals receive one offspring on average and the best individual receives a specified multiple number of offsprings.

As noted by Fogel [19] adding a large constant value to the objective function can eliminate the selection and lead to a purely random walk. To avoid this the selection may be based on ranking the individuals by objective function values.

For the multiobjective optimization, a relation called partially-less-than is applied to compare two vector valued quantities. If vector **a** is partially-less-than vector **b**, it is symbolically written as $\mathbf{a} <_p \mathbf{b}$ and is defined as [16]

$$(\mathbf{a} <_p \mathbf{b}) \Leftrightarrow (\forall i)(a_i \leq b_i) \wedge (\exists i)(a_i < b_i) \tag{13}$$

Under this condition, it is said that **b** is dominated by **a** or inferior to **a**. If a vector is not dominated by any other, it is said that the vector is nondominated or noninferior. If a vector set is not dominated by others, it is called a Pareto optimal (P-optimal) set. For multiobjective optimization, the objective is to search the Pareto optimal set. In the present work, the population is ranked on the basis of nondomination. At each generation, all nondominated individuals in the population are identified and assigned rank one. These individuals are flagged and removed from contention. The next set of nondominated individuals is then identified and assigned rank two. This rank procedure continues until the entire population is ranked. The selection probabilities of individuals are then assigned according to rank. Note that the rank process intrinsicaly avoids the possibility of a random walk.

The Algorithms — The genetic algorithms to evolve networks for single objective and multiobjective functions are presented in the following.

i. Randomly choose an initial population of P individual chromosomes b_i $(i = 1, 2, ..., P)$. Each chromosome defines a network and the associated centre locations.

ii. Decode each chromosome. Compute the connection weights from the hidden layer nodes to the output layer nodes. Compute the **AIC** value for each chromosome b_i $(i = 1, 2, ..., P)$. Set the number of generations N_g for evolution. Set counter g = 0.

iii. Determine the fitness $f(b_i)$ $(i = 1, 2, ..., P)$ of each individual chromosome.

For the single objective algorithm, scale the **AIC** value of each individual on the training set according to formula (12).

For the multiobjective algorithm, rank each individual in the population as described above.

iv. Assign a probability of reproduction p_i $(i = 1, 2, ..., P)$ to each individual according to its fitness (or rank).

v. Set counter k = 1, apply genetic operators as defined above to create offsprings.

   (a) Select two individuals according to their reproduction probabilities.

   (b) Apply crossover with given probability to the two parent strings, create two offsprings.

   (c) Apply mutation with given probability to every bit of the offsprings.

   (d) Apply deletion and addition with given probability to the offsprings.

   (e) Decode the two child chromosomes. Compute the connection weights from the hidden layer nodes to the output layer nodes. Compute the **AIC** value (or values) of each child.

   (f) Compare the two children with their parents. The two best chromosomes are kept for evolution. For the single objective, the two chromosomes with smaller **AIC** values are kept. For the multiobjective, the two non-infereor chromosomes are kept for evolution.

   (g) Set k = k + 2, if k > P, goto **step** *vi*. Otherwise, return to **step a**

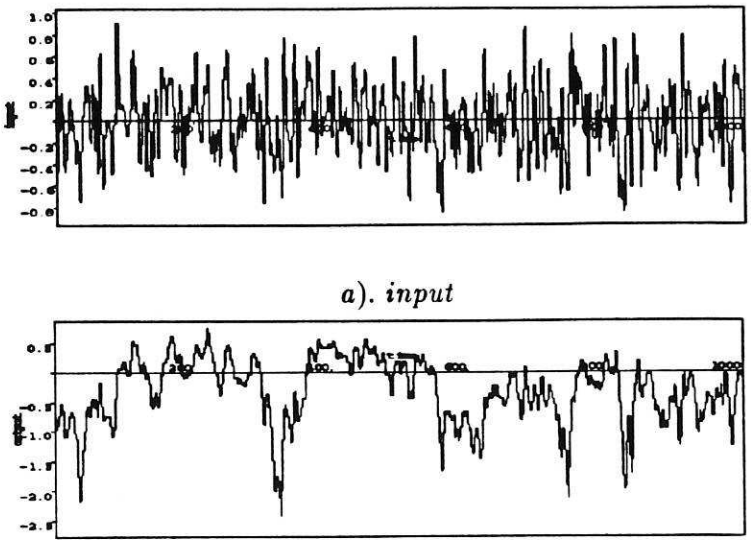
vi. Set g = g + 1, if g > N_g, stop. Otherwise, return to **step** *iii*.

5 Experimental Results

The genetic algorithms proposed above are coded for multi-output systems but a single output example will be used to demonstrate the performance of the algorithms in this paper. The example considered is a liquid level system. The system consists of a DC water pump feeding a conical flask which in turn feeds a square tank. The system input is the voltage to the pump motor and the system output is the water level in the conical flask. 1000 data samples generated in an experiment are shown in **Fig** 2. The first 500 data samples were used for training and the last 500 data samples were used for testing. The input vector to the network was chosen based on previous studies [6] [20] as

$$(y(t-1)\ y(t-2)\ y(t-3)\ u(t-1)\ u(t-2)\ u(t-3)\ u(t-4)\ u(t-5))$$

The emphasis here is on the capability of the genetic algorithm to find appropriate networks for the given objective functions rather than a "correct" network for the dynamic system. The later is a more complicated issue and will be addressed in future publications. Both the fixed length and variable length crossover were used in the expeximents. The fixed length crossover showed a higher probability of producing a premature loss of length diversity and only the results with variable length crossover will therefore be presented. In the experiments, the population size was chosen as 60. The crossover probability was 0.5 and the mutation probability was 0.02. These strategic parameters may not be optimal for the existing problem and further investigation into the selection of the strategic parameters of the algorithms will be conducted. The initial minimum string length was 15 and the maximum string length was 60. Since the training set contains 500 distinct terms, the search space therefore contains $\sum_{i=15}^{60} C_i^{500} = 3.0133 \times 10^{78}$ different networks.



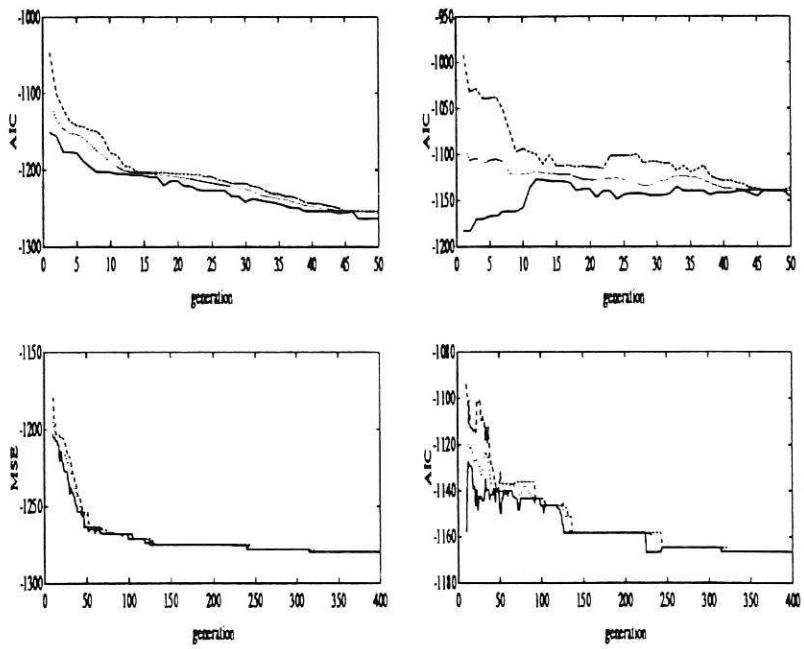
a). input

b). output

Figure 2: Input and output of a nonlinear liquid level system

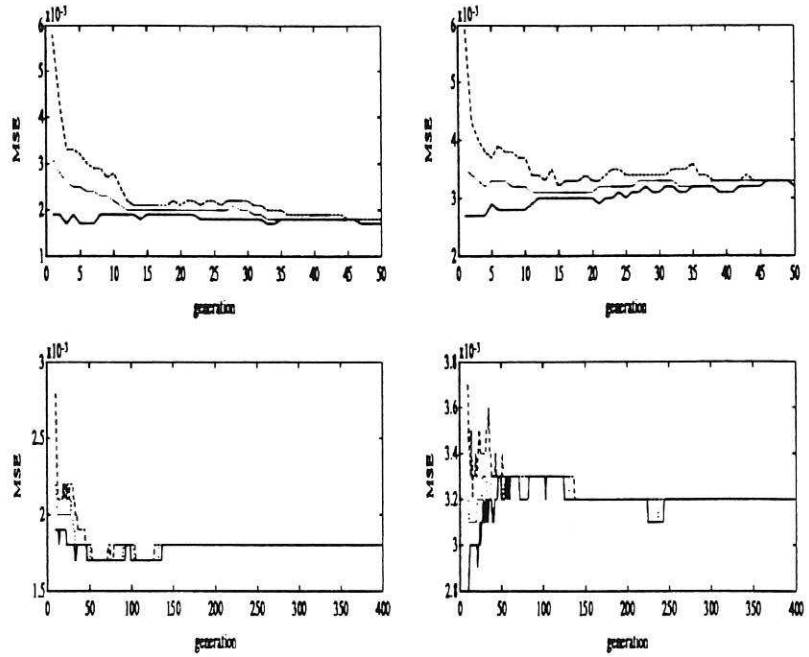
5.1 Single objective with deletion and addition

In this experiment, the AIC values on the training set were used to evaluate the fitness of individual networks. The probability of deletion and addition was 0.04. The evolution of AIC and MSE on both the training set and the test set are shown in **Fig** 3 and **Fig** 4 respectively. Although the objective function provides a compromise between network performance and network complexity and therefore results in an appropriate network for the training set, the resulting network can still have higher generalization error on the test set. Note that the lowest AIC value on the training set of the population was considerably improved through evolution but the AIC value on the test set at the 400^{th} generation is worse than that of the initial population. This may be further appreciated from the MSE values of the best network at the 400^{th} generation. The MSE values on the training set and the test set are 0.001757 and 0.003154. The evolution of the minimum and maximum string lengths of the population are shown in **Fig** 5. The algorithm automatically searchs for the appropriate network size according to the given objective. The best network at the 400^{th} generation has 21 centres. Compared with the network obtained in [20] the complexity of the genetically configured network is significantly reduced. In reference [20], a network with 41 centres achieved a training error of 0.002333 and a generalization error of 0.003051 respectively. This is probably because the genetic algorithm developed here has a lower probability of becoming trapped at a local minima. Finally, the output of the best network and the residuals are plotted in **Fig** 6.



a). AIC on training set *b). AIC on test set*

Figure 3: AIC on training set and test set (single objective with deletion and addition). top: generation 1–50, bottom: generation 10–400. '-' minimum, '...' average, '- -' maximum



a). MSE on training set *b). MSE on test set*

Figure 4: MSE on training set and test set (single objective with deletion and addition). top: generation 1–50, bottom: generation 10–400. '-' minimum, '...' average, '- -' maximum

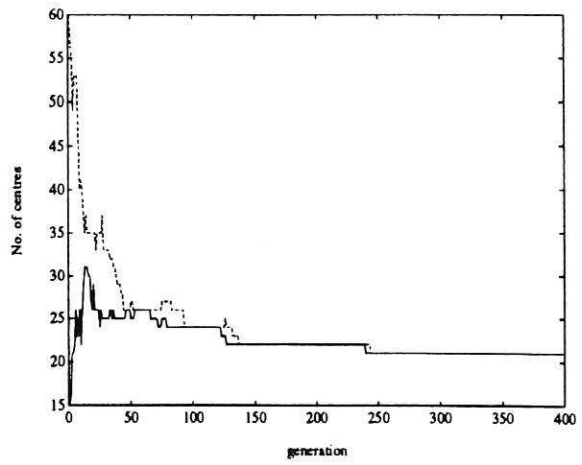


Figure 5: Maximum and minimum number of centres (single objective with deletion and addition). '-' minimum, '- -' maximum

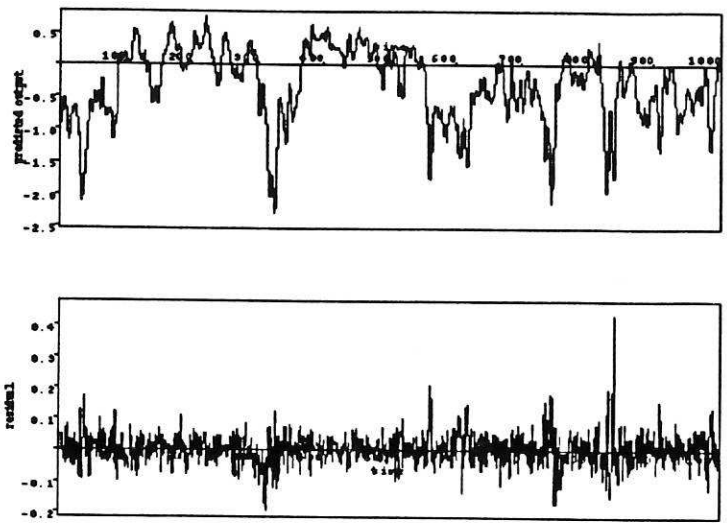


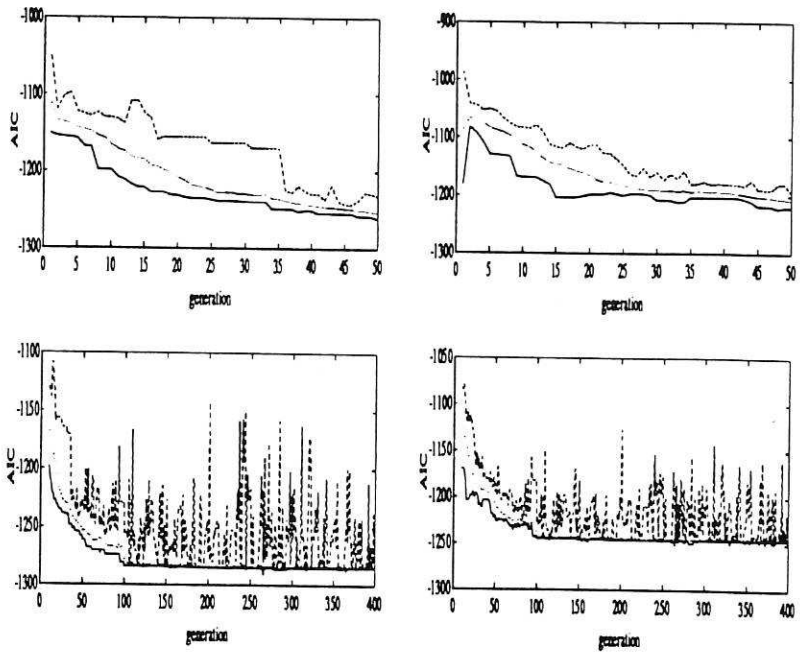
Figure 6: One step ahead predictions (top) and residuals (bottom) of the best network (single objective with deletion and addition)

5.2 Multiobjective without deletion and addition

In the previous experiment, it was shown that the network obtained by minimizing the objective function on the training set alone can have a lower training error but a higher generalization error. A possible solution would be to evolve the population to minimize the objective function on both the training set and the test set. The multiobjective function given in the previous section will be used to rank the individuals in this experiment. The centres are selected from the training set and the population evolve to minimize the objective functions in the sence of Pareto optimality. In this experiment, the deletion and addition operators are not applied to the individuals. The evolution of AIC and MSE on both the training set and the test set are shown in **Fig** 7 and **Fig** 8 respectively. Note that the AIC on both the training set and the test set are improved. The string lengths converged from a larger range (15-60) to a smaller range. One of the networks of the final generation has 14 centres and the MSE values on the training set and the test set are 0.001952 and 0.002490 respectively. The generalization error is smaller than that achieved in the single objective algorithm. The lower limit of the string length has been changed from 15 to 13. This indicates that the algorithm is not restricted by the initial limits of the string length. The predictions and the residuals of the network are shown in **Fig** 10.

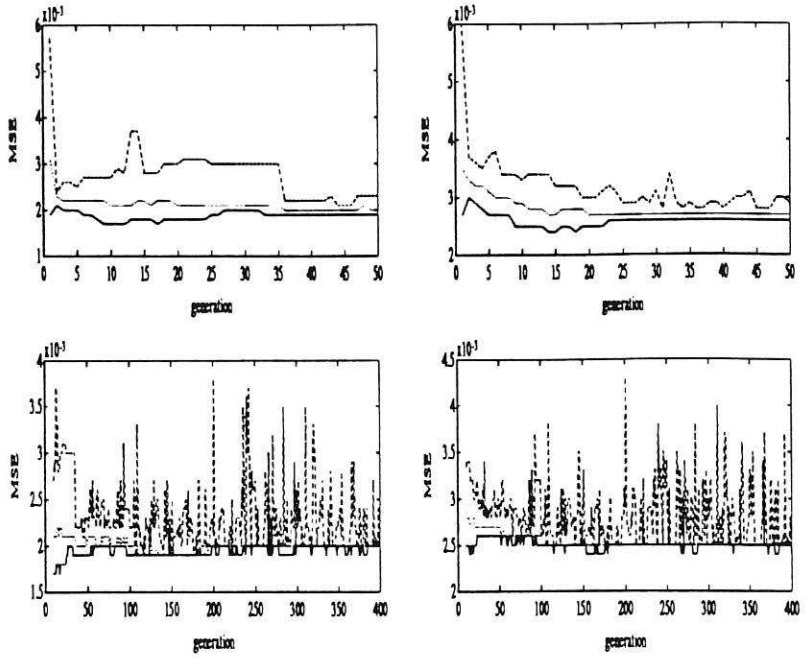
5.3 Multiobjective with deletion and addition

As mentioned in the previous section, in a variable length representation the premature loss of allele diversity may result in the premature loss of string length diversity. It may be correct to say that the premature loss of string length diversity can accelerate the prema-



a). AIC on training set *b). AIC on test set*

Figure 7: AIC on training set and test set (multiobjective without deletion and addition). top: generation 1–50, bottom: generation 10–400. '-' minimum, '...' average, '- -' maximum



a). MSE on training set *b). MSE on test set*

Figure 8: MSE on training set and test set (multiobjective without deletion and addition). top: generation 1–50, bottom: generation 10–400. '-' minimum, '...' average, '- -' maximum

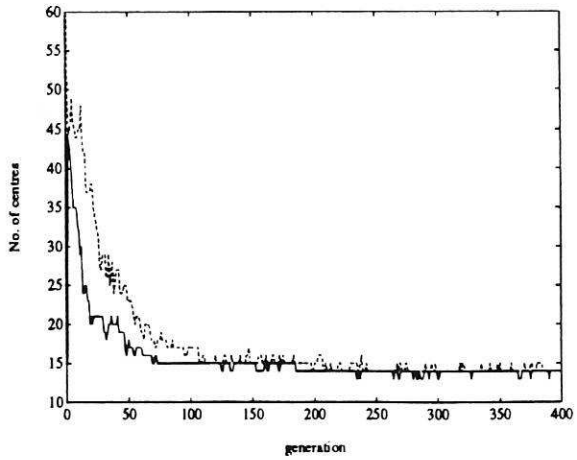


Figure 9: Maximum and minimum number of centres (multiobjective without deletion and addition). '-' minimum, '- -' maximum

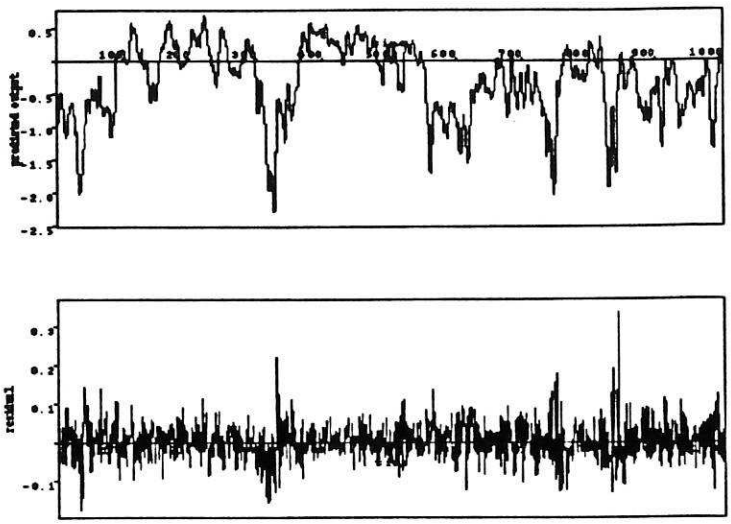
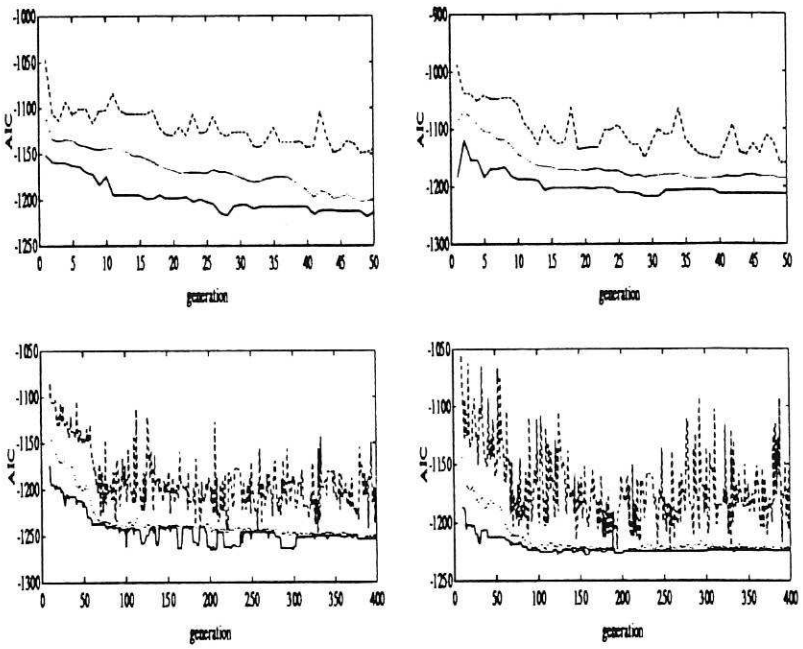


Figure 10: One step ahead predictions (top) and residuals (bottom) of the best network (multiobjective without deletion and addition)

ture loss of allele diversity. The deletion and addition operators are proposed to increase the string length diversity. The performance of these operators are demonstrated in this experiment. The strategic parameters of the genetic algorithm will be set to be the same as those in the previous experiment. The deletion and addition operators were applied to the individuals with a probability of 0.2. The evolution of AIC and MSE on both the training set and the test set are shown in **Fig** 11 and **Fig** 12 respectively. The evolution of the maximum and minimum string lengths are plotted in **Fig** 13. Compared with **Fig** 9, it may be seen that the length diversity of the population has improved significantly. The string lengths of the population nearly converged around the 100^{th} generation in the previous experiment (**Fig** 9), while the population maintained a richer length diversity throughout the evolution process in **Fig** 13. The predictions and residuals of one of the networks at the 400^{th} generation are shown in **Fig** 14. The network has 20 centres, a training error of 0.002029 and a generalization error of 0.002444. The network is more complex and the performance evaluated by the errors is slightly worse than in the previous experiment but this may be due to the slower convergence rate of the algorithm.

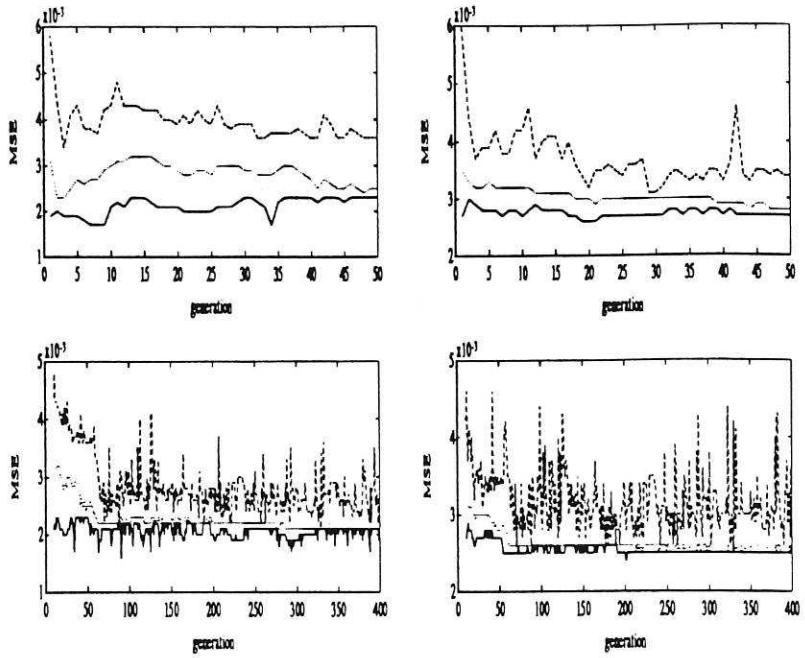
6 Conclusions

In this paper, genetic algorithms have been proposed to configure radial basis function networks. It has been demonstrated that the proposed algorithms can automatically determine appropriate network structures and network parameters according to given objective functions. Since the genetic algorithm can escape from local minima, it has a lower prob-



a). AIC on training set *b). AIC on test set*

Figure 11: AIC on training set and test set (multiobjective with deletion and addition). top: generation 1–50, bottom: generation 10–400. '-' minimum, '...' average, '- -' maximum



a). MSE on training set *b). MSE on test set*

Figure 12: MSE on training set and test set (multiobjective with deletion and addition). top: generation 1–50, bottom: generation 10–400. '-' minimum, '...' average, '- -' maximum

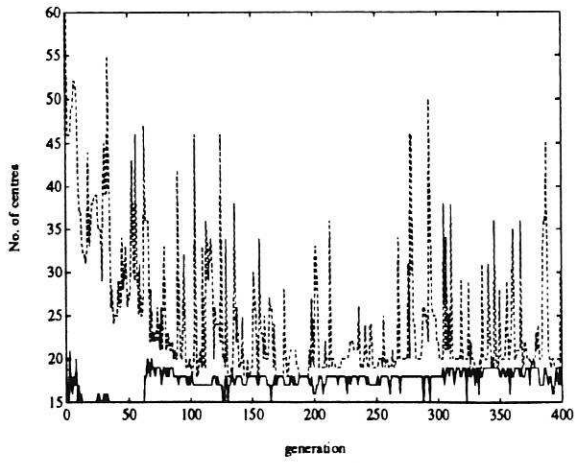


Figure 13: Maximum and minimum number of centres (multiobjective with deletion and addition). '-' minimum, '- -' maximum

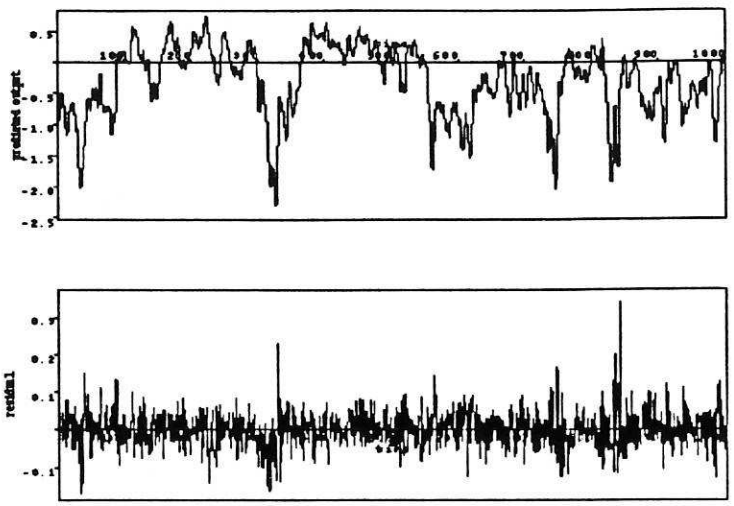


Figure 14: One step ahead predictions (top) and residuals (bottom) of the best network (multiobjective with deletion and addition)

ability of becoming trapped at structural local minima. The network complexity can be significantly reduced compared with that achieved by alternative algorithms. In addition, a group of networks can be obtained if the algorithm maintains a reasonably rich diversity of string length and allele value.

Both single objective and multiobjective functions have been proposed to evaluate network fitness. The **AIC** value provides an appropriate compromise between network complexity and network performance. However, when applied to the training set alone, the objective can result in a network with lower training error but higher generalization error. One way to avoid this is to evaluate the network on both the training set and the test set simultaneously. The generalization performance of the resulting network can be significantly improved.

The selection of strategic parameters has been a long standing problem in genetic algorithm research and has attracted attention from many researchers. It is worth noting however that strategic parameters used in the experiments may not be optimal for the given problem and further work is needed to resolve these issues.

Acknowledgements

The authors gratefully acknowledge that this work was supported by the EEC under Brite/Euram contract BREU-G91-0526 (RSJE) with the partners Professor G Tomlinson and Dr K Worden (University of Manchester), and Professor F Brancaleoni and Dr C Valente (University of Pescara, Italy).

References

[1] W. A. Light. Some Aspects of Radial Basis Function Approximation, Approximation Theory, Spline Functions and Applications, Vol. 356, 1992, pp 163 - 190.

[2] C. A. Micchelli. Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions, Constructive Approximation, 1988, Part 2, pp 11 - 22.

[3] M. J. D. Powell. Radial basis functions in 1990. In "Advances in Numerical Analysis", Vol. II, Oxford University Press, 1992, pp 105 - 210.

[4] T. Poggio, F. Girosi. Network for Approximation and Learning, Proceddings of IEEE, Vol. 78, No. 9, September 1990, 1481 - 1497.

[5] J. Moody, C. Darken. Fast learning in networks of locally-tunned processing units, Neural Computation, 1989, 1, 281 - 294.

[6] S. Chen, S. A. Billings & P. W. Grant. Recursive hybrid algorithm for non-linear system identification using radial basis function network, INT. J. Control, 1992, Vol. 55, No. 5, 1051-1070.

[7] S. N. Kaviori & V. Venkata-Subramanian. . Using Fuzzy Clustering with Ellipsoidal Units in Neural Networks for Robust Fault Classification, Computers Chem. Engng., Vol. 17, No. 8, 1993.

[8] Lei XU, A. Krzyzak & E. Oja. Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection, IEEE Trans. on Neural Networks, Vol. 4, No. 4, 1993, pp636-649.

[9] M. Vogt. Combination of Radial Basis Function Neural Networks with Optimaized Vector Quantization, Proceedings of the IEEE International conference on Neural Networks, Vol. 3, 1993, pp 1841-1846.

[10] G. L. Zheng and S. A Billings. Radial Basis Function Network Training Using a Fuzzy Clustering Scheme, Research Report No. 505, Department of Automatic Control and Systems Engineering, University of Sheffield, March 1994.

[11] S. Chen, S. A. Billings, C. F. N. Cowan & P. W. Grant. Practical identification of NARMAX models using radial basis functions, INT. J. Control, 1990, Vol. 52, No. 6, 1327-1350.

[12] T. Holcomb, M. Morari. Local Training for Radial Basis Function Networks: Towards Solving the Hidden Unit Problem, Proc. Americaan Control Conference, 1991, 2331 - 2336.

[13] Sukhan Lee and M. Kil. Rhee. A Gaussian Potential Function Network With Hierarchically Self-Organizing Learning, Neural Networks, 1991, Vol. 4, 207-224.

[14] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris and D. M. Hummels. On the Training of Radial Basis Function Classifiers, Neural Networks, 1992, Vol. 5, 595-603.

[15] D. S. Broomhead, D. Lowe. Multivariable Functional Interpolation and Adaptive Networks, Complex Systems, 1988, 2, 321 - 355.

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

[17] C. B. Lucasius and G. Kateman. Towards Solving Subset Selection Problems with the Aid of the Genetic Algorithm, Parallel Problem Solving from Nature, 2(R. Manner and B. Manderick (Editors)), Elsevier Science Publishers B. V.,1992.

[18] C. M. Fonseca, E. M. Mendes, P. J. Fleming and S. A. Billings, Non-linear Model Term Selection with Genetic Algorithms, Research Report, Department of Automatic Control and Systems Engineering, 1993.

[19] David B. Forgel. An Introduction to Simulated Evolutionary Optimization, IEEE Trans. on Neural Networks, Vol. 5, No. 1, 1994, 3-13.

[20] S. Chen, Radial Basis Functions for Signal Prediction and System Modelling, Research Report, Department of Electrical Engineering, University of Edinburgh, 1993.

List of Figures