SAWTOOTH GENETIC ALGORITHM AND ITS APPLICATION IN HAMMERSTEIN MODEL IDENTIFICATION AND RBFN BASED STOCK MARKET FORECASTING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BECHELOR OF TECHNOLOGY

IN

ELECTRONICS AND INSTRUMENTATION ENGINEERING

By

SIDDHARTHA SURUJ BORKOTOKY
Roll No : 10407004

ASHISH MALHOTRA
Roll No : 10407011

Department of Electronics & Communication Engineering
National Institute of Technology, Rourkela
Rourkela, Orissa–769008

2008

SAWTOOTH GENETIC ALGORITHM AND ITS APPLICATION IN HAMMERSTEIN MODEL IDENTIFICATION AND RBFN BASED STOCK MARKET FORECASTING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS & INSTRUMENTATION ENGINEERING

By

SIDDHARTHA SURUJ BORKOTOKY
Roll No: 10407004

ASHISH MALHOTRA
Roll No: 10407011

Under the Guidance of
PROF. GANAPATI PANDA

Department of Electronics & Communication Engineering
National Institute of Technology, Rourkela
Rourkela, Orissa – 769008
2008

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled **“SAWTOOTH GENETIC ALGORITHM AND ITS APPLICATION IN HAMMERSTEIN MODEL IDENTIFICATION AND RBFN BASED STOCK MARKET FORECASTING”** submitted by Siddhartha Suruj Borkotoky and Ashish Malhotra in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Electronics and Instrumentation Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to
any other University/ Institute for the award of any Degree or Diploma.

Dr. G. Panda
Professor and Head,
Department of E.C.E
National Institute of Technology
Rourkela – 769008

ACKNOWLEDGEMENT

I take this opportunity as a privilege to thank all individuals without whose support and guidance we could not have completed our project in this stipulated period of time. First and foremost I would like to express my deepest gratitude to my Project Supervisor Prof. G. Panda, Head of the Department, Department of Electronics and Communication Engineering, for his invaluable support, guidance, motivation and encouragement through out the period this work was carried out. His readiness for consultation at all times, his educative comments and inputs, his concern and assistance even with practical things have been extremely helpful.

I am grateful to Ms. Babita Majhi, Mr. Sitanshu Shekhar Sahoo and Mr. Mohammad Naseem for their valued suggestions and inputs during the course of the project work.

I would also like to thank all professors and lecturers for their generous help in various ways for the completion of this thesis. I also extend my thanks to my fellow students for their friendly co-operation.

Siddhartha S. Borkotoky
Roll. No. 10407004
Department of ECE
NIT Rourkela

Ashish Malhotra
Roll. No. 10407011
Department of ECE
NIT Rourkela

CONTENTS

CONTENTS

ABSTRACT

This Project work has been divided into three parts. In the first part, we deal with the sawtooth genetic algorithm. In the second part, we use this algorithm for optimization of Hammerstein model. In the third part we implemented a stock market forecasting model based on radial basis function network tuned by sawtooth genetic algorithm.

List of tables

List of figures

Chapter 1

SAWTOOTH GENETIC ALGORITHM

1.1 Introduction to genetic algorithm

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. Genetic algorithm can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non differentiable, stochastic, or highly nonlinear.

Table1.1: Comparison of Classical search algorithms and genetic algorithm

Classical Algorithm	**Genetic Algorithm**
Generates a single point at each iteration. The sequence of points approaches an optimal solution.	Generates a population of points at each iteration. The best point in the population approaches an optimal solution.
Selects the next point in sequence by deterministic computation .	Selects the next population by computation which uses random number generations.

1.2 Different operators used in genetic algorithm

1.2.1 Fitness Function and Fitness Value

The fitness value of an individual is the value of the fitness function for that individual.

It represents a measure of the proximity of the individual to the optimum solution.

1.2.2 Parents and Children

To create the next generation, the genetic algorithm selects certain individuals in the current population, called parents, and uses them to create individuals in the next generation, called children. Typically, the algorithm is more likely to select parents that have better fitness values.

1.2.3 Reproduction:

Reproduction is usually the first operator applied on population. Chromosomes are selected from the population to be parents to crossover and produce offspring. According to Darwin's evolution theory of survival of the fittest, the best ones should survive and create new offspring. That is why reproduction is also known as selection operator. There exist a number of reproduction operators in GA literature but the essential idea in all of them is the above average strings are picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner. The various methods of selecting chromosomes for parents to crossover are

1. Roulette-wheel selection
2. Boltzmann selection
3. Tournament selection
4. Rank selection
5. Steady state selection

Roulette wheel selection:

The commonly used reproduction operator is the proportionate reproductive operator where a string is selected from the mating pool with a probability proportional to the fitness. Thus ith string in the population is selected with a probability proportional to Fi where F_i is the fitness value for that string. Since the population size is usually kept fixed in a simple GA, the sum of the probabilities of each string being selected for the mating pool must be one. The probability of the ith selected string is

$$P_i = F_i / \sum F_j \quad , \quad j=1, 2,\ldots, n$$

where n is the population size.

Boltzmann selection:

Simulated annealing is a method of functional minimization or maximization. This method simulates the process of slow cooling of molten metal to achieve the minimum function value in a minimization problem. The cooling phenomenon is simulated by controlling a temperature like parameter introduced with the concept of Boltzmann probability distribution so that a system in thermal equilibrium at a temperature T has its energy distributed probabilistically according to the equation

$$P(E) = \exp(-E/kT)$$

where k is the Boltzmann constant. This expression suggests that a system at a high temperature has almost uniform probability of being at any energy state, but at a low temperature it has a small probability of being at high energy state. Therefore, by controlling the temp T and assuming that the search process follows Boltzmann probability distribution, the convergence of the algorithm is controlled.

Tournament selection:

GA uses a strategy to select the individuals from population and insert them into a mating pool. Individuals from the mating pool are used to generate new offspring, which are the basis for the next generation. As the individuals in the mating pool are the ones whose genes will be inherited by the next generation, it is desirable that the mating pool consists of good individuals. A selection strategy in GA is simply a process that favours the selection of better individuals in the population for the mating pool.

There are two important issues in the evolution process of genetic search, population diversity and selective pressure.

a. Population diversity means that the genes from the already discovered good individuals are exploited while promising the new areas of the search space continue to be explored.
b. Selective pressure is the degree to which the better individuals are favoured. A higher selective pressure ensures fast convergence towards the optimum solution, but if the selective pressure is too high there is a increasing chance of GA prematurely

converging to a local optimum solution. If the selective pressure is too low the convergence rate will be slow.

Rank selection:

Rank selection first ranks the population and taken every chromosome, receives fitness from the ranking. The worst chromosome will have fitness 1, the next 2,….and the best will have fitness N, where N is the number of chromosomes in the population. This method can lead to slow convergence because the best chromosome does not differ so much from the other.

Steady state selection:

The main idea of this selection is that bigger part of chromosome should survive to next generation. In every generation few chromosomes are selected for creating new off springs. Then some chromosomes are removed and new offspring is placed in that place. The rest of population survives a new generation.

Elitism:

In this method, first the best chromosomes are copied to new population. Elitism can very rapidly increase the performance of GA because it prevents loosing the best found solutions. If F fitness functions are positive and for minimization problem the fitness of any i^{th} individual must be subtracted from a large constant so that all fitness values are non negative and individuals get fitness values according to their actual merit. Now the new expression for fitness becomes

$$\Phi_i = (F_{max} - F_{min}) - F_i(X)$$

for minimization problem.

1.2.4 Crossover

Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability.

One Point Crossover

A crossover operator that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring.
Consider the following 2 parents which have been selected for crossover. The “|” symbol indicates the randomly chosen crossover point.

Parent 1: 11001|010
Parent 2: 00100|111

After interchanging the parent chromosomes at the crossover point, the following offspring are produced:

Offspring1: 11001|111
Offspring2: 00100|010

Two Point Crossover

A crossover operator that randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring.
Consider the following 2 parents which have been selected for crossover. The “|” symbols indicate the randomly chosen crossover points.

Parent 1: 110|010|10
Parent 2: 001|001|11

After interchanging the parent chromosomes between the crossover points, the following offspring are produced:

Offspring1: 110|001|10
Offspring2: 001|010|11

Uniform Crossover

A crossover operator that decides (with some probability – known as the mixing ratio) which parent will contribute each of the gene values in the offspring chromosomes. This allows the parent chromosomes to be mixed at the gene level rather than the segment level (as with one and two point crossover). For some problems, this additional flexibility outweighs the disadvantage of destroying building blocks.

Consider the following 2 parents which have been selected for crossover:

Parent 1: 11001010

Parent 2: 00100111

If the mixing ratio is 0.5, approximately half of the genes in the offspring will come from parent 1 and the other half will come from parent 2. Below is a possible set of offspring after uniform crossover:

Offspring1: $1_1 0_2 1_2 0_1 0_2 0_1 1_1 1_2$

Offspring2: $0_2 1_1 0_1 0_2 1_1 1_2 1_2 0_1$

Note: The subscripts indicate which parent the gene came from.

Arithmetic Crossover

A crossover operator that linearly combines two parent chromosome vectors to produce two new offspring according to the following equations:

Offspring1 = a * Parent1 + (1- a) * Parent2

Offspring2 = (1 – a) * Parent1 + a * Parent2

where a is a random weighting factor (chosen before each crossover operation).

Consider the following 2 parents (each consisting of 4 float genes) which have been selected for crossover:

Parent 1: (0.3)(1.4)(0.2)(7.4)

Parent 2: (0.5)(4.5)(0.1)(5.6)

If a = 0.7, the following two offspring would be produced:

Offspring1: (0.36)(2.33)(0.17)(6.86)
Offspring2: (0.402)(2.981)(0.149)(6.842)

Heuristic
A crossover operator that uses the fitness values of the two parent chromosomes to determine the direction of the search. The offspring are created according to the following equations:

Offspring1 = BestParent + r * (BestParent – WorstParent)
Offspring2 = BestParent

where r is a random number between 0 and 1. It is possible that Offspring1 will not be feasible. This can happen if r is chosen such that one or more of its genes fall outside the allowable upper or lower bounds. For this reason, heuristic crossover has a user settable parameter (n) for the number of times to try and find an r that results in a feasible chromosome. If a feasible chromosome is not produced after n tries, the worst parent is returned as Offspring1

1.2.5 Mutation

Mutation is a genetic operator that alters one ore more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible. Mutation is an important part of the genetic search as help helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to a user-definable mutation probability. This probability should usually be set fairly low (0.01 is a good first choice). If it is set to high, the search will turn into a primitive random search.

Flip Bit -A mutation operator that simply inverts the value of the chosen gene (0 goes to 1 and 1 goes to 0). This mutation operator can only be used for binary genes.

Boundary - A mutation operator that replaces the value of the chosen gene with either the upper or lower bound for that gene (chosen randomly). This mutation operator can only be used for integer and float genes.

Non-Uniform - A mutation operator that increases the probability that the amount of the mutation will be close to 0 as the generation number increases. This mutation operator keeps the population from stagnating in the early stages of the evolution then allows the genetic algorithm to fine tune the solution in the later stages of evolution. This mutation operator can only be used for integer and float genes.

Uniform - A mutation operator that replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

Gaussian - A mutation operator that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped if it falls outside of the user-specified lower or upper bounds for that gene. This mutation operator can only be used for integer and float genes.

1.3 Summary of the Algorithm:

The following outline summarizes how the genetic algorithm works:

1 The algorithm begins by creating a random initial population.

2 The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:

- **a** Scores each member of the current population by computing its fitness value.
- **b** Scales the raw fitness scores to convert them into a more usable range of values.
- **c** Selects members, called parents, based on their fitness.

d Some of the individuals in the current population that have lower fitness are chosen as elite. These elite individuals are passed to the next population.

e Produces children from the parents. Children are produced either by making random changes to a single parent — mutation — or by combining the vector entries of a pair of parents — crossover.

f Replaces the current population with the children to form the next generation.

g The algorithm stops when one of the stopping criteria is met.

1.4 Adaptive Genetic Algorithms:

In the standard GA, the optimum of a function is searched keeping different parameters like population size, probability of crossover and mutation etc fixed. However such a standard procedure is not always able to give the best performance when optimization of multidimensional functions having multiple local optima is required. This calls for the need to develop special GA techniques which can converge towards the global optimum more efficiently by changing its parameters in an adaptive manner during the course of the search. Several types of adaptive GAs has been proposed till now. One such GA, proposed by Srinivas and Patnaik uses variable crossover and mutation probability that vary according to the following equations:

$$P_c=k_1/(f_{max}-f_{avg})$$

$$P_m=k_2/(f_{max}-f_{avg})$$

where P_c and P_m are probabilities of crossover and mutation respectively. f_{max} and f_{avg} are maximum fitness and average fitness of a generation. k_1 and k_2 are constants having values between 0 to 1.

Another type of adaptive GA involves adaptation in terms of population size along with occasional reinitialization of the population. This includes the micro GA and the sawtooth GA.

1.4.1 Micro GA:

The micro GA, proposed by Goldberg is a small population GA which evolves for many generations. When after a number of generations the GA population converges, the evolutionary process is reinitialized by preserving the best individual and substituting the rest of the population with randomly generated individuals. The first implementation of micro GA was reported by Krishnakumar who used a population size of five individuals, tournament selection, single point crossover with probability Pc=1,elitism and no mutation. The population was considered converged when less than five percent of the population bits were different from the bits of the best individual. It was observed that micro GA can avoid premature convergence and performs better than a simple GA for selected multimodal problems.

1.4.2 Sawtooth GA:

The sawtooth GA can be viewed as an extension of micro GA. This scheme, proposed by Koumousis and Katsaras, uses a variable population size with periodic reinitialisation that follows a sawtooth scheme with a specific amplitude and period of variation. In each period, the population size decreases linearly and at the beginning of the next period randomly generated individuals are appended to the population. The scheme is characterized by the population size N_{avg}, amplitude D and period of variation T. Thus at a specific generation t, the population size N(t) is determined as

$$N(t) = \text{int}\{N_{avg}+D-2D/(T-1)[t-T\text{int}((t-1)/T)-1]\}$$

For amplitude D=0, regardless of the period T this algorithm is reduced to a constant population size GA. For bigger amplitude values D, the population size decreases with a constant decay and reinitialisation is enforced every T generations. The effect of population reinitialisation is more drastic as the amplitude D increases from 0 to N_{avg}-1. Moreover the selection of the period T is critical as it controls the duration of the decay process before reinitialisation occurs.

1.5 Study of sawtooth genetic algorithm:

The following discussion includes a study of sawtooth GA and its effectiveness in optimizing multimodal functions. The study has been carried out using different sets of average population size (N_{avg}), amplitude of variation (D) and period of variation (T), and effort has been made to find out an optimum range of values of these parameters so that the algorithm can be used for the optimization of different types of functions. In the course of the study, a comparison of the above with micro GA technique has also been made. The micro GA has been implemented using the parameters as indicated by Krishnakumar (population size = 5, probability of crossover = 1, probability of mutation = 0, tournament selection).

Six conventional test functions are used to simulate the performance of the two schemes:

1. Goldberg and Richardson test function
2. Rosenbrock test function
3. Schwefel test function
4. Rastrigin test function
5. Ackley test function
6. Griewangk test function

The first and second test functions involve four and three variables respectively whereas the rest are 10 variable functions. The experimental results thus obtained justify the assumptions regarding the superiority of sawtooth GA as compared to micro GA for the selected problems. All the simulations are carried out with the help of MATLAB. The simulation results are tabulated in the following section along with relevant graphs and figures.

Table 1.2: Comparison of standard GA, micro GA and sawtooth GA

Test Function	No. of variables	Theoretical Optimum	Standard GA	Micro GA	Sawtooth GA
Goldberg & Richardson	4	1	0.9975	1.0000	1.0000
Rosenbrock	3	0	0.0058	5.1299e-04	4.9756e-04
Rastrigin	10	0	3.0721	0.1855	1.7717e-05
Ackley	10	0	20.1201	0.5633	0.0051
Griewangk	10	0	0.1313	0.0614	7.2697e-07

Parameters used:

Standard GA:	**Micro GA:**	**Sawtooth GA:**
Population size: 5	Population size: 80	Population size: 80
Crossover probability: 1.0	Crossover probability: 0 .85	Amplitude of variation: 75
Mutation probability: 0.0	Mutation probability: 0.01	Period of variation: 40
		Crossover prob: 0 .85
		Mutation prob: 0.01

Observations on Table 1.2:

- In every case, performance of sawtooth GA has been better than the other two.
- When the number of variables is less (e.g. function 1 and 2), performance of micro GA and sawtooth GA are comparable. However as their dimensions increase, sawtooth GA starts giving better performance than micro GA.
- For the Ackley test function, standard GA tends to converge towards a local minimum. This tendency is eliminated in both micro GA and sawtooth GA.

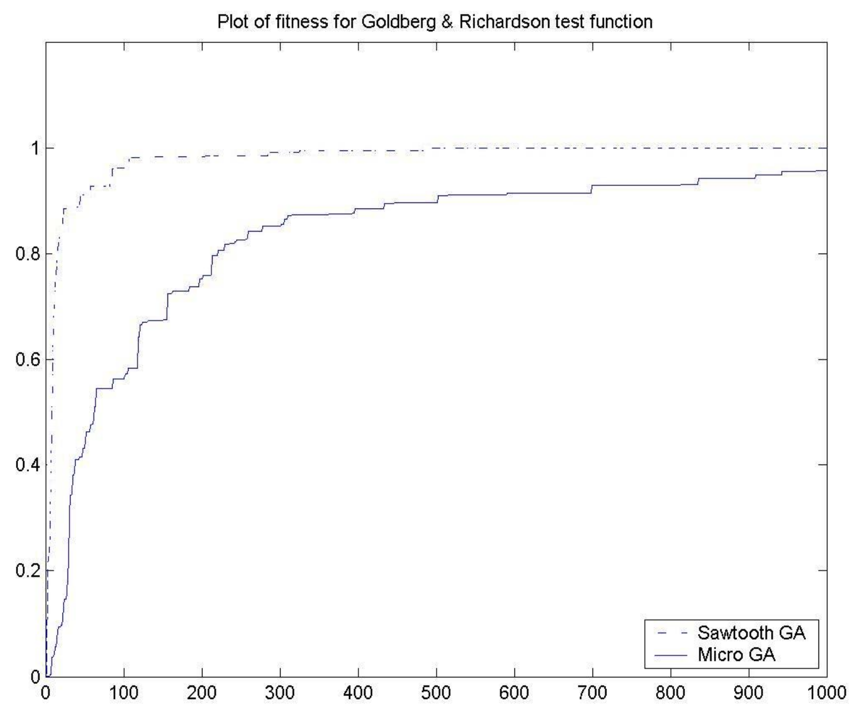


Figure 1.1: Plot of fitness for micro GA and sawtooth GA

The above figure shows the trend in which the micro GA and sawtooth GA converge towards the optimum value. The X-axis represents number of generations and Y-axis is the fitness value. It can be seen that the fitness for sawtooth GA rises very rapidly and reaches the optimum value in a much less number of iterations than micro GA.

Table 1.3: Performance of sawtooth GA for different average population

Function Name	Theoretical Optimum	Avg Pop = 16	Avg Pop = 30	Avg Pop = 40	Avg Pop = 50	Avg Pop = 80
Goldberg & Richardson	1	0.9394	0.9997	1.0000	1.0000	1.0000
Rosenbrock	0	0.1200	0.0074	5.2138e-003	7.0503e-004	6.0213e-005

Rastrigin	0	9.3886	0.9955	0.0109	3.8351e-005	1.7717e-005
Ackley	0	20.0957	0.2174	0.0369	0.0197	0.0051
Griewangk	0	0.1545	0.0295	5.8831e-005	0.0123	7.2697e-007

Observations on table 1.3:

- In the simulations above, the ratio T/N_{avg} and D/N_{avg} has been kept fixed.
- It is seen that the performance of sawtooth GA improves with increasing average population size.

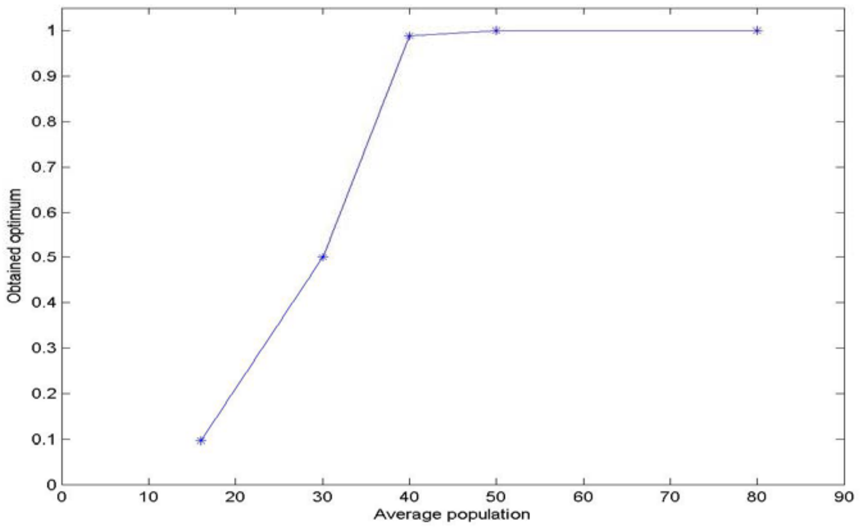


Figure 1.2: Plot of Average population and Obtained optimum

- The performance of the algorithm improves drastically with population when the population size is low but attains an almost constant value as the population size becomes larger.

Variation of performance with T and D:

- For the simulation of the given optimization problems, which were done with an average population ranging from 40 to 80, it has been observed that a large value of amplitude D gives better results. A small value of D is unable to bring the required diversity to the population and hence for multimodal functions is unable to reach the global optimum quickly. So it is required that D approaches the value of N_{avg}.

- The choice of time period of reinitialisation also has considerable impact on the performance. The best performance is achieved with a time period that is about half the average population size. Increasing T too much results in slower convergence while a very small T makes the GA unstable and make it converge towards some local maximum or minimum.

Sensitivity to changes in Pm and Pc:

From the simulations, it has been observed that the choice of Pm and Pc does not affect the performance of the GA scheme as long as Pc has a higher value (0.7 to 1) and Pm is small(around 0.01). Too small a value of Pc slows down the convergence while a large Pm may make the GA converge to a local optimum.

Computational complexity:

The sawtooth GA is found to be computationally more expensive than micro GA. Considering one period of variation of population in this scheme, the amount of operations that need to be performed is around 20 times that of micro GA. However its computational complexity is the same as that of standard GA.

1.6 Discussion:

The following inferences are drawn from the above work:

- Sawtooth GA is a better performer for optimization problems involving multimodal problems, specifically when the number of variables is large.
- The sawtooth GA is insensitive to the selection of crossover probability and mutation probability. Thus it can be used for different problems without optimizing these two parameters.
- The amplitude and period of variation affect the performance of sawtooth GA. A large value of D combined with a moderate value of T gives the optimum performance.
- Disadvantage of sawtooth GA is its computational expensiveness as compared to other types of GA techniques.

Chapter 2

HAMMERSTEIN MODEL IDENTIFICATION USING RBFN AND GENETIC ALGORITHM

2.1 Introduction

Most practical systems have inherently nonlinear characteristics such as saturation, dead-zone, in actuators or sensors. Identification of such kind of systems is greatly urgent for carrying out precise analysis, prediction or control design. The Hammerstein model is widely known and often adopted in nonlinear system identification. The model is one of the block oriented models with a nonlinear static part followed by a linear dynamic part. It has many advantages for control design or stability analysis due to the model structure. If the inverse of the static nonlinear part exists, the nonlinearity of the objective system is easily compensated by a controller implementing the inverse. Moreover the stability of the objective system is simply discussed by the linear dynamic part only. Several identification algorithms for the Hammerstein model have been investigated by using correlation theory, neural networks, orthogonal functions, polynomials , piecewise linear model , and so on. In this paper an identification method of the Hammerstein model is proposed by using radial basis function (RBF) networks and genetic algorithm (GA). Unknown nonlinear static part to be estimated is represented by the RBF network. The weighting parameters of the RBF network and the system parameters of the linear dynamic part are estimated by the linear least-squares method. The accuracy of this identification method depends sensitively on the RBF network structure, i.e. the number, centers and widths of the RBF. These adjusting parameters are properly determined with the aid of the GA, which is a probabilistic search procedure based on the mechanics of natural selection and natural genetics. The fitness value in this GA makes use of the Akaike information criterion (AIC) .
This paper is organized as follows. In section 2 the problem is formulated. In section 3 the identification method is proposed in case of the fixed RBF network structure. In section 4 the GA is applied to determination of the RBF network structure, i.e. the number, centers and widths of the RBF.

Consider a discrete-time nonlinear system described by the Hammerstein model described by the equations:

$$\begin{cases} A(q^{-1})y(k) = B(q^{-1})x(k-1) + e(k) \\ x(k) = f(u(k)) \\ A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n} \\ B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_r q^{-r} \end{cases} \qquad \text{------------------(1)}$$

where u(k) and y(k) are input and output signals, respectively. X(k) is intermediate signal that is not accessible for measurement. e(k) is measurement noise. q^{-1} denotes backward shift operator. n and r are known degrees of polynomials $A(q^{-1})$ and $B(q^{-1})$, respectively. f (.) is unknown nonlinear function. The problem is to identify the system parameters {ai} and {b j} of the linear dynamic part, and nonlinear static function f (.) from input and output data.

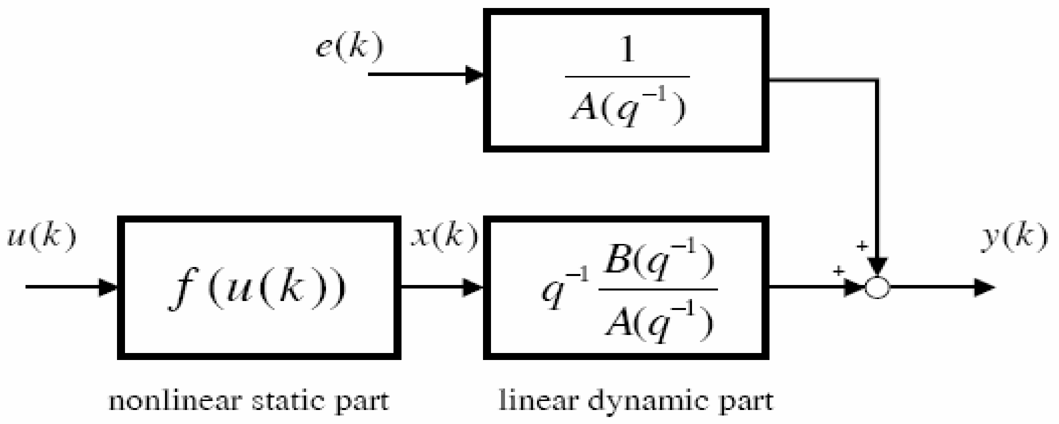


Figure 2.1 Hammerstein model

2.2 Identification

In this section the identification algorithm in case of the fixed RBF network structure is presented. The RBF network structure is properly determined by the GA in section 2. 4.
The nonlinear function is represented by using the RBF network depicted in Figure 2.2 as

$$f(u(k)) = \sum_{i=1}^{M} w_i \phi_i(u(k)) + \varepsilon(k)$$ -------------(2)

where

$$\phi_i(u(k)) = \exp\{-(\|u(k) - c_i\|^2 / d_i^2)\}$$ --------------(3)

is the Gaussian function. M is the number of the RBF. c_i and d_i are the ith center and width of the RBF, respectively. w_i is the weighting parameter associated with the ith RBF. || . || denotes the Euclidean norm. ε (k) is approximation error.

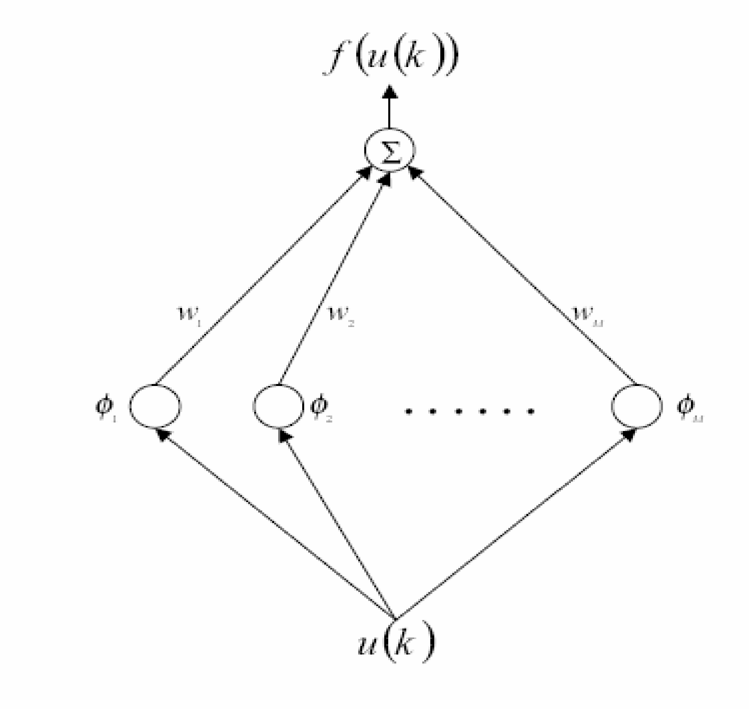


Figure 2.2 RBF network model

Substituting Eq.(2) into Eq.(1) yields

$$A(q^{-1})y(k) = \sum_{i=1}^{M} w_i B(q^{-1}) \phi_i(u(k-1)) + v(k)$$ ------------(4)

or in vector form,

$$y(k) = \boldsymbol{\varphi}^{\mathrm{T}}(k)\boldsymbol{\theta} + v(k)$$ -----------------------(5)

where v(k) = e(k)+B(q^{-1})ε (k-1) is equation error, and

$$\begin{cases} \boldsymbol{\theta} = [\boldsymbol{\theta}_a^{\mathrm{T}}, \boldsymbol{\theta}_{w_1}^{\mathrm{T}}, \boldsymbol{\theta}_{w_2}^{\mathrm{T}}, \cdots, \boldsymbol{\theta}_{w_M}^{\mathrm{T}}]^{\mathrm{T}} \\ \quad \boldsymbol{\theta}_a = [a_1, a_2, \cdots, a_n]^{\mathrm{T}} \\ \quad \boldsymbol{\theta}_{w_i} = [\theta_{w_i}(1), \theta_{w_i}(2), \cdots, \theta_{w_i}(r+1)]^{\mathrm{T}} \\ \qquad = [b_0 w_i, b_1 w_i, \cdots, b_r w_i]^{\mathrm{T}} \\ \boldsymbol{\varphi}(k) = [\boldsymbol{\varphi}_a^{\mathrm{T}}(k), \boldsymbol{\varphi}_{w_1}^{\mathrm{T}}(k), \boldsymbol{\varphi}_{w_2}^{\mathrm{T}}(k), \cdots, \boldsymbol{\varphi}_{w_M}^{\mathrm{T}}(k)]^{\mathrm{T}} \\ \quad \boldsymbol{\varphi}_a(k) = [-y(k-1), -y(k-2), \cdots, \\ \qquad\qquad -y(k-n)]^{\mathrm{T}} \\ \quad \boldsymbol{\varphi}_{w_i}(k) = [\phi_i(u(k-1)), \phi_i(u(k-2)), \cdots, \\ \qquad\qquad \phi_i(u(k-r-1))]^{\mathrm{T}} \end{cases}$$

$$(i = 1, 2, \cdots, M).$$ ---------(6)

Each parameter will be estimated as follows. First, the unknown parameter vector **θ** is easily evaluated by applying the linear least-squares method to Eq.(5):

$$\widehat{\boldsymbol{\theta}} = \left[\sum_{k=N_s+1}^{N_s+N} \boldsymbol{\varphi}(k)\boldsymbol{\varphi}^{T}(k)\right]^{-1} \left[\sum_{k=N_s+1}^{N_s+N} \boldsymbol{\varphi}(k)y(k)\right]$$ -------------(7)

where N is the number of input and output data. Thus the parameters of the linear dynamic part are estimated by

$$[\widehat{a}_1, \cdots, \widehat{a}_n, \widehat{b}_0, \cdots, \widehat{b}_r]^{\mathrm{T}} = \left[\mathbf{I}_{(n+r+1)\times(n+r+1)} : \ \mathbf{0}\ \right]\widehat{\boldsymbol{\theta}},$$ ---------------(8)

putting $\widehat{w}_1 = 1$ without loss of generality.

Next, the parameters of the nonlinear static part are obtained by using the linear least- squares technique again as

$$\widehat{w}_i = \frac{\sum_{j=1}^{r+1} \widehat{\theta}_{w1}(j)\widehat{\theta}_{wi}(j)}{\sum_{j=1}^{r+1} \widehat{\theta}^2_{w1}(j)} \quad (i = 2, 3, \cdots, M). \quad \ldots\ldots\ldots\ldots\ldots\ldots(9)$$

Thus the nonlinear static function is composed by Eq.(9) as

$$\widehat{f}(u(k)) = \sum_{i=1}^{M} \widehat{w}_i \phi_i(u(k)). \quad \text{-----------------}(10)$$

2.3 Optimization of RBF by GA

The accuracy of the above identification algorithm greatly depends on the RBF network structure, i.e. the number M, centers $\{c_i\}$ and widths $\{d_i\}$ of the RBF. The number M of RBF should be determined properly in order to avoid over parameterization and reduce the complexity of the estimated model. In this section the AIC is utilized as an objective function, and the RBF network structure is determined by the GA.

2.3.1 Coding and decoding

Note that the unknown parameter vector **θ** is estimated by the algorithm described in section 3 if candidates of M, {ci} and {di} are given. Therefore $\mathbf{\Omega} = (M,\{c_i\},\{d_i\})$ is coded into a binary bit string S and searched by the GA.

Assume that the maximum number M_{max} of the RBF is given. The string S consists of three blocks as shown in Figure 3. S_1 is a M_{max} bit binary string for the number M of the RBF. S_2 and S_3 are the blocks for the centers {ci} and widths {di} of the RBF, respectively. Both blocks have M_{max} sub-blocks. If the jth gene from the left hand in S_1 is the ith "1" from the left hand in S_1, then the jth sub-blocks from the left hand in both S_2 and S_3 are decoded for the ith center and width of the RBF, respectively, as follows:

$$c_i = \frac{c_{max} - c_{min}}{2^{L_2} - 1}\mathscr{C} + c_{min}$$

where $\mathscr{D}$ is the decimal value of the binary representation in jth sub-block in S2 and [c_{min}, c_{max}] is the search range of {ci}. In this decoding way, the number of ”1” gene in S_1 equals to the number M of the RBF.

2.4 Algorithm

First, an initial population which consists of binary bit strings as candidates of $\mathbf{\Omega}$ is generated. Then, candidates of the RBF network are constructed by using the decoded values from the strings. The candidates of unknown parameter vector $\boldsymbol{\theta}$ are estimated by the identification method described in section 3. The fitness values are calculated by using the AIC. The genetic operations, which are reproduction based on the fitness values, crossover and mutation, are repeated so that the fitness value of the population increases. In more detail the algorithm is as follows:

step 1: Initialization

Generate an initial population of Q binary bit strings for Ω randomly.

step 2: Decoding

Decode Q strings into real values $\hat{\mathbf{\Omega}}_i$

step 3: Construction of RBF network

Construct Q candidates of the RBF network using $\hat{\mathbf{\Omega}}_i$.

step 4: Identification

Identify $\hat{\boldsymbol{\theta}}_i$ and $\hat{f}_i(u(k))$ From Eqs.(7)-(10), using each candidates of the RBF network.

step 5: Fitness value calculation

Calculate the AIC:

$$AIC_i = N\log\left\{\frac{1}{N}\sum_{k=N_s+1}^{N_s+N}(y(k)-\hat{y}_i(k))^2\right\}+2P_i$$
$$(i=1,2,\cdots,Q)$$

and the fitness values F_i = -AICi, using $\hat{\boldsymbol{\Omega}}_i$, $\hat{\boldsymbol{\theta}}_i$ and $\hat{f}_i(u(k))$ $P_i = n+M_i(r+1)$ is the number of the parameters in the identification model Eq.(5). $\hat{y}_i(k)$ is the output of the estimated model.

step 6: Reproduction

Reproduce each of individual strings with the probability of $F_i/\sum_{j=1}^{Q}F_j$. Practically, the linear fitness scaling [11] is utilized to avoid undesirable premature convergence.

step 7: Crossover

Pick up two strings randomly and decide whether or not to cross them over according to the crossover probability P_c. Exchange strings at a crossing position if the crossover is required. The crossing position is chosen randomly and P_c is usually chosen greater than 50%.

step 8: Mutation

Alter a bit of string (0 or 1) according to the mutation probability P_m, which is generally such a quite low as less than a few percent.

step 9: Repetition

Repeat step 2 - step 8 from generation to generation so that the fitness value of the population increases. In simulations, the genetic operations will be repeated until prespecified Gth generation. This algorithm includes the elitest preserving strategy, in which an individual string having the best fitness value is guaranteed to survive in the next generation.

Finally, at the termination of this algorithm, the suboptimal parameters of the RBF network $\hat{\boldsymbol{\Omega}}_{best}$ is determined by the string with the best fitness value over all the past generations. So the final estimated model is constructed by $\hat{\boldsymbol{\Omega}}_{best}$, and the corresponding $\hat{\boldsymbol{\theta}}_{best}$ and $\hat{f}_{best}(u(k))$.

2.5 Experimental Results

The Hammerstein model was implemented on the following two test functions-

Function-I:

$$\begin{cases} A(q^{-1})y(k) = B(q^{-1})x(k-1) + e(k) \\ x(k) = f(u(k)) \\ \quad = \begin{cases} -2.0 & (-3.0 \leq u(k) < -1.8) \\ u(k)/0.6 + 1.0 & (-1.8 \leq u(k) < -0.6) \\ 0.0 & (-0.6 \leq u(k) < 0.6) \\ u(k)/0.6 - 1.0 & (0.6 \leq u(k) < 1.8) \\ 2.0 & (1.8 \leq u(k) \leq 3.0) \end{cases} \\ A(q^{-1}) = 1 + 0.8q^{-1} + 0.6q^{-2} \\ B(q^{-1}) = 0.4 + 0.2q^{-1} \end{cases}$$

Function-II:

$$\begin{cases} A(q^{-1})y(k) = B(q^{-1})x(k-1) + e(k) \\ x(k) = f(u(k)) = u(k) + 0.5u^3(k) \\ A(q^{-1}) = 1 + 0.8q^{-1} + 0.6q^{-2} \\ B(q^{-1}) = 0.4 + 0.2q^{-1} \end{cases}$$

The maximum number of RBF centers was set at five. Both standard GA and saw tooth GA was applied for evolving the parameters of the model and the end results were compared. The probability of crossover is 0.85 and the probability of mutation is 0.03.

Tournament selection was used in the genetic algorithm and elitism was used, which made sure that the string with the best fitness value always passed onto the next generation. The correctness of estimation was estimated by evaluating the mean average absolute error per sample. The approximation of the above two functions are shown below:

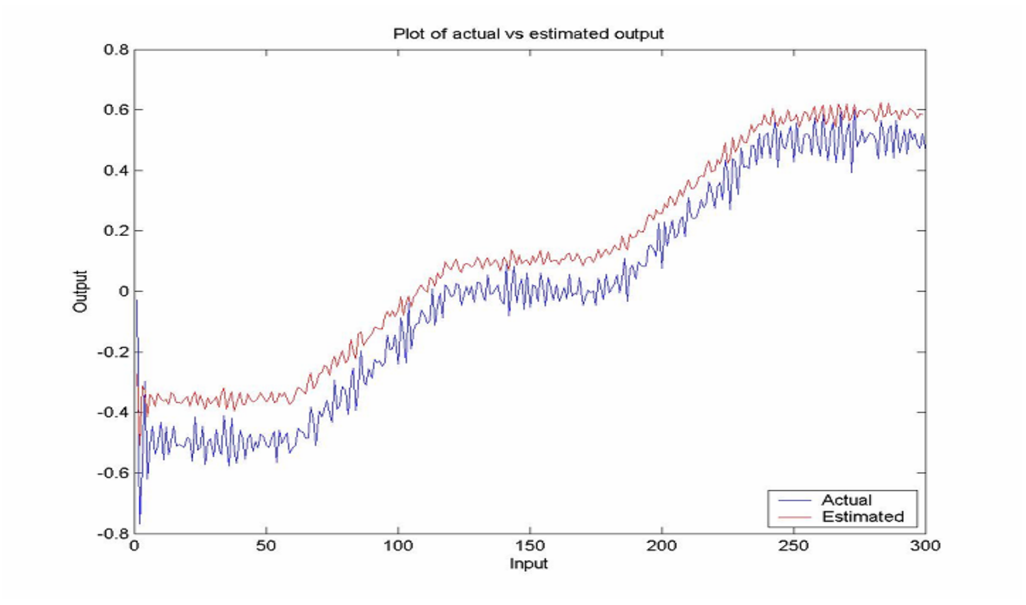


Figure 2.3: Plot of estimated vs actual output for function-I using standard GA

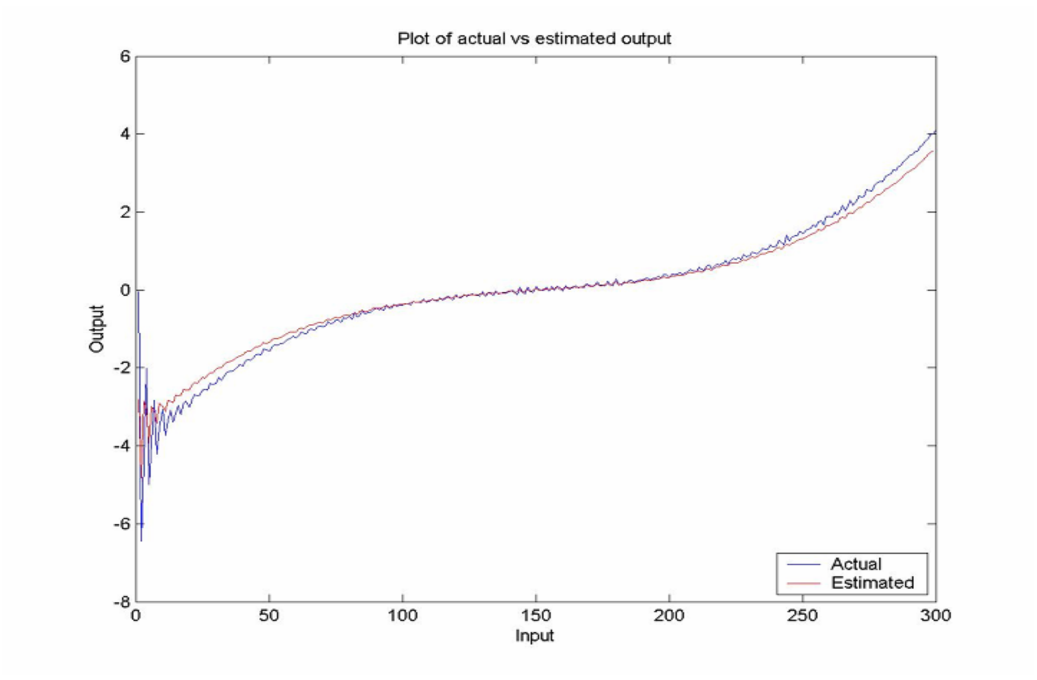


Figure2.4: Plot of estimated vs actual output for function-II using standard GA

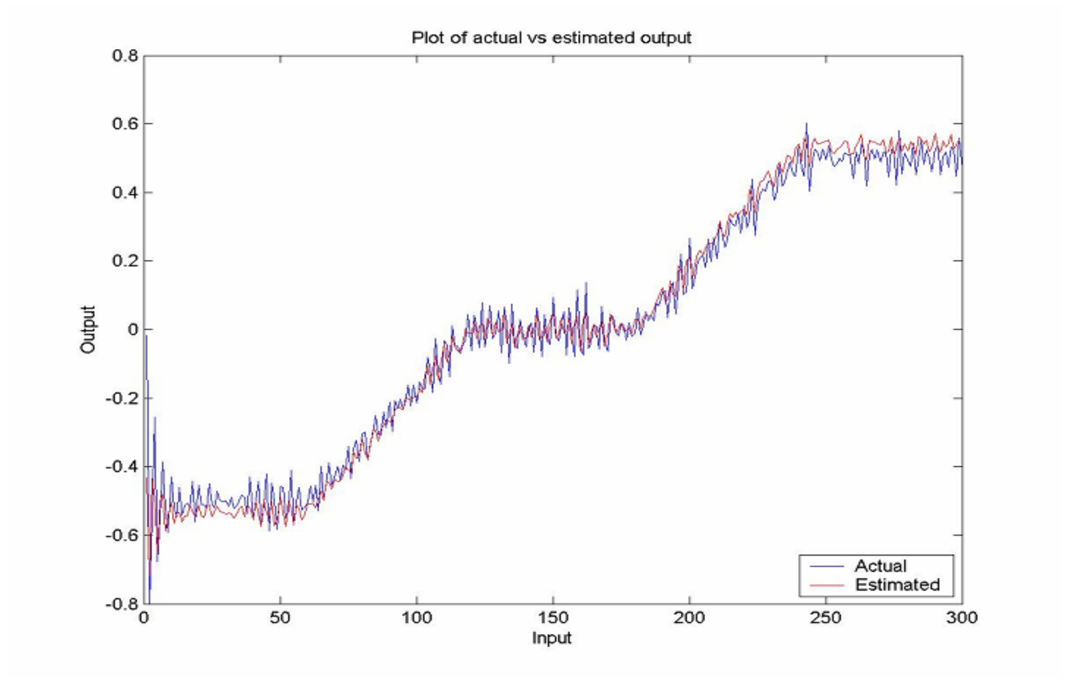


Figure 2.5: Plot of estimated vs actual output for function-I using saw tooth GA

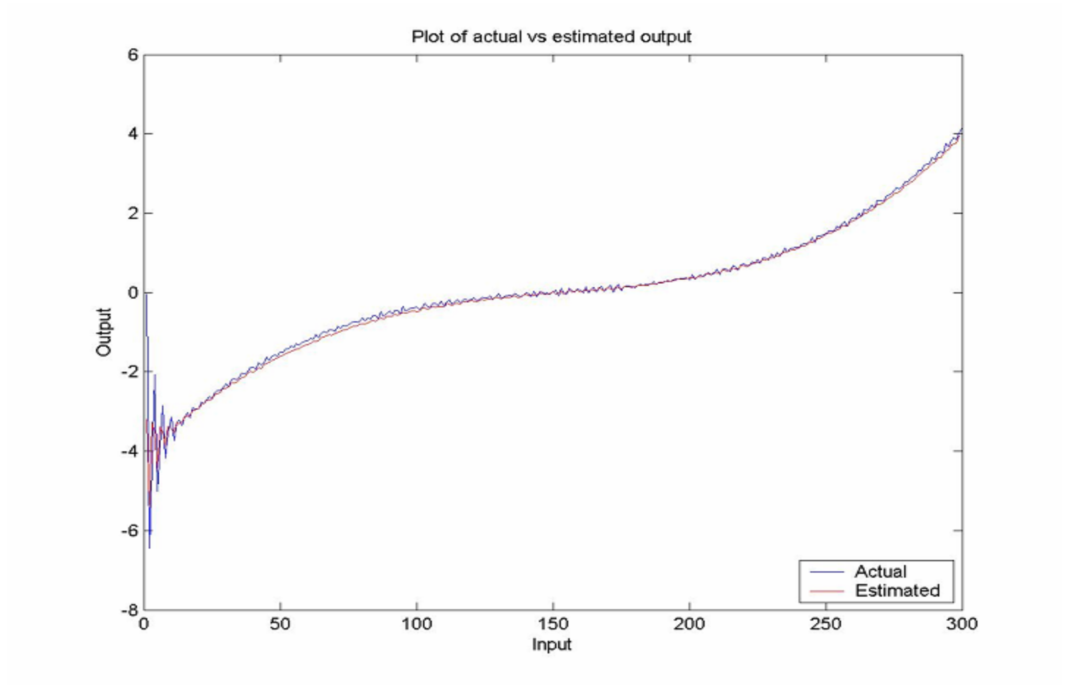


Figure 2.6: Plot of estimated vs actual output for function-II using saw tooth GA

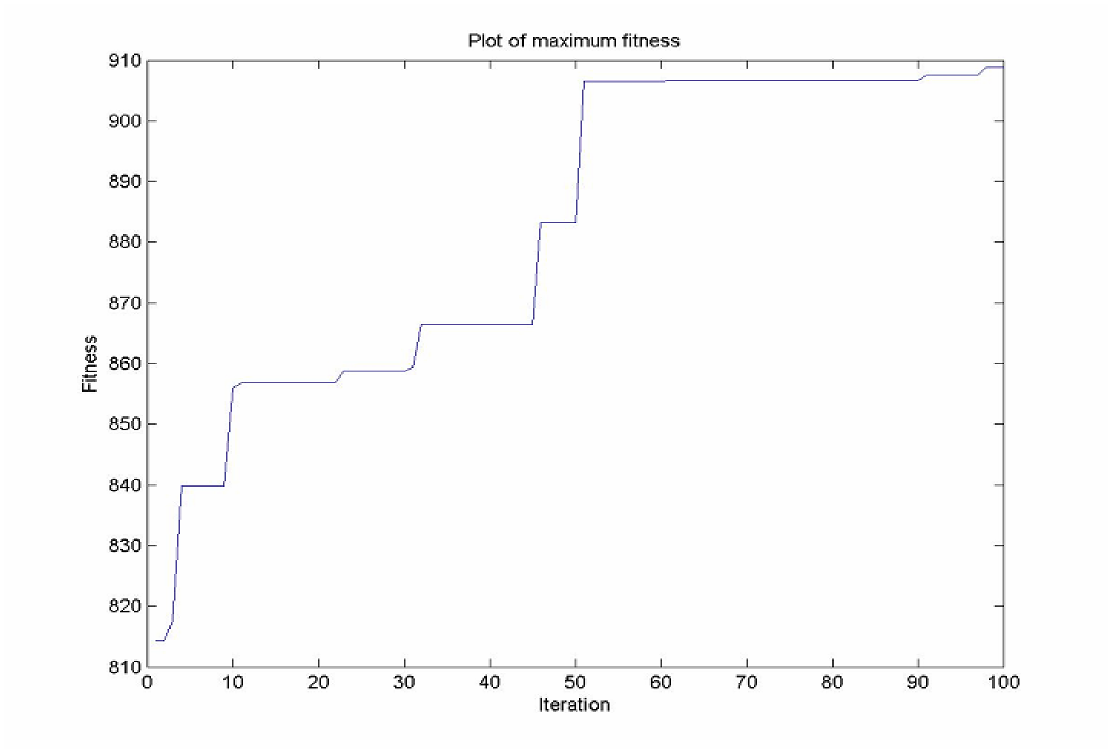


Figure 2.7: Plot of maximum fitness during training for function-I with saw tooth GA

Table 2.1: Performance comparison of standard and saw tooth GA applied for Hammerstein model identification

Test Function	Type of GA used	Mean Absolute Error
Function-I	Standard GA	0.1092
	Saw tooth GA	0.0371
Function-II	Standard GA	0.1043
	Saw tooth GA	0.0809

2.6 Discussion:

The evaluation of the mean absolute error showed that saw tooth GA as the tuning method gives more accurate results than the standard GA. In both the cases, the algorithm converges to the maximum fitness value quite fast, in about 300 iterations, which is far less than most other algorithms in use.

Chapter 3

STOCK MARKET FORECASTING USING RADIAL BASIS FUNCTION NETWORK AND GENETIC ALGORITHM

3.1 Introduction to Stock Market Prediction

Financial Forecasting or specifically Stock Market prediction is one of the hottest fields of research lately due to its commercial applications owing to the high stakes and the kinds of attractive benefits that it has to offer. Forecasting the price movements in stock markets has been a major challenge for common investors, businesses, brokers and speculators. As more and more money is being invested the investors get anxious of the future trends of the stock prices in the market. The primary area of concern is to determine the appropriate time to buy, hold or sell. In their quest to forecast, the investors assume that the future trends in the stock market are based at least in part on present and past events and data. However financial time-series is one of the most 'noisiest' and 'non-stationary' signals present and hence very difficult to forecast. The Dow Jones Industrial Average (DJIA) index was launched in 1896 with 12 stocks and is now the worlds most often quoted stock exchange index, based on a price-weighted average of 30 significant companies traded in the New York Stock Exchange (NYSE) and NASDAQ. The index gives a general indication of the behavior of the market towards different information. Another well known index, considered by researchers for prediction, is the Standard & Poor (S&P) 500. Many researchers in the past have applied various statistical and soft computing techniques such as neural networks to predict the movements in these stock indices. Generally technical indicators like moving averages and relative strength indices derived from the time series of these indices is employed in this regard. Financial time-series has high volatility and the time-series changes with time. In addition, stock market's movements are affected by many macro-economical factors such as political events, firms' policies, general economic conditions, investors' expectations, institutional investors' choices, movement of other stock market, psychology of investors, etc. Nevertheless there has been a lot of research in the field of stock market prediction across the globe on numerous stock exchanges; still it remains to be a big question whether stock markets can really be predicted and the numerous challenges that exist in its everyday application on the stock floor by the institutional investors to maximize returns. Generally there are three schools of thoughts regarding such prediction. The first school believes that no investor can achieve above 3 average trading advantages based on historical and present information. The major theories include the Random Walk Hypothesis and the Efficient Market Hypothesis. The second view is that of Fundamental Analysis. Analysts undertake in depth studies into the various macro-economic factors and look into the financial conditions and results of the industry concerned to discover the extent of correlation that may exist with the changes in the stock prices. Technical Analysis presents the third view on market price prediction. Analysts attempt to extract trends in market using past stock prices and volume information. These trends give insight into the direction taken by the stock prices which help in prediction. Technical Analysts believe that there are recurring patterns in the market behavior, which can be identified and predicted. In the process they use number of statistical parameters called Technical Indicators and charting patterns from historical data.

3.1.2 Application of Statistical and Soft Computing Techniques to Financial Forecasting

As the underlying theory behind all these techniques is totally different they generally give quite contradictory results. More importantly, these analytical tools are heavily dependent on human expertise and justification in areas like, the location of reversal (or continuation) pattern, market pattern, and trend prediction. For such reasons researchers have stressed on developing models for accurate prediction based on various statistical and soft computing techniques.

One such statistical technique employed in this regard is the Auto Regressive Integrated Moving Average (ARIMA) based model. Different time-series in practice have different frequency components. However, there is no systematic approach or a suitable class of models available in the literature to accommodate, analyze and forecast time-series with changing frequency behavior via a direct method. The virtue of ARIMA (Auto Regressive Integrated Moving Average) is well characterized by Vandaele: "… can be viewed as an approach by which timeseries data sifted trough a series of progressively finer sieves…" The aim of sifting some 4 components is to identify so called "white-noise-processes" which has merely stochastic influences on the time series.

The recent advancement in soft computing has given new dimension to the field of financial forecasting. Tools based on ANN have increasingly gained popularity due to their inherent capabilities to approximate any nonlinear function to a high degree of accuracy. Neural networks are less sensitive to error term assumptions and they can tolerate noise, chaotic components. Banks and Financial Institutions are investing heavily in development of neural network models and have started to deploy it in the financial trading arena. Its ability to 'learn' from the past and produce a generalized model to forecast future prices, freedom to incorporate fundamental and technical analysis into a forecasting model and ability to adapt according to the market conditions are some of the main reasons for its popularity. Radial Basis Function (RBF), Recurrent Neural Network (RNN) and Backpropagation in Multilayer Perceptron (MLP) are the three most popular Artificial Neural Network (ANN) tool for the task. On top of these, evolutionary approaches such as Genetic Algorithm (GA), confluence of statistics and ANN, are receiving attention as well.

A lot of research has gone into the development of models based on a range of intelligent soft computing techniques over the last two decades. Early models employed the Multi Layer Perceptron (MLP) architecture using Backpropagation algorithm, while a lot of recent work is based on evolutionary optimization techniques such as Genetic Algorithms (GA). This section describes briefly some of work that has gone into the field of application of ANN to stock price prediction. In Japan, technology major Fujitsu and investment company, Nikko Securities joined hands to develop a stock market prediction system for TOPIX, the Tokyo based stock index, using modular neural network architecture. Various economic and technical parameters were taken as input to the modular neural network consisting of multiple MLP used in parallel. A study was done on the effect of change of network parameters of an ANN Backpropagation model on the stock price prediction problem. The paper gives insights into the role of the learning rate, momentum, activation function and the number of hidden neurons to the prediction. In addition to ANN using Backpropagation, the Probabilistic Neural Network (PNN) has also been employed to stock prediction. In their work, the model is used to draw up a conservative thirty day stock price prediction of a specific stock: Apple Computers Inc. Due to their bulky nature owing to the large training data, the PNN are not popular among forecasters. In the process lots of newer architectures came to the fore. Ornes & Sklansky in their paper present a Visual Neural Network (VNN), which combines the ability of multi expert networks to give low prediction error rates with visual explanatory power of nonlinear dimensionality reduction. They conclude that the VNN is a powerful means of interactive neural network design, which provides both better prediction accuracy and good visual explanatory ability.

Another architecture introduced to the prediction problem is the Multi Branch Neural Network (MBNN) proposed by (Yamshita, Hirasawa & Hu, 2005) and applied to the TOPIX (Tokyo 7 Stock Exchange). The simulations show that MBNN, based on the concept of Universal Learning Networks (ULN), have higher accuracy of prediction than conventional NNs. In their paper, (Chen, Dong & Zhao, 2005) investigate how the seemingly chaotic behavior of stock market could be well represented using Local Linear Wavelet Neural Network (LLWNN) technique. They considered the NASDAQ-100 index and S&P CNX NIFTY index (India). The LLWNN is optimized by using Estimation of Distribution Algorithm (EDA). Results show that the LLWNN model performs marginally better than conventional NN models. Hybrid architectures are also being deployed in recent times. (Raymond Lee, 2004) propose a Hybrid Radial Basis Function Recurrent Network (HRBFN) stock prediction system called the iJADE stock advisor. The stock advisor was applied to major Hong Kong stocks and produced promising results in terms of efficiency, accuracy and mobility. Another Hybrid AI approach to the implementation of trading strategies in the S&P 500 index futures market is proposed by (Tsiah, Hsu & Lai,). The Hybrid AI approach integrates the rule-based systems techniques with Reasoning Neural Networks (RN) to highlight the advantages and overcome the limitations of both the techniques. They demonstrate that the integrated futures trading system (IFTS) based on this hybrid model outperforms other conventional NN. There are instances of application of fuzzy logic based models to the stock market prediction as well. Hiemstra proposes a fuzzy logic forecast support system to predict the stock prices using parameters such as inflation, GNP growth, interest rate trends and market valuations. According to the paper, the potential benefits of a fuzzy logic forecast support are better decision making due to the model-based approach, knowledge management and knowledge accumulation. Another effort towards the development of fuzzy models for stock markets has been made by (Alaa Sheta, 2006) using Takagi-Sugeno (TS) fuzzy models. Sheta uses the model for two non-linear processes, one pertaining to NASA and the other to prediction of next week S&P 500 index levels. The two steps involved in the process are 1) the determination of the membership functions in the rule antecedents using the model input data; 2) the estimation of the consequence parameters. Parameters are estimated using least square estimation. 8 The application of evolutionary optimization techniques such as Genetic Algorithm has given an entirely new dimension to the field of stock market prediction. (Badawy, Abdelazim & Darwish) conducted simulations using GA to find the optimal combination of technical parameters to predict Egyptian stocks accurately. Tan, Quek & Ng, (2005) introduce a novel technique known as Genetic Complementary Learning (GCL) to stock market prediction and give comparisons to demonstrate the superior performance of the method. GCL algorithm is a confluence of GA and hippocampal complementary learning. Another paper introducing Genetic algorithm approach to instance selection (GAIS) (Kyoungjae- Kim, 2006) for ANN in financial data mining has been reported. Kim introduces this technique to select effective training instances out a large training data set to ensure efficient and fast training for stock market prediction networks. The GA also evolves the weights that mitigate the well known limitations of the gradient descent algorithm. The study demonstrates enhances prediction performance at reduced training time. A hybrid model proposed by (Kuo, Chen & Hwang, 2001) integrates GA based fuzzy logic and ANN. The model involves both quantitative factors (technical parameters) and qualitative factors such as political and psychological factors. Evaluation results indicate that the neural network considering both the quantitative and qualitative factors excels the neural network considering only the quantitative factors both in the clarity of buying-selling points and buying selling performance. Another hybrid model involving GA proposed by (Hassan, Nath & Kirley, 2006) utilizes the strengths of Hidden Markov Models (HMM), ANN and GA to forecast financial market behavior. Using ANN, the daily stock prices are transformed to independent sets of values that become input to HMM. The job of the GA is to optimize the initial parameters of HMM. The trained HMM is then used to identify and locate similar patterns in the historical data. A similar study investigates the effectiveness of a hybrid approach based on Time Delay Neural Networks (TDNN) and GA (Kim & Shin, 2006). The GA is used to optimize the number of time delays in the neural network to obtain the optimum prediction performance.

Other studies and research in the field of stock market prediction using soft computing techniques include comparative investigation of both the ANN and the statistical ARIMA model (Schumann & Lohrbach, 1994) for the German stock index (DAX).The ANN method uses the four layer counter propagation network. The paper compares the results provided by both the methods and concludes that the efficient market hypothesis does no hold good. A Data Compression Techniques for stock prediction (Azhar, Badros & Glodjo, 1994) has been reported that uses the vector quantization method as an example of lossy data compression and Lempel-Ziv method as an example of lossless data compression technique to predict most of the well known indices across the globe.

3.2 Data preprocessing and technical parameters

The data for the stock market prediction experiment has been collected for two stock indices namely Dow Jones Industrial Average (DJIA), USA, Standards & Poor's 500 Index (S&P 500), USA. The time series data of all the stock indices were collected from 3rd January 1994 to 23rd October 2006. Thus there were 3228 data patterns for both DJIA and S&P 500 index. The data collected for the stock indices consisted of the closing price, opening price, and lowest value in the day, highest value in the day and the total volume of stocks traded in each day. (Note that one day's closing price of the index can be slightly different from next day's opening price, due to introduction of after hours trading between institutions private exchanges). The proposed forecasting model is developed to forecast the closing price of the index in each day of the forecasting period.

Different technical and fundamental indicators are used as inputs to the network. Technical indicators are any class of metrics whose value is derived from generic price activity in a stock or asset. Technical indicators look to predict the future price levels, or simply the general price direction, of a security by looking at past patterns. Out of the many technical indicators used by traders, 10 indicators have been chosen as input to the network which has been used before by many researchers for stock market forecasting problems. The details of the parameters and how they are calculated from the available data is given below:

• Simple Moving Average (SMA):

It's the simple average of the values by taking a window of the specified period. The various SMAs used in the experiment are:

1. 10 days (SMA10)
2. 20 days (SMA20)
3. 30 days (SMA30)

• Exponential Moving Average (EMA):

It is also an average of the values in the specified period but it gives more weight to recent values. Thus it approaches the actual values more closely.

Formula Used:

EMA= (P * A) + (previous EMA * (1 – A)) (10)

P=> current price
A=> smoothing factor = 2/(1+N)
N=> no. of time periods

• Accumulation/Distribution Line(ADO):
It measures money flow in the security. It attempts to measure the ratio of buying to selling by comparing price movements of a period to the volume of that period.
A/DO = ((Close – Low) – (High – Close))/ (High – Low) * Period's Volume (11)
Every day's ADO has been taken in the experiment.

• Stochastic Oscillator(STOC):
Stochastic Oscillator is a momentum indicator that shows the location of the current close relative to the high/low range over a set of number of periods. Closing levels that are consistently near the top of the range indicates accumulation (buying pressure) and those near the bottom of the range indicate distribution (selling pressure).
There are two lines: %K and %D
Formula Used:
%K = [(Today's Close – Lowest low in K periods)/ (Highest high in K periods – Lowest low in K periods)] * 100 (12)
%D is the SMA of %K for a particular period.
For this study: %K = 10 days and % D = 3 days

• On Balance Volume (OBV):
It is a momentum indicator that relates volume to price change.
Calculation of OBV:
If today's close > Yesterday's Close
OBV = Yesterday's OBV + Today's Volume
If today's close > Yesterday's Close
OBV= Yesterday's OBV – Today's volume

• Williams %R(WILLIAMS):
It is a momentum indicator that measures overbought/oversold levels.

Calculation of Williams %R =

(Highest high in n periods – Today’s close)*100 (13)

(Highest high in n-periods – Lowest low in n-periods)

For this experiment: n= 9 days

• Relative Strength Index (RSI) :

It calculates the internal strength of the security. It has been used in most of the research papers.

Basic formula for RSI calculation:

RSI = 100 – (100/(1+ (U/D)) (14)

For this study the periods have been taken as 9 days (RSI9) and 14 days (RSI14).

• Price Rate of Change (PROC):

The PROC indicator displays the difference between the current price and closing price x-time periods ago.

Calculation:

(Today’s close – Close x-periods ago) * 100 (15)

(Close x-periods ago)

Through experimental results it’s found that x=12 is considered best for technical analysis.

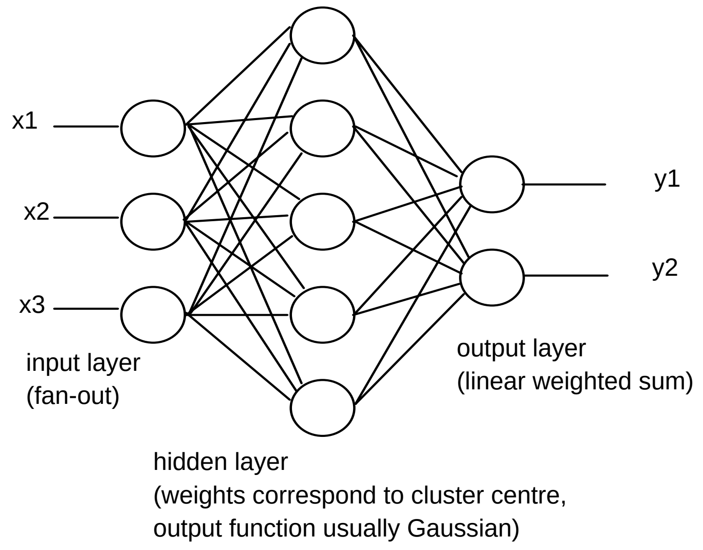
• Closing Price (CPACC) and High Price (HPACC) Acceleration:

It’s the acceleration of the closing prices and the high prices in the given period. Apart from these technical parameters which depend on the past value of the data for forecasting, it has been shown by Nial O’ Connor and Michael G. Madden (2006) that there are Fundamental Analysis Factors as well which affect the stock market and hence forecasting can be improved by incorporating them. Fundamental analysis is the study of economic, industry, and company conditions in an effort to determine the value of a company's stock. Fundamental analysis typically focuses on key statistics in a company's financial statements to determine if the stock price is correctly valued.

Most fundamental information focuses on economic, industry, and company statistics. Some of the fundamental factors included in this project work are: Monthly average oil price, Quarterly Gross Domestic Product (GDP) growth rate, Quarterly corporate dividend rate, Monthly interest rates and inflation figures in terms of Commodity Price Index (CPI).

3.3 Experiment Model Setup

We use the radial basis function network (RBFN) architecture for the stock market prediction model. The structure of the RBF network is as shown below:



The inputs are applied to the hidden layer of the RBF network, which computes the Euclidean distance of the inputs from the centers. The centers must have the same dimensionality as the inputs. A linear weighted sum of the hidden layer is produced at the output node of the structure. For an RBFN using Gaussian centers, the output of each center is given by

$$\phi_i(u(k)) = \exp\{-(||u(k) - c_i||^2/d_i^2)\}$$

where, u(k) is the input pattern, c_i is the center and d_i is the spread parameter.

The final output is given by

$$\widehat{f}(u(k)) = \sum_{i=1}^{M} \widehat{w}_i \phi_i(u(k)).$$

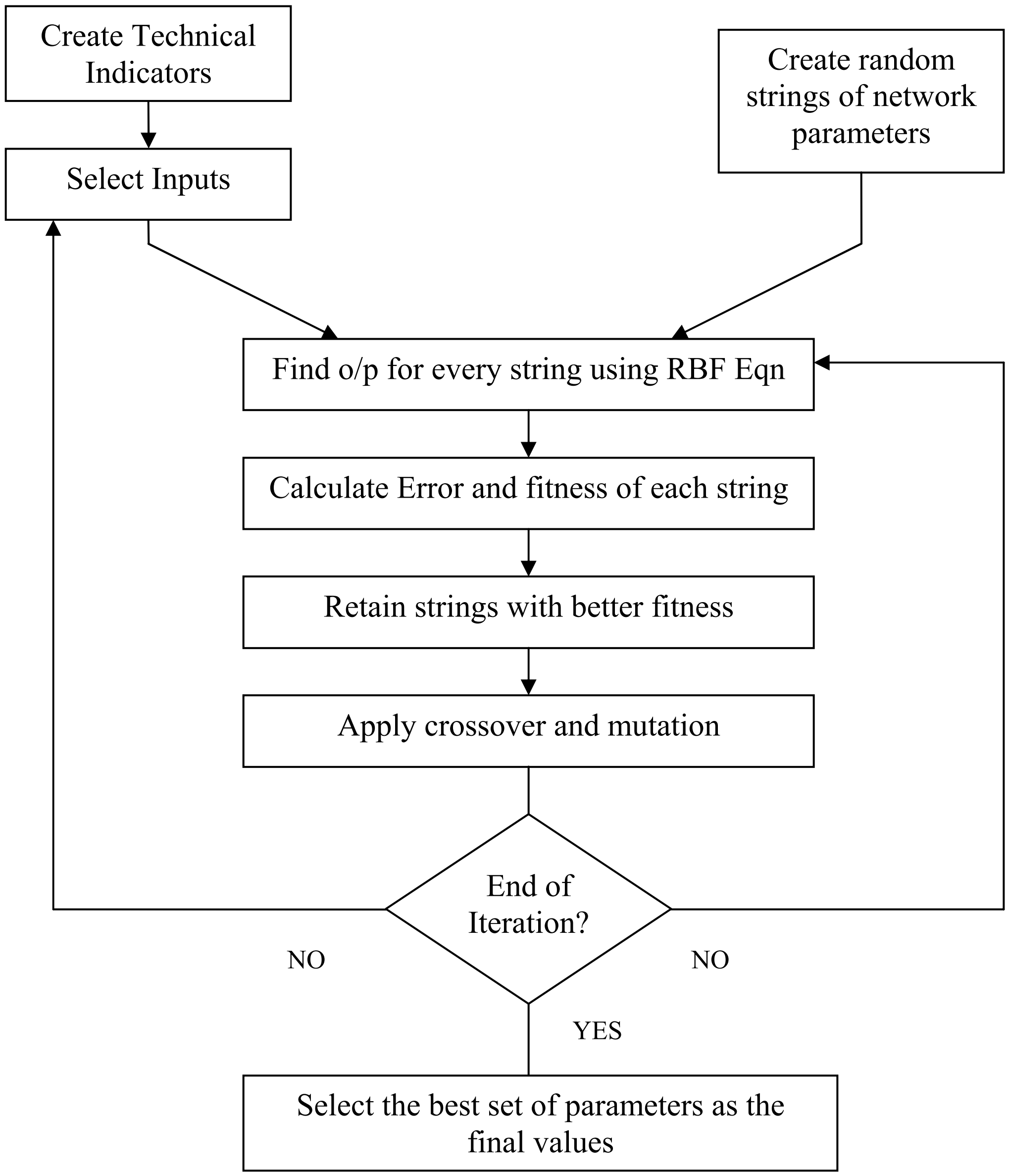
Where, M is the number of centers and w_i are the connecting weights. The values of the centers, spread and weights have to be trained adaptively so that the network output matches a required pattern. In the present case, genetic algorithm is used to optimize the RBF structure. Here, we take a population of random weights, centers and biases represented as binary strings and apply

our input pattern to each of these sets. Depending upon the accuracy of the output of each network, a fitness value is assigned to it and the strings having better fitness are retained and the others are rejected. Than crossover and mutation operators are applied on the selected strings as explained in section 1.3. The process is carried out iteratively for several generations.
The inputs to the RBFN are chosen from the technical indicators mentioned in the previous section. The number of RBF centers is kept variable, and the performance of the model is estimated for different number of centers. The coordinates of the centers and the values of the connecting weights in the output layer are optimized using binary coded genetic algorithm (GA).
The inputs have to be normalized for the proper behavior of the network. The inputs are normalized to values between +1 and -1. This can be done by a number of normalization techniques. One of the popular techniques we used was expressing the data in terms of the maximum and minimum of the data set.
All the values are normalized by using the following equation
$Y = [2*X - (Max + Min)]/(Max + Min)$
Y: - normalized values.
X: - present value.
The total data set of a particular stock market index is split up into two, one for training of the network and the rest for testing the performance of the network after freezing the weights. In this experiment we take approx 500 daily statistical data of the stock index as training set. Then around 500 data that follows are used for testing the network.

3.4 Training Process

The training of the network takes place in the following fashion. The parameter update is epoch based. The initial centers are taken as random values in the range 1 to -1. The weights are also initialized to be random numbers between 3 to -3. An initial population of 155 such weights and centers are created. The input data set are normalized prior to the network training. The inputs are applied to the whole population of weights and centers. The average error for the whole data set with respect to every member of the population is computed, and a fitness value is assigned to every member. Depending upon the fitness value, the best member strings are retained by tournament selection method along with elitism. The genetic operators crossover and mutation are also applied to the members in each iteration. The probability of crossover is 0.85 and the probability of mutation is 0.03. Here, the cost function is the fitness value of the member strings, and the maximum fitness value for each iteration is computed and plotted. Each of the iterations involves training the network with the 500-odd patterns, calculation of fitness function, crossover, mutation and selection. The iteration can be stopped when it is observed that there is no further significant improvement in the fitness values. There exists a trade-off between the time taken and quality of training. High number of iterations tends to give better training of the network at the cost of time taken to train.

Flowchart of the algorithm :


Create Technical Indicators
Select Inputs
Create random strings of network parameters
Find o/p for every string using RBF Eqn
Calculate Error and fitness of each string
Retain strings with better fitness
Apply crossover and mutation
End of Iteration?
NO
NO
YES
Select the best set of parameters as the final values

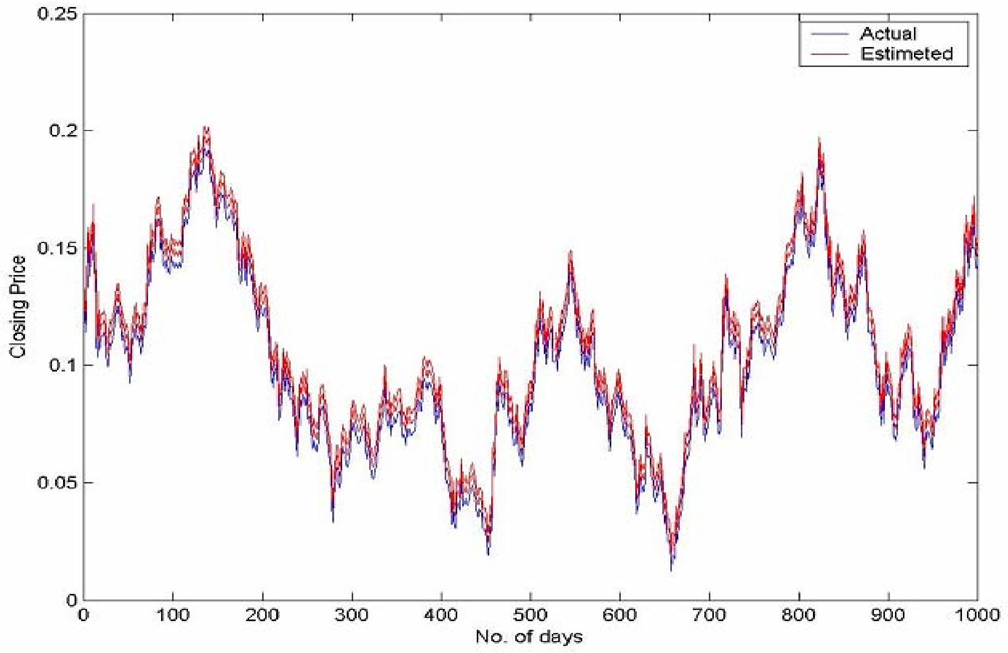


Figure 3.1: Plot of predicted vs actual closing price (Training)

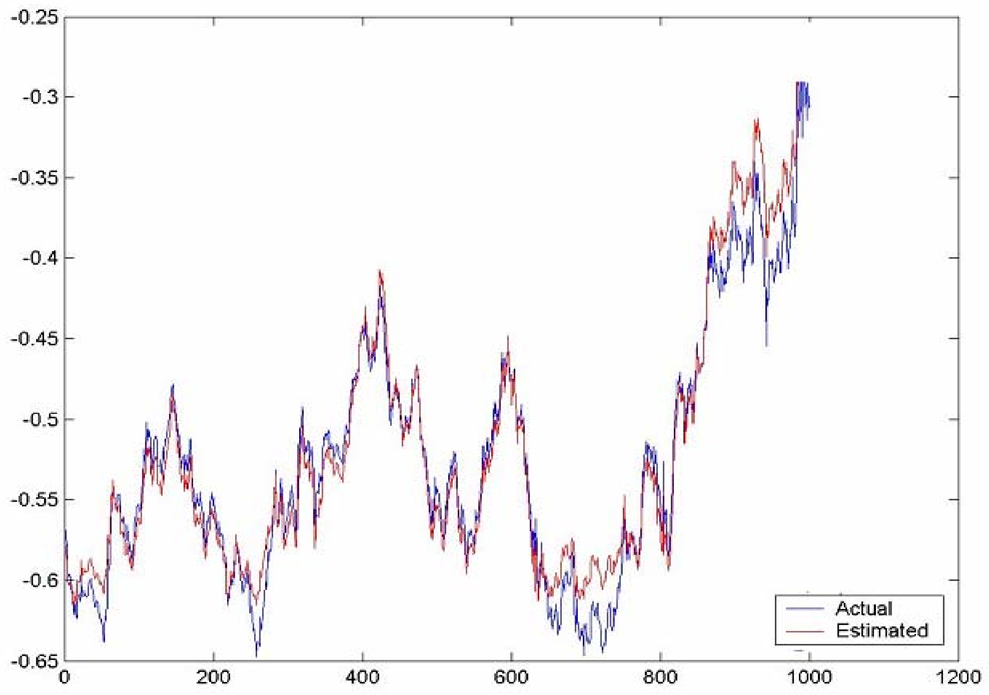


Figure 3.2: Plot of predicted vs actual closing price (Testing)

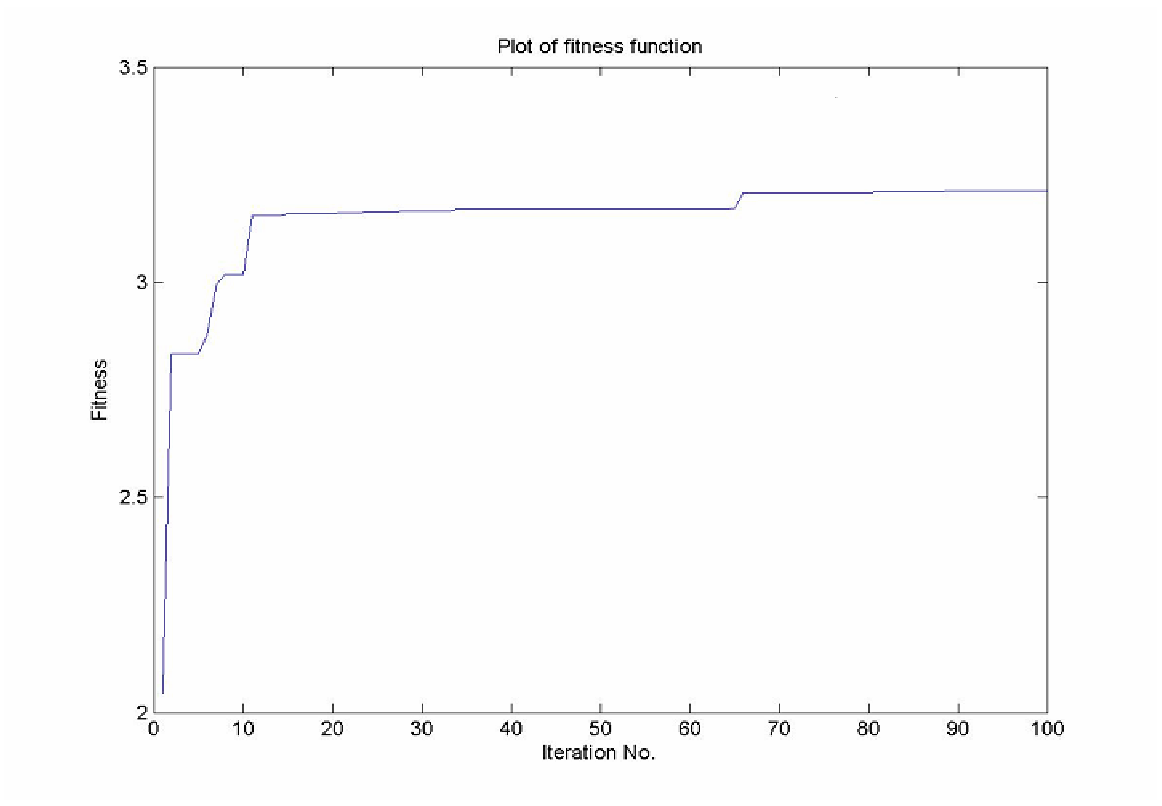


Figure 3.3: Plot of fitness function during training

3.5 Testing Process

At the end of the training process of the network, the weights are frozen for testing the network on inputs that were set apart from the training set. The testing set patterns are the input to the network and the output, the predicted index close price is compared with desired output or actual close price. The percentage of error is recorded for each data set. The criteria for judging the quality of prediction shown by the model is the mean of all the percentage error of the testing data set. The Mean Absolute Percentage Error (MAPE) is used to gauge the performance of the trained prediction model for the test data. The effort is to minimize the MAPE for testing patterns in the quest for finding a better model for forecasting stock index price movements The MAPE is given as

$$MAPE = \frac{1}{N}\sum_{j=1}^{N} \left| \frac{y_j - \hat{y}_j}{y_j} \right| \times 100$$

3.6 Simulation Results

Table 3.1 One day in advance prediction using Saw tooth GA tuned RBFN

Input Variables to RBFN	Training Period	Testing Period	MAPE
EMA20,EMA30,ADO,RSI9	500 Days	400 Days	16.6556
EMA20,RSI9,RSI27,PROC12	500 Days	400 Days	17.5619
PROC12,HPACC,STOC,Williams	500 Days	400 Days	10.1542
EMA10,EMA20,Closing Price	500 Days	400 Days	2.1781
EMA10,EMA20,EMA30,Closing Price	500 Days	400 Days	1.1931

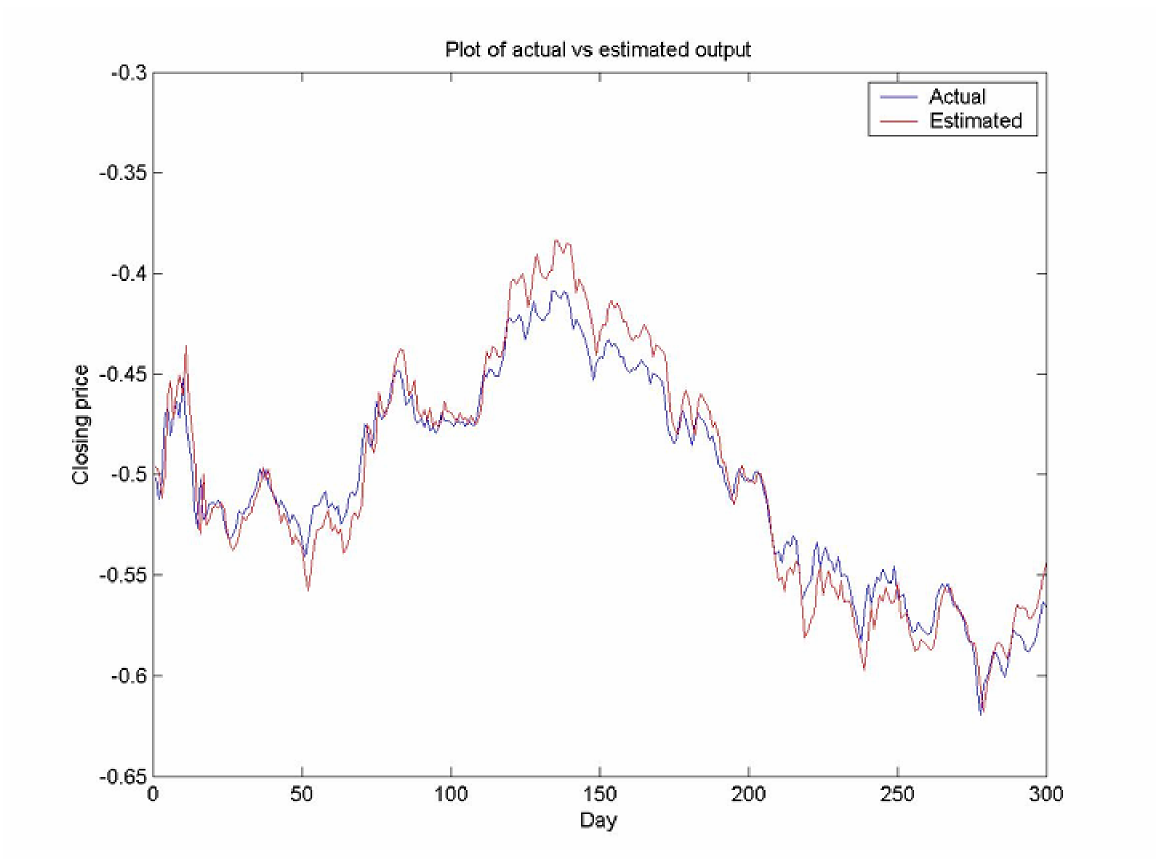


Figure3.4 : Plot of predicted vs actual closing price for testing dataset of Bombay Stock Exchange (One day in advance prediction)

Table 3.2 One week in advance prediction using saw tooth GA tuned RBFN

Input Variables to RBFN	Training Period	Testing Period	MAPE
EMA20,EMA30,ADO,RSI9	500 Days	300 Days	18.4214
EMA20,RSI9,RSI27,PROC12	500 Days	250 Days	27.8923
PROC12,HPACC,STOC,Williams	500 Days	250 Days	9.1542
EMA10,EMA20,Closing Price	500 Days	500 Days	3.1781
EMA10,EMA20,EMA30,Closing Price	500 Days	500 Days	2.9867

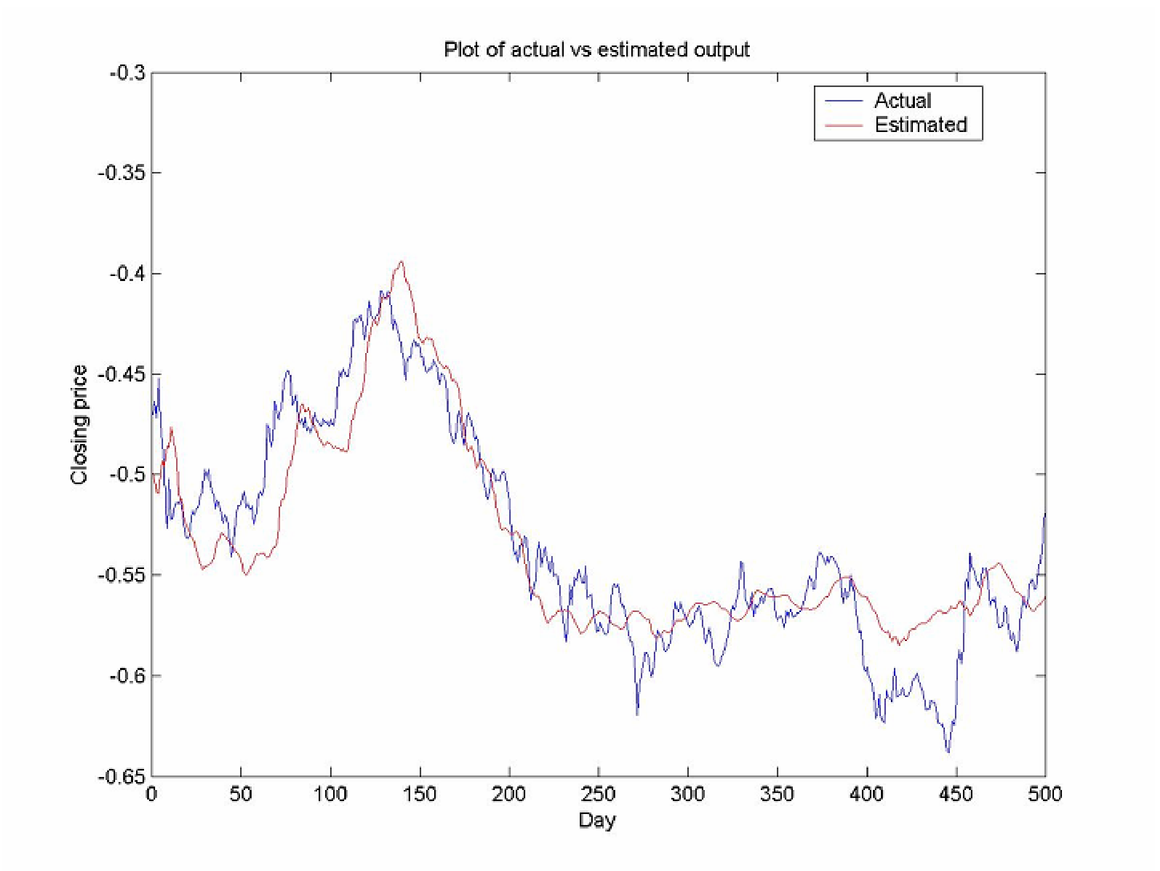


Figure 3.5 : Plot of predicted vs actual closing price for testing dataset of Bombay Stock Exchange (One week in advance prediction)

Table 3.3 One month in advance prediction using saw tooth GA tuned RBFN

Input Variables to RBFN	Training Period	Testing Period	MAPE
EMA20,EMA30,ADO,RSI9	500 Days	300 Days	15.2719
EMA20,RSI9,RSI27,PROC12	500 Days	250 Days	10.6474
PROC12,HPACC,STOC,Williams	500 Days	250 Days	8.1753
EMA10,EMA20,Closing Price	500 Days	500 Days	4.5399
EMA10,EMA20,EMA30,Closing Price	500 Days	800 Days	4.3560

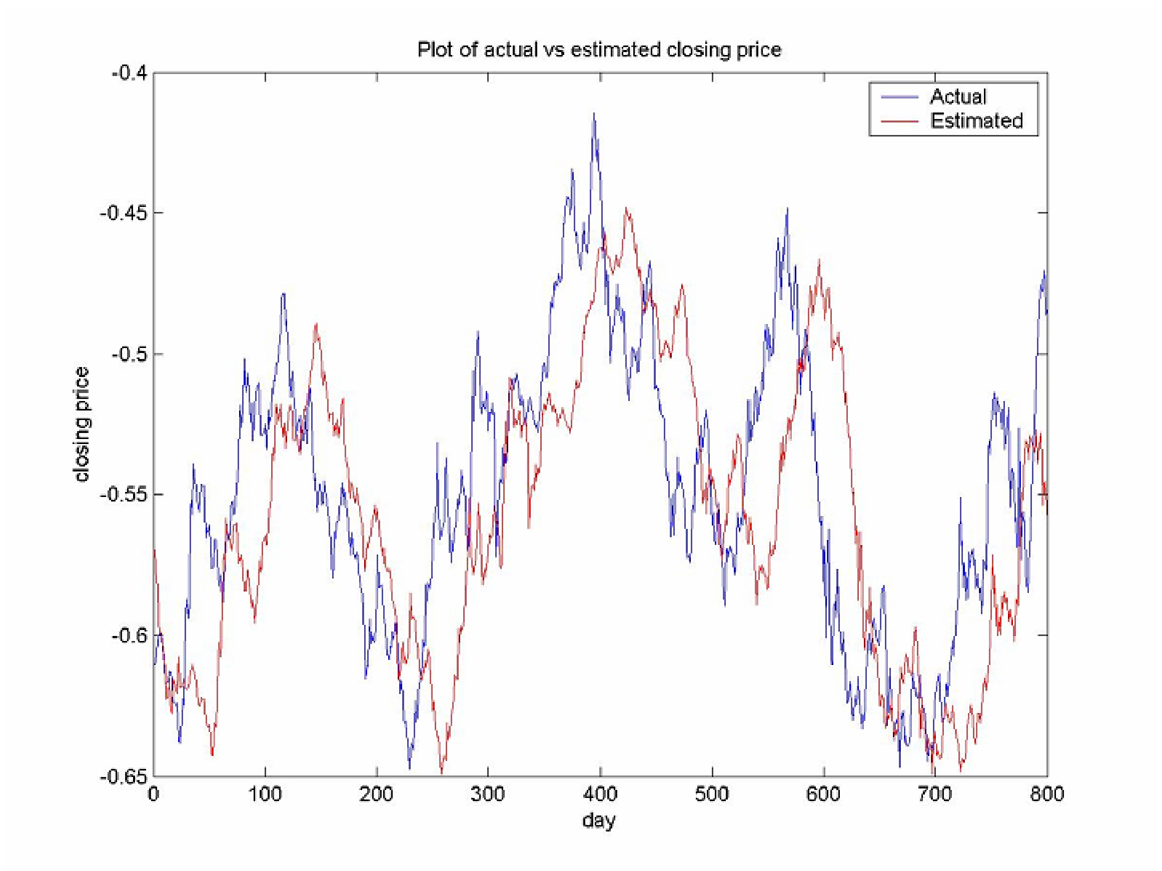


Figure 3.6 : Plot of predicted vs actual closing price for testing dataset of Bombay Stock Exchange (One day in advance prediction)

Table 3.4 Two months in advance prediction using saw tooth GA tuned RBFN

Input Variables to RBFN	Training Period	Testing Period	MAPE
EMA20,EMA30,ADO,RSI9	500 Days	150 Days	23.3463
EMA20,RSI9,RSI27,PROC12	500 Days	250 Days	28.2493
PROC12,HPACC,STOC,Williams	500 Days	250 Days	13.2816
EMA10,EMA20,Closing Price	500 Days	500 Days	9.7724
EMA10,EMA20,EMA30,Closing Price	500 Days	500 Days	8.6971

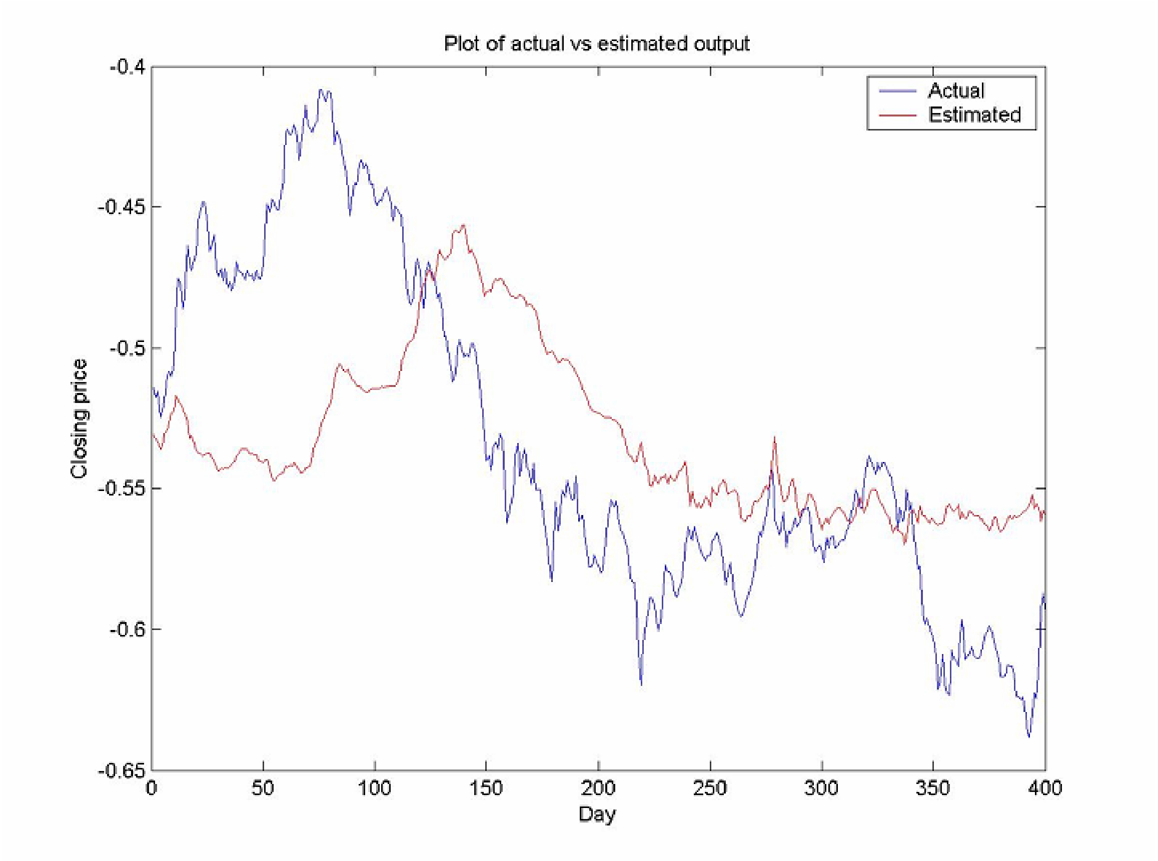


Figure 3.7 : Plot of predicted vs actual closing price for testing dataset of Bombay Stock Exchange (2 months in advance prediction)

Table 3.5: Performance Comparison of saw tooth GA and standard GA tuned RBFN

Duration	MAPE for Saw tooth GA	MAPE for Standard GA
1-Day in Advance	1.0943	2.5696
1-Week in Advance	2.9867	3.1116
1-Month in Advance	5.6507	6.2803
2-Months in Advance	8.6971	9.0685

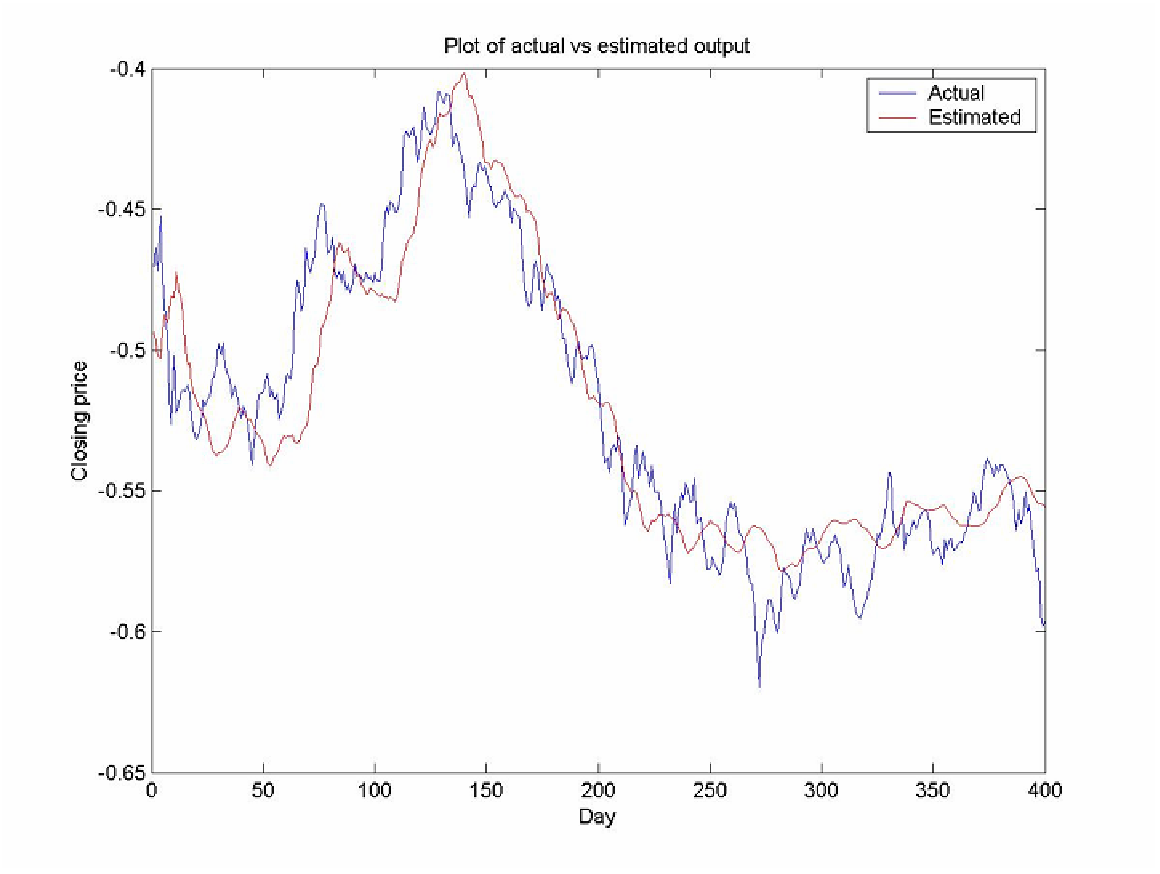


Figure 3.8 : One day in advance prediction using standard GA tuned RBF network

Table 3.6 : Comparison of saw tooth GA tuned RBFN with different number of centers

Duration	No of RBF Centers	MAPE
1-Day Ahead	3	1.9310
	5	1.9043
	7	1.9187
1-Week Ahead	3	6.4210
	5	3.6866
	7	2.9867
	9	2.9965
1-Month Ahead	3	6.7231
	5	6.1827
	7	5.6507
	9	7.4896
2-Months Ahead	3	9.8171
	5	8.7311
	7	8.6891
	9	8.5631

3.7 Discussion

The one-day in advance prediction model was implemented in a trial and error basis by selecting different combination of inputs and different number of centers so that a model giving the least value of MAPE can be obtained. It was observed that for our model, the best performance is obtained when the exponential moving average over 10, 20 and 30 days are used as inputs along with closing price. Apart from this four input model, other input combinations gave varying results, which were usually inferior to the results obtained from the aforesaid input parameters, and also took a long time (larger number of iterations) to converge to a good solution. The best MAPE obtained in this case was 1.1931% with the use of five RBF centers. It was also observed that the prediction obtained from the network was consistent over a large set of test data. For example, the training was performed using 400 data samples whereas the testing gave the MAPE of 1.1931% over a length of 500 test data.

The one week in advance predictions also gave the best results with the technical indicators EMA10, EMA20, EMA30 and closing price. The results showed a little deterioration from those obtained for one day ahead predictions. Similar tests were carried out for predicting the stock price one month and two months in advance. It can be observed that the prediction performance degrades as the time gap between the available data and the predicted data increases. The best MAPE obtained for one week in advance prediction was 2.9867% with the use of seven RBF centers. The corresponding best values for one month and two months in advance predictions were 4.356% and 8.697% respectively and were obtained using seven and nine RBFs in the respective cases.

The effect of the number of RBF centers was also studied. It was observed that as the time gap of prediction increases, more number of centers tends to give better results. But using too many centers also degrades the performance and increases computational complexity. For one day in advance predictions, five centers gave the optimum results. On increasing the centers beyond five, performance deteriorates for the same inputs and same number of iterations. For one week ahead the best output was achieved using seven RBF centers. The one month and two months in advance predictions were carried out using three, five, seven and nine centers. Among these, seven centers gave the least MAPE in case of one month ahead forecasting. However, for two months in advance predictions, the performance improved monotonically with increasing number of centers.

Finally, the performance of the network was compared with that of the RBFN tuned using standard GA. It was observed that the saw tooth model performed better in all the four cases for equal computational cost. This conforms to the earlier discussion about the superiority of the saw tooth GA over its conventional counterpart.

Chapter 4

CONCLUSION

A variant of genetic algorithm that uses the variation of population size in a saw tooth manner has been analyzed and applied to the Hammerstein model identification problem. A radial basis function network based stock market forecasting model using saw tooth GA as the update algorithm has also been proposed. It has been experimentally shown that saw tooth GA is more efficient than conventional GA using fixed population for solving multi-dimensional problems having several local optima. It more predictably reaches the global optimum with equal computational complexity and also converges much faster than standard GA. This superiority is also experimentally demonstrated for the Hammerstein model and the stock market prediction model where use of saw tooth GA for learning yields less error as compared to standard GA.

REFRENCES

1. V. K. Koumousis and C. P. Katsaras “A sawtooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance,” IEEE Transaction on Evolutionary Computation, vol. 10, pp. 19, 2006.

2. D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Mass.: Addison-Wesley, 1989.

3. J. Vesterstroem and R. Thomsen “A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems,” Proc. Congress of Evolutionary Computation, vol. 2, pp. 1980, 2004.

4. D.E Goldberg,"Real-coded genetic algorithms, virtual codes, and blocking," Univ. Illinois at Urbana Champaign, Technical Report No. 90001, 1990.

5. G. Syswerda,"Uniform crossover in genetic algorithms," Proc. Third Internationall Conference on Genetic Algorithms, pp. 2-9, 1989.

6. M. Srinivas and L. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithm," IEEE Trans. System, Man, and Cybernetics, vol. 24, no. 4, Apr. 1994.

7. T. Hachino, Katsuhisa Deguchi and Hitoshi Takata, “”Identification of Hammerstein model using radial basis function networks and genetic algorithm”, 5th Asian Control Conference, 2004.

8. T. Hatanaka, K. Uosaki and M. Koga, Evolutionary Computation Approach to Hammerstein Model Identification,Proc. of the 4th Asian Control Conference, pp.1730-1735, 2002.

9. H. Al-duwaish and M. N. Karim, A New Method for the Identification of Hammerstein Model, Automatica, Vol.33, No.10, pp.1871-1875, 1997.

10. H. Akaike, A New Look on the Statistical Model Identification, IEEE Trans. on Automatic Control, Vol.19, pp.716-723, 1974.

11. T.Z. Tan, C. Quck, G.S. Nag, "Brain Inspired Genetic Complimentary Learning for Stock Market Prediction"

12. Y. Wang –F(2002) "Predicting stock price using fuzzy grey prediction system" Expert System with Applications,22, P33-39.

13. S. Taylor, "Modeling Financial Time Series", John Wiley & Sons (1986).

14. Y-H. Pao, "Adaptive Pattern Recognition & Neural Networks", Reading, MA; Addison-Wesley, 1989.

15. Tan, H.; Prokhorov, D.V.; Wunsch, D.C., II; "Conservative thirty calendar day stock prediction using a probabilistic neural network", Computational Intelligence for Financial Engineering, 1995., Proceedings of the IEEE/IAFE 1995 9-11 April 1995

16. Lee, R.S.T.; " iJADE stock advisor: an intelligent agent based stock prediction system using hybrid RBF recurrent network" , IEEE Transactions on Systems, Man and Cybernetics, Part A, Volume 34, Issue 3, May 2004

17. Kyoung-jae Kim; "Artificial neural networks with evolutionary instance selection for financial forecasting", Expert Systems with Applications 30, 2006