M.Tech. Thesis

Evolution of Fuzzy Logic System using Genetic Algorithms and Performance Analysis for Various Number of Fuzzy Rules

Submitted in partial fulfillment of the requirements for the degree of Master of Technology in Electronics & Communication Engineering

by

Shivani Kakkar (1322788)

Under the Supervision of

Dr. Satvir Singh

PUNJAB TECHNICAL UNIVERSITY

Jalandhar-Kapurthala Highway, Jalandhar

SHAHEED BHAGAT SINGH
STATE TECHNICAL CAMPUS
Moga Road (NH-95), Ferozepur-152004 (PB) INDIA

October 2015

CERTIFICATE

I, **Shivani Kakkar (1322788)**, hereby declare that the work being presented in this thesis on Evolution of Fuzzy Logic System using Genetic Algorithms and Performance Analysis for Various Number of Fuzzy Rules is an authentic record of my own work carried out by me during my course under the supervision of Dr. Satvir Singh. This is submitted to the Department of ECE at Shaheed Bhagat Singh State Technical Campus, Ferozepur (affiliated to Punjab Technical University, Jalandhar) as partial fulfillment of requirements for award of the degree of Master of Technology in Electronics & Communication Engineering.

Shivani Kakkar (1322788)

To the best of my knowledge, this thesis has not been submitted to Punjab Technical University, Jalandhar or to any other university or institute for award of any other degree or diploma. It is further understood that by this certificate, the undersigned do/does not endorse or approve any statement made, opinion expressed or conclusion drawn therein, however, approve the thesis only for the purpose for which it is submitted.

Dr. Satvir Singh [Supervisor]

The M.Tech Viva-Voce Examination of Shivani Kakkar (1322788) is held at Department of ECE, SBS State Technical Campus, Ferozepur on .....

External Examiner
Name: .....

Dr. Sanjeev Dewra
M.Tech. Coordinator, ECE

Hardship often prepares an ordinary person for an extraordinary destiny.

- C.S. Lewis

Dedicated to

My Family & Guide

ACKNOWLEDGEMENTS

Apart from the efforts of myself, the success of Masters dissertation depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I would like to express the deepest appreciation to my supervisor, **Dr. Satvir Singh**, Associate Professor, Department of Electronics & Communication Engineering, SBS State Technical Campus, Ferozepur (Punjab), India, who has the attitude and the substance of a genius: he continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this dissertation would not have been possible. I cant say thank you enough for his tremendous support and help. I feel motivated and encouraged every time I attend his meeting. Without his encouragement and guidance this project would not have materialized.

Dr. Satvir Singh's mature research advice in the very initial stage never let me down in research throughout the research period. I could learn the technique of organizing and writing quality research matter only because of his erudite teachings throughout the project. This in fact has left a permanent impression on my personality and written and verbal communication. There are several other persons who made important contributions during this period. First and foremost, I want to express my heartfelt gratitude to **Mr. Sarabjit Singh**, Assistant Professor, Computer Science and Engineering, SBS State Technical Campus, Ferozepur (Punjab) for helping me in programming. The guidance and support received from all the members who contributed to this project, was vital for the success of the project. I am grateful for their constant support and help.

My sincere thanks to **Dr. T. S. Sidhu**, Director, SBS State Technical Campus, Ferozepur (Punjab) and to **Dr. Sanjeev Dewra**, M.tech Co-ordinator ECE, SBS State Technical

Campus, Ferozepur and to **Mrs. Rajni**, Associate Professor, ECE, SBS State Technical Campus, Ferozepur (Punjab).

I wish to acknowledge the magnificent support I have received from my senior friend **Mr. Durlabh Chauhan**, **Ms. Rashmi Sharan Sinha** and my fellow friends **Ms. Jaspreet Kaur**, **Mr. Harpreet Singh**, **Ms. Amandeep Kaur** in the form of useful discussions throughout this work.

Most profound regards to my mother **Smt. Anju Kakkar** and my father **Sh. Ashok Kakkar**, who confined their needs all for my sake. Many a times, they took loan to meet my college expenses during my course of UG and PG studies; it is these sacrifices and constant blessings that kept me motivated and committed, until I reached this end.

Finally, I must thank GOD for giving me the environment to study, people to help, opportunities to encash and potential to succeed.

Place: SBS STC Ferozepur
Date: October 19, 2015

Shivani Kakkar
(1322788)

LIST OF THESIS OUTCOMES

International Conferences Publications

1. Shivani Kakkar, Satvir Singh, Sarabjeet Singh and Vijay Kumar Banga, "Investigations on the performance of Fuzzy Logic System when evolved using Genetic Algorithm for different number of Fuzzy rules", in *Proceedings of 9th International Conference on Instrumentation, Electrical and Electronics Engineering (**ICIEEE**),* Chandigarh, India, 16 August, 2015. [Online http://googlescholar.com]

2. Shivani Kakkar and Satvir Singh, "Fuzzy Logic using GPGP", in *Proceedings of **ICCCS** International Conference on Communication, Computing and Systems (**ICCCS**),* Firozpur, India, 8-9 August, 2014. [Online http://drsatvir.com]

International Journals Publications/Submissions

1. Shivani Kakkar, Satvir Singh, Sarabjeet Singh, Jaspreet Kaur and Vijay Kumar Banga "Evolution of Fuzzy Logic System using Genetic Algorithms for time-series forecasting", in *International Journal of Current Engineering and Scientific ResearchIJCESR, Volume-2, Issue-7, Pages 97-103.* [Online http://googlescholar.com]

2. Jaspreet Kaur, Satvir Singh, Sarabjeet Singh, Shivani Kakkar and Vijay Kumar Banga "Comparitive Study of GA and PSO Algorithm", in *International Journal of Current Engineering and Scientific ResearchIJCESR, Volume-2, Issue-7, Page 91-96.* [Online http://googlescholar.com]

ABSTRACT

Most of the real world problems are NP hard in nature those are difficult to be solved with traditional mathematical approaches. Optimization is a global search technique which is applicable to almost every domain of engineering, sciences, management and social life. Traditional mathematic based search techniques are outperformed by the evolutionary algorithms. Most of the evolutionary algorithms have been developed on the basis of meta-heuristics of natural systems. These evolutionary algorithms are stochastic in nature and capable of evolving near optimal solutions in reasonable amount of time.

Fuzzy logic is another paradigm that belongs to computational intelligence and is capable of handling uncertain/imprecise data. Most of Fuzzy Logic Systems (FLSs) are developed based on two different approaches, firstly model-based approach in which objective information is represented by mathematical models and subjective information is represented by linguistics statements that then are converted into rules secondly, model-free approach in which rules are expected from numerical data and are then combined using linguistic information collected from experts. Designing an FLS is itself an NP hard problem. It contains four components Rules, Fuzzifier, Inference engine, Defuzzifier. Once rules have been established an FLS can be viewed as mapping from inputs to the output and this can be expressed quantitatively as y=f(x) where f is highly non-linear and high order differential mapping of multiple crisp inputs with that of single (multiple) crisp outputs(s). The linguistic knowledge collected from experts is represented in the form of fuzzy rules. The rules are constructed using the collection of fuzzy if/then statements depending upon the possible combinations of input and output variables. The if part of the fuzzy rule represents the antecedent and then part represent the consequent. Both antecedents and consequents are represented using fuzzy sets

(defined by membership functions). The curve of membership function defines the degree of truth for each element in the input space.

Mackey-Glass time series forecasting is a benchmark problem which is used to study the behavior of dynamically varying systems like weather and climate. The application is used to predict the value at time period t depending upon the values at previous time periods (t-1), (t-2), (t-3), (t-4). Therefore the FLS is designed with 4-inputs and 1-output value. Each input value has 4 Type-1 Gaussian membership functions. Following the different possible combinations of the input membership functions the no. of possible rules becomes 4^4=256 with each rule comprising 4 antecedent membership functions from each input i.e. total of 4×256=1024 membership functions. Further each Gaussian membership function is defined by 2 parameters, i.e., mean (m) and sigma (?). Hence we have to deal with total 2×1024=2048 number of parameters in order to design our FLS. Generating fuzzy rules is the most important requirement to design a FLS. As we does not need to use all the possible rules so for the good performance of the system we need to search the best possible rules from all the possible rules. Searching for the best possible rule set from such a high dimension search space using conventional optimizations techniques becomes highly complicated. Genetic Algorithm (GA) is an effective optimization tool inspired by the process of evolution in natural systems. It is a powerful technique to search suitable solutions to various optimization problems. So instead of using conventional methods GA is one of the most popular Evolutionary Algorithm for searching optimal solution.

The work presented in this thesis is about the methodology to evolve optimal design of FLS where fuzzy sets are designed using fuzzy c-mean clustering algorithm and fuzzy sets are evolved by GA. Initially the randomly generated rule sets are encoded as GA population of chromosomes. Fitness of each chromosome is calculated using a fitness function which is problem specific. The fitness function used for time-series forecasting problem is the Mean Square Error (MSE). MSE is calculated by comparing the forecasted outputs from the FLS with that of the actual outputs. Lesser is the value of MSE, higher is the fitness of the chromosome. In order to select the best chromosomes for the next generation Roulette wheel concept is used. Using Roulette wheel selection the chromosomes with higher fitness are more likely to be selected for the succeeding generation. After selection the crossover operator interchanges the gene values between the parent chromosomes to produce better offspring. The values of the genes are varied using mutation operators. The newly generated population is fed back to the FLS as a new improved rule set. The population again undergoes selection, crossover and mutation to produce new improved population. So, GA iteratively

improves the performance of the system by reducing the value MSE hence increasing the value of fitness level. The system was evolved for different number of rules i.e. 2, 4, 6, 8, 10, 15, 20, 25, 30. It can be observed from the simulation results that the system with 10 rules evolved with least value of MSE and hence performed better. The average value of MSE obtained was found to be 1.15410^{-2} using 10 rules in the rule set. Therefore the designed system was found to be the best for time-series forecasting with least value of MSE.

Although the designed system for evolution of the best solution is very effective for solving time-series problem, their execution time can become a limiting factor as it involves large number of parameters that are to be determined making it computationally intensive.

Place: Ferozepur — Shivani Kakkar

Date: October 19, 2015 — (1322788)

ABBREVIATIONS

Abbreviations	Description
AI	Artificial Intelligence
ANN	Artificial Neural Networks
CGA	Cellular Genetic Algorithm
CI	Computational Intelligence
CUDA	Compute Unified Design Architecture
EA	Evolutionary Algorithm
FIS	Fuzzy Inference System
FL	Fuzzy Logic
FLS	Fuzzy Logic Systems
FOU	Footprint of Uncertainty
GA	Genetic Algorithm
GPU	Graphic Processing Unit
LMF	Lower Membership Function
MGTS	Mackey Glass Time Series
MINLP	Mixed Integer Non-Linear Programming
MRAFC-GA	Model Reference Adaptive Fuzzy-GA Controller
MSE	Mean Square Error
PSO	Parical Swarm Optimization

Abbreviations	Description
RCGA	Real Coded Genetic Algorithm
SNR	Signal to Noise Ratio
TS	Tournament Selection
UMF	Upper Membership Function

NOTATIONS

Symbols	Description
p_i	Selection probability of an ith individual
m	Mean
σ	Sigma
T_s	Tournament size
f_i	Fitness value of an ith individual
μ	Membership function
$\mu(\mathbf{x})$	Clipping level of a membership funtion after implication
y_i	ith output obtained from the designed system
d	True output from the system

LIST OF FIGURES

2.1 Block diagram of GA 6

3.1 Traditional Logic 12
3.2 Fuzzy Logic Membership function representation 12
3.3 Block diagram of Type-1 FLS 13
3.4 Block diagram of Type-2 FLS 14
3.5 Block diagram of Type-2 FLS 15

4.1 Flow diagram of Genetic Algorithms 18
4.2 Tournament Selection 20
4.3 Roulette-Wheel Selection 21
4.4 one-point crossover 22
4.5 Two-point crossover 22
4.6 Uniform crossover 22

5.1 Graphical representation of Mackey Glass time series 27

6.1 GA based design flow of FLS's 31
6.2 Roulette Wheel Selection 33
6.3 Uniform Crossover 33

7.1 Minimum MSE obtained when FLS evolved for various number of rules. . . . 36
7.2 Minimization of MSE for 1000 iterations. 37
7.3 Minimization of MSE for 1000 iterations. 37
7.4 Comparison between the Actual and the Forecasted time series 38

LIST OF TABLES

7.1 Comparative MSEs with different rulebase sizes 36

CONTENTS

CERTIFICATE i

ACKNOWLEDGEMENTS v

LIST OF THESIS OUTCOMES vii

ABSTRACT viii

ABBREVIATIONS xi

NOTATIONS xiii

LIST OF FIGURES xiv

LIST OF TABLES xv

CONTENTS xvi

1 INTRODUCTION 1

1.1 Introduction . . . 1

1.1.1 Fuzzy Logic System . . . 1

1.1.2 Evolving FLS using GA . . . 2

1.2 Mackey-Glass Time-Series . . . 3

1.3 Objectives . . . 3

1.4 Methodology . . . 4

1.5 Thesis Outline . . . 4

2 LITERATURE SURVEY 5

2.1 Introduction . . . 5

2.2 Genetic Algorithms . . . 5

2.3 Variants of GA . . . 6

2.3.1 Real Coded Genetic Algorithm (RCGA) . . . 6

2.3.2 Binary Coded Genetic Algorithm (BCGA) . . . 7

2.3.3 Cellular Genetic Algorithm (CGA) . . . 7
2.3.4 Mixed Integer Non-Linear Programming (MINLP) . . . 7
2.4 Fuzzy Logic Systems . . . 8
2.4.1 Application of Type-2 FLS for Time-Series Forecasting . . . 8
2.4.2 Designing Fuzzy Logic Systems . . . 9
2.5 Evolutionary FLS using GA . . . 9
2.5.1 Implementation of Evolutionary Fuzzy Systems . . . 9
2.5.2 Fuzzy logic synthesis with GA . . . 9
2.5.3 Design of intelligent fuzzy logic controllers using GA . . . 10
2.6 Conclusion . . . 10

3 FUZZY LOGIC SYSTEMS 11
3.1 Introduction . . . 11
3.1.1 Type-1 FLS . . . 13
3.1.2 Type-2 FLS . . . 13
3.1.3 Interval Type-2 FLS . . . 15
3.2 Conclusion . . . 16

4 GENETIC ALGORITHMS 17
4.1 Introduction . . . 17
4.1.1 Initialization of Population . . . 19
4.1.2 Fitness Function . . . 19
4.1.3 GA Operators . . . 19
4.1.3.1 Selection of Parent Pool . . . 19
4.1.3.2 Crossover . . . 21
4.1.3.3 Mutation . . . 23
4.1.3.4 Elite . . . 23
4.2 Conclusion . . . 23

5 MACKEY-GLASS TIME SERIES 25
5.1 Time-Series Forecasting . . . 25
5.2 Mackey Glass Time-Series (MGTS) . . . 26
5.3 Conclusion . . . 28

6 IMPLEMENTATION 29
6.1 Introduction . . . 29
6.1.1 Genetic Algorithm (GA) . . . 29
6.1.2 Fuzzy Logic System (FLS) . . . 30
6.2 Implementation . . . 30
6.3 Conclusion . . . 34

7 RESULTS AND DISCUSSION 35
7.1 Introduction . . . 35
7.2 Experimental Setup . . . 35

7.2.1 Overall Performance Analysis 36
7.2.2 Performance Evaluation for 1000 Iterations 37
7.2.3 Performance Evaluation for 10,000 Iterations 37
7.2.4 Result Findings 37
7.3 Significant Investigation Findings 38

8 CONCLUSION AND FUTURE SCOPE **39**
8.1 Introduction 39
8.2 Concluding statements 40
8.3 Future Scope 40

REFERENCES **44**

INDEX **45**

CHAPTER 1

INTRODUCTION

Optimization is a global search technique which is applicable to almost every domain of science, management and social life. Evolutionary computation have been developed on the basis of meta-heuristics of natural systems. These are stochastic in nature and are found to evolve near optimal solutions to various optimization problem in reasonable amount of time. Genetic Algorithm (GA) is one of the amazing techniques of evolutionary computation which is based on the process of evolution in various natural systems. Fuzzy Logic Systems (FLS) which is capable of handling uncertainty/imprecision lacks self learning and generalization of rules [Cordón et al., 1996]. FLS when integrated with GA for the process of evolution of rules outperforms the traditional methods of optimization. This thesis work presents the implementation of evolving a FLS using GA for Mackey-Glass time series forecasting problem and also the performance analysis of the designed FLS is discussed for different number of IF-THEN fuzzy rules.

1.1 Introduction

1.1.1 Fuzzy Logic System

FLS introduced by Lotfi A. Zadeh [Mendel, 1995], [Zadeh, 1974] is that class of computational intelligence which provides us with a provision to deal with uncertainty or the knowledge which do not have well defined sharp boundaries. Fuzzy Logic (FL) has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. FLS [Cordón et al., 1996] employ the use of linguistics variables which

are used to define FL rules. A linguistic variable such as temperature may be represented as high or low depending upon its numeric value. Therefore FLS are highly non-linear in nature which have the capability to in fer the complex non-linear relationship between input and output variables [Mendel and Mouzouris, 1997]. FLS constitute four parts, i.e, fuzzifier, rulebase, inference engine and defuzzifier. Going through various levels of uncertainty for every crisp input we get a crisp output from an FLS. Designing an FLS is considered as a complex task where we need to decide the most optimal set of input/output pairs from the large number of possible input/output pairs in the form of rules. Traditional optimization techniques have certain limitations which can be easily outperformed by the evolutionary algorithm techniques like GA.

1.1.2 Evolving FLS using GA

GA introduced in [Holland, 1975], [Holland, 1973] is a powerful technique inspired by the Darwinian Theory of Survival of Fittest in various natural systems [De Giovanni and Pezzella, 2010]. GA is domain independent paradigm which has the capability to search near optimal solutions to arbitrary optimization problems. So, GA starts with the random initialization of population of parent chromosomes for the purpose of automatic evolution of the parent chromosomes using various GA operators like Selection, Crossover, Mutation and Elite. GA [[Sinha and Singh, 2014] is widely used as an optimization tools in various fields such as medical, engineering and finance [Kumar et al., 2012] can also be used for the purpose of automatic evolution using crossover, mutation and survival of the fittest [Sinha and Singh, 2014]. GA starts with initialization of random population consisting of vectors of chromosomes [Shi et al., 1999]. After that the fitness for each chromosome in the population is calculated using some standard fitness function. A new population is generated by applying crossover and mutation over selected chromosomes. Selection is most commonly performed using a "Roulette wheel" mechanism. The next step crossover involves the interchanging of some values of 2 parent chromosomes depending upon the crossover probability [Shi et al., 1999] is performed which stands for changing the values of the elements of the population randomly using mutation probability. The all new population is generated now which is copied back to the initial population in order to calculate the new fitness values. The new population will yield the improved values of fitness. The whole algorithm repeats until some required condition is not met. GA iterate number of times to generate the new improved population of offspring chromosomes until the terminating criteria is not met.

Evolving FLS using GA involves producing the best possible set of rules required to design an FLS. Initially the fuzzy logic rules which are to be evolved are generated randomly. Randomly generated rule sets are encoded as GA population in order to search for the best rule set from the present pool of rule sets.

1.2 Mackey-Glass Time-Series

Mackey-Glass time series is a benchmark problem which is used to predict the behavior of various dynamically varying systems like weather and climate. It is required to predict the values at time period (t) depending upon the sample values at previous time periods, i.e, $(t-4)$, $(t-3)$, $(t-2)$ and $(t-1)$. We collect 1000 such data pairs. The first 500 are used for training while the others are used for checking the performance of the system.

FLS is trained using first 500 input and output pairs. Therefore the FLS is designed with 4-inputs and 1-output value. Each input value has four Type-1 Gaussian membership functions. Following the different possible combinations of the input membership functions the no. of possible rules becomes 4^4=256 with each rule comprising 4 antecedent membership functions from each input, i.e, total of 4×256=1024 membership functions. Further each Gaussian membership function is defined by 2 parameters, i.e., mean (m) and sigma (σ). Hence we have to deal with total 2×1024=2048 number of parameters in order to design our FLS. Generating fuzzy rules is the most important requirement to design a FLS. As we does not need to use all the possible rules so for the good performance of the system we need to search the best possible rules from all the possible rules. Searching for the best possible rule set from such a high dimension search space using conventional optimizations techniques becomes highly complicated.

GA is an effective optimization tool inspired by the process of evolution in natural systems. It is a powerful technique to search suitable solutions to various optimization problems. So instead of using conventional methods GA is one of the most popular Evolutionary Algorithm for searching optimal solution.

1.3 Objectives

The primary objectives of this research work are summarized as follows:

1. To implement an FLS for 20 randomly generated different rule sets for time-series forecasting.
2. To implement GA in order to evolve the near optimal fuzzy rule set amongst 20 different rule sets depending upon fitness evaluation.
3. To implement an FLS using optimal rule set obtained using GA for time-series forecasting problem and to compare the performance of an FLS for different number of rules in the rule base.

1.4 Methodology

The methodology of this thesis is summarized below:

1. In order to evolve FLS using GA firstly we need to study deeply about FLS and GA.
2. Second requirement is to understand their various key features and strategy parameters to design an FLS and GA.
3. After brief study about both FLS and GA we need to collect the training as well as testing data for time series forecasting problem. Finally a program is designed for evolving FLS using GA to forecast the approximate outputs of the series.

1.5 Thesis Outline

After the brief introduction to M.Tech. thesis given in this chapter, Chapter 2 starts with the literature survey giving an overview of Genetic Algorithms, its variants and the brief introduction on how to design an FLS for time-series forecasting and also it presents the use of evolving FLS using GA.

Chapter 3 is devoted to FLS, literature of FLS including its types and flow diagrams. Chapter 4 is dedicated to study of GA, its literature, algorithms flow and its variants reported till date.

Chapter 5 is dedicated to detailed study of Mackey-Glass time series and its uses.

Chapter 6 devotes understanding of various design parameters of FLS and GA. Secondly, implementation flow of Evolutionary FLS using GA for time-series forecasting is discussed.

Chapter 7 represents simulation results of convergence performance of the algorithms. The best results in tabulated form are also represented in this chapter. Here, forecasted outputs obtained during simulation results are also represented for better understanding.

Lastly, conclusion and future scopes of this research are discussed in Chapter 8

CHAPTER 2

LITERATURE SURVEY

This chapter presents brief literature survey on designing FLS and various types of GA. It also covers the investigation study on evolutionary FLS using GA and the application of Type-2 FLS to the forecasting of Mackey-Glass time series.

2.1 Introduction

Evolutionary computation is that class of Artificial Intelligence (AI) which uses mechanism inspired by the process of natural evolution like reproduction, crossover, selection etc. Evolutionary Algorithm (EA) technique is used in all those fields in which we need to find the approximate solution to the problems that cannot be solved easily using other techniques. Optimization problems are included into this category. In optimization problems candidate solutions are used as individuals of the populations and the performance of quality of the solution is determined by the fitness function. Population then evolves after iterating through the above operators. Evolutionary computation is successful in finding the approximate solution to the problem in the various fields such engineering, medicine, science, robotics etc.

2.2 Genetic Algorithms

Amongst various EA techniques GA is one of the most popular EA technique. It is a search technique discovered by Holland in 1975 [Holland, 1973] which are based on the process of the natural evolution such as natural selection in various biological species [De Giovanni and

Pezzella, 2010]. GA we can randomly generate a pool of candidate solutions, i.e, elements of the function domain and apply the fitness function specified by an application for which we are evolving the system. Higher the fitness value of the particular solution better is the solution. Based on this fitness some of the better candidates are chosen to seed the next generation by applying crossover operation to them. Crossover is the operator applied to the two or more selected candidates,i.e, parent chromosomes and results one or more new candidates (the children).

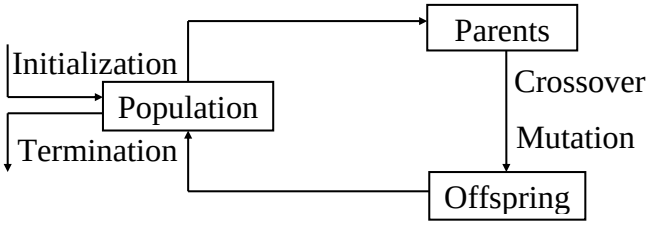


FIGURE 2.1: Block diagram of GA

Mutation is applied to one candidate and result in one new candidate, i.e, offspring. This process can be iterated until a candidate with sufficient quality is found or a previously set computational limit is reached. The block diagram depicting GA is shown in Fig 2.1.

2.3 Variants of GA

GA is an efficient Algorithm in solving problems based upon swarm optimization. Over the last few decades, many different alternatives of GA has been introduced for solving optimization problem. The comprehensive study of their performance is shown below.

2.3.1 Real Coded Genetic Algorithm (RCGA)

This section presents a theory of convergence for Real Coded Genetic Algorithms (RCGA) that use floating point or other high-cardinality codings in their chromosomes. The theory is consistent with the theory of schemata and postulates that selection dominates early GA performance and restricts subsequent search to intervals with above-average function value dimension by dimension [Goldberg, 1990]. These intervals may be further subdivided on the basis of their attraction under genetic hill climbing. Each of these subintervals is called a virtual character, and the collection of characters along a given dimension is called a virtual alphabet. It is the virtual alphabet that is searched during the recombinative phase of the GA, and in many problems this is sufficient to ensure that good solutions are found. Although the theory helps suggest why many problems have been solved using RCGA, it also suggests

that real-coded GA's can be blocked from further progress in those situations when local optima separate the virtual characters from the global optimum [Goldberg, 1990].

2.3.2 Binary Coded Genetic Algorithm (BCGA)

The Binary Coded Genetic Algorithm (BCGA) is a probabilistic search algorithm that iteratively transforms a set (called a population) of mathematical objects (typically fixed-length binary character strings), each with an associated fitness value, into a new population of off-spring objects using the Darwinian principle of natural selection and using operations that are patterned after naturally occurring genetic operations, such as crossover and mutation. Following the model of evolution, they establish a population of individuals, where each individual corresponds to a point in the search space. An objective function is applied to each individual to rate their fitness. Using well conceived operators, a next generation is formed based upon the survival of the fittest. Therefore, the evolution of individuals from generation to generation tends to result in fitter in individuals, solutions, in the search space [Bridges and Goldberg, 1987].

2.3.3 Cellular Genetic Algorithm (CGA)

Cellular Genetic Algorithms (CGAs) are a kind of EA's. CGA's are robust search algorithms inspired by the analogy of natural evolution from the point of view of the individual solution. They have demonstrated to be particularly effective optimization techniques solving many practical problems in science and engineering. The basic algorithm (CGA) shows high performance and because of its swarm intelligence structure (i.e., emergent behavior and decentralized control flow). CGA is able of keeping a high diversity in the population until reaching the region containing the global optimum [Alba and Dorronsoro, 2009].

2.3.4 Mixed Integer Non-Linear Programming (MINLP)

Mixed Integer Non-Linear Programming (MINLP) problems are the most generalized form of single-objective global optimization problems. They contain both continuous and integer decision variables, and involve non-linear objective function and constraints setting no limit to the complexity of the problems. MINLPs deals with three major parameters. 1) They involve both discrete (integer) and continuous (floating point) variables. 2) Objective function and constraints are non-linear, generating potential non-convexities. 3) They can involve active equality and inequality constraints. Many real world constrained optimization problems are modeled as MINLPs e.g. heat and mass exchange networks, batch plant design and scheduling, design of interplanetary spacecraft trajectories etc [Costa and Oliveira, 2001].

2.4 Fuzzy Logic Systems

FLS system is a non-linear mapping of an input data into output data [Mendel and Mouzouris, 1997]. It has the ability to handle problems with imprecise and incomplete data also it can handle numerical as well as linguistic knowledge simultaneously. Unlike standard traditional logic that deals with only 0 and 1 FLS deals with degree of truthfulness nd falsefulness of a particular element. Fuzzy Inference System (FIS) consists of a number of conditional if-then statements called rules, e.g., if "weather is cold" then "heat is on" We can supply as many rules as necessary to describe a system. FLS rely on membership functions to carry out computations. Therefore in order to deal with uncertainty we deal with Type-1 FLS but there are some cases where things mean different to different people and in those cases we have to deal with uncertainty over uncertainty, e.g., if few people says that a person with height above 5 feet is considered as tall and others say that a person above 6 is considered as tall. So there are mixed views about the height of a person. Therefore their is an uncertainty in the boundaries of the membership function for tall. So, blurred boundaries of a membership function depict a Type-2 Fuzzy set. In all those cases where their is ambiguity over the boundaries or location of fuzzy sets are known as Type-2 FLS. Unclear boundaries refers to a noise in the fuzzified inputs. So, in 1999 Karnik and Mendel [Karnik and Mendel, 1999] presented an approach to forecast time-series by incorporating information about noise strength into Type-2 FLS.

2.4.1 Application of Type-2 FLS for Time-Series Forecasting

In 1999 Karnik and Mendel [Karnik and Mendel, 1999] trained type-1 FLS with noisy data, and demonstrated how the uncertainty introduced in the FLS due to this noise can be modeled using type-2 fuzzy sets. They showed that an interval type-2 FLS can be used to obtain bounds on the output, as well as a better crisp prediction, by tuning its parameters. The former serves as a sort of confidence interval and cannot be obtained using a type-1 FLS, regardless of how many rules one uses in that system. Firstly Type-1 FLS was designed and then type-2 FLS was created from it by using information available about the noise in the training data. Performance of the system was evaluated using Mean Square Error (MSE) obtained between the crisp outputs obtained and the desired outputs of the time-series. Minimum value of MSE obtained from this system was 0.0134 for 0db Signal to Noise Ratio (SNR) using this system. There can, however, be other design approaches. For example, they could start with a type-2 system, and tune its parameters directly using the training data which becomes the future work of this paper.

2.4.2 Designing Fuzzy Logic Systems

In 1997 J.M. Mendel and George C. Mouzouris [Mendel and Mouzouris, 1997] presented a formulation of FLS that can be used to construct non-parametric models of non-linear processes, given only input-output data. The system can learn non-linear mapping by being presented a sequence of input signal and desired response pairs, which are used in conjunction with some optimization algorithm to determine the values of the system parameters. Given a set of input-output pairs, the task of learning is essentially equivalent to determining a system that provides an optimal fit to input-output pairs with respect to the cost function or fitness function. The fitness function used in this case was MSE. In this work several design methods with different properties and features were discussed and their performances were compared using an example on the predictive modeling of a non-linear chaotic system.

2.5 Evolutionary FLS using GA

Fuzzy Systems are being used successfully in an increasing number of application area. They use linguistic rules to decide the system. One of the most important requirement in designing any fuzzy system is the generation of the fuzzy rules. Evolutionary fuzzy systems are those systems in which fuzzy rule sets including the number of rules inside a rule set are evolved using EA [Shi et al., 1999].

2.5.1 Implementation of Evolutionary Fuzzy Systems

In 1999 Yuhui Shi and Russel Eberhart [Shi et al., 1999] implemented an evolutionary fuzzy system in which the fuzzy rule sets including the number of fuzzy rules inside a rule set are evolved using GA. The method described appeared to be useful for a wide range of classification and diagnostic problems.

2.5.2 Fuzzy logic synthesis with GA

Thrift considers the application of a genetic-based learning algorithm to systems based on Fuzzy Logic. In this paper, Fuzzy Logic controller has been designed. Discrete nature of fuzzy strategies make them prime candidate for discovery by genetic algorithms. GA is employed here to create fuzzy rules. The application using FLS here was found to experience acceleration so the usefulness of GA approach as the system becomes more complex could become more apparent [Thrift, 1991].

2.5.3 Design of intelligent fuzzy logic controllers using GA

Hwang and Thompson presents a methodology for combining genetic algorithms and fuzzy algorithms for learning the optimal rules for a FLS. With the aid of genetic algorithms, optimal rules of fuzzy logic controllers can be designed without human operators' experience and/or control engineers knowledge [Hwang and Thompson, 1994]. The approach presented here maintains the shape of membership functions and searches the optimal control rules based on a fitness value which is defined in terms of a performance criterion. Applications of the method to a Fuzzy Logic Controller using Genetic Algorithm (FLS-GA) and a model reference adaptive fuzzy-GA controller (MRAFC) are presented to illustrate the effectiveness of the design procedure [Hwang and Thompson, 1994].

2.6 Conclusion

In this chapter, we present the different methodologies to design FLS based applications. Also we present the application of Type-2 FLS for time-series forecasting problem which is a benchmark problem. It was found that the evolution of an FLS using GA proves to be a very effective in order to find out the best optimal solution to the problem. The application using FLS was found to experience acceleration when evolved using GA so the usefulness of GA approach increases as the system becomes more complex. Therefore, in this work we have implemented the concept of evolving Type-1 FLS using GA for time-series forecasting problem. We will discuss briefly about FLS and its types in the next chapter.

CHAPTER 3

FUZZY LOGIC SYSTEMS

FLS is required to map input space to output space using FL. When we talk about uncertainty we usually deal with FL. So, FL is a kind of knowledge which helps us to deal with uncertainty. Traditional logic which deals with either true or false statements. But in the real world problems not every decision is either true or false. So, in order to deal with the real world problems we deal with FL.

3.1 Introduction

FLS introduced by Lotfi. A. Zadeh [Mendel, 1995], [Zadeh, 1974] in order to provide us with a provision to deal with uncertainty or the knowledge which do not have well defined sharp boundaries. In FLS, fuzzy sets are characterized by membership functions mapped between [0,1]. FL provides us with membership functions or we can say truthfulness or falsehoods, i.e, up to what degree a particular value is true or false for a particular fuzzy set. Most words and evaluations we use in our daily reasoning are not clearly defined in a mathematical manner. So we require FL to deal with the real world problems. Fig. 3.1 clearly differentiate between the traditional logic and FL.

As shown in Fig. 3.2 according to traditional logic a person with height 5 feet and above is considered as tall. There is an abrupt change in the degree of membership, i.e, it is either 0 or 1. There is no value lying between 0 and 1 but unlike traditional logic as shown in case of FL the membership degree for short height and long height changes gradually depending upon the value of the height. So, there lies certain degree of uncertainty in case of FL. The curve of membership function defines the degree of truth for each element in the input space.

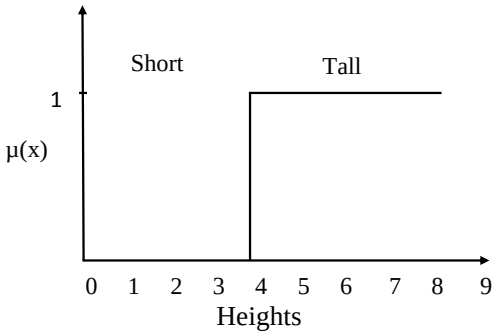


FIGURE 3.1: Traditional Logic

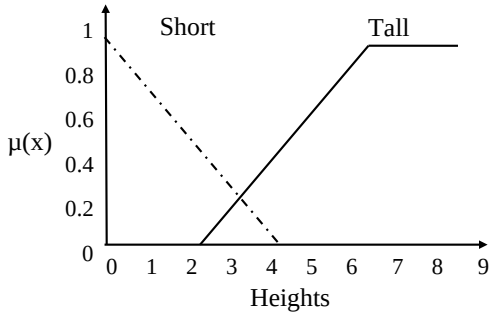


FIGURE 3.2: Fuzzy Logic Membership function representation

Most of the FLS are developed based on two different approaches, firstly model-based approach in which objective information is represented by mathematical models and subjective information represented by linguistic statements that then are converted into rules. Secondly, model-free approach in which rules are expected from numerical data and are then combined using linguistic information collected from experts. Designing an FLS is itself an NP hard problem. The linguistic knowledge collected from experts is represented in the form of fuzzy rules. The rules are constructed using the collection of fuzzy IF-THEN statements depending upon the possible combinations of input and output variables. For example "if process is too hot and process is heating rapidly" "then the command is cool the process quickly". Here the if part of the fuzzy rules represents an antecedent and then part represents the consequent. Both antecedent and consequent are represented using fuzzy sets defined by membership functions. Different types of membership functions are Gaussian, triangular, trapezoidal, piecewise linear and singleton where the choice of membership function depend upon the type of application we are using. Once rules have been established an FLS can be viewed as mapping from inputs to the output and this can be expressed quantitatively as $y = f(x)$ where f is highly non-linear and high order differential mapping of multiple crisp inputs with that of single (multiple) crisp outputs(s).

3.1.1 Type-1 FLS

The basic architecture of type one FLS is shown in Fig. 3.3:

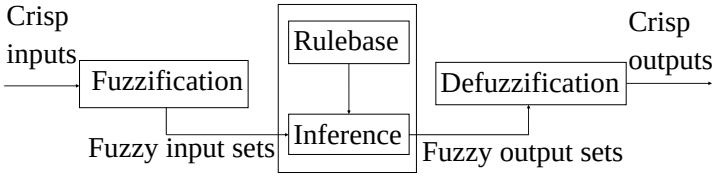


FIGURE 3.3: Block diagram of Type-1 FLS

The different steps involved in designing of an FLS are explained below:

1. *Fuzzification*: Fuzzification is the process of fuzzifying the inputs that means various input values are provided with their particular degree of membership in the corresponding membership function of their fuzzy sets.

2. *Rulebase*: The complete set of rules of a single FLS is called as Rulebase.

3. *Inference Engine*: In the process of Inference truth value for each rule is computed and applied to the consequent part of each rule. If the antecedent partially true in accordance to the input provided then the output fuzzy set is truncated using implication operator. The inputs to the aggregation process are list of truncated output functions returned by the implication process for each rule. All the truncated fuzzy sets assigned to each output variables are combined together to form a single fuzzy set using aggregation operator.

4. *Defuzzification*: After performing inference on the set of if-then rules the resulting fuzzified output set is used to obtain single crisp output using defuzzification. Various defuzzification techniques are used like height defuzzification, center of gravity, etc.

Therefore, Type-1 FLS does not have the ability to handle uncertainty over uncertainty. In order to deal with such kind of uncertainty another type of FLS known as Type-2 FLS are used.

3.1.2 Type-2 FLS

Fuzzy sets models words that are being used in rulebase and inference engine. However, word mean different thing to different people and, therefore, are uncertain. Membership degree of a Type-1 fuzzy set cannot capture uncertainties about the words. Hence, another type of fuzzy set, i.e., Type-2 fuzzy sets came into existence which is capable of handling such

uncertainties. For such a fuzzy set membership value corresponding to some crisp input is not a crisp value rather a Type-1 fuzzy set called secondary membership [Karnik and Mendel, 2001b], [Singh and Kakkar, 2014], [Castillo and Melin, 2008]. This concept can be extended to Type-n fuzzy sets. Computations based on Type-2 fuzzy sets are very intensive, however, when secondary membership is assumed unity the computational burden reduces drastically. This is another variant to fuzzy set representation and is known as Interval Type 2 fuzzy sets [Singh and Kakkar, 2014], [Singh and Kakkar, 2014], [Castillo and Melin, 2008].

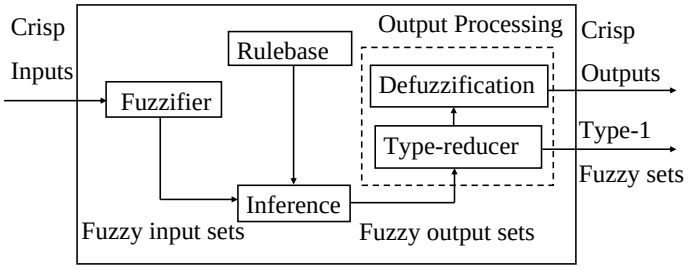


FIGURE 3.4: Block diagram of Type-2 FLS

T2 FLS are an extension of T1 FLS in which uncertainty is represented by an additional dimension. This ancillary third dimension in T2 FLS gives more degrees of freedom for better representation of uncertainty compared to T1 fuzzy sets. T2 FSs are useful in circumstances where it is difficult to determine the exact membership function for a FS. Using T2 FLS provides the capability of handling a higher level of uncertainty and provides a number of missing components that have held back successful deployment of fuzzy systems in human decision making. A T2 FLS includes fuzzifier, rule base, fuzzy inference engine, and output processor as shown in Fig. The output processor includes type-reducer and defuzzifier which generates a T1 FS output (from the type-reducer) or a crisp number (from the defuzzifier). A T2 FLS is characterized using T2 FSs for antecedents and/or consequents and IF-THEN rules. Block diagram of Type-2 FLS is shown in Fig.3.4. It can be explained as below:

1. *Fuzzification*: As shown in figure crisp inputs are first transformed into fuzzy sets in the fuzzifier block because it is fuzzy sets and not numbers that activate the rules. Fuzzy sets obtained in this case are Type-2 Fuzzy sets that are three-dimensional.

2. *Inference Engine*: After measurements are fuzzified, the resulting input fuzzy sets are mapped into fuzzy output sets by the Inference block. This is accomplished by first quantifying each rule using fuzzy set theory, and by then using the mathematics of fuzzy sets to establish the output of each rule, with the help of an inference mechanism. If there are M rules then the fuzzy input sets to the Inference block will activate only a subset of those rules, where the subset contains at least one rule and usually way fewer

than M rules. Inference is done one rule at a time. So, at the output of the Inference block, there will be one or more fired-rule fuzzy output sets.

3. *Outout Processing*: The fired-rule output fuzzy sets have to be converted into a number, and this is done in the Fig. 3.4 Output Processing block. Output processing block consist of two parts, i.e, Type-reduction part where type-2 fuzzy set is reduced to type-1 fuzzy set. There are as many type-reduction methods as there are type-1 defuzzification methods. An algorithm developed by Karnik and Mendel [Karnik and Mendel, 2001a], [Liang and Mendel, 2000] now known as the KM Algorithm is used for type-reduction. Although this algorithm is iterative, it is very fast. The second step of Output Processing, which occurs after type-reduction, is called defuzzification which is used to obtain crisp output from the fuzzified output.

3.1.3 Interval Type-2 FLS

Generalized T2 FLSs are computationally more intensive as compared to T1 FLS as former includes Fuzzy sets those are 3-dimensional in nature. Things do simplify when secondary membership functions are considered as interval sets, i.e., the secondary membership values are either 0 or 1 and set are referred as IT2 FSs or simply IT2 FSs. IT2 FSs have received the most investigational interests as they involve mathematics that is simpler than that of generalized T2 FSs. Therefore, literature available about IT2 FSs is more as compared to that of generalized T2 FSs. Now a days, both kinds of fuzzy sets are being actively investigated by an ever-growing number of researchers around the world. IT2 FSs have widely been accepted as they provide more freedom degree in modeling higher orders of uncertainty than T1 FSs. This property has been the driving force behind more of the advancements in theories and applications of IT2 FSs and FLSs.

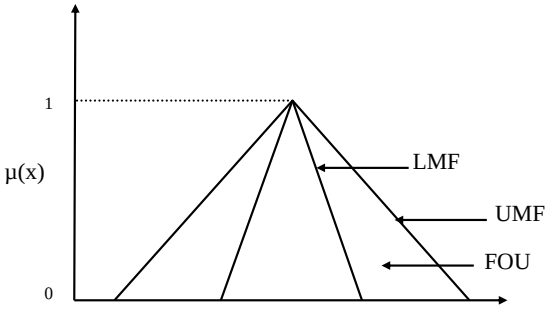


FIGURE 3.5: Block diagram of Type-2 FLS

IT2 FSs are represented by upper and lower bounds of uncertainty called Upper Membership Function (UMF) and Lower Membership Function (LMF) as shown in Fig. 3.5. The region between upper and lower bounds of Uncertainty is termed as Footprint of uncertainty (FOU).

3.2 Conclusion

Fuzzy Logic was conceived as a better method for sorting and handling data but has proven to be a excellent choice for many control system applications since it mimics human control logic. It can be built into anything from small, hand-held products to large computerized process control systems. It uses an imprecise but very descriptive language to deal with input data more like a human operator. It is very robust and forgiving of operator and data input and often works when first implemented with little or no tuning. After giving the brief explanation regarding FLS the next chapter deals with the GA.

CHAPTER 4

GENETIC ALGORITHMS

This chapter introduces GA and there use for optimization. In the field of artificial intelligence, GA is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a meta-heuristic) is routinely used to generate useful solutions to optimization and search problems [Lin and Mitchell, 2005]. GA belong to the larger class of EA, which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

4.1 Introduction

GA were formally introduced in the United States in the 1970s by John Holland at University of Michigan [Holland, 1973]. The continuing performance improvements of computational systems has made them attractive for many types of optimization. In particular, GA work very well on mixed (continuous and discrete), combinatorial problems. They are less susceptible to getting stuck at local optima than gradient search methods, however, they tend to be computationally intensive. Fig. 4.1. shows the detailed flow diagram of GA.

GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles. For example, in a problem such as the traveling salesman problem, a chromosome represents a route, and a gene may represent a city. In contrast to traditional

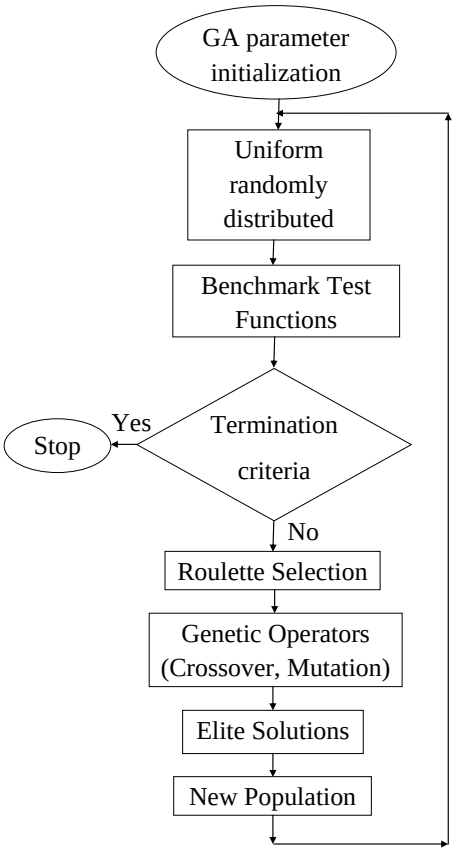


FIGURE 4.1: Flow diagram of Genetic Algorithms

optimization techniques, GAs work with coding of parameters, rather than the parameters themselves. To evolve good solutions and to implement natural selection, we need a measure for distinguishing good solutions from bad solutions. The measure could be an objective function that is a mathematical model or a computer simulation, or it can be a subjective function where humans choose better solutions over worse ones. In essence, the fitness measure must determine a candidate solutions relative fitness, which will subsequently be used by the GA to guide the evolution of good solutions. Another important concept of GAs is the notion of population. Unlike traditional search methods, genetic algorithms rely on a population of candidate solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms. For example, small population sizes might lead to premature convergence and yield substandard solutions. On the other hand, large population sizes lead to unnecessary expenditure of valuable computational time. Once the problem is encoded in a chromosomal manner and a fitness measure for discriminating good solutions from bad ones has been chosen, we can start to evolve solutions to the search problem using the following steps:

4.1.1 Initialization of Population

GA starts with initialization of randomly generated population consisting of vectors of chromosomes [Shi et al., 1999]. The initial population of candidate solutions is usually generated randomly across the search space. It is commonly done by seeding the population with random values. However, domain-specific knowledge or other information can be easily incorporated.

The population size depends on the nature of the problem, however, typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

4.1.2 Fitness Function

Fitness of each chromosome is calculated using a fitness function which is problem specific. Fitness function is a particular type of objective function that is used to find the figure of merit, i.e, how close a given design solution is to achieving a set of aims. The calculation of fitness values is conceptually simple. It can, however, be quite complex to implement in a way that optimize the efficiency of GAs search of the problem space. It is the fitness that guides the search of the problem space.

Fitness value is proportional to the performance measurement of the function being optimized. In each generation, the priority of the genetic strings is ranked according to the fitness values calculated based on a fitness function. Higher the fitness of the chromosome, better is its performance. Through either maximizing or minimizing the fitness values generation by generation, the genetic string with global optimum could be found.

4.1.3 GA Operators

Various GA operators like selection, crossover, mutation operators are discussed in brief in the following subsections:

4.1.3.1 Selection of Parent Pool

Selection is the stage of a GA in which individual genomes are chosen from a population for later breeding. A new population is formed by selecting from members of the current

population using stochastic process that is weighted by each their fitness value. The higher the fitness, the more likely it is that the chromosome will be selected for the new generation. Various selection techniques used in GAs are discussed as below:

1. ***Tournament Selection***

   Tournament Selection (TS) is probably the most popular selection method in GA [Goldberg and Deb, 1991]. In TS n individuals are selected randomly from the large population, and the selected individuals compete against each other. The individual with the highest fitness wins and will be included as one of the next generation of population. The no of individuals competing in each tournament is referred to as tournament size, commonly set to 2. This technique also gives us a chance to all individuals to be selected and thus is preserves diversity [Goldberg and Deb, 1991], [Noraini and Geraghty, 2011]. The mechanism of TS is shown in Fig. 4.2.

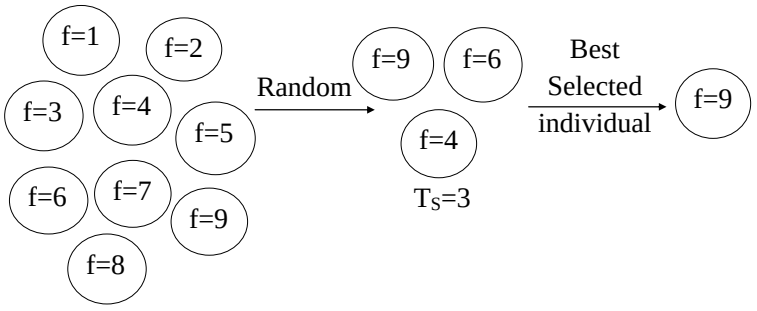
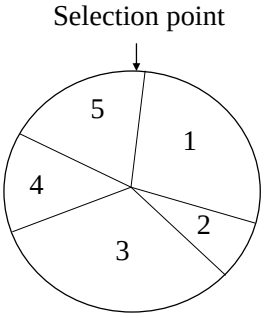


FIGURE 4.2: Tournament Selection

   In the above figure, f(1), f(2)....., f(9) are the individuals from which selection has to be made, the tournament size (T_s) is set to 3, which means 3 chromosomes are competing each other for being selected as one of the individual of the next generation. Only the best chromosomes among them is selected to reproduce.

2. ***Roulette Wheel Selection***

   This selection method is called the fitness proportionate selection and it is the most commonly used selection technique. In this method individuals are selected with a probability that is directly proportional to their fitness value, i.e, an individual selection corresponds to a portion of a roulette wheel. The probability of selecting a parent can be seen as spinning a roulette wheel with a size of the segment for each parent being proportional to its fitness. Those with a largest fitness have more probability of being chosen [Noraini and Geraghty, 2011]. The fittest individual occupies the largest segment, where as the least fit have smaller segment. When the wheel is spun the wheel will finally stop and the pointer attached to it will point on one of the segment, most



Population	Fitness
1	25
2	5
3	40
4	10
5	20

FIGURE 4.3: Roulette-Wheel Selection

probably on one of the widest one. Each time the wheel is spun the better individuals will be chosen more often then the poorer ones, thus fulfilling the requirement of survival of fittest. The roulette wheel selection mechanism is depicted in Fig. 4.3.

The selection probability (P_i) for individual is defined as:

$$P_i = \frac{f_i}{\sum_{i=1}^{n} f_i} \tag{4.1}$$

where f_1, f_2, f_3,.....f_n are the fitness values of the individuals 1,2,3.....n.

3. ***Rank Selection***

   Rank-based roulette wheel selection is the selection strategy where the probability of a chromosome being selected is based on its fitness rank relative to the entire population. Rank-based selection schemes first sort individuals in the population according to their fitness and then computes selection probability according to their ranks rather then fitness values. Hence rank based selection can maintain a constant pressure in the evolutionary search where it introduces a uniform scaling across the population and is not influenced by super individuals or the spreading of fitness values at all as in roulette wheel selection. The performance of the selection scheme depends greatly on the mapping function [Whitley et al., 1989], [Noraini and Geraghty, 2011].

4.1.3.2 Crossover

Crossover is a process of taking more than one parent solutions producing a child solution from them. This operator is used to merge the genetic information of two individuals, i.e, using this operator we are able to exchange the genetic information between the two parent chromosomes randomly chosen in order to produce better offspring [Gwiazda, 2006], [Spears et al., 1992].

Various crossover techniques are discussed below:

1. ***One-point Crossover***

   In case of one point crossover a single crossover point on both the parent chromosome string is selected. All the data beyond that point in either of the parent strings is swapped between two parent chromosomes. The resulting chromosomes are the children. One-point crossover is shown in Fig. 4.4:

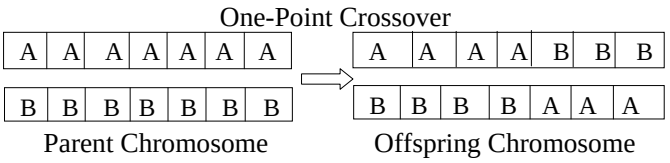


FIGURE 4.4: one-point crossover

2. ***Two-point crossover***

   Two-point crossover calls for 2-points to be selected on the parent chromosome strings. Everything between the two points is swapped between the parent organisms, rendering two children chromosomes. Two-point crossover is shown in Fig. 4.5:

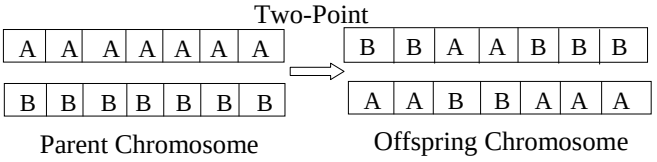


FIGURE 4.5: Two-point crossover

3. ***Uniform Crossover***

   Uniform Crossover uses a fixed mixing ratio between two parents. Unlike one- and two-point crossover, the uniform crossover enables the parent chromosomes to contribute the gene level rather then the segment level. If the mixing ratio is 0.5, the offspring has approximately half of the genes from first parent and the other half from second parent. Uniform crossover is shown in Fig. 4.6:

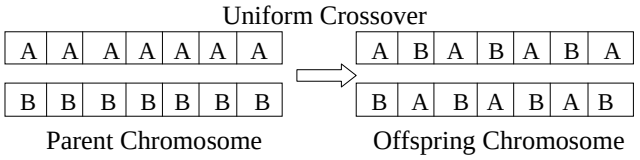


FIGURE 4.6: Uniform crossover

4.1.3.3 Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of GA chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation. Mutation occurs during evolution according to a user definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. This mutation procedure, based on the biological point mutation, is called single point mutation. Other types are inversion and floating point mutation. When the gene encoding is restrictive as in permutation problems, mutations are swaps, inversions, and scrambles [Spears et al., 1992].

The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This reasoning also explains the fact that most GA systems avoid only taking the fittest of the population in generating the next but rather a random selection with a weighting toward those that are fitter [Spears et al., 1992].

4.1.3.4 Elite

When creating new population by crossover and mutation, we have a big chance, that we will loose the best chromosome. Retaining the best individuals in a generation unchanged in the next generation, is called elitism or elitist selection. It is a successful variant of the general process of constructing a new population.

4.2 Conclusion

GAs are used for a number of different application areas. An example of this would be multidimensional optimization problems in which the character string of the chromosome can be used to encode the values for the different parameters being optimized.

In practice, therefore, we can implement this genetic model of computation by having arrays of bits or characters to represent the chromosomes. Simple bit manipulation operations allow the implementation of crossover, mutation and other operations. Although a substantial amount of research has been performed on variable length strings and other structures, the majority of work with GA is focussed on fixed-length character strings. We should focus on both this aspect of fixed-length and the need to encode the representation of the solution being sought as a character string, since these are crucial aspects that distinguish genetic programming, which does not have a fixed length representation and there is typically no encoding of the problem. After briefly discussing about the FLS and GA the next chapter is about the application for which we are using FLS and GA. The next chapter gives the brief introduction about the time-series forecasting problem.

CHAPTER 5

MACKEY-GLASS TIME SERIES

The emphasis in this chapter is on time series analysis and forecasting. This chapter deals with the use of data to forecast future events.

5.1 Time-Series Forecasting

A time series is a sequence of data points, typically consisting of successive measurements made over a time interval. Examples of time series are ocean tides, counts of sunspots etc. Time series are used in statistics, signal processing, pattern recognition, econometrics, mathematical finance, weather forecasting, intelligent transport and trajectory forecasting, earthquake prediction, electroencephalography, control engineering, astronomy, communications engineering, and largely in any domain of applied science and engineering which involves temporal measurements [Hamilton, 1994]. Time series forecasting is the use of a model to predict future values based on previously observed values. Time series data have a natural temporal ordering. This makes time series analysis distinct from cross-sectional studies, in which there is no natural ordering of the observations (e.g., explaining people's wages by reference to their respective education levels, where the individuals data could be entered in any order) [Lin et al., 2003].

Time series analysis is also distinct from spatial data analysis where the observations typically relate to geographical locations (e.g. accounting for house prices by the location as well as the intrinsic characteristics of the houses). A stochastic model for a time series will generally reflect the fact that observations close together in time will be more closely related than observations further apart. In addition, time series models will often make use of the natural

one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values (see time reversibility). A time series is a collection of data recorded over a period of time-weekly, monthly, quarterly, or yearly. An analysis of history-a time series-can be used by management to make current decisions and plans based on long-term forecasting. We usually assume past patterns will continue into the future. Long-term forecasts extend more than 1 year into the future; 5-, 10-, 15-, and 20-year projections are common [Lin et al., 2003], [Hamilton, 1994].

5.2 Mackey Glass Time-Series (MGTS)

Chaos theory is the field of study in mathematics that studies the behavior of dynamical systems that are highly sensitive to initial conditionsa response popularly referred to as the butterfly effect. Small differences in initial conditions (such as those due to rounding errors in numerical computation) yield widely diverging outcomes for such dynamical systems, rendering long-term prediction impossible in general [Kellert, 1994]. This happens even though these systems are deterministic, meaning that their future behavior is fully determined by their initial conditions, with no random elements involved [Kellert, 1994]. In other words, the deterministic nature of these systems does not make them predictable [Kellert, 1994], [Werndl, 2009]. This behavior is known as deterministic chaos, or simply chaos. The theory was summarized by Edward Lorenz as: [Danforth, 2013] Chaos: When the present determines the future, but the approximate present does not approximately determine the future.

Chaotic behavior exists in many natural systems, such as weather and climate [Casdagli, 1989], [Lorenz, 1963], [Ivancevic and Ivancevic, 2008]. This behavior can be studied through analysis of a chaotic mathematical model, or through analytical techniques such as recurrence plots and Poincar maps. Chaos theory has applications in several disciplines, including meteorology, sociology, physics, engineering, economics, biology, and philosophy.

Chaos theory concerns deterministic systems whose behavior can in principle be predicted. Chaotic systems are predictable for a while and then appear to become random. The amount of time for which the behavior of a chaotic system can be effectively predicted depends on three things: 1) How much uncertainty we are willing to tolerate in the forecast. 2) How accurately we are able to measure its current state. 3) A time scale depending on the dynamics of the system. Chaos is based on the principle that simple deterministic laws can exhibit complex external behavior; although it is quite difficult to reveal such simple laws from the system's external behavior only. This is mainly due to the dissipation and sensitivity to initial conditions properties and structure parameters of the chaotic systems [Wei, 2002]. However, most time series of practical relevance are of nonlinear and chaotic

nature which makes conventional, linear prediction methods inapplicable. Hence, a number of nonlinear prediction methods have been developed [Mayer et al., 1999].

We suppose that know the time series be MGTS and want to reach accurate prediction from time series system. At first, equations and properties of MGTS is investigated. Then the unknown parameters of time series structure by GA have been estimated. MGTS also known as Chaotic series is basically a benchmark problem required for time series forecasting in order to study the behavior of various dynamic systems like (weather and climate). So, MGTS is required to study the future behavior of the system depending upon the present behavior of the system i.e. present determine the future. MGTS can be represented by the figure. MGTS is generated using delay differential equation (5.1)

$$\frac{dx(t)}{dt} = \frac{0.2x\big(t-\tau\big)}{1+x^{10}\big(t-\tau\big)} - 0.1x\big(t\big) \tag{5.1}$$

The application is used to predict the values at time (t) depending upon the values at previous time periods $(t-1)$, $(t-2)$, $(t-3)$, $(t-4)$. The graph of MGTS is shown in Fig. 5.1.

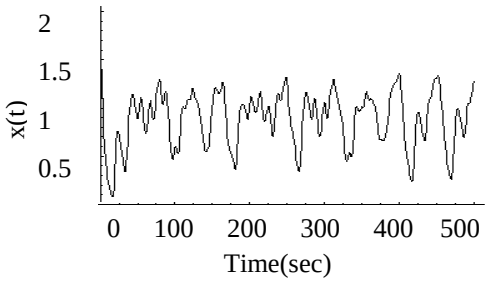


FIGURE 5.1: Graphical representation of Mackey Glass time series

Firstly, the system is trained using first 500 inputs obtained from time-series data and then the system is tested for next 500 inputs. The chaotic nature of MGTS makes its prediction uncertain because of the noise in the inputs available from the time-series data. In 1999 Karnik et.al. and Mendel et.al. [Karnik and Mendel, 1999] presented approach to forecast time series by incorporating information about noise strength into Type-2 FLS. Using Type-2 FLS the bounds on the outputs was obtained within which true output is likely to lie. Artificial Neural Networks (ANN) and Polynomials are two methods for global modeling. But they both cannot give simple and elegant model representations. They are less powerful in revealing the system dynamic laws and are difficult to be integrated with the pre-discovered knowledge on chaotic systems. But some other methods such as genetic algorithm, Dynamic Programming, and Swarm Optimization are also used. These methods are based on some prior knowledge of chaotic systems.

The FLS is designed with four antecedent and one consequent, where each antecedent has four Gaussian shaped fuzzy sets. Further each Gaussian membership function is defined by 2 parameters, i.e., mean (m) and sigma (σ). Hence we have to deal with total $2 \times 4 \times 1024 = 2048$ number of parameters in order to design our FLS. Generating fuzzy rules is the most important requirement to design a FLS. As we does not need to use all the possible rules so for the good performance of the system we need to search the best possible rules from all the possible rules. Searching for the best possible rule set from such a high dimension search space using conventional optimizations techniques becomes highly complicated. GA is an effective optimization tool inspired by the process of evolution in natural systems. It is a powerful technique to search suitable solutions to various optimization problems. So instead of using conventional methods GA is one of the most popular Evolutionary Algorithm for searching optimal solution. The work presented in this thesis is about the methodology to evolve optimal design of FLS where fuzzy sets are designed using fuzzy c-mean clustering algorithm and fuzzy sets are evolved by GA for MGTS forecasting.

5.3 Conclusion

In this thesis, Fuzzy System has been developed for forecasting the given time series. GA is suitable for evolving a fuzzy system with known structure. Various possible future outputs can be predicted with the help of the designed fuzzy system. Next chapter discuses the methodology for implementing the designed process.

CHAPTER 6

IMPLEMENTATION

This chapter presents method of evolve FLS and GA to solve the problem of forecasting the Mackey-Glass chaotic time series. In this chapter the detailed methodology to evolve FL rules automatically using GA has been discussed also the methodology employed for the minimization of the fitness function, i.e., MSE has been discussed and at last the best rulebase corresponding to the least MSE has been fixed into the designed FLS in order to get the output closer to the desired output.

6.1 Introduction

6.1.1 Genetic Algorithm (GA)

GA developed by John Holland in 1970 [Holland, 1973] is a global search algorithm inspired by natural mechanism of genetical improvement in biological species [De Giovanni and Pezzella, 2010], described by Darwinian Theory of Survival of Fittest [Sinha and Singh, 2014]. GA is widely used as an optimization tools in various fields such as medical, engineering and finance [Kumar et al., 2012] and can also be used for the purpose of automatic evolution using crossover, mutation and survival of the fittest [Sinha and Singh, 2014]. GA is widely used as an optimization tools in various fields such as medical, engineering and finance and can also be used for the purpose of automatic evolution using crossover, mutation and survival of the fittest [Kumar et al., 2012]. GA starts with initialization of randomly generated population consisting of vectors of chromosomes [Shi et al., 1999]. After that the fitness for each chromosome in the population is calculated using some standard fitness function. A

new population is generated by applying crossover and mutation over selected chromosomes. Selection is most commonly performed using a "Roulette wheel" mechanism. The next step crossover involves the interchanging of some values of 2 parent chromosomes depending upon the crossover probability [Shi et al., 1999], [Bastian and Hayashi, 1995]. Finally mutation is performed which stands for changing the values of the elements of the population randomly using mutation probability. The all new population is generated now which is copied back to the initial population in order to calculate the new fitness values. The new population will yield the improved values of fitness. The whole algorithm repeats until some required condition is not met.

6.1.2 Fuzzy Logic System (FLS)

FLS introduced by Lotfi. A. Zadeh [Mendel, 1995], [Zadeh, 1974] was discovered in order to provide us with a provision to deal with uncertainty or the knowledge which do not have well defined sharp boundaries. In FLS, fuzzy sets are characterized by membership functions mapped between [0,1] [Dubois, 1980]. For example temperature can be represented in the form of different fuzzy sets depending upon its range such as too cold, cold, warm, hot, too hot [Singh and Kakkar, 2014]. Different types of membership functions are Gaussian, triangular, trapezoidal, piecewise linear and singleton where the choice of membership function depend upon the type of application we are using. Every Fuzzy Logic System use a set of If-then rules called Rulebase where if part contain a condition and then part contain a conclusion. For example if the temperature is hot then the command is cool. Inference is performed by evaluating and combining various fuzzy rules using fuzzy set operations in order to get the output fuzzy set from which a single crisp output is obtained using defuzzification.

6.2 Implementation

GA have demonstrated to be a robust and very powerful tool to perform tasks such as the generation of fuzzy rule base, optimization of fuzzy rule bases Cordón et al. [2001]. All these tasks can be considered as optimization or search processes within large solution spaces Bastian and Hayashi [1995], [Yuan and Zhuang, 1996]. The implementation involves the methodology to evolve optimal design of FLS where fuzzy sets are designed using fuzzy c-mean clustering algorithm and fuzzy sets are evolved by GA. The task of evolution involves random generation of fuzzy rule base as well as optimization of fuzzy rule base from a large search space. The detailed flowchart is shown in Fig.6.1. that depicts the evolutionary process flow of an FLS using GA.

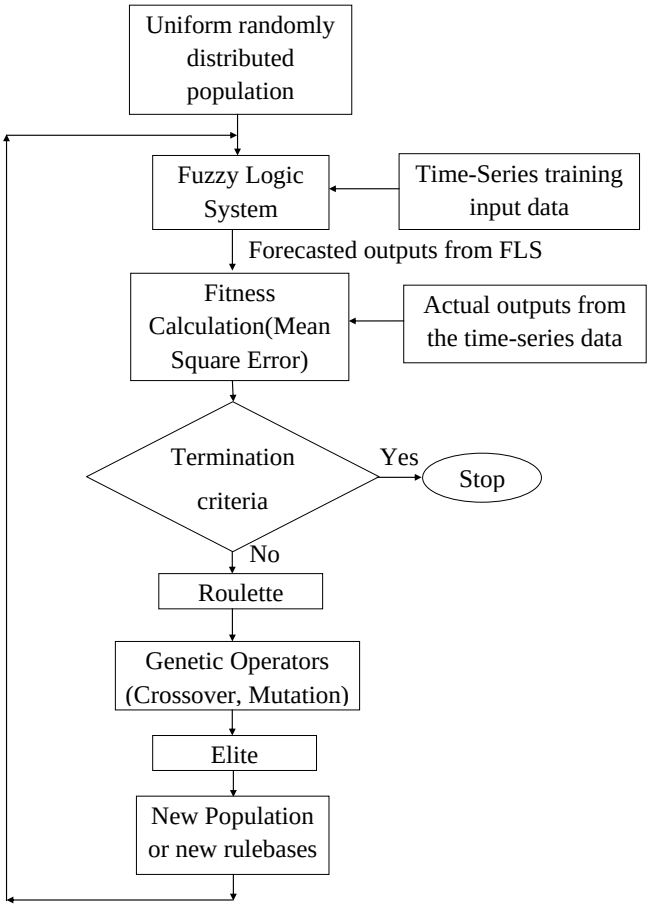


FIGURE 6.1: GA based design flow of FLS's

1. ***Initialization of Population***

   In the approach of evolving rules using GA, the 20 different rulebases with each rulebase containing 10 rules are randomly generated keeping in mind the range of antecedents and consequents of the rulebase of the FLS system to be designed for time-series forecasting. The randomly generated rulebases are then copied into the GA population with each rulebase representing each chromosome of the population. So, in the start of the algorithm each chromosome of GA population is encoded with the rule set of FLS [Shi et al., 1999]. At this step we cannot decide the number of rules to be included in the rulebase so we have just assumed the number of rules to be included in each rulebase.

2. ***Fitness Function***

   The fitness function here is evaluated using a type-1 FLS designed using four input and one output variables for time-series prediction. Here, fuzzy sets are designed using fuzzy c-mean clustering algorithm and fuzzy sets are evolved by GA. The FCM algorithm attempts to partition a finite collection of n elements X = x_1, x_2, x_3,.....,x_n into a collection of c fuzzy clusters with respect to some given criterion. Given a finite set

of data, the algorithm returns a list of c cluster centers C==c_1, c_2, c_3,.....,c_n and a partition matrix W = $w_{i,j}$ ∈[0, 1], i = 1, . . . , n, j = 1, . . . , c, where each element w_{ij} tells the degree to which element x_i belongs to cluster c_j. Each input variable is fuzzified using four gaussian membership functions characterized by two parameters, i.e., mean (m) and standard deviation (σ). Gaussian membership function is represented by (6.1).

$$\mu(x) = \exp\left(\frac{-(x-m)^2}{2\sigma^2}\right) \tag{6.1}$$

where x stands for the input values. The rule sets randomly generated and encoded as GA population are fired using inputs available from the training data. Min and max operatotrs are employed for implication and aggregation. Defuzzification is performed with the Height Defuzzification method. Output of Height Defuzzification is given by (6.2).

$$y = \frac{\sum_{i=1}^{M} C_i \mu(x_i)}{\sum_{i=1}^{M} \mu(x_i)} \tag{6.2}$$

where M stands for number of rules, C represents the location of singleton consequent fuzzy sets and $\mu(x_i)$ stands for the clipping level after implication. Corresponding to 500 inputs available from the training data of time-series we will obtain 500 outputs from the designed FLS. From time-series prediction problem the commonly used fitness function is MSE which is calculated using (6.3).

$$MSE = \frac{\sum_{i=1}^{N} (y_i - d_i)^2}{N} \tag{6.3}$$

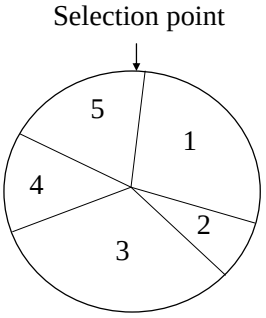
where N is the number training data, y is the output obtained from the designed FLS and d is the true output from the system.

3. ***Selection***

   Selection is basically carried out in order to prefer the fittest solutions for the next generation. Here the method used for selection is "Roulette Wheel" selection where on rotating the Wheel once the probability of each parent chromosome being chosen depends upon its fitness. The more will be the fitness of the parent chromosome, the more area will it cover on the "Roulette Wheel" hence has the more probability of being selected [Shi et al., 1999], [Sinha and Singh, 2014]. Hence selection favors the concept of the "Survival of the Fittest". The mechanism of selection is shown in Fig.6.2.

4. ***Crossover***

   After selection the crossover between the parent chromosomes is carried out resulting into two new better off springs [Shi et al., 1999]. In order to carry out crossover



Population	Fitness
1	25
2	5
3	40
4	10
5	20

FIGURE 6.2: Roulette Wheel Selection

between two individuals the crossover probability was first defined at 0.9 and a uniform random number (r) between 0 and 1 was generated. If the uniform random number, r generated is less than the crossover probability then the crossover between the two individuals takes place otherwise no crossover takes place and the original copies of parent chromosomes is reproduced in the next generation. Uniform crossover as shown in Fig.6.3 is carried out at gene level using a mixing ratio of 0.8.

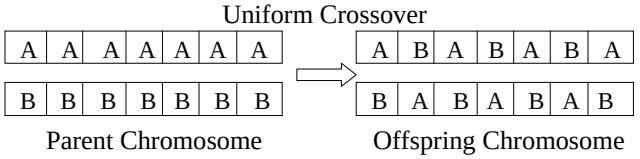


FIGURE 6.3: Uniform Crossover

5. ***Mutation***

   After crossover the Mutation is carried out in order to reproduce better and better chromosomes with more improved solutions then that produced by crossover as with one point crossover their is a probability of two parents having same string at a given gene and the population will have the same string at a given gene and the population will have the same string forever if mutation is not carried out. Here the mutation is carried out by varying the elements of the parent chromosomes. The elements here are varied by 0.08 depending upon the uniform random number generated either greater then or smaller then the mutation probability.

6. ***Elite***

   Elite solutions are applied in order to replace the worst chromosome in the population with the best one preserved earlier depending upon its fitness.

7. ***Validation***

   The validation check is applied on the newly generated population depending upon the range of the antecedents and the consequents of a FLS. The range for antecedent in

this case is kept between 1 to 4 and for consequents the range is kept between 1-10. Any element lying beyond this range will be changed to the nearest number in range using validation check. Last but not the least the newly generated population is fed back to the FLS as a new improved rule set. The population again undergoes selection, crossover and mutation to produce new improved population with reduced vale of MSE. So, GA iteratively improves the performance of the system by reducing the value of MSE hence increasing the value of fitness level.

6.3 Conclusion

Implementation include flow of algorithm to find global optimal solution. The major implementation of algorithm consists generation of initial population, randomly generated numbers to find global best solution, selection of parent solution, implementation of genetic operators and elite solution and finally coping child population back to parent population. Results obtained from the designed system are depicted in the next chapter.

CHAPTER 7

RESULTS AND DISCUSSION

This chapter presents the simulation results obtained from the designed systems. The system performance was compared for different number of rules and the one with the best performance was fixed for the purpose of time-series forecasting. At last the forecasted outputs were compared with the true outputs obtained from the time-series data.

7.1 Introduction

The system was evolved for different number of rules i.e. 2, 4, 6, 8, 10, 15, 20, 25, 30, i.e, the dimension size was kept 10, 20, 40, 50, 75, 100, 125, 150 as the number of antecedents in each rule were four and single consequent. The experiments were performed for 1000 and 10,000 iterations. The system evolved with better performance when the number of rules used were 10, i.e, the dimension size was $10 \times 5 = 50$.

7.2 Experimental Setup

The experiments were conducted on Intel (R) Core (TM) i3-2328M CPU@2.20 GHz, Window 7 (Professional). The system code was written in Visual Studio C++ (2012 Release Mode) and was compiled on nvcc compiler.

7.2.1 Overall Performance Analysis

The designed FLS system was evolved for different number of rules keeping the inputs and GA parameters constant and it was observed that on evolving the system using GA the MSE which is the performance parameter of the system was minimized up to approximately 1.154×10^{-2} for 1000 iterations and 1.141×10^{-2} for 10,000 iterations and the resulting time series obtained were very close to the desired time-series. It was also found from the simulation results that the system with 10 rules, i.e, with dimension size $10 \times 5 = 50$ evolved with least value of MSE and hence performed better. The average value of MSE obtained was found to be 1.1475×10^{-2} using 10 rules in the rule set.

TABLE 7.1: Comparative MSEs with different rulebase sizes

Number of Fuzzy Rules	Mean Square Error		
	After 1000 Iterations	After 10,000 Iterations	Average
2	0.09962	0.02032	0.05997
4	0.02746	0.02764	0.02755
6	0.01852	0.01842	0.01847
8	0.01458	0.01314	0.01386
10	0.01154	0.01141	0.011475
15	0.01895	0.01907	0.01901
20	0.04457	0.04327	0.04392
25	0.07151	0.06945	0.07048
30	0.09594	0.09309	0.094515

The graphical representation of data shown in Table 7.1 is depicted in Fig.7.1.The forecasted outputs obtained from the designed FLS using 10 rules in a Rule base were compared with the actual Time-Series outputs.

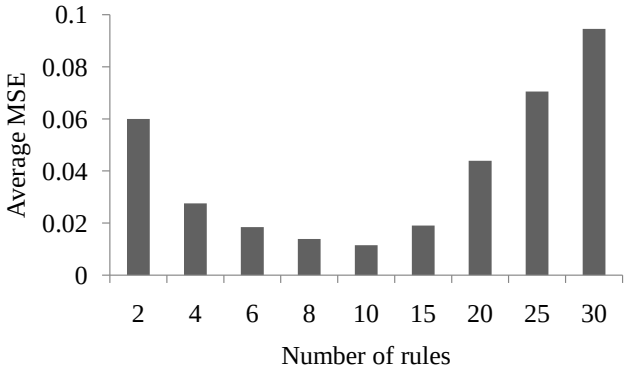


FIGURE 7.1: Minimum MSE obtained when FLS evolved for various number of rules.

7.2.2 Performance Evaluation for 1000 Iterations

MSE between the actual and the forecasted outputs of the time-series was minimized upto 1.154×10^{-2} when system was evolved for 1000 iterations. The graphical representation of the minimization of the MSE for 1000 iterations is shown in Fig. 7.2.

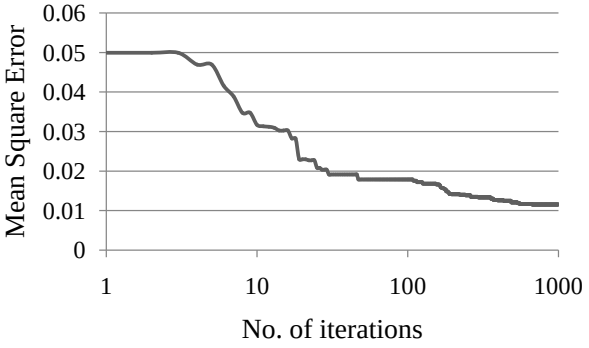


FIGURE 7.2: Minimization of MSE for 1000 iterations.

7.2.3 Performance Evaluation for 10,000 Iterations

MSE was further minimized upto 1.141×10^{-2} when evolved for 10,000 iterations. The graphical representation of the minimization of MSE for 10,000 iterations is shown in Fig. 7.3.

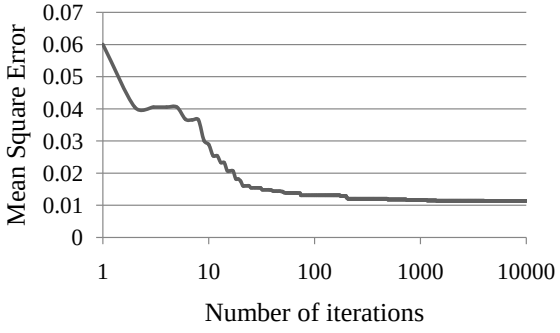


FIGURE 7.3: Minimization of MSE for 1000 iterations.

7.2.4 Result Findings

Fig. 7.4 shows the comparison between the actual outputs and the forecasted outputs obtained from the designed system. We see that the actual time-series values lie almost close

to the forecasted time series, however, there lies a scope of further improvement in the time-series waveform as there are some limitations in the forecasted outputs obtained from the designed system.

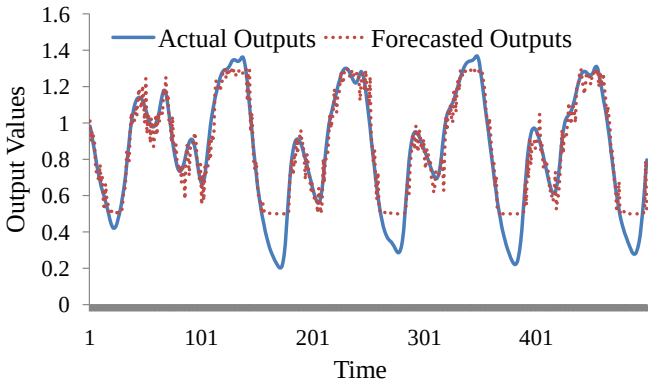


FIGURE 7.4: Comparison between the Actual and the Forecasted time series

7.3 Significant Investigation Findings

Significant improvements in the designed system can be highly useful to predict the future behavior of the various dynamic systems in future. Not only using GA we can even evolve the system using various other techniques like Particle Swarm Optimization (PSO). Various applications involving the use of FLS can employ the technique of integration of FLS and GA with highly efficient outputs.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

Research is an iterative process very similar to GA where researchers keep testing ideas based on their previous successes and the successes observed by other researchers in the area. The work in this thesis is no exception. Various research observations are presented at the end of each chapter as conclusions but limited to the scope of that chapter only. This chapter aims to conclude the thesis, as a whole, and to aggregate all the offshoots found throughout the work.

8.1 Introduction

One of the most important advantages of evolving Fuzzy Logic System using GA is that the functions parameterized in a way which is interpretable for humans. The presented application is an intelligent system designed to predict the time-series outputs. It offers many advantages such as handling imprecision.

In previous chapters the demonstration of implementation of evolving FLS using GA has been shown. The way of evolving rules in type-1 FLS using GA has been proposed. Also the designed system is used for time-series prediction.

8.2 Concluding statements

Following are brief statements to conclude this thesis as a whole:

1. It was found from the designed system that the forecasted outputs were very close to the actual outputs of the time-series.
2. So, the designed FLS with 0db SNR proved to be the best time-series prediction and forecasting outputs nearer to the actual outputs in a reasonable amount of time.
3. Using the designed system we are able to forecast the outputs in the reasonable amount of time. Therefore, it is highly advantageous to evolve the FLS using GA for various other applications.

8.3 Future Scope

Research is a never lasting process. However, this project has to be terminated here. Our research will continue and followings may be our future research agenda:

1. Although the designed system for evolution of the best solution is very effective for solving time-series problem, their execution time can become a limiting factor as it involves large number of parameters that are to be determined making it computationally intensive. In future implementing the same system on Graphic Processing Unit using Compute Unified Design Architecture can help in increasing the execution speed of the system.
2. The designed fuzzy logic system can also be evolved using some different evolutionary algorithm technique like Particle Swarm Optimization
3. The architecture of evolving rule based model using GA approach can also be applied to various other domains and the methodology can be applied in many other applications involving wide range of classification.

REFERENCES

Alba, E. and Dorronsoro, B. (2009). *Cellular genetic algorithms*, volume 42. Springer Science & Business Media.

Bastian, A. and Hayashi, I. (1995). An anticipating hybrid genetic algorithm for fuzzy modeling. 7(5):997–1006.

Bridges, C. L. and Goldberg, D. E. (1987). An analysis of reproduction and crossover in a binary-coded genetic algorithm. *Grefenstette*, 878:9–13.

Casdagli, M. (1989). Nonlinear prediction of chaotic time series. *Physica D: Nonlinear Phenomena*, 35(3):335–356.

Castillo, O. and Melin, P. (2008). *3 Type-2 Fuzzy Logic*. Springer.

Cordón, O., Herrera, E., Gomide, E., Hoffman, E., and Magdalena, L. (2001). Ten years of genetic fuzzy systems: current framework and new trends. In *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, volume 3, pages 1241–1246. IEEE.

Cordón, O., Herrera, F., and Lozano, M. (1996). On the bidirectional integration of genetic algorithms and fuzzy logic. In *Proceedings of the Second Online Workshop on Evolutionary Computation (WEC2)*, pages 13–16.

Costa, L. and Oliveira, P. (2001). Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers & Chemical Engineering*, 25(2):257–266.

Danforth, C. M. (2013). Chaos in an atmosphere hanging on a wall. *Mathematics of Planet Earth*.

De Giovanni, L. and Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. *European journal of operational research*, 200(2):395–408.

Dubois, D. J. (1980). *Fuzzy sets and systems: theory and applications*, volume 144. Academic press.

Goldberg, D. E. (1990). Real-coded genetic algorithms, virtual alphabets, and blocking. *Urbana*, 51:61801.

Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93.

Gwiazda, T. D. (2006). *Crossover for single-objective numerical optimization problems*, volume 1. Tomasz Gwiazda.

Hamilton, J. D. (1994). *Time series analysis*, volume 2. Princeton university press Princeton.

Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105.

Hwang, W.-R. and Thompson, W. E. (1994). Design of intelligent fuzzy logic controllers using genetic algorithms. In *Fuzzy Systems, 1994. IEEE World Conference on Computational Intelligence., Proceedings of the Third IEEE Conference on*, pages 1383–1388. IEEE.

Ivancevic, V. G. and Ivancevic, T. T. (2008). *Complex nonlinearity: chaos, phase transitions, topology change, and path integrals*. Springer Science & Business Media.

Karnik, N. N. and Mendel, J. M. (1999). Applications of type-2 fuzzy logic systems to forecasting of time-series. *Information Sciences*, 120(1):89–111.

Karnik, N. N. and Mendel, J. M. (2001a). Centroid of a type-2 fuzzy set. *Information Sciences*, 132(1):195–220.

Karnik, N. N. and Mendel, J. M. (2001b). Operations on Type-2 Fuzzy Sets. *International Journal on Fuzzy Sets & Systems*, 122:327 – 348.

Kellert, S. H. (1994). *In the wake of chaos: Unpredictable order in dynamical systems*. University of Chicago press.

Kumar, A., Khosla, A., Saini, J. S., and Singh, S. (2012). Meta-heuristic range based node localization algorithm for wireless sensor networks. In *Localization and GNSS (ICL-GNSS), 2012 International Conference on*, pages 1–7. IEEE.

Liang, Q. and Mendel, J. M. (2000). Interval type-2 fuzzy logic systems. In *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, volume 1, pages 328–333. IEEE.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM.

Lin, Y. and Mitchell, K. E. (2005). 1.2 the ncep stage ii/iv hourly precipitation analyses: Development and applications. In *19th Conf. Hydrology, American Meteorological Society, San Diego, CA, USA*. Citeseer.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141.

Mayer, H., Schwaiger, R., et al. (1999). Evolutionary and coevolutionary approaches to time series prediction using generalized multi-layer perceptrons. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1. IEEE.

Mendel, J. M. (1995). Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345–377.

Mendel, J. M. and Mouzouris, G. C. (1997). Designing fuzzy logic systems. *IEEE Transactions on Circuits and Systems-Part II-Analog and Digital Signal Processing*, 44(11):885–895.

Noraini, M. R. and Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving tsp.

Shi, Y., Eberhart, R., and Chen, Y. (1999). Implementation of evolutionary fuzzy systems. *Fuzzy Systems, IEEE Transactions on*, 7(2):109–119.

Singh, S. and Kakkar, S. (2014). Fuzzy systems using gpgpu - a survey. In *International Conference on Communication, Computing and Systems*, pages 238–241, Ferozepur, Punjab, India.

Sinha, R. S. and Singh, S. (2014). Optimization techniques on gpu. In *International Multi Track Conference on Science, Engineering & Technical Innovations*, pages 566–568, CT Group of Institutions, Jalandhar.

Spears, W. M. et al. (1992). Crossover or mutation. *Foundations of genetic algorithms*, 2:221–237.

Thrift, P. R. (1991). Fuzzy logic synthesis with genetic algorithms. In *ICGA*, pages 509–513.

Wei, R. (2002). A new wavelet model for identification of discrete chaotic systems and qualitative analysis of model. *Acta Electronica Sinica*, 30(1):73–75.

Werndl, C. (2009). What are the new implications of chaos for unpredictability? *The British Journal for the Philosophy of Science*, 60(1):195–220.

Whitley, L. D. et al. (1989). The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *ICGA*, pages 116–123.

Yuan, Y. and Zhuang, H. (1996). A genetic algorithm for generating fuzzy classification rules. *Fuzzy sets and systems*, 84(1):1–19.

Zadeh, L. A. (1974). *The concept of a linguistic variable and its application to approximate reasoning*. Springer.

INDEX

BCGA, 7

CGAs, 7
Crossover, 21

Elite, 23
Experimental Setup, 35

Fitness Function, 19
FLC-GA, 10
Future Scope, 40
FUZZY LOGIC SYSTEMS, 11
Fuzzy Logic Systems, 11

GA Operators, 19

Interval Type-2 FLS, 15

Methodology, 4
MGTS, 26
MINLP, 7
MRAFC-GA, 10
Mutation, 23

Objectives, 3

Rank Selection, 21
RCGA, 6
Roulette Wheel Selection, 20

Thesis Outline, 4
Time-Series Forecasting, 25
Tournament Selection, 20
Type-1 FLS, 13
Type-2 FLS, 13

Variants of GA, 6