Chapter 5

Soft-Computing Techniques and Development Tools

Soft-Computing Techniques and Development Tools

5.1 Overview of Soft-Computing Techniques

Real world problems have to deal with systems which are non-linear, time-varying in nature with uncertainty and high complexity. The computing of such systems is study of algorithmic processes which describe and transform information: their theory, analysis, design, efficiency, implementation, and application. Conventional computing/Hard computing requires exact mathematical model and lot of computation time [1]. For such problems, methods which are computationally intelligent, possess human like expertise and can adapt to the changing environment, can be used effectively and efficiently. Soft Computing is an evolving collection of artificial intelligence methodologies aiming to exploit the tolerance for imprecision and uncertainty that is inherent in human thinking and in real life problems, to deliver robust, efficient and optimal solutions and to further explore and capture the available design knowledge [2]. Soft computing utilizes computation, reasoning and inference to reduce computational cost by exploiting tolerance for imprecision, uncertainty, partial truth and approximation. Numerous Soft Computing-based methods and applications have been reported in the literature in a variety of scientific domains. The advances in application of Soft Computing Techniques in various demanding domains has promoted its use in industrial applications [3,4].

Soft Computing with its roots in fuzzy logic, artificial neural network, and evolutionary computation has become one of the most important research field applied to numerous engineering areas such as Aircraft, Communication networks, computer science, power systems and control applications. Soft Computing Techniques comprises of core methodologies: Fuzzy Systems (FS), including Fuzzy Logic (FL); Evolutionary Computation (EC), including Genetic Algorithms (GAs); Artificial Neural Networks (ANN), including Neural Computing (NC); Machine Learning (ML); and Probabilistic Reasoning (PR) [5]. Where PR and FL systems are based on knowledge-driven reasoning, whereas, ANN and EC, are data-driven search and optimization approaches. List of various problem Solving Techniques are as shown in Figure 5.1.

Soft Computing Techniques consists of rich knowledge representation, knowledge acquisition and knowledge processing for solving various applications. These techniques can be deployed as individual tools or be integrated in unified and hybrid architectures. The fusion of Soft Computing techniques causes a paradigm shift in engineering and science fields, which could not be solved with the conventional computational tools. Soft Computing has gain importance in the application fields for wireless communication in the last decade. Wireless communication systems are associated with

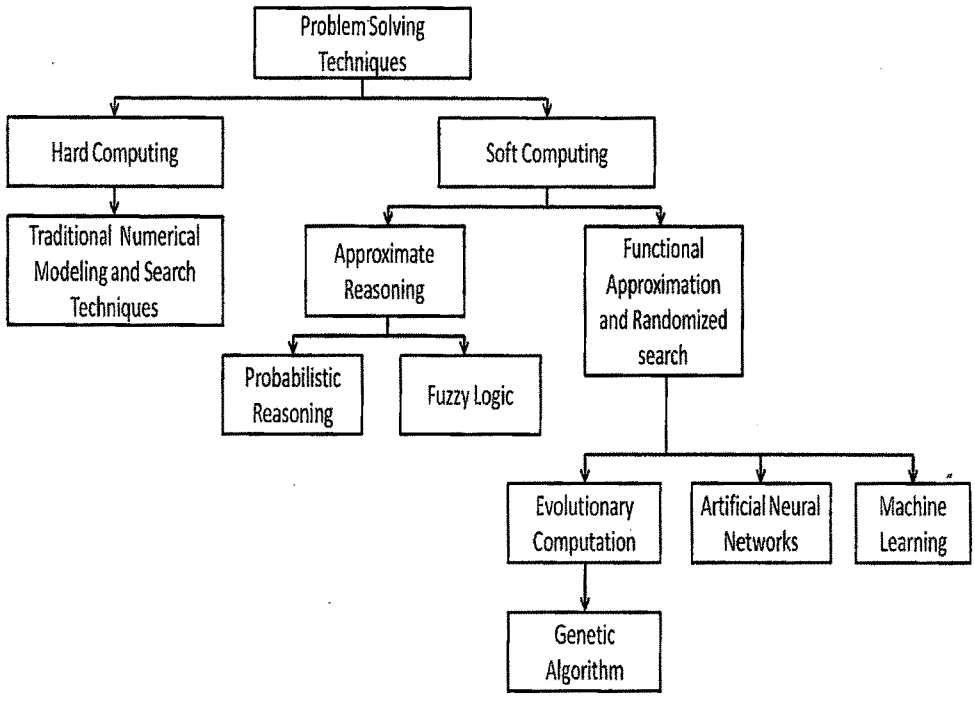

Figure 5.1: Various Problem Solving Techniques

much uncertainty and imprecision due to a number of stochastic processes such as time-varying characteristics of the wireless channel caused by the mobility of transmitters, receivers, objects in the environment and mobility of users. This reality has fueled numerous applications of soft computing techniques in mobile and wireless communications [6]. The role of soft computing in the domain of wireless systems can be classified into three broad categories, namely, optimization, prediction and uncertainty management [7]. Evolutionary algorithms are mostly used for optimization. Neural networks and other learning systems are used for different types of prediction tasks. Uncertainties arising due to incomplete modeling and measurements are handled using fuzzy logic, either in stand-alone manner or in conjunction with the optimization and prediction algorithms.

This chapter discusses the theoretical understanding of core Soft Computing Techniques i.e FL, ANN and GA in detail. Also the simulation of these systems with MATLAB based Toolboxes is presented. Various methods of using the tools for application oriented programming techniques is briefly discussed. The advances of application of Soft Computing Techniques in the vast field wireless communication is reviewed and presented in the chapter.

5.2 Fuzzy Logic System

Fuzzy systems are based on fuzzy logic, a generalization of traditional Boolean logic which is extended to handle the concept of partial truth i.e values between "complete true" and "complete False". Fuzzy Logic provides a set of mathematical methods for representing information in a way that resembles natural human reasoning and deals with system uncertainty and vagueness [9]. Concepts of fuzzy sets, fuzzy logic and fuzzy control have been introduced and developed by L.Zadeh in a series of articles spanning several years [10–13]. Fuzziness is imprecision or vagueness, a fuzzy proposition may be true to some degree. For example we use a linguistic variable like short, tall, very tall for HEIGHT or maybe young, old for AGE. Fuzzy logic is viewed as a formal mathematical theory for the representation of uncertainty [14, 15].

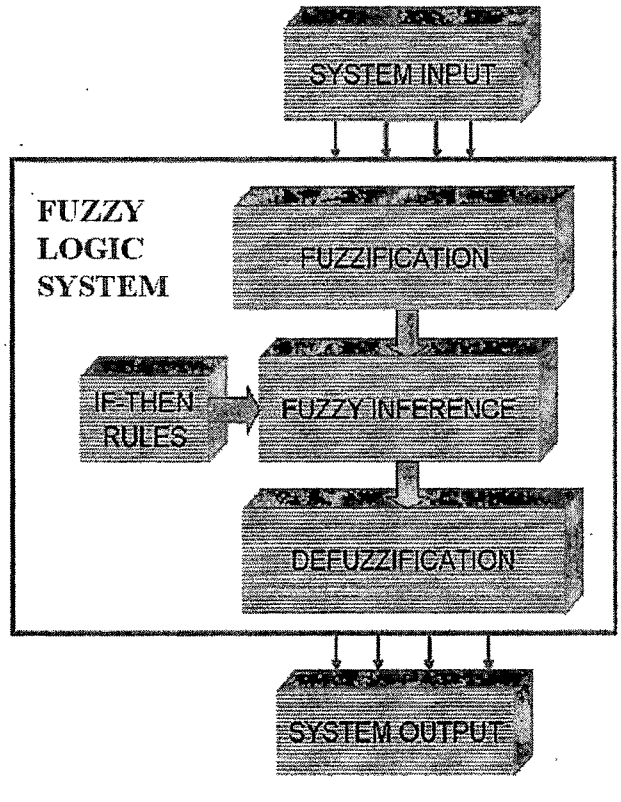

Figure 5.2: Fuzzy Logic System

A Fuzzy Logic System is an expert system that uses a collection of fuzzy membership functions and fuzzy IF-THEN rule base, instead of Boolean logic, to reason about data. The rules in a Fuzzy Logic System are of a form as following:

IF (x is LOW) AND (y is HIGH) THEN (z is MEDIUM),

IF (premise) THEN (Conclusion)

where x and y are input variables for known data values, z is an output variable for an output data to be computed, LOW is a membership function (fuzzy subset) defined on the set of x, HIGH is a membership function defined on the set of y,and MEDIUM is a membership function defined on the set of z.The antecedent (the rules premise, between IF and THEN) describes to what degree the rule applies, while the consequent (the rules conclusion, following THEN) assigns a membership function to each of one or more output variables. The set of rules in a Fuzzy Logic System is known as the rule base or knowledge base. Figure 5.2 shows the Fuzzy Logic System Block Diagram.

Algorithm 5.1 Fuzzy Logic Algorithm

- Initialization:
  - Define Linguistic variables and terms
  - Construct Membership Functions
  - Construct Rule Base
- Fuzzification: Convert crisp input data to fuzzy values using the membership functions
- Inference: Evaluate the rules in the rule base and combine the results in rule base
- Defuzzification: Convert output data to non-fuzzy values

The Fuzzy Inference Process consists of following steps [16]:

- Fuzzification: The membership functions defined on the input variables are applied to their actual values, to determine the degree of truth for each rule premise.
- Fuzzy Inference Engine: The truth value for the premise of each rule is computed, and applied to the conclusion part of each rule. This results in one fuzzy subset to be assigned to each output variable for each rule. The aggregation method min or product is used as inference rules. After Inference, the composition of all fuzzy sets is carried out. Under composition, all of the fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable. Usually max or sum is used.
- Defuzzification: It convert the fuzzy output set to a crisp number. There are many defuzzification methods [17, 18]. The Fuzzy Logic Algorithm steps is as given in Algorithm 5.1.

For modeling, simulation and design of Fuzzy Logic Systems usage of simulation software packages has become apart of engineering practice. MATLAB consists of Fuzzy Logic Toolbox

that allows designers to create fuzzy systems, the description of the features of the Fuzzy Logic Toolbox is described in next Section.

5.2.1 MATLAB Simulation-Fuzzy Logic Toolbox

The Fuzzy Logic Systems can be designed and simulated using MATLAB Fuzzy Logic Toolbox [19]. The Fuzzy Logic Toolbox, provides functions and GUI based editors for building Fuzzy Inference System (FIS).

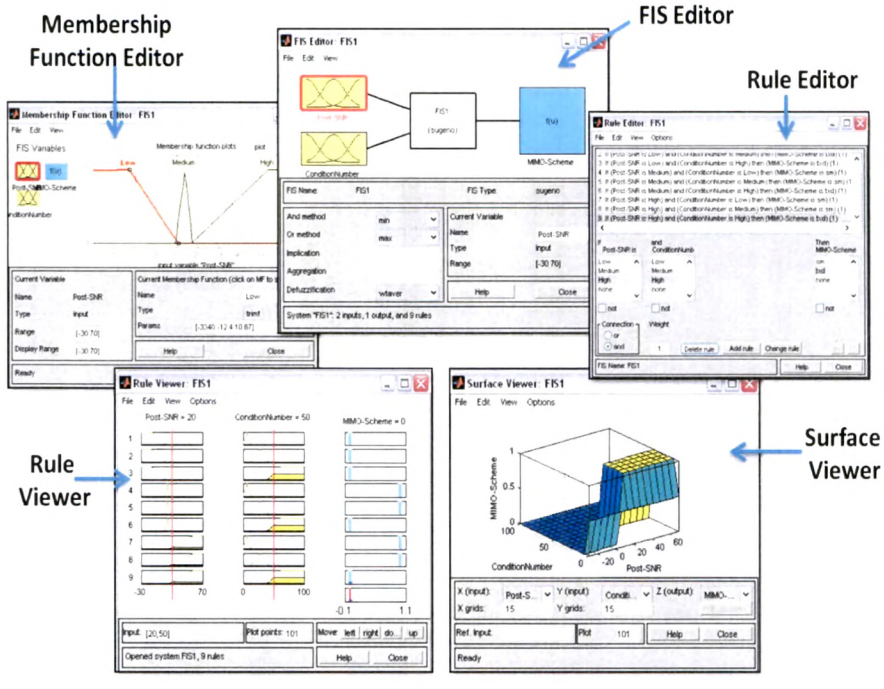

Figure 5.3: Fuzzy Inference System Editors and Viewers

FIS Editor Blocks	Description
FIS Editor	Display general Information about FIS
Membership Function Editor	Display and edit the MFs associated with the input and output variables of FIS
Rule Base Editor	View and edit fuzzy rules
Rule Viewer	View detailed behavior of a FIS to help es diagnose the behavior of specific rul
Surface Viewer	Generates a 3-D surface from two input variables and the output of FIS

Table 5.1: List of FIS Editor Blocks and description

FIS editor is a GUI which contains editors and viewers for building rule sets, defining membership functions and for analyzing the behavior of FIS. The toolbox also has ability to embed FIS

in Simulink model for simulation and to generate C code or stand-alone executable fuzzy inference engines. The editors and viewers of FIS Editor is as shown in Figure 5.3. Description of each editors and viewers are listed in Table 5.1.

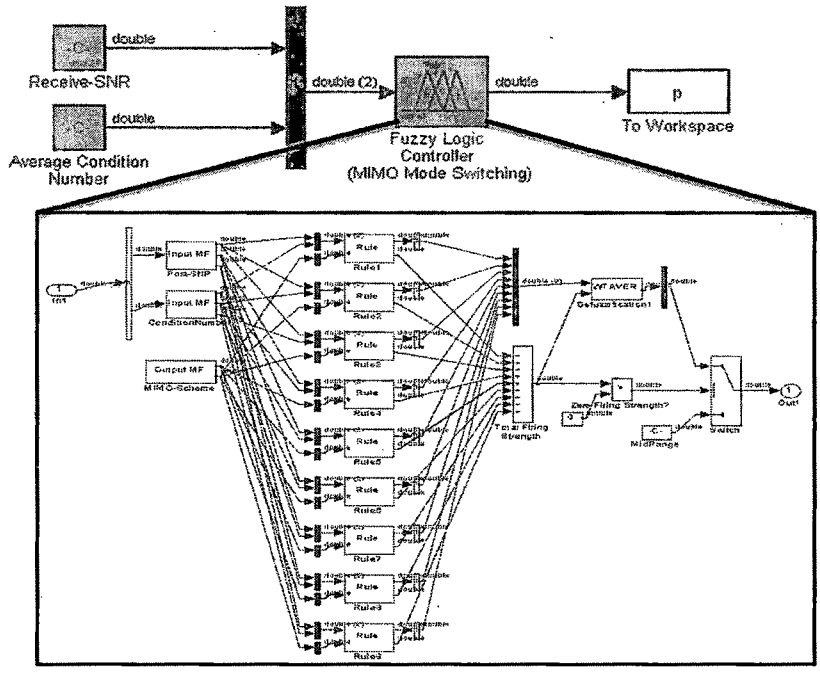

Figure 5.4: FLC in Simulink

MATLAB Function	Description
newfis	Create new Fuzzy Inference System
readfis	Load FIS from File
evalfis	Perform Fuzzy Inference Calculations
addvar	Add variable to FIS
addmf	Add MF's tro FIS
addrule	Add rule to FIS
defuzz	Defuzzify Membershipo Functions

Table 5.2: List of MATLAB Functions for Designing FIS

FIS performance can be evaluated using the Fuzzy Logic Controller (FLC) block in a Simulink model. The Fuzzy Logic Controller block automatically generates a hierarchical block diagram representation for Fuzzy Logic controller designed. This representation uses only built-in Simulink Blocks, enabling efficient code generation. The Fuzzy Logic blocks available in Simulink and the detailed representation is as shown in Figure 5.4.

The FIS model can also be designed using the programming functions provided in the Fuzzy Logic Toolbox. Table 5.2 lists few MATLAB Functions to design FIS System.

5.3 Artificial Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the biological neural networks, which consists of massively parallel computing systems with large number of simple processors with many interconnections [20]. ANN methodologies consists of basic architecture known as "Neurons". A neuron or nerve cell is a special biological cells that processes information in human brain. Brief description on Biological Neurons and Neural Networks can be found in [21]. ANNs are applied to solve various challenging problems like Classification [22], Clustering, Function Approximation [23–25], Prediction/Forecasting, Medical Imaging Application [27],Optimization and Control related applications [26].

5.3.1 McCulloch and Pitts Model of Neuron

The science of ANN has its first significance appearance during the 1940's, when researchers McCulloch and Pitts [28, 29] tried to emulate the functions of human brain by developing physical model of biological neuron and their interconnections [30]. Their work was focus on a simple neuron, which were considered to be binary with fixed thresholds as shown in Figure 5.5.

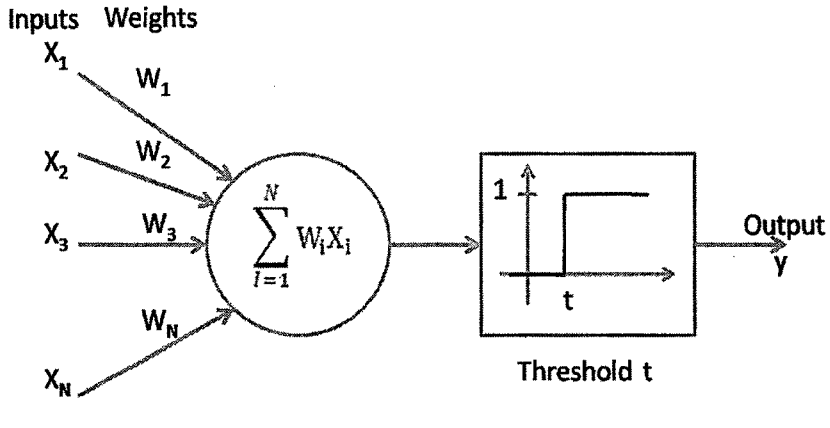

Figure 5.5: McCulloch-Pitts model neuron

The threshold unit receives input from N other units. Input from i^{th} unit is termed as x_i, and the associated weight is w_i. The total input to a unit is the weighted sum over all inputs

$$\sum_{i=1}^{N} W_i X_i = W_1 X_1 + W_2 X_2 + ... + W_N X_N \tag{5.1}$$

If this value is below threshold t, the output of the unit is 1 and 0 otherwise. The McCulloch-Pitts model of a neuron is so simple that it only generates a binary output and also the weight and threshold values are fixed. But, it has substantial computing potential. The neural computing algorithm has diverse features for various applications . Thus, we need to obtain the neural model with more flexible computational features. Based on the McCulloch-Pitts model described previously, the general form an artificial neuron can be described in two stages shown in Figure 5.7.

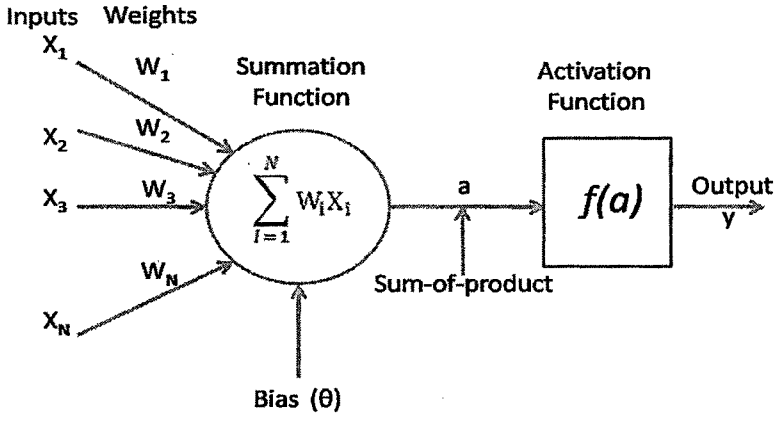

Figure 5.6: Artificial Neuron

In the first stage, the linear combination of inputs is calculated. Each value of input array is associated with its weight value, which is normally between 0 and 1. Also, the summation function often takes an extra input value Theta with weight value of 1 to represent threshold or bias of a neuron. The summation function will be then performed as:

$$a = \sum_{i=1}^{N} W_i X_i + \theta \tag{5.2}$$

The sum-of-product value is then passed into the second stage to perform the activation function which generates the output from the neuron. The activation function "squashes" the amplitude the output in the range of [0,1] or [-1,1] alternately. The behavior of the activation function will describe the characteristics of a neuron model.

5.3.2 ANN Network Architectures

ANNs can be viewed as weighted directed graphs in which artificial neurons are nodes and their weights are connections between neuron outputs and neuron inputs. Based on the connection pattern (architecture), ANNs can be classified into two categories [31,32]:

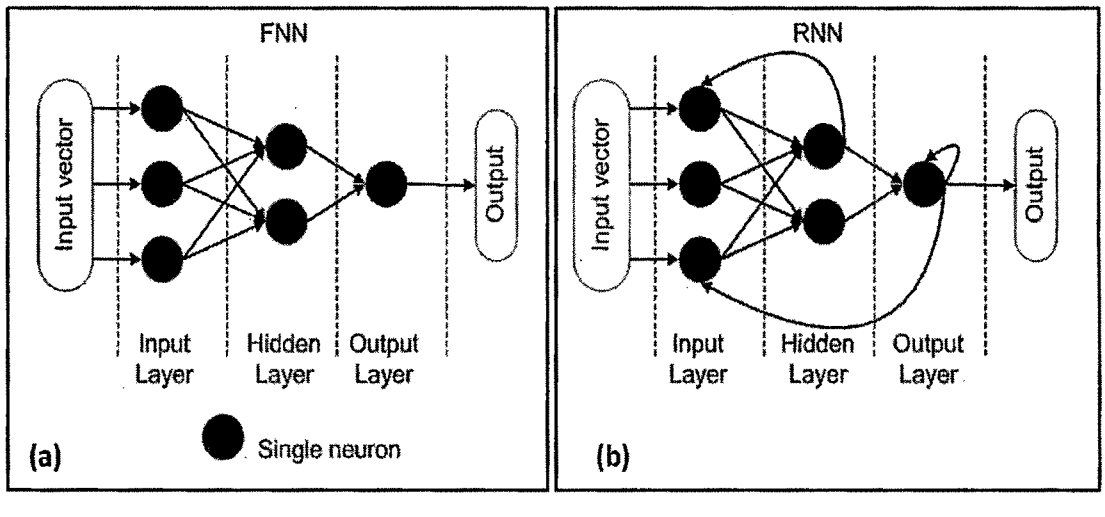

Figure 5.7: (a) Feed-forward (FNN) and (b) Recurrent Neural Network (RNN) Architectures

- **Feed-forward Networks**: In this type of network neurons are organized into layers that have unidirectional connection between them. This networks are static in nature as they have no feedback and hence no delays. the output is calculated directly from the input through feedforward connections. They are memory-less networks as its response to an input is independent of the previous network state. Types of Feed-forward networks include: Single-Layer perceptron , Multilayer perceptron and Radial Basis Function networks. Figure 5.7 (a). shows a single layer Feed-forward neural network with n inputs and m outputs.

- **Recurrent or Feedback Networks**: These networks are dynamic in nature i.e. the outut depends on the current and previous inputs, outputs and states of network. Due to feedback paths, the input to each neuron is modified according to the feedback value and the network enters into a new state. Types of recurrent networks are: Competitive networks, Kohonen's Self-organizing Maps (SOM), Hopfield Network and ART models. Figure 5.7 (b). shows a Recurrent Neural network' which consists of feedback paths from output to input neurons. There can be neurons with self-feedback links.

The ability to learn is a fundamental trait of intelligence of ANN. In learning process, ANN updates network architecture and connection weights from training patterns. ANN's ability to learn from examples makes it attractive for various applications in research field. ANNs learn the underlying rules (like input-output relation) from the given collection of training data. Learning algorithms [33] adjusts the weights of ANN using learning rules. Based on learning process there are three types of learning paradigms:

- **Supervised learning**: also known as learning with a "teacher", means the network is provifed with the correct output for every input pattern. Connection weights are then determined so as the allow the network to produce output very close to the correct answers. Examples of supervised learning algorithms are Boltzmann learning algorithm, Learning vector quantization, Back-propagation Adaline algorithm and Perceptron learning Algorithms.

- **Unsupervised Learning**: also know as learning without "teacher", do not require correct output answers for each input pattern in the training set provided. It explores the network structure in data or correlations between input patterns, and organizes input patterns into categories from these correlations. Unsupervised Learning algorithms include Principal Component Analysis, Associative memory Learning, Kohonen's SOM, Adaptive resonance theory (ART) algorithms.

- **Hybrid learning**: It combines supervised and unsupervised learning i.e. part of the weights are determined through supervised learning and the remaining are obtained through unsupervised learning. Radial Basis Function (RBF) Learning algorithm used for learning in RBF networks using Error-correction and competitive learning rule is an example of Hybrid learning.

5.3.3 Designing Neural Network

The ANN design process follows number of systematic steps [34–36]. It can be design using following steps:

1. **Collection of data**: The data for which the neural network is to be designed, the data for training the networks. It is termed as Inputs and Targets for the Neural networks [data preparation for neural network].

2. **Designing the network**: It includes defining the architecture of Neural Network. it includes defining the number of layers, number of nodes in each layer, defining Transfer functions for each

layer, defining the training algorithm to train the network. Option parts of design includes: Error function for neural network, plotting the data, number of echos.

3. **Training the network**: once the network is designed, it has to be trained to optimize the error function. This process determines the best set of weights and biases for the collected data.

4. **Testing the network**: Finally the designed net is to be tested for accuracy and generalization.

5.3.4 MATLAB Simulation: Neural Network Toolbox

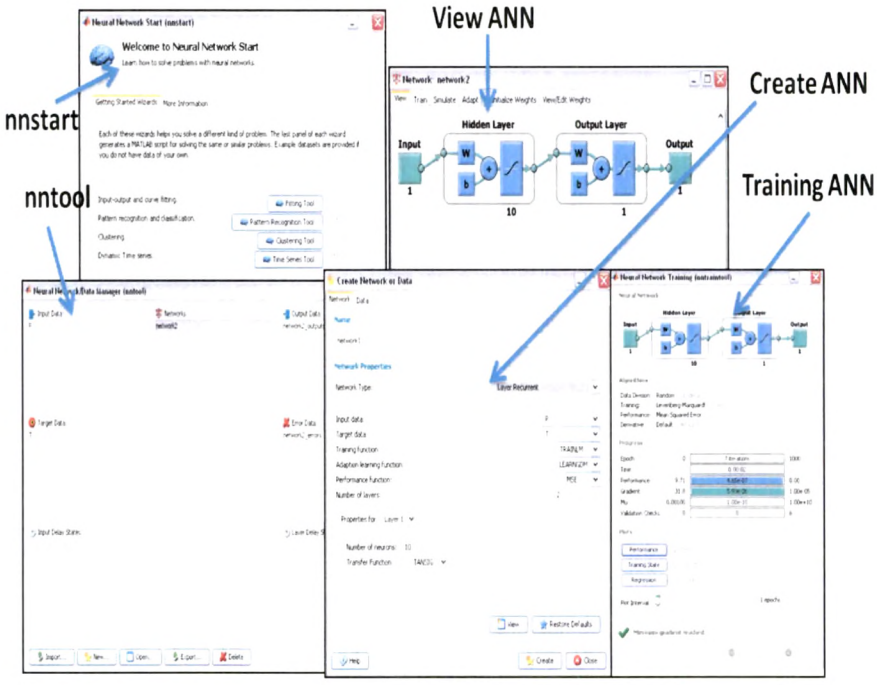

Figure 5.8: Neural Network Toolbox

The MATLAB Neural Network Toolbox [37] provides tools for design, visualization and simulation of ANN. It supports many network paradigms and provides GUI which enables user to design networks. Neural Network Toolbox supports a variety of supervised and unsupervised network architectures. With the toolboxs modular approach to building networks, custom network architectures for specific problem can be developed. The network architecture including all inputs, layers, outputs, and interconnections can be viewed. The features of GUI are as shown in Figure 5.8. The Neural network Start GUI provides examples and data sets for designing neural networks. It consists of Fitting Tool, Pattern recognition Tool, Clustering Tool and Time Series Tool for designing ANN for various Applications.

The toolbox can also be used by basic command-line operations. The command-line operations offer more flexibility than the GUIs, but with some added complexity. Various ANN architectures can be designed using command line functions like *feedforwardnet* for creating a feed-forward neural network. Number of parameters for ANN like number of hidden layer, activation functions, input nodes, training algorithm can be set using numerous MATLAB functions provided in the Toolbox. Example of sample code for designing, viewing, evaluating, training and creating simulink model for Feedforward network with 5 hidden layers is as below:

```
[x,t]=simplefit_dataset;   % data set for training ANN
net=feedforwardnet(5)        % Creates FNN
net=train(net,x,t);     % Training of net
view(net) % View the net designed
y=net(x); % Evaluate net for x input
perf=perform(net,y,t) % Calculate network performance
gensim(net)  % Generates simulink model for the net
```

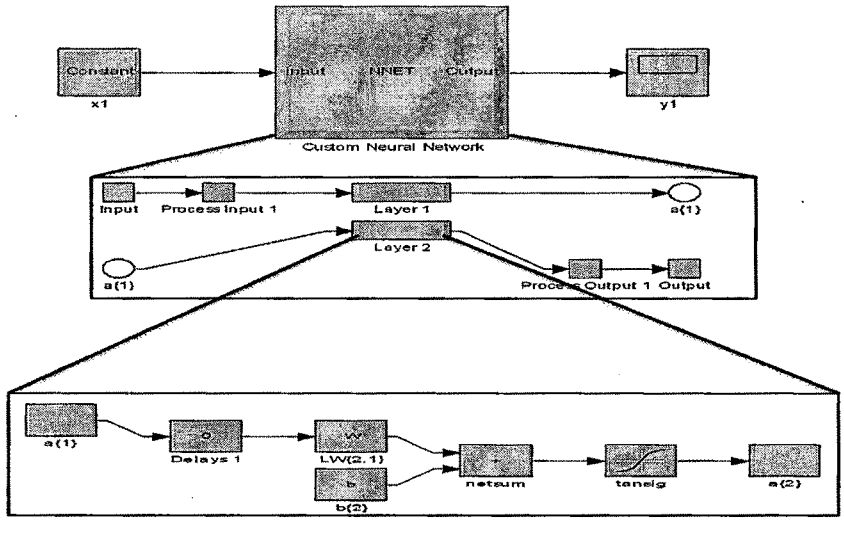

Figure 5.9: Neural Network Simulink model

Alternatively, Networks can be created and trained in the MATLAB environment and automatically generate network simulation blocks for use with Simulink using *gensim* command. This approach also enables users to view networks graphically. Example of Custom neural network created by the gensim command for the sample code of feedforward network is as shown in Figure 5.9.

Neural Network Toolbox provides set of blocks for building neural networks in Simulink. These blocks are divided into four libraries:

- Transfer function blocks, which take a net input vector and generate a corresponding output vector
- Net input function blocks, which take any number of weighted input vectors, weight-layer output vectors, and bias vectors, and return a net input vector
- Weight function blocks, which apply a neuron's weight vector to an input vector (or a layer output vector) to get a weighted input value for a neuron
- Data pre-processing blocks, which map input and output data into the ranges best suited for the neural network to handle directly

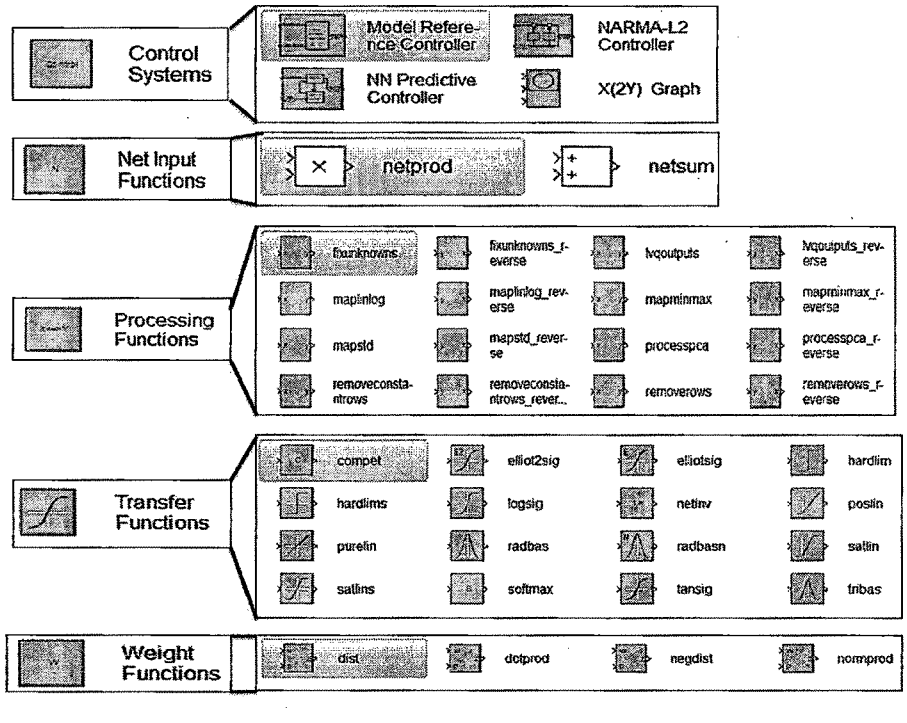

Figure 5.10: Neural Network Toolbox Simulink Blocks

Figure 5.10, shows the set of blocks provides by Neural Network Toolbox for simulating Neural Networks in Simulink. Neural Network Simulink blocks consists of Control Systems, Net Input functions, numerous Pre and Post processing Functions, various Transfer Functions and Weight functions for designing Neural Networks for various applications. Using these blocks ANN Architecture is designed and its performance and analysis is carried out for system analysis.

5.4 Genetic Algorithms

Since 1950s several researchers have studied Evolutionary Systems as an optimization tool for engineering problems. The basic idea in all these systems were to evolve a population of candidate solutions of a given problem, using operators inspired by natural genetic variation and natural selection [39]. In 1970's, the pioneering work of J.H. Holland proved to be significant contribution for various engineering and scientific applications. Holland's book "Adaptation in Natural and Artificial Systems" [40] was instrumental in creating the flourishing field of research in Genetic Algorithms. The well known applications of GA include scheduling, sequencing, reliability design, and image processing [41].

5.4.1 Introduction

Genetic Algorithms are inspired by the mechanism of natural selection, which is a biological process in whcih stronger individuals are more likely to be winners in a competing environment [42]. GA assumes that the solution of a problem is an individual, which can be represented by a set of parameters. These parameters are known as genes of the chromosomes and can be represented by string of binary values. GAs is a search technique which start with an initial set of random solutions known as population. Each individual in population is called chromosomes, which is a string of binary values. The chromosomes evolve through successive iterations, called generations. During each iteration the chromosome evolve using some measures of fitness. Then the next generation is created, where the new chromosomes called as off-springs, are formed by either merging two chromosomes from current generation using a crossover operator or modifying a chromosome using a mutation operator. New generation is formed by selection, based on the fitness values, some of the parents and off springs are rejected to keep the population size constant. After several iterations the algorithm converges to the best chromosome, which represents the optimum or sup-optimum solution to the problem [43]. Figure 5.11. shows the basic structure of Genetic Algorithms. The Standard genetic Algorithm [42,44] is given in Algorithm 5.4.1.

Genetic Algorithm tools are available to evaluate the optimization problem. The GA toolkits and libraries available [45,46] based on C++ programming include GALib, The Genetic Algorithm Utility Library (GAUL), Open Beagle and Java based toolkits include Java Genetic Algorithms Package (JGAP), JAVA API for Genetic Algorithms and JAVA GALib. MATLAB also provides Global Optimization Toolbox for solving Optimization Problems. It consists of Genetic Algorithm Solver as one of the method to solve optimization problems, discussed in following section.

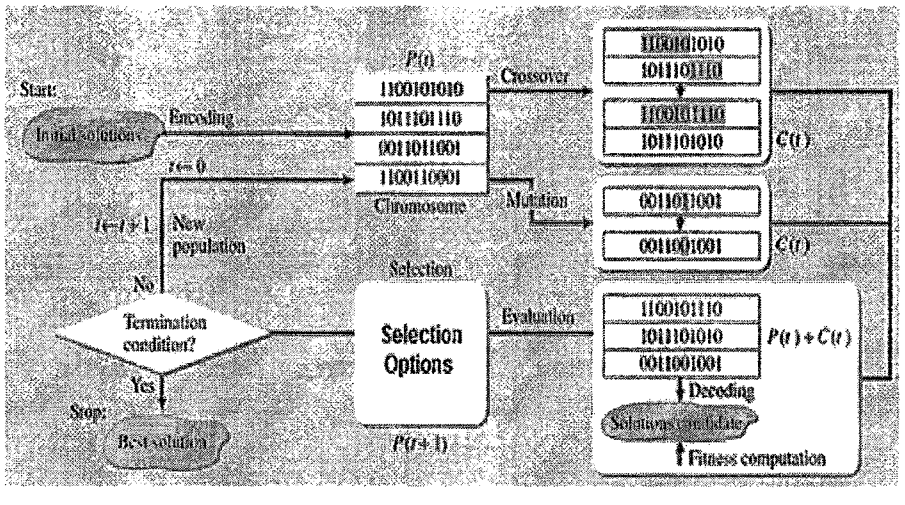

Figure 5.11: Genetic Algorithm Basic Structure

Algorithm 5.2 Standard Genetic Algorithm

- Input Initial Parameters: Fitness Function, Population size, Crossover operator, Mutation operator and stopping criteria.
- Initialize a random population of individuals;
- Evaluate fitness of all initial individual of population;

while $stoppingcriterianotfullfilled$ **do**

- Select individuals for reproduction;
- Create offsprings by crossing individuals;
- Eventually Mutate some individuals;
- Evaluate its new fitness;
- Select survivors from actual fitness;
- Compute New Generation.

end while

- Plot SNR v/s Throughput

5.4.2 MATLAB Simulation: Global Optimization Toolbox

MATLAB provides Global Optimization Toolbox which consists of methods that search for global solutions to problems that contain multiple maxima or minima [47]. Methods include global search, multistart, pattern search, genetic algorithm, and simulated annealing solvers. The Genetic Algorithm solves both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual using rules modeled on gene combinations in biological reproduction. The steps for the genetic algorithm optimization techniques are as follows:

1) Random Initial Population is created using the population options like population size and creation function specified.

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

- Computing fitness value of each member of the current population.
- Selects members, called *parents*, who contribute their genes to their *children*, based on their fitness.
- Some of the individuals in the current population that have best fitness are chosen as *elite*. These elite individuals are passed to the next population.
- Produces *children* from the parents. Children are produced either by *mutation* or *crossover*.
- Replaces the current population with the children to form the next generation.

3) The algorithm stops when one of the stopping criteria like number of generations, time limit, fitness limit is met.

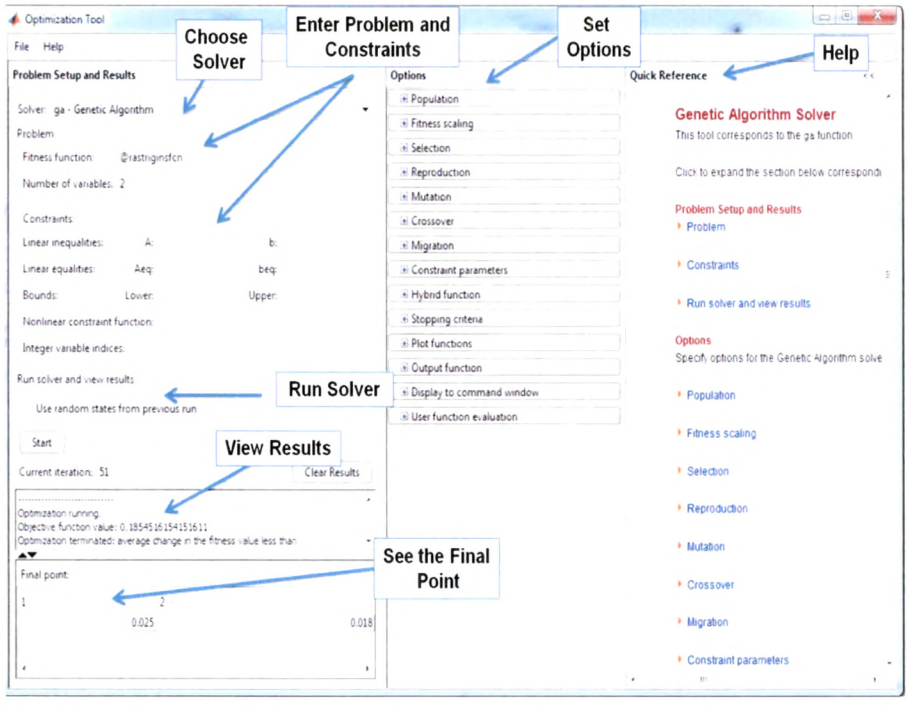

Figure 5.12: Optimization App in Global Optimization Toolbox

The Optimization App provided by the Global Optimization Toolbox is open by using the command: *optimtool* in MATLAB Command Window. This command opens the Optimization App, as shown in Figure 5.12.

Genetic Algorithm Solver options like Population, Fitness Scaling, Mutation options, Crossover options and Stopping criteria is selected based on the optimization function and application. Using Optimization App various plots like best fitness, best individual, Distance, Range and Score Diversity is generated for performance analysis of Genetic Algorithm.

Genetic Algorithm for mixed-integer or continuous-variable optimization, constrained or unconstrained optimization problems can be also solved using MATLAB functions and by specifying algorithm options. Customize Genetic algorithm can be created by modifying the initial population and fitness scaling options or by defining parent selection, crossover and mutation functions. Table 5.3 lists the MATLAB functions for solving problems using Genetic Algorithm.

Functions	Description
ga	Find minimum of function using Genetic Algorithm
gaoptimget	Obtain values of GA options structure
gaoptimset	Create GA options sturcture

Table 5.3: List of MATLAB Functions for Genetic Algorithm

The genetic algorithm is executed y calling the gaoptimset function and providing with fitness function to optimize and other Genetic Algorithm options.

5.5 Applications of Soft Computing Techniques in Wireless Communication

Wireless communications is rapidly evolving sector with challenges to meet the demands for higher performance and efficiency. Wireless communication systems are associated with much uncertainty and imprecision due to a number of stochastic processes such as time-varying characteristics of the wireless channel caused by the mobility of transmitters, receivers, objects in the environment and mobility of users. This reality has fueled numerous applications of soft computing techniques in mobile and wireless communications [48, 49]. The role of soft computing in the domain of wireless systems can be classified into three broad categories, namely, optimization, prediction and uncertainty management [50]. Evolutionary algorithms are mostly used for optimization. Neural networks and other learning systems are used for different types of prediction tasks. Uncertainties arising due to incomplete modeling and measurements are handled using fuzzy logic, either in stand-alone manner or in conjunction with the optimization and prediction algorithms.

Fuzzy Logic have been applied in numerous areas of Wireless communication such as in channel estimation, channel equalization and decoding [51]. Evolutionary Algorithms (EAs) have been

frequently applied to telecommunication problems in hardware design, network design and data transmission. Genetic Algorithms have been successful in solving various optimization problems in the field of Wireless Communication. Genetic Algorithms has been applied to hardware design applications like antenna design, network routing and assignment [52].It has also been applied to Wireless Sensor Network to optimize system performance [53]. Artificial neural networks have been applied in high-speed communication networks [54], Antenna Design [55]. Literature review of various field areas where soft-computing techniques are applied in Wireless Communication Systems and Wireless Sensor Networks is given in [48,50].

5.6 Concluding Remarks

This chapter discusses various problem solving techniques. The theoretical background for core soft-computing techniques i.e Fuzzy Logic, ANN and Genetic Algorithm is summarized. MATLAB based toolboxes available for designing and testing the performance of application of soft-computing techniques in various areas is discussed in detail. Procedural steps for designing and programming the soft-computing techniques and its usage using SIMULINK is summarized.