Chapter 7

ANN Based AODV, GA

This chapter describes optimization of transmission power of the node in WANET using Reactive routing protocol AODV. Optimization of the routing protocol using Artificial Neural Network (ANN), decides the hello interval value from the transmission power and mobility of the node. For the optimized ANN architecture its weight and training is done using Genetic Algorithm (GA).

Neural computing is a new approach to information processing. It is capable of imitating brain's ability to make decisions and draw conclusions when presented with complex, noisy, irrelevant or partial information. Thus it is the domain in which an attempt is made to make compute think, react and compute.

7.1 Artificial Neural Network: An Introduction

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. An ANN is configured for a specific application, through a learning process. Learning in biological system involves adjustment to the synaptic connections that exist between the neurons.

Artificial Neural Networks (ANNs) are non-linear mapping structures based on the function of the human brain. They are powerful tools for modelling, especially for the underlying data relationship is unknown. ANNs can identify and learn correlated patterns between input data sets and corresponding target values. After training ANNs can be used to predict input data. ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy [1]. Thus they are ideally suited for the modelling of complex and often non-linear data. ANNs has great capacity in predictive modelling i.e. all the characters describing the unknown situation can be presented to the trained ANNs, and then prediction of systems is guaranteed.

An ANN is a computational structure that is inspired by observed process in natural networks of biological neurons in the brain. It consists of simple computational units called neurons, which are highly interconnected. ANNs have become the focus of much attention, largely because of their wide range of applicability and the ease with which they can treat complicated problems. ANNs are parallel computational models comprised of densely interconnected adaptive processing units. These networks are fine-grained parallel implementations of nonlinear static or dynamic systems. A very important feature of others networks is their adaptive nature, where "learning by example" replaces "programming" in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is readily available. ANNs are now being increasingly recognized in the area of classification and predictions, where regression model and other related statistical techniques have traditionally been employed. The most widely used learning algorithm in an ANN is the backpropagation algorithm. There are various types of ANNs like Multilayered Perceptron, Radial Basis Function and Kohonen networks. These networks are "neural" in the sense that they may have been inspired by neuroscience but not necessarily because they are faithful models of biological neural or cognitive phenomena. In fact majority of the networks are more closely related to traditional mathematical and/or statistical models such as non-parametric pattern classifiers, clustering algorithms, nonlinear filters and statistical regression models than they are to neurobiology models.

ANNs have been used for a wide variety of application where statistical methods are traditionally employed. The problems which were normally solved through classical statistical methods, such a discriminate analysis, logistic regression, bayes analysis, multiple regression, and ARIMA time series models are being tackled by ANNs. It is, therefore, time to recognize ANN as a powerful tool for data analysis.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This can then be used to provide projections given new situations of interest. Other advantages include:

- **Adaptive learning:** An ability to learn how to do tasks based on the data given for training or initial experience.
- **Self-Organization:** An ANN can create its own organization or representation of the information it receives during learning time.
- **Real Time Operation:** ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- **Fault Tolerance via Redundant Information Coding:** Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

ANN is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. An artificial neuron is a device with many inputs and one output. Each connection has a weight factor and these weights are adjusted in a training process. There are many types of neural networks for various applications viz. Back Propagation, Feed Forward, Multilayered Perceptron. A commonly used is multilayered perceptrons (MLP). The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not. A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained.

Each interconnection in an ANN has a strength that is expressed by a number referred to as weight. This is accomplished by adjusting the weights of given interconnection according to some learning algorithm. Learning methods in neural networks can be broadly classified into three basic types

1. Supervised learning
2. Unsupervised learning
3. Reinforced learning

In MLP, the supervised learning will be used for adjusting the weights. The graphic representation of this learning is given in **Figure 7.1.**

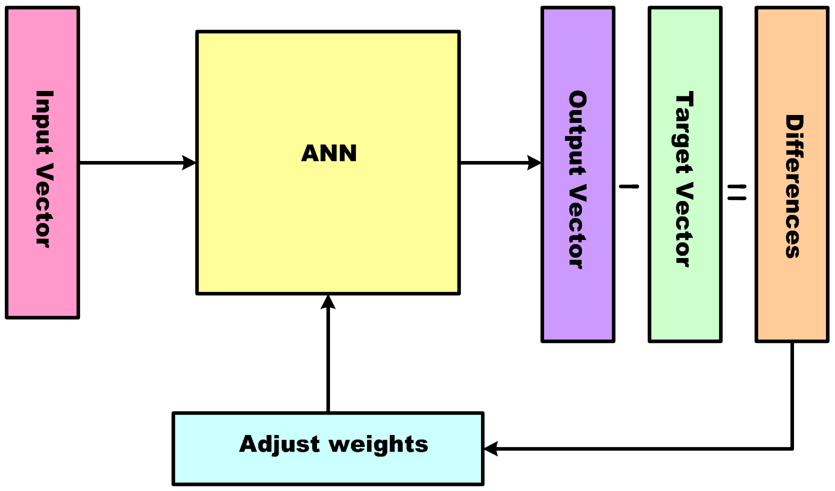


Figure 7.1: A learning Cycle in the ANN model

7.2 ANN Based AODV

ANN is attractive due to its information processing characteristic such as nonlinearity, high parallelism, fault tolerance as well as capability to generalize and handle imprecise information **[2].** These characteristics have made ANN suitable for solving a variety of problems. ANNs take advantage of the highly nonlinear properties of their architecture and are able to replicate precisely a variety of dynamics given appropriate training. Large amount of experimental or simulation data are required to train ANNs. Although able to represent complex nonlinearity, ANNs give little insight into the system physics.

[3] describes learning vector quantization neural networks that have the ability to identify patterns of network attacks to enhance the security of wireless mobile ad hoc networks (MANET) by

developing a distributed intrusion detection capability.[4] reviews the overview of the dynamic shortest path routing problem and the various neural networks to solve it. Different shortest path optimization problems can be solved by using various neural networks algorithms. The neural networks are the best candidates for the optimization of dynamic shortest path routing problems due to their fastness in computation comparing to other soft computing and metaheuristics algorithms. Wireless network resource use depends in large part on the mobility of network users. The ability to predict this mobility at least in part enables the network to anticipate resource use in the future and take precautionary measures if necessary. [5] Presents a neural network prediction system that is able to capture some of the patterns exhibited by users moving in a wireless environment and can then predict the future behaviour of these users.

The performance of mobile ad hoc networks (MANETs) depends upon a number of dynamic factors that ultimately influence protocol and overall system performance. Adaptive protocols have been proposed that adjust their operation based on the values of factors, such as traffic load, node mobility, and link quality. [6] Has proposed self-controller to determine a set of factor values that will maximize system performance or satisfy specific performance requirements. The model-based controller adapts or reconfigures system-wide parameters or protocol operation as a function of the dynamically changing network state. [7] Has proposed, a recurrent neural network for long-term time series prediction, since mobility prediction is a particular problem of time series prediction. [8] Proposes a novel reverse on-demand routing protocol for mobile ad hoc networks based on best route selection with recurrent neural networks. In path discovery phase, the source node select best route that has high stability between available routes.

Continually broadcasting the Hello messages helped to get a clearer view of the local network topology, it also produced some drawbacks for the whole network in general. Increased number of these messages consumes network resources and bandwidth, increases collisions and interferences with data and control messages, and consumes the limited nodes' battery life during sending and receiving operations. On the other hand, the decreased number of Hello messages results in a time gap between a link failure event and its detection. In essence, it means that the protocol designer has to trade-off sending these messages carefully to represent the real needs for connectivity updating.

The potential benefits of neural networks extend beyond the high computation rates provided by massive parallelism. The neural network models are specified by the net topology, node characteristics, and training or learning rules. These rules specify an initial set of weights and indicate how weights should be adapted during use to improve performance. The neural Network is trained to decide the hello interval based on information of transmission power and speed/mobility of nodes.

7.2.1 ANN Configuration

Artificial Neural Networks [1] have been shown to have the potential to perform well for classification problems in many different environments, including business, science and engineering. The majority of the studies rely on a gradient algorithm, typically a variation of backpropagation, to obtain the weights of the model. Although, the limitations of gradient search techniques applied to complex nonlinear optimization problems, such as the artificial neural network, are well known, many researchers still choose to use these methods for network optimization [9].

A feedforward ANN is the topology has no closed paths. Its input nodes are the ones with no arcs to them, and its output nodes have no arcs away from them. All other nodes are hidden nodes. When the states of all the input nodes are set, all the other nodes in the network can also set their states as values propagate through the network. The operation of a feedforward network consists of calculating outputs given a set of inputs in this manner. A layered feedforward network is one such that any path from an input node to an output node traverses the same number of arcs. The nth layer of such a network consists of all nodes which are n arc traversals from an input node. A hidden layer is one which contains hidden nodes. Such a network is fully connected if each node in layer i is connected to all nodes in layer i+l for all i.

Layered feedforward networks have a training algorithm called backpropagation exists which can often find a good set of weights (and biases) in a reasonable amount of tune **[10]**. Backpropagation is a variation on gradient search. It generally uses a least-squares optimality Machine Learning criterion. The key to backpropagation is a method for calculating the gradient of the error with respect to the weights for a given input by propagating error backwards through the network.

Since it is possible to generate I/O training pairs from the existing environment and classical procedures, we have selected a feed forward ANN with multi layers perceptron model, with 2 nodes in the input Layer, 10 nodes in hidden layer and 1 node in the output layer. The input layer neurons have linear activation characteristics while the hidden and output layers have a sigmoid tan-type activation function to produce bipolar outputs.

Network dimension must satisfy at least two criteria:

- The network must be able to learn the input data;
- The network must be able to generalize for similar input data that were not in training set.

[1] **To be Publish** **(under Publication)** Paper entitled **"Performance Optimization of Reactive Routing Protocol for Mobile Ad-Hoc Network using Artificial Neural Network"** in the **International Journal of Sensors and Actuators, IGI Publication**.

7.2.2 Training Algorithm

Figure 7.2(a) shows the inputs and output parameter to the ANN, hello interval is decided by the ANN based on the change in inputs Tx power and mobility of the nodes in the WANET. **Figure 7.2(b)** describes the detail layered architecture of ANN with 2 nodes in input layer, 4 nodes in hidden layer and 1 node in output layer.

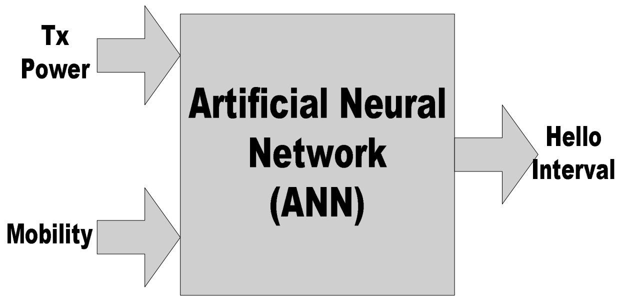


Figure 7.2(a): ANN with respective inputs and outputs

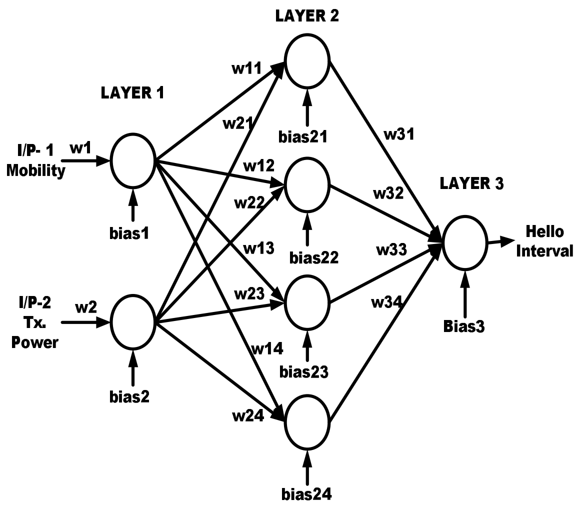


Figure 7.2(b): ANN layered Architecture with neurons

Fuzzy AODV [9-11] was used to generate the training pairs for I/O pairs (P, T) for training ANN [12]. The MATLAB code segment to generate training pairs decided by FIS is given as follows.

```
for txpower = 8:1:14
   for mobility = 5:1:15
      txpower
      mobility
      fis=readfis('hello.fis');
      HI=evalfis([txpower,mobility],fis)
   end
end
```

Neural networks are adjusted, or trained, so that a particular input leads to a specific target output. The ANN weights are adjusted based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are needed to train a network. Process of optimization of MANET using Artificial Neural Network is shown in **Figure 7.3**.

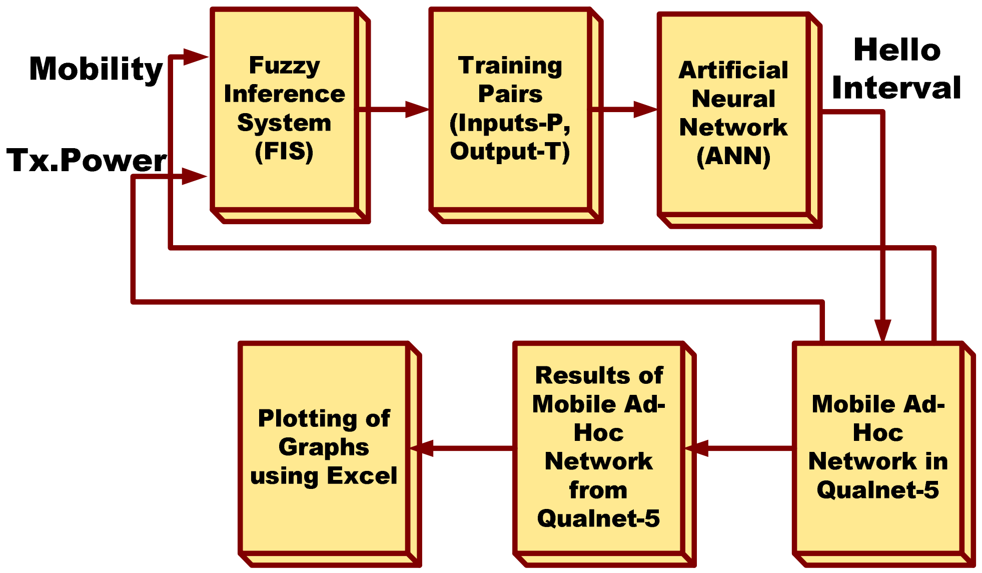

Mobility
Tx.Power
Fuzzy Inference System (FIS)
Training Pairs (Inputs-P, Output-T)
Artificial Neural Network (ANN)
Hello Interval
Mobile Ad-Hoc Network in Qualnet-5
Results of Mobile Ad-Hoc Network from Qualnet-5
Plotting of Graphs using Excel

Figure 7.3: Process of Optimization of MANET

Algorithm for training and obtaining results by using parameters decided from ANN is as follow.

- Obtain the input and output parameters to be optimized.
- Obtain the training pairs for ANN. In this paper it was generated by Fuzzy Inference system.
- After training, obtain the trained ANN.
- Give the inputs to the ANN
- Obtain the output from ANN.
- Open Qualnet and created wireless network.
- Set the input parameters mobility and transmitted power of the nodes in wireless network.
- Also set parameters decided by ANN into the wireless network in this paper it is HELLO INTERVAL value.
- Simulate the network created in the Qualnet for desired parameter value.
- Observe and analyze the results.
- Plot the results in Excel for the comparison of Traditional and proposed method.

P for input and T for target output considered. Conventional Back propagation training method is used. The MATLAB [11, 12] code segment to generate the ANN block is listed below:

```
net=newff(minmax(P),[2,10,1],{'tansig','purelin','tansig'},
'trainlm');
net.trainParam.show = 50;
netnn.trainParam.lr = 0.0001;
net.trainParam.epochs = 20;
net.trainParam.goal = 0.01;
[net,tr]=train(net, P, T);
Y= sim(net, P);
```

Training of neural network tries to achieve the goal of 1e-2 within 20 epochs; the goal performance is 0.0473868.The **Figure.7.4** shows the training of the neural network.

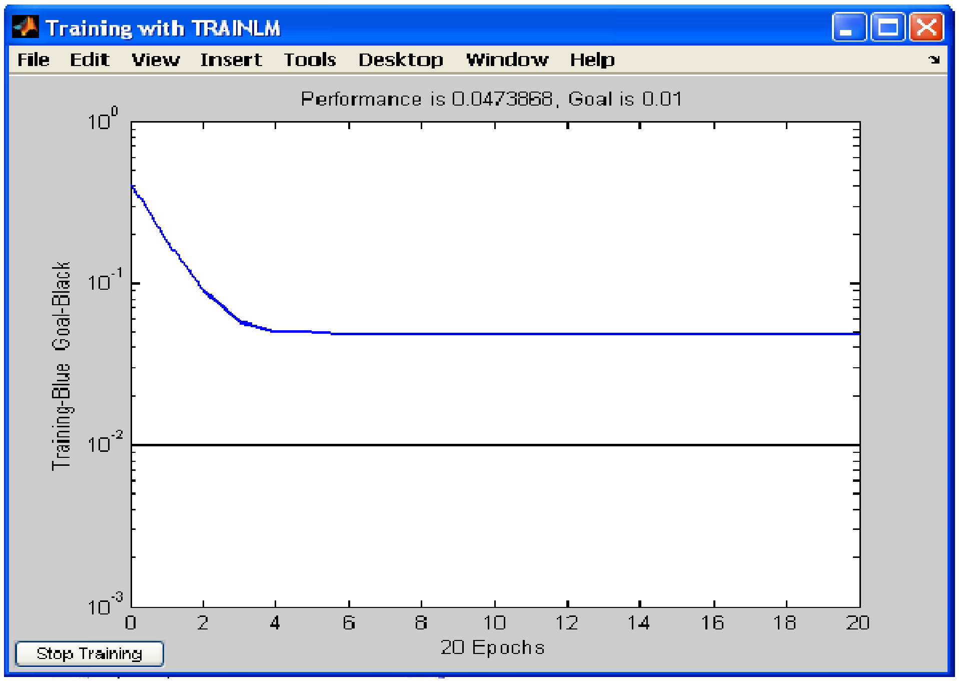
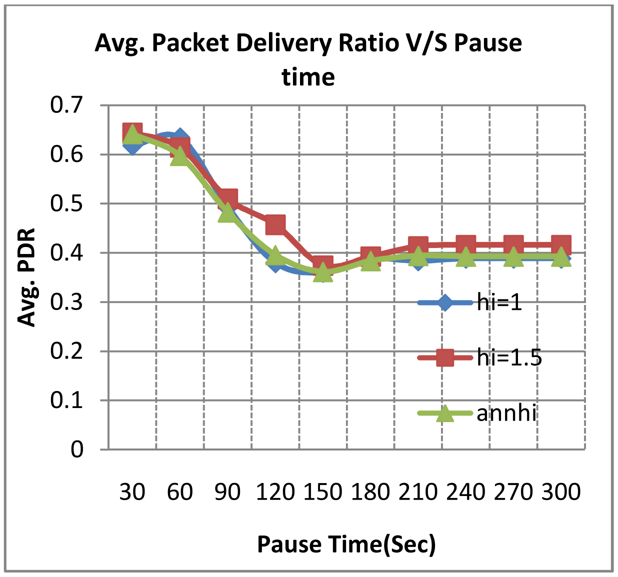


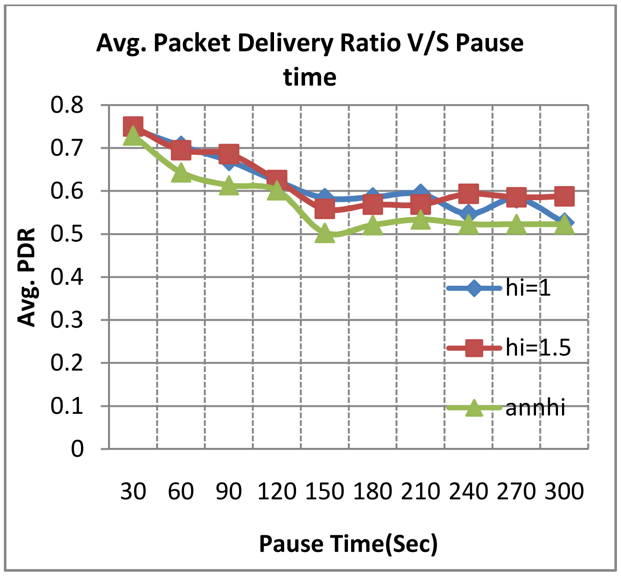
Figure 7.4: Training of Artificial Neural Network Hello Interval (HI)

7.2.3 Implementation of ANN-AODV[2]

The simulation results were carried out using Qualnet 5simulator and MATLAB to evaluate the performance of the AODV routing protocol on the MANET. MANET with the 50 nodes was created in Qualnet 5 simulator with the parameters selection as described in **Section 6.5.** The simulation results obtained using hybrid simulation in MATLAB and QUALNET are as follows shown in figures using different performance metrics used for wireless ad hoc network to evaluate the performance of routing protocol.

[2] **Published in Proceeding of International conference as well as on IEEE website Paper entitled "Development and Simulation of Artificial Neural Network based decision on parametric values for Performance Optimization of Reactive Routing Protocol for MANET using Qualnet" called "COMPUTATIONAL INTELLIGENCE AND COMMUNICATION NETWORKS CICN 2010" page 167-171 held at State University the R.G.P.V., Bhopal, India on 26-28 Nov 2010 organized by MIR Labs (USA) and RGPV Bhopal and co-sponsored by IEEE SMC Society, Mumbai Section.**

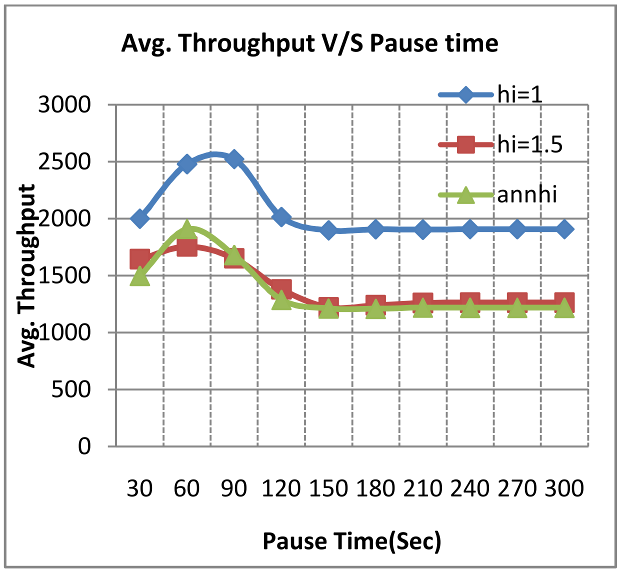


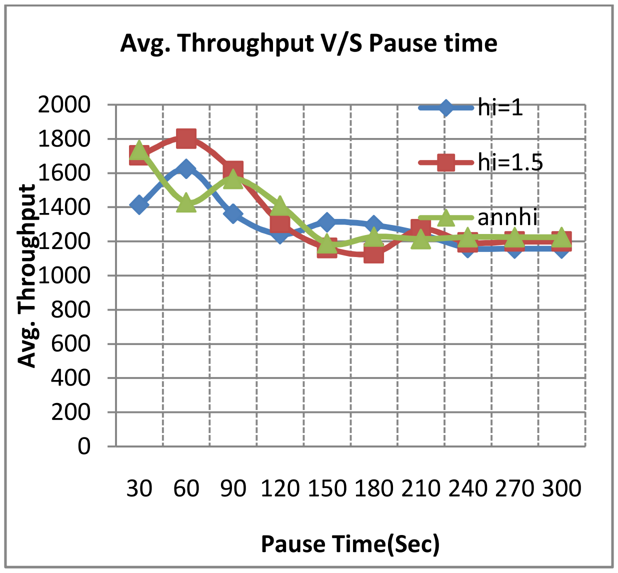


(a) Nodes = 30 **(b) Nodes = 40**

Figure 7.5: Average Packet Deelivery Ratio V/S Pause Time

Figure 7.5 shows the effect in Packet Delivery Ratio of the AODV protocol with respect to change in the pause time or change in mobility of the nodes present in the wireless ad hoc network for the number of nodes 30 and 40.The Packet Delivery Ratio Performance for the hello interval determined using ANN is better than the standard HI=1 when the mobility is higher.

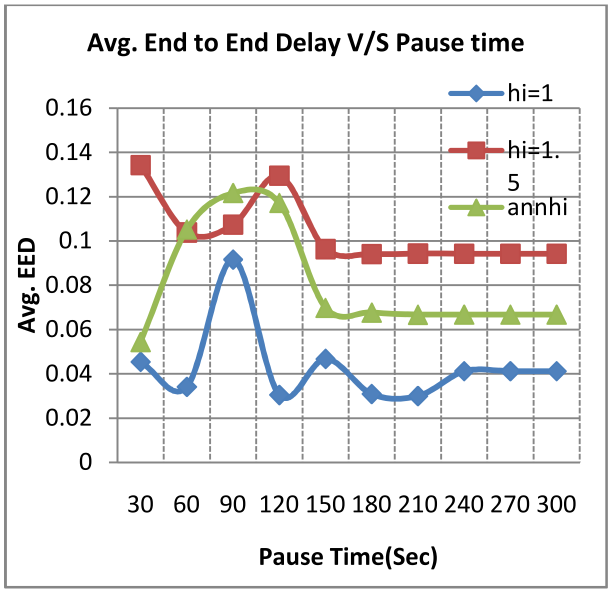


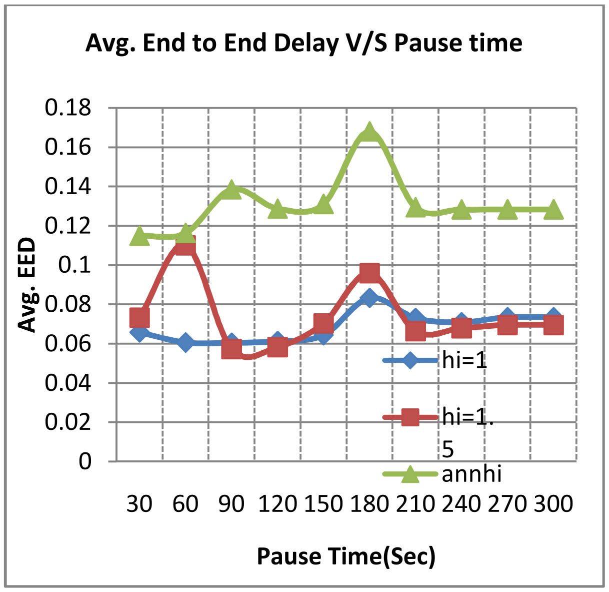


(a) Nodes = 30 **(b) Nodes = 40**

Figure 7.6: Average Throughput V/S Pause Time

The Throughput and Received packets performance for the ANN Hello interval is almost in between the standard hello interval which is shown in the **Figure 7.6** for the nodes 30 and 40.

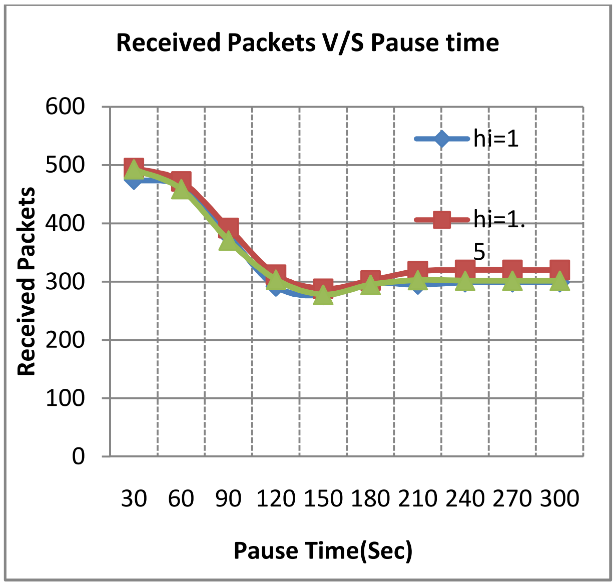


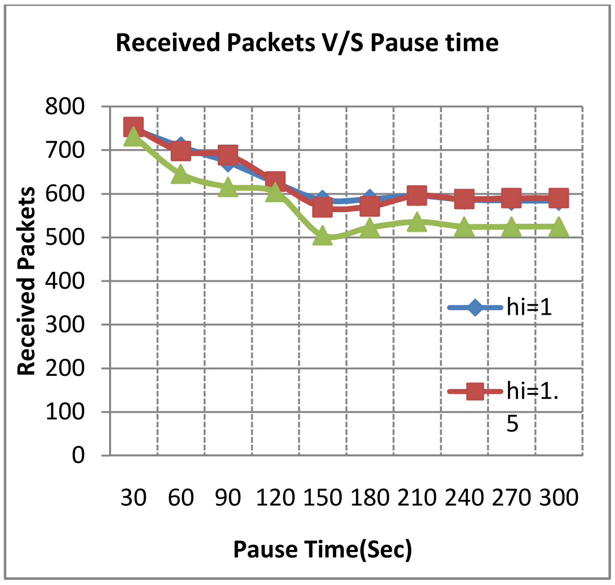


(a) Nodes = 30 (b) Nodes = 40

Figure 7.7: Average End to End Delay V/S Pause Time

Here the Average end to end delay is large in ANN HI in the case of nodes=40 while it is less in nodes=30 than the standard fix hello interval, which can be observed from the **Figure 7.7**.





(a) Nodes = 30 (b) Nodes = 40

Figure 7.8 : Received Packets V/S Pause Time

The performance of Received Packets v/s pause time can be observed from **Figure 7.8**. From the figure we can observe that hello interval decided by ANN has almost the same performance as for the fixed value of hello interval of 1sec and 1.5 sec. As the number of nodes increases the fixed value of hello interval performs better than the ANN based hello interval.

7.3 Genetic Algorithm (GA): an Introduction

During the last decade, GAs has been applied in a variety of areas, with varying degrees of success within each. A significant contribution has been made within control systems engineering, in science and engineering as adaptive algorithms for solving practical problems. Certain classes of problem are particularly suited to being tackled using a GA based approach. GAs has been used to optimize both structure and parameter values for both controllers and plant models. Hybrid approaches have been proved popular, with GAs being integrated in fuzzy logic and neural computing schemes. However, enhanced synergy of GAs with Fuzzy Inference System(FIS) and Artificial Neural Networks (ANN) and proper fitness functions to form hybrid AI systems, will harness fully the strengths of each technique, thus providing GAs a niche role in future intelligent systems design.

The key operators of a GA are selection, crossover, mutation and population size, with secondary parameters including variable encoding and decoding, and population-update. Genetic algorithms are algorithms for optimization and learning based loosely on several features of biological evolution. They require five components:

- A way of encoding solutions to the problem on chromosomes.
- An evaluation function that returns a rating for each chromosome given to it.
- A way of initializing the population of chromosomes.
- Operators that may be applied to parents when they reproduce to alter their genetic composition. Included might be mutation, crossover (i.e. recombination of genetic material), and domain-specific operators.
- Parameter settings for the algorithm, the operators, and so forth.

Neural networks and genetic algorithms are two techniques for optimization and learning, each with its own strengths and weaknesses. The two have generally evolved along separate paths. However, recently there have been attempts to combine the two technologies. Whitley (1988) attempted unsuccessfully to train feedforward neural networks using genetic algorithms. [13] Describe a different genetic algorithm for training feedforward networks. It not only succeeds in its task but it outperforms backpropagation, the standard training algorithm on a difficult example. This success comes from tailoring the genetic algorithm to the domain of training neural networks.

[14] Presents different procedures for the optimization of ANN with aim to: solve the time-consuming of learning process, enhancing generalizing ability, achieving robust and accurate model, and to reduce the computational complexity. A Genetic Algorithm (GA) has been used to optimize

operational parameters (input variables), and to optimize neural network architecture (i.e. number of hidden layer and neurons per layer), weight, types, training algorithms, activation functions, learning rate, momentum rate, number of iterations, and dataset partitioning ratio. A hybrid neural network and genetic algorithm model for the determination of optimal operational parameter settings. The preliminary result of the model has indicated that the new model can optimize operational parameters precisely and quickly, subsequently, satisfactory performance.

The proposed procedure used in this work for training the parameters of ANN. To illustrate the crossover operation in this task, two chromosomes may be selected from the population according to its fitness values. These chromosomes may have different number of hidden nodes and consequently different number of weights. The unused position of the string would take the zero value. The applicability of GAs with real-coding has made its existing operators to be used without any modification while handling these variable strings length. The procedure as shown in **Figure 7.9** goes as follows.

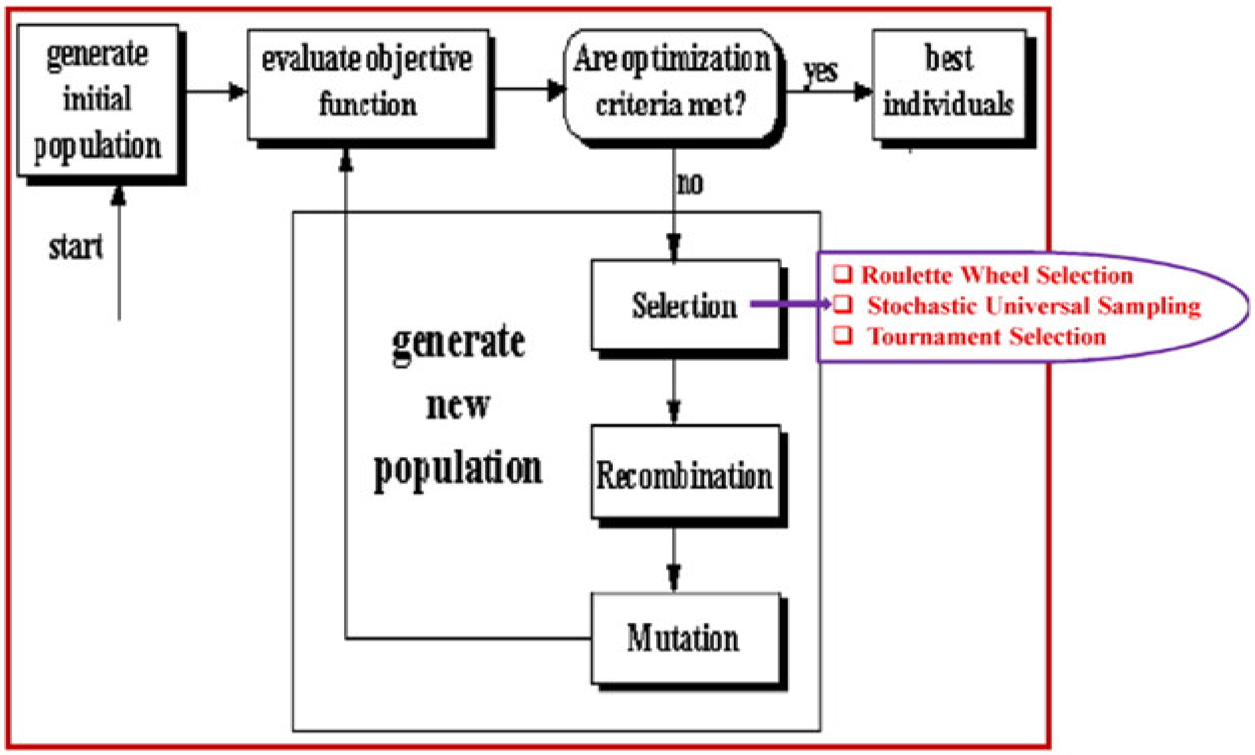


Figure 7.9: Genetic Tuning Algorithm

1. Initialize the genetic operators: *Cross over probability*, *Mutation probability, number of population and* the *maximum number of generations*.
2. Generate an initial population of the network at random. The number of hidden nodes and the initial connection density for each network is uniformly generated at random within certain

ranges. The random initial weights are uniformly distributed inside a small range, typically between -1 and +1.

3. First take the number of hidden nodes, *nh* as an integer number from the end of each chromosome from the population. Then, partially train each network on the training set to evaluate the objective function (for example *MSE*), and then calculate the fitness function.
4. Place in descending order, all chromosomes in the current population, (the first one is the fittest).
5. Select individuals' chromosomes using hybrid selection method (Roulette Wheel plus deterministic selection). The real coded genetic operators of mutation and crossover (single point) are applied.
6. Stop if a maximum number of generations of GAs are achieved, otherwise increase the generations by one and go to Step3 [15].

For instance a meta-learning evolutionary neural network is presented [16] to combine the learning of the connection weights and topology on predicting some time series problems. While in an intrusion detection method which is presented in [17], the evolutionary artificial neural network is used to find the optimal network topology and weights. GA is actually used o initialize the network's weights and the BP to perform a local search and train the network. In [18] the authors employ GA to estimate the weights and the number of neurons in the hidden layer of a Radial Basis Function (RBF) network and it is then used to predict time series data. In [19] a grammatical evolution method called Context Free Grammer (CFG) is presented to construct and train the Neural Network topology along with the network parameters (input vector, weights, bias). The combination of a CFG and a genetic algorithm is known as grammatical evolution and has the benefit of allowing easy shaping of the resulting search space. Mind Evolutionary Computation (MEC) is another method for weight optimization of a neural network which is based on simlartaxis and dissimilation operators of weights optimization [20].

7.4 GA based ANN

To develop an accurate process model using ANN, the learning process or training and validation are among the important steps. In the training process, a set of input-output patterns is repeated to the ANN. From that, weights of all the interconnections between neurons are adjusted until the specified input yields the desired output. Through these activities, the ANN learns the correct input-output response behaviour. For validation, the ANN is subjected to input patterns unseen during training, and introduces adjustment to make the system more reliable and robust. It is also used to determine the stopping point before over fitting occurs. A typical fitting criterion may be introduced to emphasis the model validity. Such criterion may be mean square error (MSE), sum square error (SSE) which is calculated between the target and the network output. Research on using genetic algorithms for neural networks learning is increasing. Typically, genetic search is used for the weights optimization on a pre-specified neural network

Genetic algorithms are stochastic search techniques that guide a population of solutions towards an optimum using the principles of evolution and natural genetics. In recent years, genetic algorithms have become a popular optimization tool for many areas of research, including the field of system control, control design, science and engineering. Significant research exists concerning genetic algorithms for control design and off-line controller analysis.

Genetic algorithms are inspired by the evolution of populations. In a particular environment, individuals which better fit the environment will be able to survive and hand down their chromosomes to their descendants, while less fit individuals will become extinct. The aim of genetic algorithms is to use simple representations to encode complex structures and simple operations to improve these structures. Genetic algorithms therefore are characterized by their representation and operators. In the original genetic algorithm an individual chromosome is represented by a binary string. The bits of each string are called genes and their varying values alleles. A group of individual chromosomes are called a population. Basic genetic operators include reproduction, crossover and mutation. Genetic algorithms are especially capable of handling problems in which the objective function is discontinuous or non differentiable, non-convex, multimodal or noisy. Since the algorithms operate on a population instead of a single point in the search space, they climb many peaks in parallel and therefore reduce the probability of finding local minima.

Genetic algorithms and artificial neural networks are both techniques for learning and optimization, which have been adopted from biological systems. Neural networks use inductive learning and in general require examples, while GAs uses deductive learning and require objective evaluation function. A synergism between the two techniques has been recognized which can be applied to enhance each technique performance in what may be referred to as evolutionary neural networks. An area that has attracted the most interest is the use of GAs as an alternative learning technique in place of gradient descent methods, such as, error backpropagation. Supervised learning algorithms suffer from the possibility of getting trapped on suboptimal solutions. GAs enables the learning process to escape from entrapment in local minima in instances where the backpropagation algorithm converges prematurely. Furthermore, because GA does not function in the task domain they may be used for weight learning in recurrent networks where suitable training algorithms are still a problem. Studies have been attempted to take advantage of both techniques. Algorithms, which combine GAs and error backpropagation, have been shown to exhibit better convergence properties than the pure backpropagation. The GA is used to rapidly locate the region of optimal performance and then gradient descent backpropagation can be applied in this region. GAs have also been studied as generalized structure/parameter learning in neural systems. This type of learning combines as complimentary tools both inductive learning through synaptic weight adjustment and deductive learning through the modification of network topology to obtain automatic adaptation of system knowledge of the domain environment. Such hybrid systems are capable of finding both the weights and the architecture of a neural network, including number of layers, the processing elements per layer and the connectivity between processing elements. In summary, GAs has been used in the area of neural networks for three main tasks: -Training the weights of the connections, designing the structure of the network and finding an optimal learning rule.

Pham and Karaboga **[21]** look at using a genetic algorithm to train the weights of recurrent neural networks, assuming that the structure of the network has been decided. Structure including, the number of layers, the type and number of neurons, the pattern of connections, the permissible ranges of trainable connection weights, and the values of constant connection weights, if any. The initial set of solutions is produced by a random number generator. Each solution in the population is a string comprising n elements, where n is the number of trainable connections. They use binary encoding, where each element is 16 bits long and holds the value of a trainable connection. They found that 16 bits gave adequate resolution for both feed forward and feedback connections. From the point of view of the GA, all connection weights are handled in the same way, i.e. training of feedback connections is carried out identically to training the feedforward connection, (different from the backpropagation algorithm). **[22]** Also use genetic algorithms for the training of recurrent neural networks, pointing out that when the backpropagation gradient descent algorithm is applied to recurrent neural networks; it is more complicated than for feedforward networks due to the many attractors in the state space. It also investigated the use of genetic algorithms for automated selection of parameters in an ad hoc networking system. It provides experimental results demonstrating that the genetic algorithm can optimize for different classes of operating conditions. In it also compare genetic algorithm optimization against hand-tuning in a complex, realistic scenario and show how the genetic algorithm provides better performance.

A sequence of input signals is fed to both plant and neural network and the output signals from both are compared. The absolute difference is computed, and the sum of all errors for the whole sequence is used as a measure of fitness for the particular network under consideration **(Figure. 7.10).** Genetic operators were applied to create a new population.

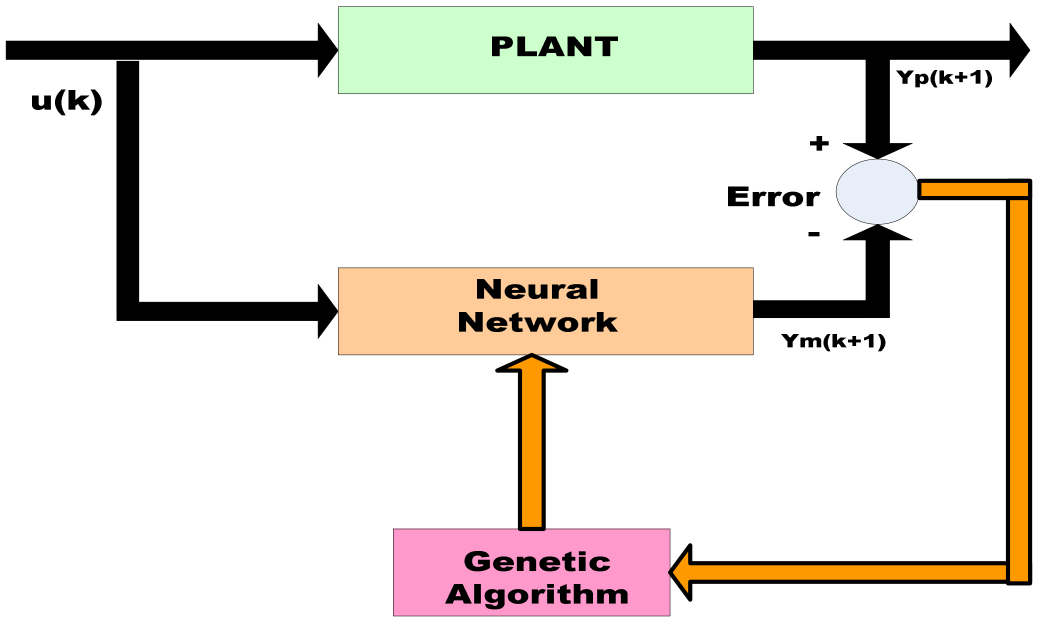


Figure 7.10: Schemes for Training a Neural Network to Identify a Plant

7.4.1 Chromosomes

A chromosome should include information on solution that it represents. Each chromosome is set up with binary series. Each number that named bit in this series can represent a characteristic of the solution or, a serial, on its own, would indicate a number. Expressing the chromosomes with the set of numbers in the binary series is the most common representation form; however, also integer and real numbers can be used. The reason for selecting for binary series is as the following: first of all it is simple, and secondly, it is processed by the computer easier and faster. Due to the floating point values of weights here we have defined real numbers for chromosome.

The reproduction process is a process which is applied according to certain selection criteria to reproduce new generation. A selection criterion takes the compatibility as a basis and selects the compatible members. At the stage, it is possible that more compatible new members will emerge from those members who will be subjected to crosswise and fission. All members may be selected in terms of compatibility or some are selected randomly and transferred to next generation. Crossover can be applied after the decision for representation of chromosomes taken.

Following **Figure 7.11** shows the weights to be trained by the genetic algorithm for each layer of ANN. Layer 1 of ANN is input layer with the 2 nodes, which includes two biases and four weights. Layer 2 of ANN is hidden layer with the 4 nodes, 4 bias values and 8 weights. Layer 3 is output layer consist of one node with one bias value and four weight values.

LAYER 1(2nodes)
INPUT LAYER

LAYER 2(4 nodes)
HIDDEN LAYER

LAYER 3(1 node)
OUTPUT NODE

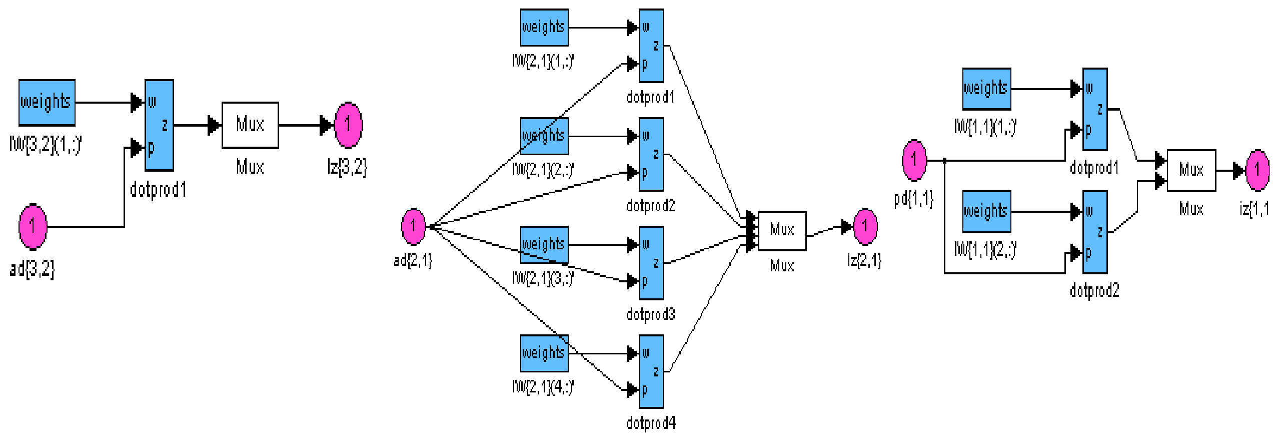


Figure 7.11: Weights of Each Layer of ANN

Figure 7.12 shows schematics of used real coded GA chromosome including weights of input to hidden layer, biases of hidden layers , weights of context layer , weights of hidden to output layer, biases of output layer (1=single output).

Bias (Input Layer)	**Bias (Hidden Layer)**	**Bias (Output Layer)**	**Weights (Input Layer)**	**Weights (Hidden Layers)**	**Weights (Output Layer)**

Bias (Layer1)		**Bias (Layer2)**				**Bias (Layer3)**	**Weights Layer1**			
1	2	3	4	5	6	7	8	9	10	11
B11	B12	B21	B22	B23	B24	B31	W11	W21	W12	W22
11.1372	**-2.705**	**-3.536**	**-3.704**	**1.232**	**1.533**	**0.363**	**0.281**	**0.596**	**0.236**	**-0.448**

Weights(Layer 2)								**Weights(Layer3)**			
12	13	14	15	16	17	18	19	20	21	22	23
W211	W231	W212	W232	W221	W241	W222	W242	W31	W32	W33	W34
1.875	**0.408**	**-0.270**	**0.718**	**-1.749**	**-0.651**	**-0.379**	**-0.256**	**-0.717**	**0.247**	**-0.662**	**0.745**

Figure 7.12: Arrangement of Weights and Biases in the GA Chromosome

The control parameter maintained in simulations were as follows in **Table 7.1,**

Control Parameter	**Values**
Population size	**50**
Generation gap	**0.98**
Crossover rate	**0.85**
Mutation rate	**0.01**
Max. Generations	**30**

Table 7.1: parameters for GA

The major drawback of the GA compared to the back propagation algorithm is that the GA is inherently slower as it has to operate on the weights of a population of neural network whereas the Back propagation algorithm deals only with the weights of one network.

The present work describes a technique of an Elman neural network trained by genetic algorithm. We used encoding strategy for weights and biases, where real coded chromosomes were used to manage the large number of weights and biases needed. **Following code segment shows** used real coded GA chromosome including weights of input to hidden layer (HN*Inputs), biases of hidden layers (HN), weights of context layer (HN*HN), weights of hidden to output layer (HN), biases of output layer (1=single output).

```
BSLBL1 = [0 -4]; %layer 1 bias lower bounds
BSUBL1 = [12 4]; %layer 1 bias upper bounds

BSLBL2 = [-4 -4 -4 -4]; % layer 2 bias lower bounds
BSUBL2 = [4 4 4 4];     %layer 2 bias upper bounds

BSLBL3 = 0; %layer 3 bias lower bounds
BSUBL3 = 2; %layer 3 bias upper bounds

%Input weight lower bound, and upper bound

IWLBC1 = [-1 -1 -1 -1];%layer 1 weights lower bounds [w11 w21 w12 w22]
IWUBC1 = [1 1 1 1];    %layer 1 weights upper bounds [w11 w21 w12 w22]
%Layer weights

LWLBL2 = [0 -1 -3 -3 -2 -2 -3 -1];% lower bounds weights layer 2 [w211 w231
w212 w232 w221 w241 w222 w242]
LWUBL2 = [2 1 3 3 2 2 3 1];% upper bounds weights layer2

LWLBL3 = [-4 -1 -1 -3];% lower bounds weights of layer 3[w211 w231 w212 w232
w221 w241 w222 w242]
LWUBL3 = [4 1 1 3];%upper bounds of layer 3 weights

%arrangement of chromosomes
FieldDR = [BSLBL1 BSLBL2 BSLBL3 IWLBC1 LWLBL2 LWLBL3;
           BSUBL1 BSUBL2 BSUBL3 IWUBC1 LWUBL2 LWUBL3]
```

7.4.2 Flow chart

Above algorithmic steps are well defined in the flow chart shown in **Figure 7.13**.

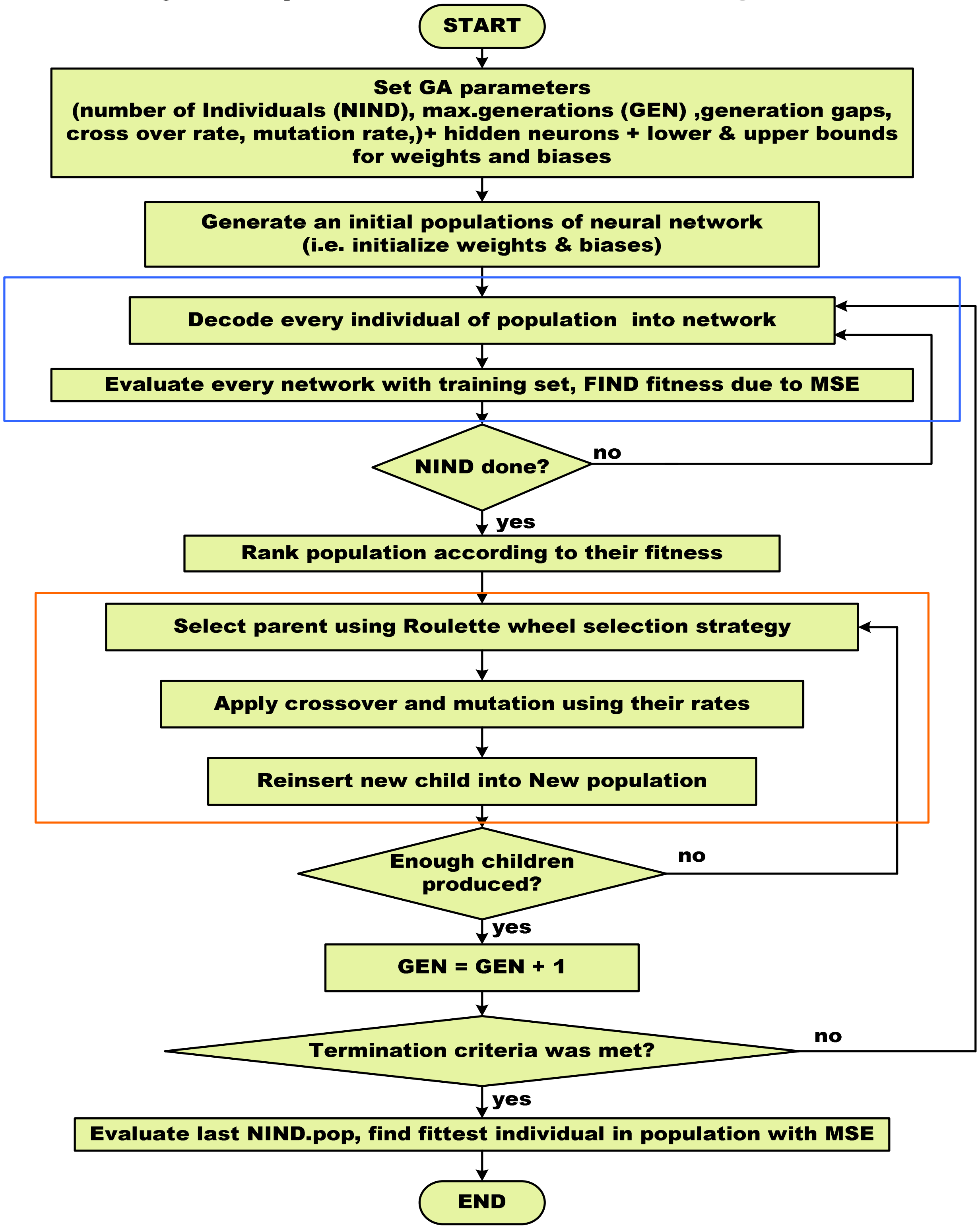

START
Set GA parameters
(number of Individuals (NIND), max.generations (GEN) ,generation gaps, cross over rate, mutation rate,)+ hidden neurons + lower & upper bounds for weights and biases
Generate an initial populations of neural network (i.e. initialize weights & biases)
Decode every individual of population into network
Evaluate every network with training set, FIND fitness due to MSE
NIND done?
no
yes
Rank population according to their fitness
Select parent using Roulette wheel selection strategy
Apply crossover and mutation using their rates
Reinsert new child into New population
Enough children produced?
no
yes
GEN = GEN + 1
Termination criteria was met?
no
yes
Evaluate last NIND.pop, find fittest individual in population with MSE
END

Figure 7.13: Flow Chart of the Proposed Algorithm

7.4.3 Proposed Genetic Algorithm

Step 1: Represent the problem variable domain as a chromosome of a fixed length, choose the size of a chromosome of population NIND (number of individuals), the crossover probability XOVRATE and the mutation probability MUTRATE, generation gap GGAP, termination criteria MAXGEN, see flowchart, shown in **Figure 7.13.**

```
Nind= 50;%number of individuals
HN =7;%number of hidden nodes
MAXGEN =30; %maximum generations
GGAP = 0.98;%generation gaps
XOV_RATE = 0.85;% cross over rate
MUT_RATE = 0.01;% mutation rate
```

Step 2: Define a fitness function to measure the performance, or fitness, of an individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction, mean square error (MSE) was chosen here.

Step 3: Randomly generate an initial network population of chromosomes of size S, see **Figure 7.12.**

```
Chrom = rand(Nind,Nvar) .* Range + Lower;
```

Step 4: Decode every individual created of NIND into network it represents.

Step 5: Evaluate every network of population we got with training data, calculate:

Error = Target - predicted,

Then MSE to find fitness, objective is to minimize MSE.

Step 6: Rank population of networks (chromosomes) generated according to their fitness.

```
FitnV = ranking(ObjV);
```

Step 7: Select a pair of chromosomes for mating from the current population, Roulette Wheel. Parent chromosomes are selected with a probability related to their fitness.

```
ObjV = ffnnga_evaluation(Chrom, Nind, HN);
min(ObjV)
```

Step 8: Create a pair of offspring chromosomes by applying the genetic operator - crossover and mutation.

Step 9: Insert the created offspring chromosomes in the new population.

```
%Evaluate offspring
SIND = [size(SelCh)]*[1;0];
ObjSel = ffnnga_evaluation1(SelCh, SIND, HN);
[Chrom ObjV] = reins(Chrom, SelCh, 1, 1, ObjV, ObjSel);
```

```
GA_Perf(gen) = min(ObjV);
gen = gen+1;
[Best_Perf Index] = min(ObjV)
```

Step 10: Repeat Step 7 until the size of the new chromosome population becomes equal to the size of the initial population, NIND.

Step 11: Replace the initial (parent) chromosome population with the new (offspring) population.

Step 12: Go to Step 4, to evaluate new created offspring and repeat the process until the termination criterion is satisfied, maximum number of generation is met (MAXGEN).

7.4.4 Files Used for GA Based ANN

MATLAB files and functions crated to optimize the ANN using Genetic algorithm are listed below in **Table 7.2.**

Sr.No.	Name	Purpose
1.	**Mynnffga.m**	**program of ANN training by Genetic Algorithm**
2.	**ffnnga_evaluation.m**	**Evaluates the fitness function**
	AODV_Crtrp.m	**This function creates a population of given size of random real-values.**
3.	**AODVMutate.m**	**This function takes a matrix OldChrom containing the representation of the individuals in the current population, mutates the individuals and returns the resulting population.**
4.	**aodvranking.m**	**This function performs ranking of individuals**
5.	**aodvrecombine.m**	**This function performs recombination between pairs of individuals and returns the new individuals after mating. The function handles multiple populations and calls the low-level recombination function for the actual recombination process.**
6.	**Aodv_Xovmp.m**	**applies crossover to consecutive pairs of individuals with probability Px and returns the resulting population**
7.	**aodvSelect.m**	**This function performs universal selection. The function handles multiple populations and calls the low level selection function for the actual selection process**
8.	**aodvReins.m**	**This function reinserts offspring in the population.**
9.	**Susaodv.m**	**This function performs selection with STOCHASTIC UNIVERSAL SAMPLING.**
10.	**RWSAODV.m**	**This function selects a given number of individuals Nsel from a population. FitnV is a column vector containing the fitness values of the individuals in the population.**

Table 7.2: MATLAB files related to Genetic Algorithm

Sr. No.	Tx Power(mW)	Mobility (m/sec)	Hello Interval(sec) (Target)	ANN	GA – ANN (Relative Difference)	GA – ANN (Relative Difference)
1	8	5	0.7902	0.6103	0.5704 (0.2198)	0.5082 (0.2820)
2	9	7	0.4911	0.5702	0.5410 (0.0499)	0.4831 (0.008)
3	10	9	0.6210	0.5430	0.5233 (0.0977)	0.4675 (0.1535)
4	11	11	0.4996	0.5551	0.5378 (0.0382)	0.4916 (0.008)
5	12	13	0.7824	0.7798	0.7480 (0.0344)	0.7654 (0.017)
6	13	14	0.9983	0.9993	0.9988 (0.0005)	0.9996 (0.0013)
7	14	15	0.9983	1	1 (0.0017)	1 (0.0017)

Table 7.3: Result Comparison of GA based with Traditional ANN

Result of Genetic algorithm based ANN for training of two trial is shown in **Table 7.3**. From the table we can observe that the GA based ANN gives the nearest result to the target output compared to the traditional trained ANN.

Summary

In this chapter ANN based Hello Interval parameter optimization of reactive routing protocol AODV has been described. Feed forward type Multilayer Perceptron (MLP) neural network is used to decide interval between hello messages instead of using fixed interval of 1 sec or 1.5 sec. comparison of the performance of Routing protocol and WANET is compared using ANN based HI, fixed 1sec and 1.5 sec. ANN is generated and trained traditionally. Genetic Algorithm is used to train the ANN by adjusting weights and biases of the layers of MLP. Also the comparison of traditional trained ANN and GA based ANN is done using Mean Square Error.