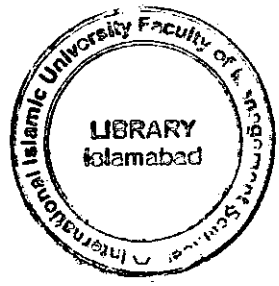
Channel Encoding of Wireless Medium Based on Convolutional Codes Using Neural Networks and Genetic Algorithm

Submitted by

Imran Khan
80-CS/MS/02
Syed Muhammad Saqlain
81-CS/MS/02



Supervised by

Mr. Muhammad Sher

Dr. M. Sikander Hayat Khiyal

Department of Computer Science
Faculty of Applied Sciences
International Islamic University, Islamabad.
(2005)

A dissertation submitted to the
Department of Computer Science
Faculty of Applied Sciences
International Islamic University Islamabad
as a partial fulfillment of the requirement
for the award of the degree of
MS in Computer Science

International Islamic University, Islamabad

31-10
~~September 15~~, 2005

Final Approval

It is certified that we have read the Thesis titled **"Channel Encoding of Wireless Medium Based on Convolutional codes using Neural Networks and Genetic Algorithm"** submitted by Mr. Imran Khan, Reg. No. 80-CS/MS/02 and Mr. Syed Muhammad Saqlain, Reg. No. 81-CS/MS/02 and it is our judgment that this Thesis is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the Degree of MS in Computer Science.

Committee

External Examiner

Dr. Abdul Sattar
H. #. 143, St. #. 60, I-8/3,
Islamabad

Internal Examiner

Dr. Syed Afaq ~~Ahmed~~
Head of Dept. of Telecommunication &
Computer Engineering,
International Islamic University, Islamabad.

Supervisor

Dr. Sikandar Hayat Khiyal
Head of Dept. of Computer Sciences,
International Islamic University, Islamabad.

Mr. Muhammad Sher
Assistant Professor,
Dept. of Computer Sciences,
International Islamic University, Islamabad

on leave abroad

Dedication

We dedicate our this thesis Particularly to our parents and siblings who always prayed for us, Generally to Muslim Ummah and to Zaib who was always behind us encouraging right from start to end of the successful Research.

Imran Khan
Syed Muhammad Saqlain

ACKNOWLEDGEMENTS

All praise to Almighty Allah, the most merciful and compassionate, who enabled us to complete this research work.

We express our gratitude to our kind supervisors *Mr. Muhammad Sher and Dr. Sikandar Hayat Khiyal* who were the men behind the idea of **Channel Encoding of Wireless Medium Based on Convolutional codes using Neural Networks and Genetic Algorithm.** Their motivations lead us to this success and who kept our morale high by their suggestions and appreciation. They were available to us whenever and for whatever we consulted them. Without their precious guidance and help we could never be able to develop such research work.

We shall always remember the cooperation of our friends whether that Co-Operation was technical or moral support was there. Particularly Syed Husnain Abbas Naqvi for his Technical and moral Support, Also the encouragement by Shakeel Ahmed Khan, Saqib Zulfiqar, Ayaz Akhtar, Syed Kashif Abbas, Malik M. Asif, Neyyer Islam, Malik Asim, Murad, Tahir Malik, Ch. Abid, Ali Nadeem and Zaib had a lot meanings to us.

I (Saqlain) am also very thankful to my Senior Office Colleagues particularly Mr. Rizwan Asghar, Mr. Saqib Masood and Mr Tanveer Afsar for their Co-Operation in Completing this Thesis Successfully.

And last but not the least we would like to acknowledge the support of our family members. We would like to admit that we owe all our achievements to our truly, sincere and most loving parents, brothers and sisters, who mean the most to us, and whose prayers are a source of determination for us.

Imran Khan
80-CS/MS/02

Syed Muhammad Saqlain
81-CS/MS/02

DECLARATION

We, hereby declare that this software, neither as a whole nor as a part thereof has been copied out from any source. It is further declared that we have developed this Software and the accompanied thesis entirely on the basis of our personal efforts made under the sincere guidance of our teachers. If any part of this thesis is proved to be copied out or found to be reported, we shall standby the consequences. No portion of the work presented in this thesis has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Imran Khan
80-CS/MS/02

Syed Muhammad Saqlain
81-CS/MS/02

Project in Brief

Project Title:	Channel Encoding of Wireless Medium Based on Convolutional Codes Using Neural Networks and Genetic Algorithm
Organization:	Department of Computer Sciences, International Islamic University, Islamabad.
Undertaken By:	Imran Khan 80-CS/MS/02 Syed Muhammad Saqlain 81-CS/MS/02
Supervised By:	**Mr. Muhammad Sher** Assistant Professor, Department of Computer Science, International Islamic University, Islamabad. **Dr. Sikandar Hayat Khiyal** Head Department of Computer Science Faculty of Applied Sciences International Islamic University Islamabad
Tool Used:	C++, Visual C++, Matlab
Operating System:	Windows 2000 Server / Linux, UNIX, etc.
System Used:	IBM Compatible
Date Started:	15, September 2003.
Date Completed:	15, September 2005.

Abstract

Convolutional codes are generated by continuous performing logic operations on a moving, limited sequence of bits(m bits, say) contained in the message stream. For each bit in the message stream, the encoder produces a fix number (two or more) of bits whose values depend on the value of the present message bit and value of preceding m-1 bits. Convolutional codes employ relatively simple encoders whose span may extend over many bits; the decoders are quite complex. Fundamentally different from block codes, information bits are not collected in a discrete sequence to which check bits are added to form a block or codeword. Instead, a structured mapping technique is used to convert the sequence of information input bits (k) to encoder output bits. Decoding requires the receiver to undo the mapping and determine the most likely known codeword for an errored word

There are different decoding Algorithms used for the decoding of Convolutional Codes all of which have their pits and falls in context of computation delay and the error rate. The Viterbi Algorithm is the optimum decoding algorithm (in the sense of maximum-likelihood decoding of the entire sequence) for Convolutional codes. However, it requires the computation of 2kk metrics at each node of the trellis and the storage of 2k(k-1) surviving sequences, each of which may be 5kKbits long.

The Intelligent systems are the most demanding field in this era. Here we are having two of the most successful applications of AI, Artificial Neural Network and Genetic Algorithm.

This thesis presents the two algorithms for the decoding of Convolutional codes, one using Artificial Neural Network and other using Genetic Algorithm.

Table of Contents

CHAPTER 1
INTRODUCTION

1. Introduction

Over the years, there has been a tremendous growth in digital communications especially in the fields of cellular/PCS, satellite, and computer communication. In these communication systems, the information is represented as a sequence of binary bits. The binary bits are then mapped (modulated) to analog signal waveforms and transmitted over a communication channel. The communication channel introduces noise and interference to corrupt the transmitted signal. At the receiver, the channel corrupted transmitted signal is mapped back to binary bits. The received binary information is an estimate of the transmitted binary information. Bit errors may result due to the transmission and the number of bit errors depends on the amount of noise and interference in the communication channel [1].

Channel coding is often used in digital communication systems to protect the digital information from noise and interference and reduce the number of bit errors. Channel coding is mostly accomplished by selectively introducing redundant bits into the transmitted information stream. These additional bits will allow detection and correction of bit errors in the received data stream and provide more reliable information transmission. The cost of using channel coding to protect the information is a reduction in data rate or an expansion in bandwidth.

1.1 Types of coding

Channel encoding can be of two forms; waveform coding and structured sequential coding.

1.1.1 Waveform Coding

Waveform coding transforms the source data and renders the detection process less subject to errors and thereby improves transmission performance. Waveform coding is some kind of approximately lossless coding, as it deals with speech signal as any kind of ordinary data. The resulting signal is close as possible as the original one. Codecs using this technique have generally low complexity and give high quality at rates $\geq$ 16 Kbps. The simplest form of waveform coding is Pulse Code Modulation (PCM), which involves sampling and quantizing the input waveform. Narrow-band

speech is typically band-limited to 4 KHz and sampled at 8 KHz. Many codecs try to predict the value of the next sample from the previous samples. This is because there is correlation between speech samples due to the nature of speech signal. An error signal is computed from the original and predicted signals. As in most cases, this error signal is small with respect to the original one, it will have lower variance than the original one. Hence, fewer bits are required to encode them. This is the basis of Differential Pulse Code Modulation (DPCM) codecs. They quantize the difference between the original and predicted (from the past samples) signals. The notion of adaptive coding is an enhancement to DPCM coding. This is done by making the predictor and quantizer adaptive so that they change to match the characteristics of the speech being coded. The most known codec using this technique is the Adaptive DPCM (ADPCM) codecs. It is also possible to encode in the frequency domain instead of the time domain (as the above mentioned techniques). In Sub-Band Coding (SBC), the original speech signal is divided into a number of frequency bands, or sub-bands. Each one is coded independently using any time domain coding technique like ADPCM encoder. One of the advantages of doing this is that all sub-band frequencies do not influence in the same way the perceptual quality of the signal. Hence, more bits are used to encode the sub-bands having more perceptually important effect on the quality than those where the noise at these frequencies is less perceptually important. Adaptive bit allocation schemes may be used to further exploit these ideas. SBC produces good quality at bit rates ranging form 16 to 32 Kbps. However, they are very complex with respect to the DPCM codecs. As in video spatial coding, Discrete Cosine Transformation (DCT) is used in speech coding techniques. The type of coding employing this technique is the Adaptive Transform Coding (ATC). Blocks of speech signal is divided into a large numbers of frequency bands. The number of bits used to code each transformation coefficient is adapted depending on the spectral properties of the speech. Good signal quality is maintained using ATC coding at bit rates of about 16 Kbps [1].

1.1.2 Structured Sequential Coding

Structured sequential coding (linear block coding) represents a method of inserting structured redundancy into the source data so that transmission or channel errors can be identified and corrected. Structured sequences are one of two types: block coding and convolutional coding [8].

1.1.2.1 Block codes

Block codes accept a block of k information bits and produce a block of n coded bits. By predetermined rules, n-k redundant bits are added to the k information bits to form the n coded bits. Commonly, these codes are referred to as (n,k) block codes. Some of the commonly used block codes are Hamming codes, Golay codes, BCH codes, and Reed Solomon codes (uses nonbinary symbols) [12].

1.1.2.1.1 Hamming Codes

The Hamming codes are a very interesting and useful family. For example, codes from this family are used in computers for managing disk storage and Ramdom Access Memory (RAM). They are all single-error-correcting

The Hamming code of order r over a field of order q is a linear (n,k)-code, where $n=(q^r-1)/(q-1)$ and $k=n-r$, with parity check matrix H_r an $r\times n$ matrix such that the columns of H_r are non-zero and no two columns are scalar multiples of each other [9].

1.1.2.1.2 Cyclic Codes

Cyclic codes are a sub-class of linear block codes, considered to be the most important class. Where we can derive many other codes and too which many other codes are equivalent. A linear code C is cyclic if $x_0 \ldots x_n$ in C implies $x_n \ldots x_{n-1}$ in C. A cyclic codes can easily be created by performing right cyclic shifts on an initial codeword, developing a new code-word with each shift, which continues until the original codeword is obtained. Put simply, all the elements of a codeword are shifted one space to the right and the element on the end moves to the beginning of the codeword [15].

codewords of C_{23} because C_{23} has Hamming distance 8. Finally it turns out that the Hamming distance for C_{23} is d=7. Thus C_{23} is a binary linear (23,12,7)-code. Now we check that it satisfies the sphere packing bound as equality:

$$|C|\cdot\sum_{i=0}^{3}\binom{n}{i}(q-1)^i=2^{12}(1+\binom{23}{1}(2-1)+\binom{23}{2}(2-1)^2+\binom{23}{3}(2-1)^3)=2^{23}$$

Thus C_{23} is 3-perfect. It is called the binary Golay (23,12,7) code [16].

The ternary Golay (12,6,6) code and its 'family': Begin by letting C_{12} be the ternary linear (12,6)-code with generator matrix $G=[I_6|A]$ where

$$A=\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 1 & 0 & 1 & 2 \\ 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 0 \end{pmatrix}$$

Deciding that d=6 for this code requires some detailed checking.

Now let C_{11} denote the code obtained from C_{12} by deleting the last entry from every codeword. This is equivalent to deleting the last column from the generator matrix G. Then C_{11} has length n=11. Arguing as for the code C_{23} we see that C_{11} is a linear code and its dimension is k=6. On checking the codewords it turns out that the Hamming distance for C_{11} is d=5. Thus C_{11} is a ternary linear (11,6,5)-code.

Now we check that it satisfies the sphere packing bound as equality: [16]

$$|C|\cdot\sum_{i=0}^{2}\binom{n}{i}(q-1)^i=3^6(1+\binom{11}{1}(3-1)+\binom{11}{2}(3-1)^2)=3^{11}$$

Thus C_{11} is 2-perfect. It is called the ternary Golay (11,6,5) code.

The (r+1,r) parity check code

The (r+1,r) parity check code C over a finite field F consists of all r-tuples in F^r with an extra entry added in such a way that the sum of all r+1 entries is 0. Thus The length of this code is n=r+1, and the number of codewords

$$c=(c_1,\ldots,c_r,c_{r+1})\in C \Leftrightarrow \sum_{i=1}^{r+1} c_i=0.$$

is $|F^r|$.It is easy to check that C is a subspace of F^{r+1} of dimension r, and so C is a linear (r+1,r)-code. The r vectors (1,0,0,...,0,−1), (0,1,0,...,0,−1), ...,(0,0,0,...,1,−1) all lie in C and form a basis for C. Hence is a generator matrix for C. The dual code C is

$$G=\begin{pmatrix} & & 1\\ & I_r & :\\ & & 1\end{pmatrix}$$

a linear code of dimension r+1−dimC=1 and contains $\mathbf{h}=(1,1\ldots,1)$ since $G\mathbf{h}^T=\mathbf{0}$. Hence is a parity check matrix for C. It is easy to see that the Hamming distance d for C is d=2.

$$H=\begin{pmatrix} & & \\ 1 & 1 & \ldots\ 1\\ & & \end{pmatrix}$$

The r-fold repetition code C of dimension k over a finite field F has as codewords all vectors $\mathbf{c} \in F^{kr}$ of the form $\mathbf{c}=(\mathbf{x},...,\mathbf{x})$ where $\mathbf{x} \in F^k$ and $\mathbf{x}$ is repeated r times to form the codeword $\mathbf{c}$. This code is easily proved to be a subspace of F^{rk} of dimension k. Hence C is a linear (rk,k)-code over F. If we form k codewords in this way by taking $\mathbf{x}$ in turn to be each of the k standard basis vectors for F^k, we will generate a basis for C. Thus the k×(rk) matrix is a generator matrix for C.

$$G=\begin{pmatrix} I_k & I_k & \dots & I_k \end{pmatrix}$$

The dual code $C^{\perp}$ is a linear code of dimension rk–dimC=(r–1)k. Thus a parity check matrix for C will be an (r–1)k×rk matrix over F. Perhaps you can work out in general what form it might take. You might start by checking that, for r=3 the matrix

$$H=\begin{pmatrix} I_k & -I_k & 0 \\ I_k & 0 & -I_k \end{pmatrix}$$

is a parity check matrix for the triple repetition (3k,k) code C. (Just check that H has rank 2k and $GH^T=0$.) For the (rk,k) repetition code over F, the Hamming distance (which is the minimum weight of non-zero codewords) is clearly equal to k.

1.1.2.1.4 Hadamard codes

These codes form a family of non-linear codes with very good error correcting properties. Each of these codes is constructed from an n×n Hadamard matrix H_n (which we define below). Let's say that $C(H_n)$ is the code associated with a Hadamard matrix H_n [11]. Then $C(H_n)$ has relatively few codewords, only 2n, but has large Hamming distance n/2. (Usually n is even, see below.) It was an Hadamard code constructed from a certain Hadamard matrix of order n=32 that was used to transmit black and white photographs from the 1972 Mariner space probe to Mars. This code $C(H_{32})$ has $64=2^6$ codewords and could therefore be used to encode all 6-bit strings; it is e-error correcting where e=[(16−1)/2]=7

An Hadamard matrix H_n of order n is an n×n matrix with entries in {1,−1} such that

$$H_n^T H_n = H_n H_n^T = nI_n.$$

This means that the dot product of any two distinct rows vectors (or column vectors) of H_n is equal to 0, that is, any two distinct row vectors (or column vectors) of H_n are orthogonal.

1.1.2.2 Convolutional Codes

Convolutional encoding is a multiplexing of two or more different convolutions of the same source data onto a channel. This is accomplished in a continuous manner with the use of shift registers and mod-2 adders. These mod-2 adders are XOR gates whose inputs are various combinations of the shift register states and whose outputs are multiplexed together to form the output stream. The input data stream enters the encoder at a rate of b/n bits per second, where b is the speed of the output stream and n is the number of mod-2 adders. This indicates a time redundancy since the input speed is less than that of the output. Some Convolutional

encoders divide the input data into two separate entities or more, allowing for non integral multiple time redundancy. In practice, encoders are commonly divided like this to match a fixed input rate with a fixed output rate, but they are somewhat more difficult to analyse. The generalization of a single input stream and an output stream running at an integral multiple of the input's rate is sufficient [13].

1.1.2.2.1 Coding and Decoding with Convolutional Code.

Convolutional codes are commonly specified by three parameters i.e. (n, k, m).

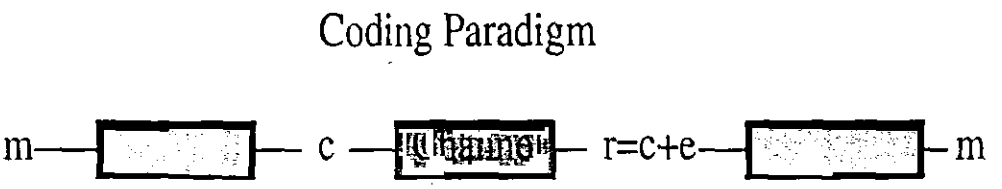


Figure 1.1 Coding paradigm

n = number of output bits

k = number of input bits

m = number of memory register

The quantity of k/n called the code rate is a measure of then efficiency of the code. Commonly k and n parameters range from 1 to 8, m from 2 to 10 and the code rate from 1/8 to 7/8 except for deep space applications, where code rate as low as 1/100 or even longer have been employed.

Often the manufactures of convolutional code chip specify the code by parameters (n, k, L) [14]. The quantity L is called the constraint length of the code and is defined by

Constraint Length L = k (m-1)

The constraint L represents the number of bits in the encoder memory that affect the generation of the n output bits. The constraint length L is also referred to by the

capital letter K, which can be confusing with the lower case k, which represent the number of input bits. In some places K is defined as equal to product of the k and m. Often in commercial specs, the codes are specified by (r, K), where r = the code rate k/n and K is constraint length. The constraint length K however is equal to L-1.

1.1.2.2.2 Encoder Structure

A convolutional code introduces redundant bits into the data stream through the use of linear shift registers as shown in Figure 2.2.

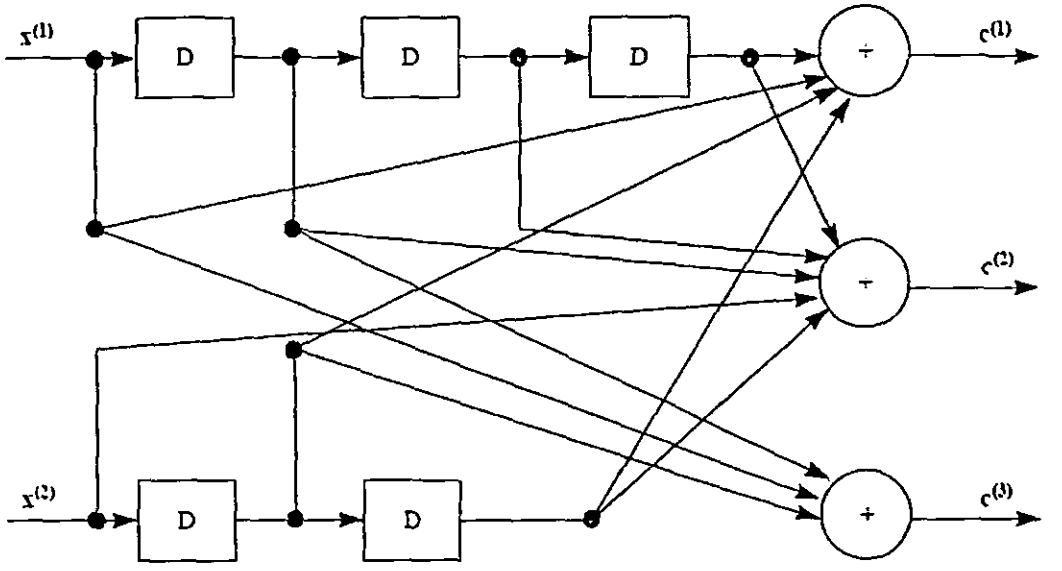


Figure 1.2 Example of convolutional encoder where $x_{(i)}$ is an input information bit stream and $c_{(i)}$ is an output encoded bit stream

The information bits are input into shift registers and the output encoded bits are obtained by modulo-2 addition of the input information bits and the contents of the shift registers [12]. The connections to the modulo-2 adders were developed heuristically with no algebraic or combinatorial foundation. The code rate r for a convolutional code is defined as

$$r = \frac{k}{n}$$

Where k is the number of parallel input information bits and n is the number of parallel output encoded bits at one time interval. The constraint length K for convolutional codes defined as

$$K = m + 1$$

Where m is the maximum number of stages (memory size) in any shift register. The shift registers store the state information of the convolutional encoder and the constraint length relates the number of bits upon which the output depends. For the convolutional encoder shown in Figure 2.2, the code rate r=2/3, the maximum memory size m=3, and the constraint length K=4.

A convolutional code can become very complicated with various code rates and constraint lengths. As a result, a simple convolutional code will be used to describe the code properties as shown in Figure 2.3.

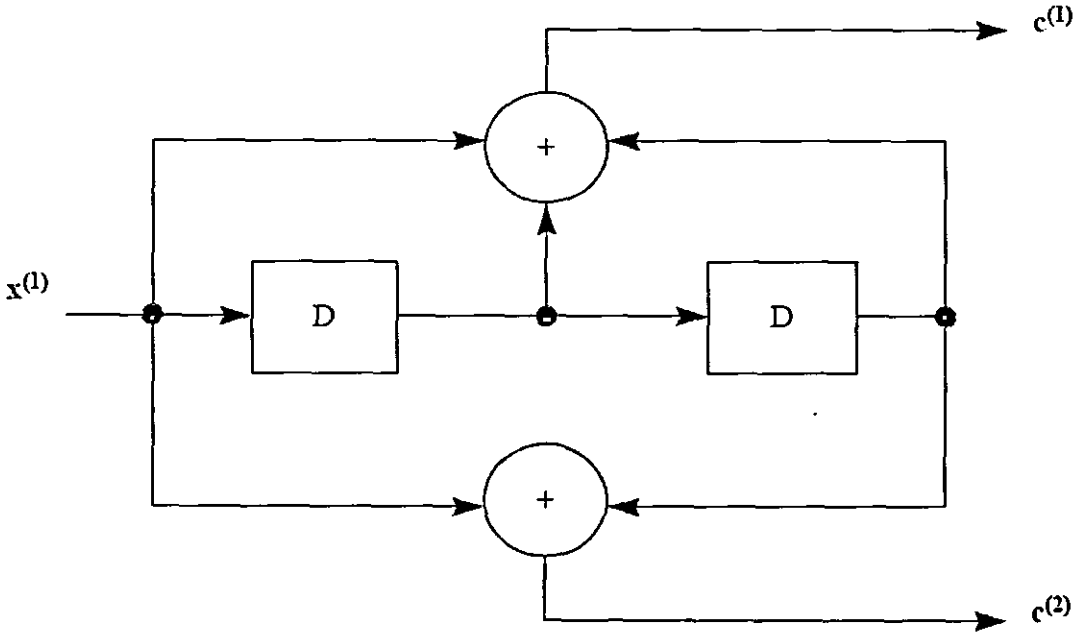


Figure 1.3 Convolutional encoder with k=1, n=2, r=1/2, m=2, and K=3.

1.1.2.2.3 Encoder Representations

The encoder can be represented in several different but equivalent ways i.e,

1. Generator Representation
2. Tree Diagram Representation
3. State Diagram Representation
4. Trellis Diagram Representation

1.1.2.2.4 Generator Representation

Generator representation shows the hardware connection of the shift register taps to the modulo-2 adders. A generator vector represents the position of the taps for an output. A "1" represents a connection and a "0" represents no connection. For example, the two generator vectors for the encoder in Figure 2.3 are $g_1 = [111]$ and $g_2 = [101]$ where the subscripts 1 and 2 denote the corresponding output terminals [15].

1.1.2.2.5 Tree Diagram Representation

The tree diagram representation shows all possible information and encoded sequences for the convolutional encoder. Figure 2.4 shows the tree diagram for the encoder in Figure 2.3 for four input bit intervals.

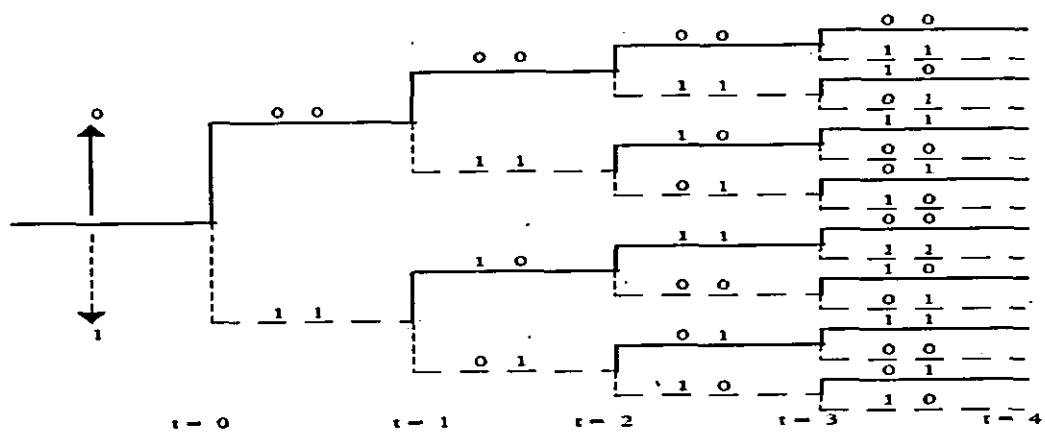


Figure 1.4 Tree diagram representation of the encoder in Figure 1.3 for four input bit intervals.

In the tree diagram, a solid line represents input information bit 0 and a dashed line represents input information bit 1. The corresponding output encoded bits are shown on the branches of the tree. An input information sequence defines a specific path through the tree diagram from left to right. For example, the input information sequence **x**= {1011} produces the output encoded sequence **c**= {11, 10, 00, 01}. Each input information bit corresponds to branching either upward (for input information bit 0) or downward (for input information bit 1) at a tree node.

1.1.2.2.6 State Diagram Representation

The state diagram shows the state information of a convolutional encoder [16]. The state information of a convolutional encoder is stored in the shift registers. Figure 2.5 shows the state diagram of the encoder in Figure 2.3.

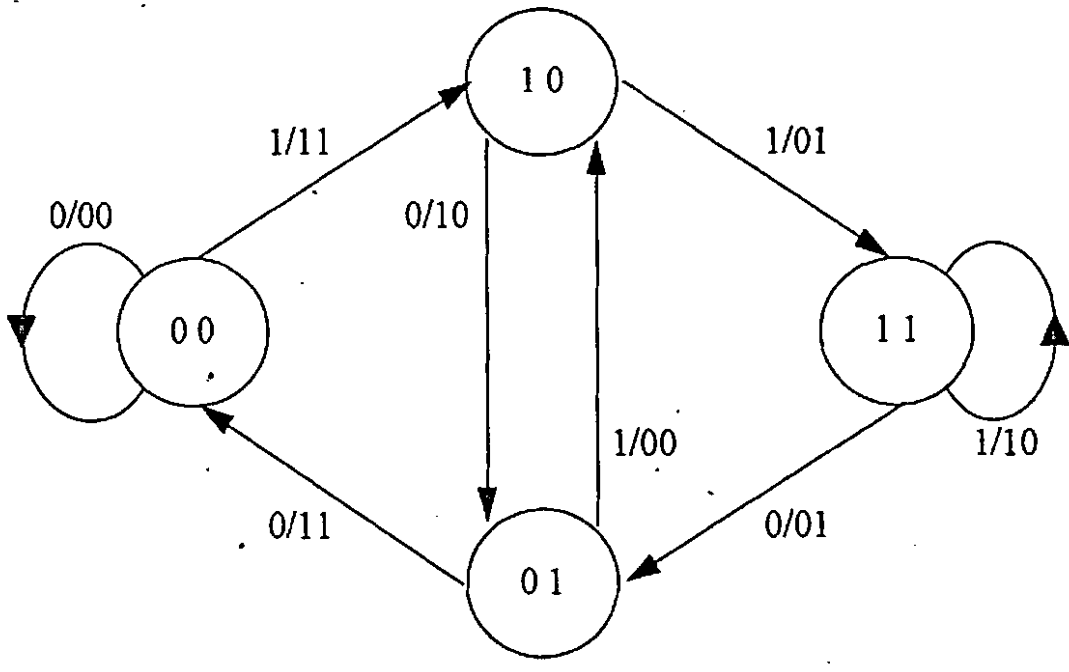


Figure 1.5 State diagram representation of the encoder in Figure 1.3

In the state diagram, the state information of the encoder is shown in the circles. Each new input information bit causes a transition from one state to another. The path information between the states, denoted as **x/c**, represents input information

bit x and output encoded bits c. It is customary to begin convolutional encoding from the all zero state. For example, the input information sequence x={1011} (begin from the all zero state) leads to the state transition sequence s={10, 01, 10, 11} and produces the output encoded sequence c={11, 10, 00, 01}. Figure 2.6 shows the path taken through the state diagram for the given example.

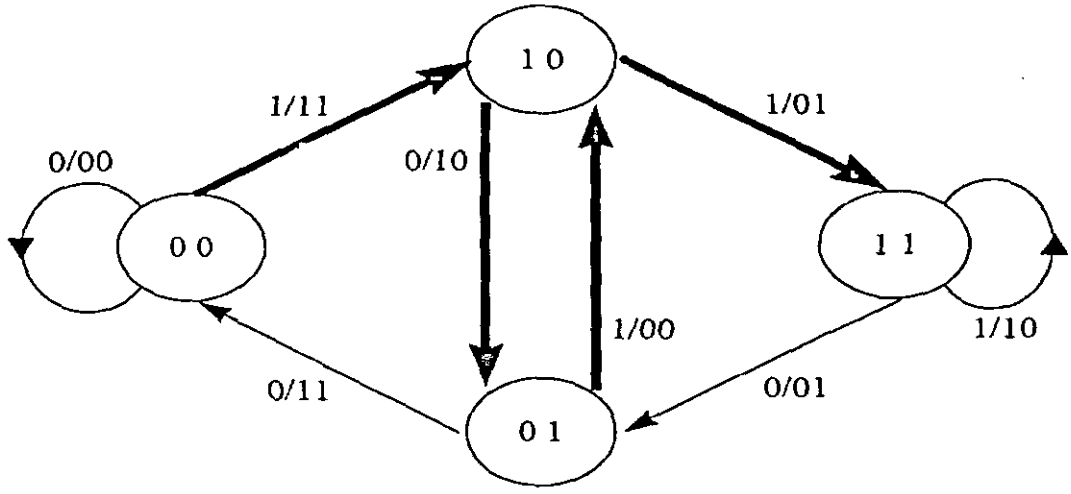


Figure 1.6 The state transitions (path) for input information sequence {1011}

1.1.2.2.7 Trellis Diagram Representation

The trellis diagram is basically a redrawing of the state diagram. It shows all possible state transitions at each time step [10]. Frequently, a legend accompanies the trellis diagram to show the state transitions and the corresponding input and output bit mappings (x/c). This compact representation is very helpful for decoding convolutional codes as discussed later. Figure 2.7 shows the trellis diagram for the encoder in Figure 2.3.

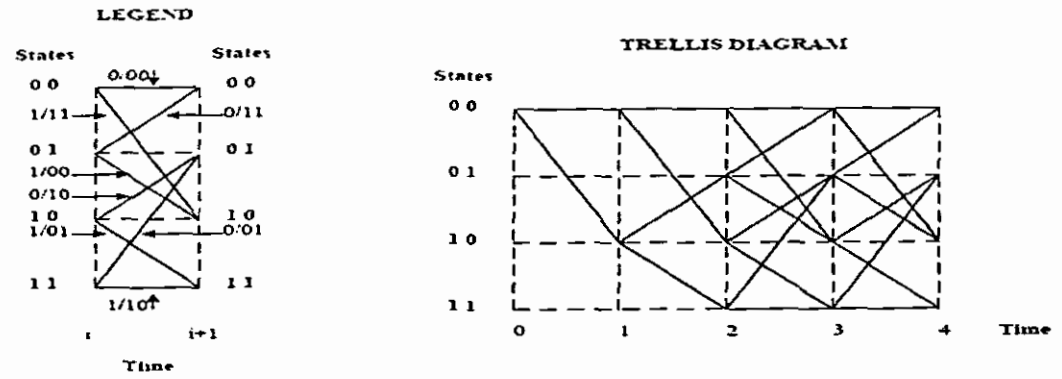


Figure 1.7 Trellis diagram representation of the encoder in Figure 1.3 for four input bit intervals

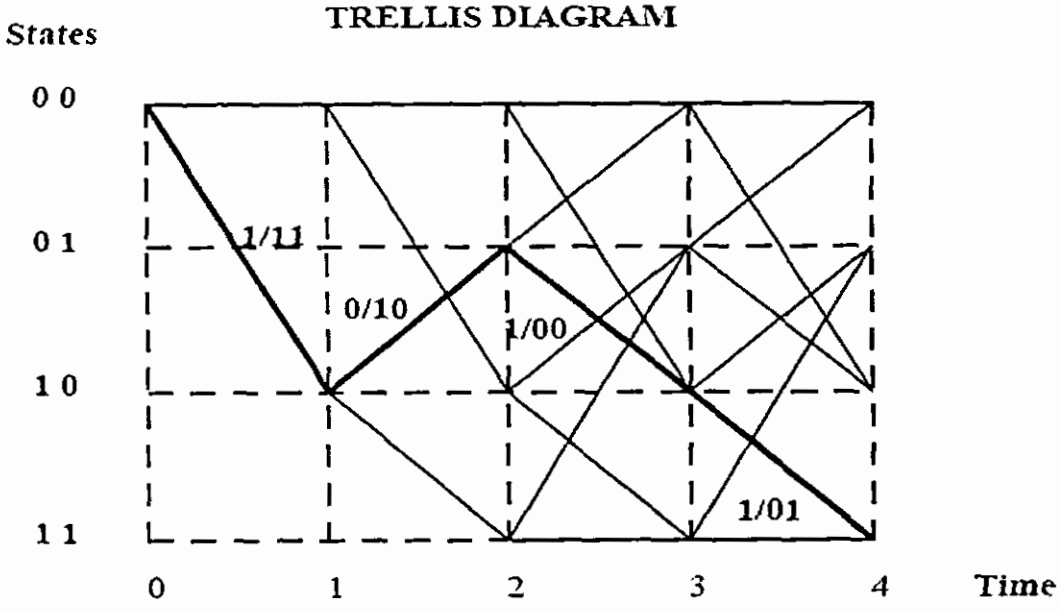


Figure 1.8 Trellis path for the state transitions in Figure 1.6

1.1.2.2.8 Catastrophic Convolutional Code

Catastrophic convolutional code causes a large number of bit errors when only a small number of channel bit errors is received. This type of code needs to be avoided and can be identified by the state diagram. A state diagram having a loop in which a nonzero information sequence corresponds to an all-zero output sequence identifies a catastrophic convolutional code. Figure 2.9 shows two examples of such code [11].

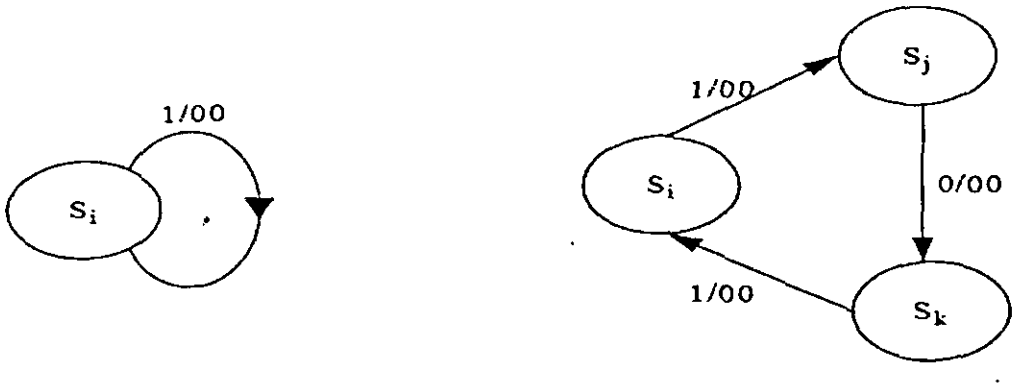


Figure 1.9 Examples of catastrophic convolutional code.

1.1.2.2.9 Hard-Decision and Soft-Decision Decoding

Hard-decision and soft-decision decoding refer to the type of quantization used on the received bits. Hard-decision decoding uses 1-bit quantization on the received channel values. Soft-decision decoding uses multi-bit quantization on the received channel values. For the ideal soft-decision decoding (infinite-bit quantization), the received channel values are directly used in the channel decoder. Figure 2.10 shows hard- and soft-decision decoding.

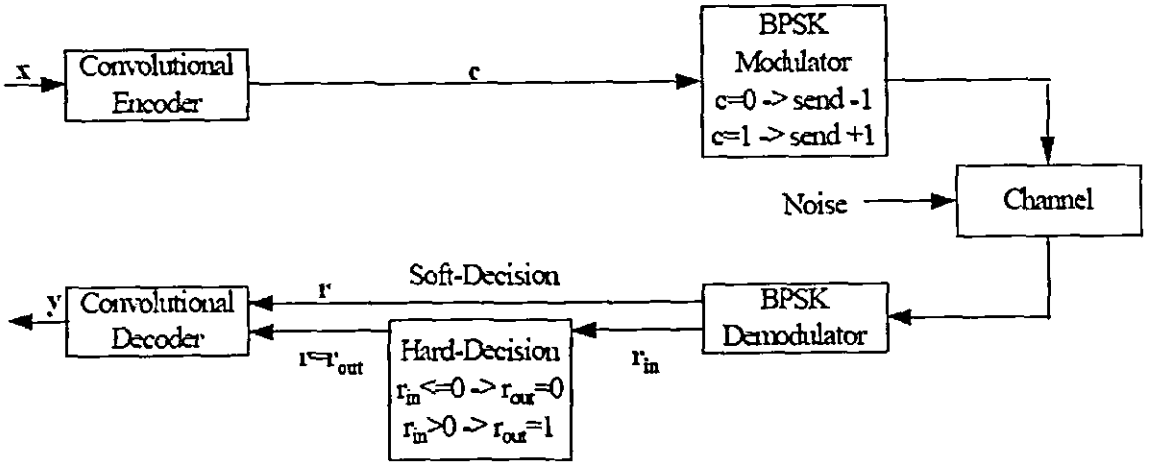


Figure 1.10 Hard- and Soft-decision decoding

1.1.2.2.10 Performance Analysis of Convolutional Code

The performance of convolutional codes can be quantified through analytical means or by computer simulation. The analytical approach is based on the transfer function of the convolutional code which is obtained from the state diagram. The process of obtaining the transfer function and other related performance measures are described below [13].

1.1.2.2.10.1 Transfer Function of Convolutional Code

The analysis of convolutional codes is generally difficult to perform because traditional algebraic and combinatorial techniques cannot be applied. These heuristically constructed codes can be analyzed through their transfer functions. By utilizing the state diagram, the transfer function can be obtained. With the transfer function, code properties such as distance properties and the error rate performance can be easily calculated. To obtain the transfer function, the following rules are applied:

1. Break the all-zero (initial) state of the state diagram into a start state and an end state. This will be called the modified state diagram.
2. For every branch of the modified state diagram, assign the symbol D with its exponent equal to the Hamming weight of the output bits.
3. For every branch of the modified state diagram, assign the symbol J.
4. Assign the symbol N to the branch of the modified state diagram, if the branch transition is caused by an input bit 1

For the state diagram in Figure 1.5, the modified state diagram is shown in Figure 1.11

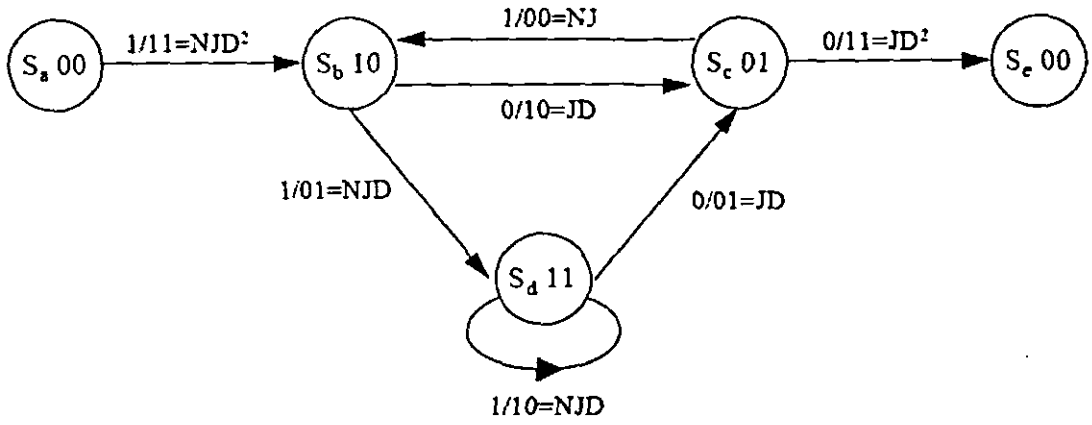


Figure 1.11 The modified state diagram of Figure 1.5 where S_a is the start state and S_e is the end state.

Nodal equations are obtained for all the states except for the start state in Figure 2.12. These results are

$$S_b = NJD^2 S_a + NJS_c$$
$$S_c = JDS_b + JDS_d$$
$$S_d = NJDS_b + NJDS_d$$
$$S_e = JD^2 S_c$$

The transfer function is defined to be

$$T(D,N,J) = \frac{S_{end}(D,N,J)}{S_{start}(D,N,J)}$$

and for Figure 2.11,

$$T(D,N,J) = \frac{S_e}{S_a}$$

1.1.2.2.10.2 Decoding Depth

The decoding depth is a window in time that makes a decision on the bits at the beginning of the window and accepts bits at the end of the window for metric computations. This scheme gives up the optimum ML decoding at the expense of using less memory and smaller decoding delay. It has been experimentally found that

if the decoding depth is 5 times greater than the constraint length K then the error introduced by the decoding depth is negligible [13].

1.1.2.2.10.3 Degree of Quantization

For soft-decision Viterbi decoding, the degree of the quantization on the received signal can affect the decoder performance. The performance of the Viterbi decoder improves with higher bit quantization. It has been found that an eight-level quantizer degrades the performance only slightly with respect to the infinite bit quantized case [13].

1.1.2.2.11 Euclidean Distance

The use of convolutional encoders in combination with their decoder counterparts suggests something beyond Hamming distance as a measure of the competence of a code set (of polynomials.) The argument thus far has assumed a channel of perfect flat response. No channels of this virtue are naturally occurring, and convolutional codes are not applied to channels that are approximated by such clean response anyway. The channel is certainly considered before the selection of a convolutional code. The decision is primarily made on the basis of Euclidean distance between resulting waveforms at the receiver. The effects of the channel are applied to the calculation of the expected results, and the code is classified as having an aptitude for that particular channel. The minimum Euclidean distance between any two waveforms at the receiver originating from two distinct information messages in the encoder becomes the measure of success. The Euclidean distance is the energy of the difference signal.

1.1.2.2.12 Manchester Encoding

Manchester encoding was implemented in the final stage of the encoder. This coding scheme ensures that a signal level change will occur for every clock cycle. The bandwidth is doubled by this technique, but the advantage is the resulting spectral shift of the output away from DC The DC spectral density drops to zero, and the

output spectrum is shifted to the clock frequency. The necessity for this shift stems from the transmitter response characteristic. The transmitter does not respond to frequencies below about 100 Hz, so it is advantageous to center the spectral density over the maximum response frequency of 1 kHz [1].

In some way, the Manchester encoder is a conceptual extension of the convolutional encoder.It performs a time redundancy and transformation on the input data. The receiver has more error control since the expected input must show a change at least once per clock cycle. The Manchester coding could be embedded in the convolutional encoder by doubling the number of interleaves, copying the last half of the convolutions and inverting them.

CHAPTER 2
GENETIC ALGORITHM

2. Genetic Algorithms

Genetic Algorithms are a family of computational models inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome like data structure and apply recombination operators to these structures so as to preserve critical information. Genetic algorithms are often viewed as function optimizers, although the range of problems to which genetic algorithms have been applied is quite broad [4].

An implementation of a genetic algorithm begins with a population of typically random chromosomes. One then evaluates these structures and allocates reproductive opportunities in such a way that those chromosomes which represent a better solution to the target problem are given more chances to "reproduce" than those chromosomes which are poorer solutions. The "goodness" of a solution is typically defined with respect to the current population.

In a broader usage of the term, a genetic algorithm is any population based model that uses selection and recombination operators to generate new sample points in a search space.

Genetic programming is a branch of genetic algorithms. The main difference between genetic programming and genetic algorithms is the representation of the solution. Genetic programming creates computer programs in the lisp or scheme computer languages as the solution. Genetic algorithms create a string of numbers that represent the solution. Genetic programming uses four steps to solve problems [3]:

1) Generate an initial population of random compositions of the functions and terminals of the problem (computer programs).

2) Execute each program in the population and assign it a fitness value according to how well it solves the problem.

3) Create a new population of computer programs.

i) Copy the best existing programs

would simply be the amount of time that the clock is wrong. Unfortunately, few problems have such an easy fitness function; most cases require a slight modification of the problem in order to find the fitness.

2.1.1 Gun Firing Program

A more complicated example consists of training a genetic program to fire a gun to hit a moving target [3]. The fitness function is the distance that the bullet is off from the target. The program has to learn to take into account a number of variables, such as wind velocity, type of gun used, distance to the target, height of the target, velocity and acceleration of the target. This problem represents the type of problem for which genetic programs are best. It is a simple fitness function with a large number of variables.

2.1.2 Water Sprinkler System

Consider a program to control the flow of water through a system of water sprinklers. The fitness function is the correct amount of water evenly distributed over the surface. Unfortunately, there is no one variable encompassing this measurement. Thus, the problem must be modified to find a numerical fitness. One possible solution is placing water-collecting measuring devices at certain intervals on the surface. The fitness could then be the standard deviation in water level from all the measuring devices [3]. Another possible fitness measure could be the difference between the lowest measured water level and the ideal amount of water; however, this number would not account in any way the water marks at other measuring devices, which may not be at

the ideal mark.

2.1.3 Maze Solving Program

If one were to create a program to find the solution to a maze, first, the program would have to be trained with several known mazes. The ideal solution from the start

is the total sum of all the members of the population, then the probability that the solution Si will be copied to the next generation is [Koza 1992]:

$$\frac{f(s_i(t))}{\sum_{j=1}^{M} f(s_j(t))}$$

Another method for selecting the solution to be copied is tournament selection. Typically the genetic program chooses two solutions random. The solution with the higher fitness will win. This method simulates biological mating patterns in which, two members of the same sex compete to mate with a third one of a different sex. Finally, the third method is done by rank. In rank selection, selection is based on the rank, (not the numerical value) of the fitness values of the solutions of the population

.

The creation of the offsprings from the crossover operation is accomplish by deleting the crossover fragment of the first parent and then inserting the crossover fragment of the second parent. The second offspring is produced in a symmetric manner. For example consider the two S-expressions in Figure 2, written in a modified scheme programming language and represented in a tree.

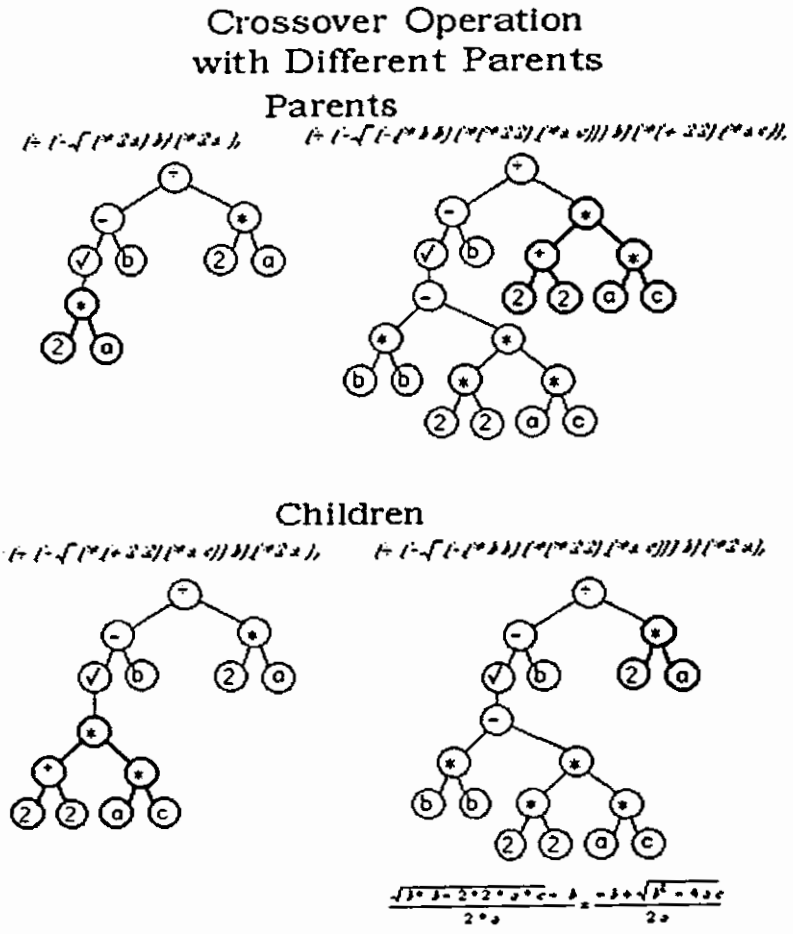


Figure 2.2 Crossover Operation with different Parents

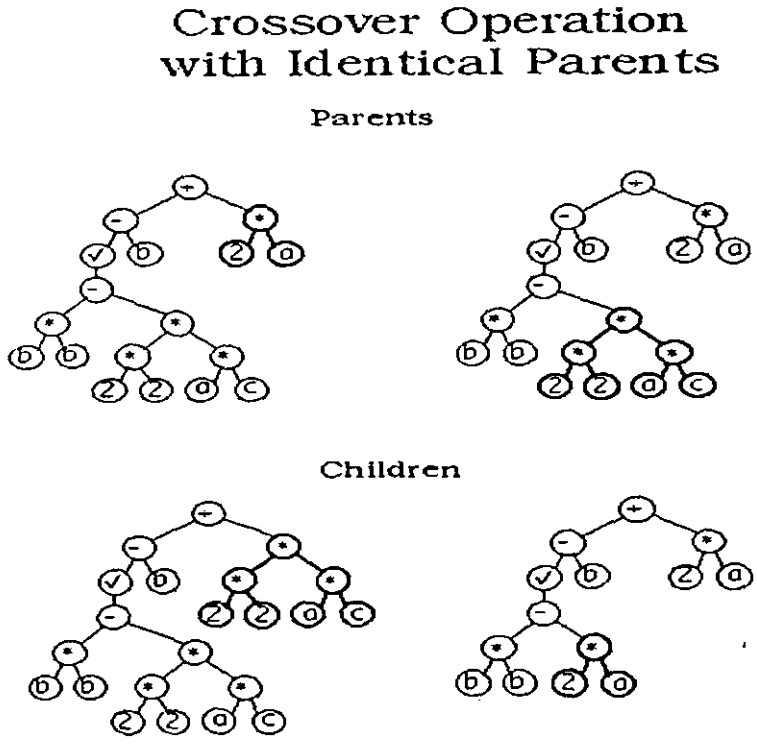


Figure 2.3 Crossover Operation with Identical parents

An important improvement that a genetic programming displays over genetic algorithms is its ability to create two new solutions from the same solution. In the Figure 2.3 the same parent is used twice to create two new children.

2.4 Mutation

Mutation is another important feature of genetic programming. Two types of mutations are possible [3]. In the first kind a function can only replace a function or a terminal can only replace a terminal. In the second kind an entire subtree can replace another subtree. Figure 2.4 explains the concept of mutation:

Mutation

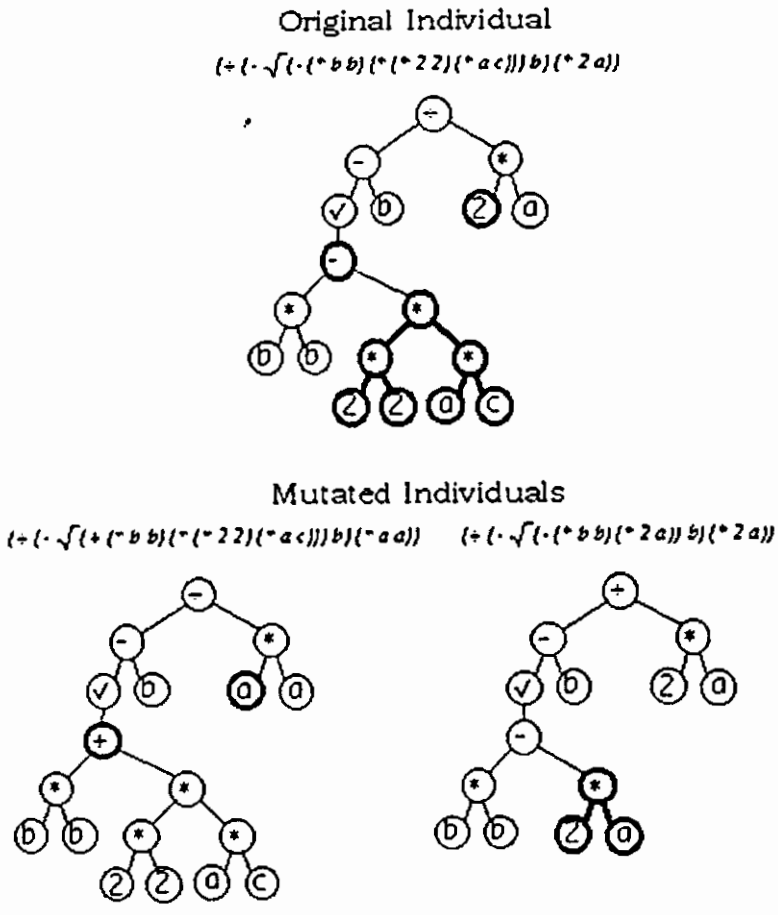


Figure 2.4 Mutation

CHAPTER 3

ARTIFICIAL NEURAL NETWORK

3. Artificial Neural Networks

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. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons[7]. This is true of ANNs as well.

3.1 Usage of Neural networks

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 expert can then be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include:

1. **Adaptive learning:** An ability to learn how to do tasks based on the data given for training or initial experience [6].

2. **Self-Organisation:** An ANN can create its own organisation or representation of the information it receives during learning time [6].

3. **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 [6].

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

3.3 A Simple Neuron

An artificial neuron is a device with many inputs and one output. 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.

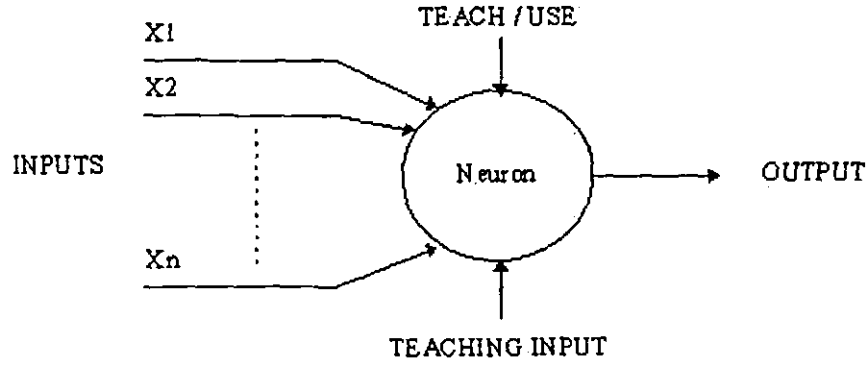


Fig 3.1 A simple neuron

3.4 Firing rules

The firing rule is an important concept in neural networks and accounts for their high flexibility. 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. A simple firing rule can be implemented by using Hamming distance technique. The rule goes as follows:

The difference between the two truth tables is called the *generalization of the neuron.* Therefore the firing rule gives the neuron a sense of similarity and enables it to respond 'sensibly' to patterns not seen during training [5].

3.5 Architecture of neural networks

There are different types of neural network architecture.

3.5.1 Feed-forward networks

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

3.5.2 Feedback networks

Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found [5]. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.

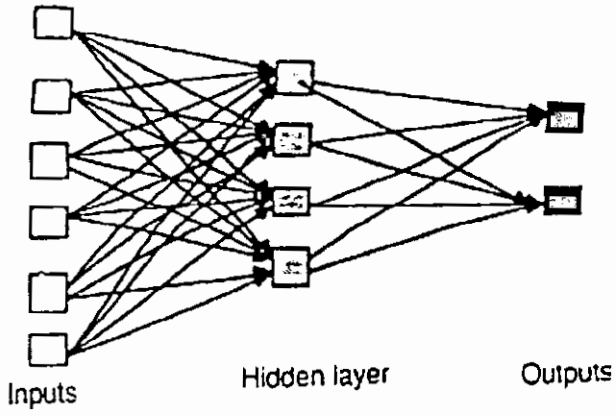


Figure 3.2 An example of a simple feedforward network

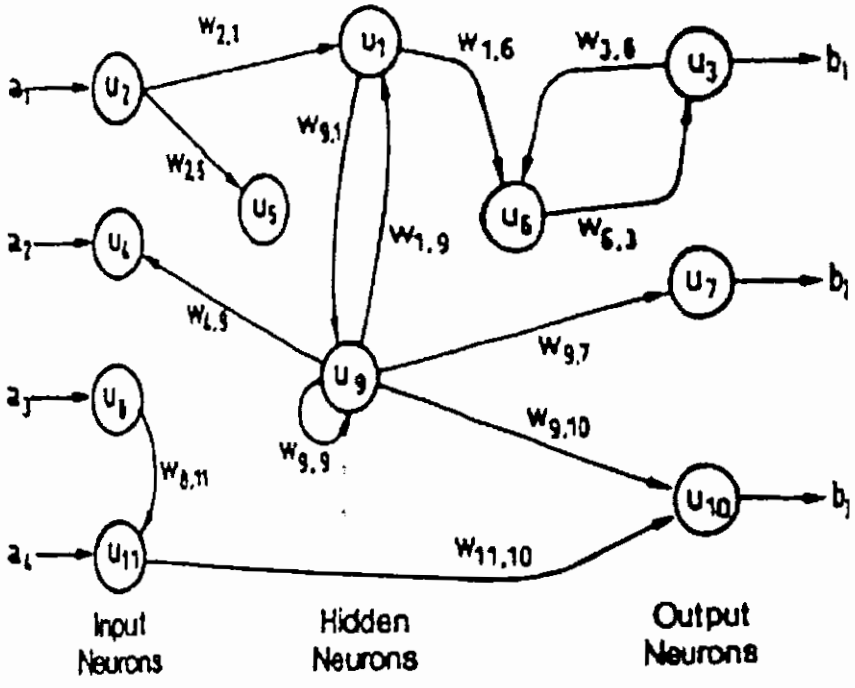


Figure 3.3 An example of a complicated network

3.6 Network layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "**input**" units is connected to a layer of "**hidden**" units, which is connected to a layer of "**output**" units.

1. The activity of the input units represents the raw information that is fed into the network.

2. The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.

3. The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

3.7 Perceptrons

The most influential work on neural nets in the 60's went under the heading of 'perceptrons' a term coined by Frank Rosenblatt [6]. The perceptron turns out to be an MCP model (neuron with weighted inputs) with some additional, fixed, pre--processing. Units labelled A1, A2, Aj , Ap are called association units and their task is to extract specific, localised featured from the input images. Perceptrons mimic the

basic idea behind the mammalian visual system. They were mainly used in pattern recognition even though their capabilities extended a lot more.

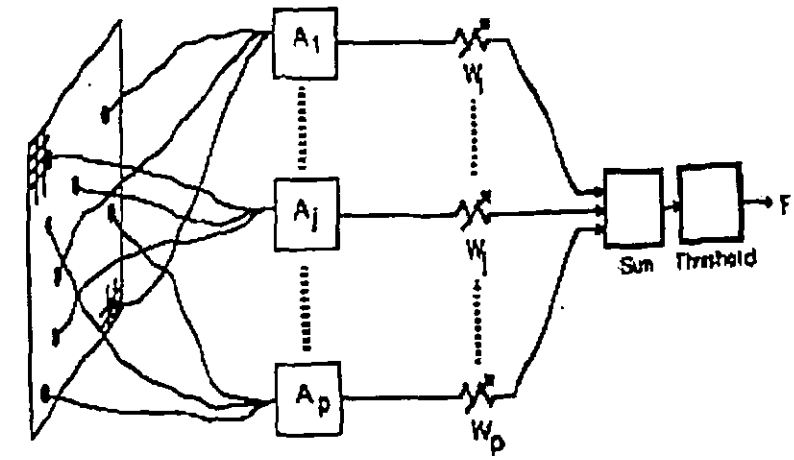


Figure 3.4 Perceptrons

3.8 The Learning Process

The memorization of patterns and the subsequent response of the network can be categorized into two general paradigms [5]:

1. **associative mapping** in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms:

2. **auto-association:** an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern competition, ie to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns.

3. **hetero-association:** is related to two recall mechanisms:

4. **Nearest-neighbor** recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented, and

5. **Interpolative** recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping is classification, ie when there is a fixed set of categories into which the input patterns are to be classified.

6. **Regularity detection** in which units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.

Every neural network possesses knowledge which is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights.

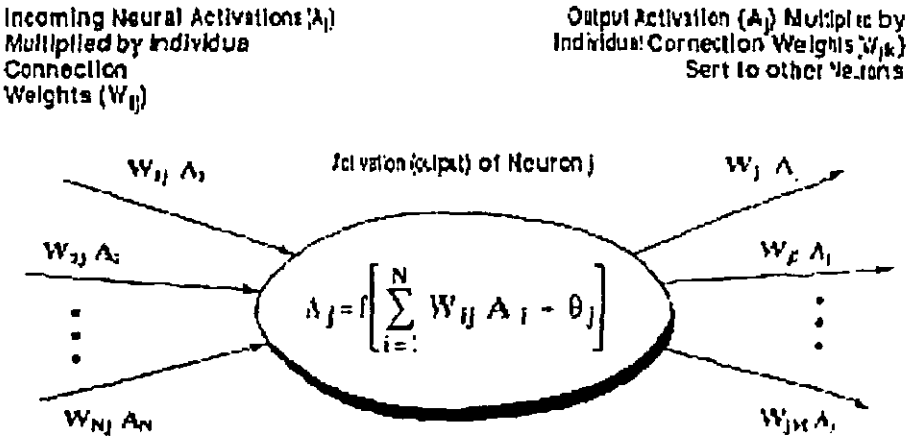


Figure 3.5 network learning is performed

Information is stored in the weight matrix W of a neural network. Learning is the determination of the weights. Following the way learning is performed, we can distinguish two major categories of neural networks:

1. **Fixed networks** in which the weights cannot be changed, ie dW/dt=0. In such networks, the weights are fixed a priori according to the problem to solve.

2. **Adaptive networks** which are able to change their weights, ie dW/dt not= 0.

All learning methods used for adaptive neural networks can be classified into two major categories:

1. **Supervised learning** which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning.
   An important issue concerning supervised learning is the problem of error convergence, ie the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms, is the least mean square (LMS) convergence.

2. **Unsupervised learning** uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian leaning and competitive learning. Ano2.2 From Human Neurons to Artificial Neuron Esther aspect of learning concerns the distinction or not of a separate phase, during which the network is trained, and a subsequent operation phase. We say that a neural network learns off-line if the learning phase and the operation phase are distinct. A neural network learns on-line if it learns and operates at

the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed on-line.

3.9 Transfer Function

The behavior of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories [6]:

1. linear (or ramp)
2. threshold
3. sigmoid

For **linear units**, the output activity is proportional to the total weighted output.

For **threshold units**, the output are set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

For **sigmoid units**, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurones than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another (see figure 4.1), and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

We can teach a three-layer network to perform a particular task by using the following procedure:

1. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.

2. We determine how closely the actual output of the network matches the desired output.

3. We change the weight of each connection so that the network produces a better approximation of the desired output.

CHAPTER 4
IMPLEMENTATION

4. Implementation

Our project Channel Encoding of wireless Medium based on convolutional Codes using ANN (Artificial Neural Network) and GA (Genetic Algorithm) is simulated using two tools the interfacing is all done by using Visual C++ and also the Genetic Algorithm and Artificial Neural Network are also implemented using VC++ and the in the second phase where Results are simulated we used MATLAB tool.

Now we will define in this chapter the pseducode of methods, Algorithms and processes used for implementing the project.

4.1 Decoding Convolutional Codes using Genetic Algorithm

The criteria of our algorithm for using Genetic Algorithm step by step are given as.

1. We will assume the case that before the first values enters into the encoder all the registers are with the value '0',e.g., we are at state 00 for k=3 encoder
2. From code rate(k/n) provided we determined the no of module 2-adders, this will determine the no of output values for one value entered into the encoder.
3. Pad the k-n zeros to the output sequence obtained at ith time so that length(ui)=k say it is InRegVal.
4. Provide this InRegVal to Genetic algorithm system which determines the optimum register contents at the encoder for which the output was generated say it OutRegVal.
5. Take first (k/2)+1 for even register encoder and (k+1)/2 for an odd register encoder. So take these first bits of OutRegVal keeping in mind the above conditions.
6. Compare Values obtained from step 5 with the states shown in state diagram, If there exist in state diagram's transition states from previous step find

6. Compare Values obtained from step 5 with the states shown in state diagram, If there exist in state diagram's transition states from previous step find whether there would be '1' or '0'.If there is no match it means there have occurred an error.

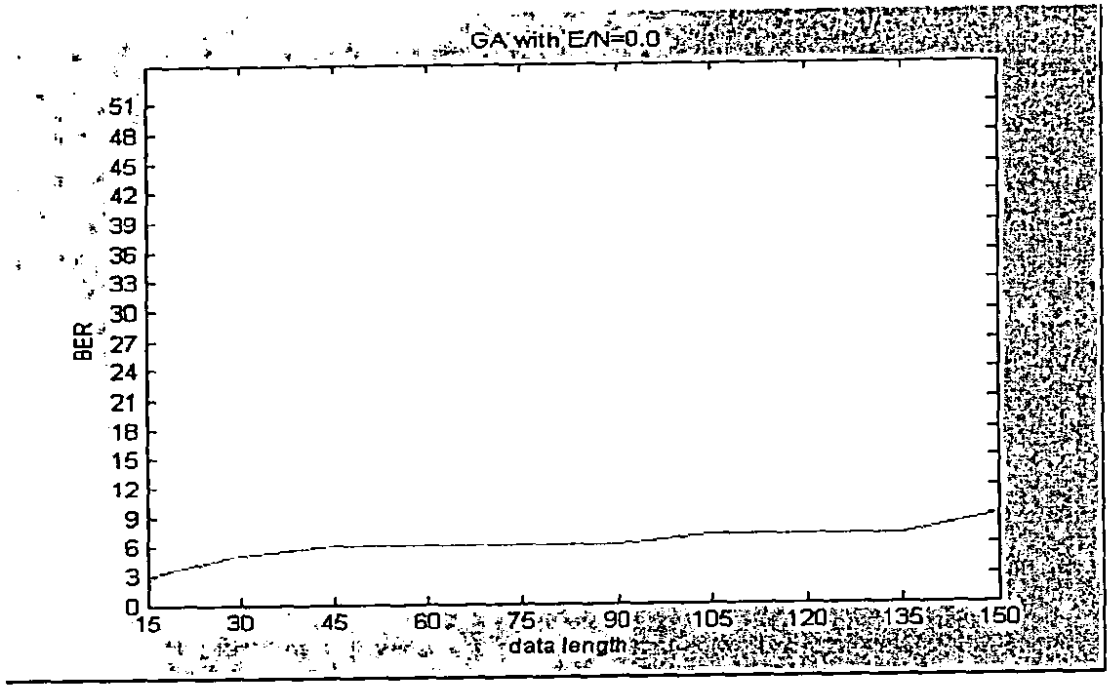
Parameters used for ANN

1. Feed Forward with back Propagation Artificial Neural network is used.
2. ANN has three layers first layer is input layer, second is Hidden layer and third one is Output layer. Input and hidden layers have (n+k) *2 neurons while output layer have n neurons.
3. This Network is Adaptive Network.
4. Supervised Learning is used based on Lookup table of the Encoder.
5. Delta rule is used for the adjustment of the weights of the output layer while back propagation is used for the hidden layer.

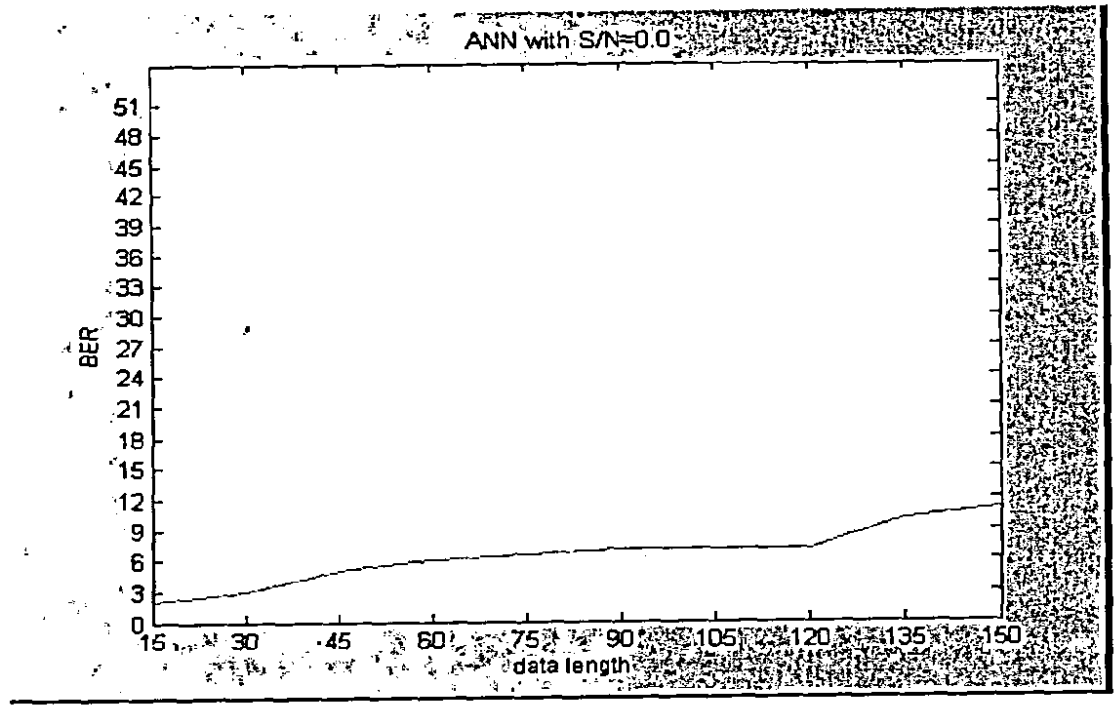
4.3 Error Correction

When we have an error we randomly take bit '0' or '1' but when we decode the next sequence of data and get the decoded bit, firstly we see the value at second position which we got from the 5th point and also see that from the previous state this state comes or not if not we move back and change the previous bit so the error is corrected.

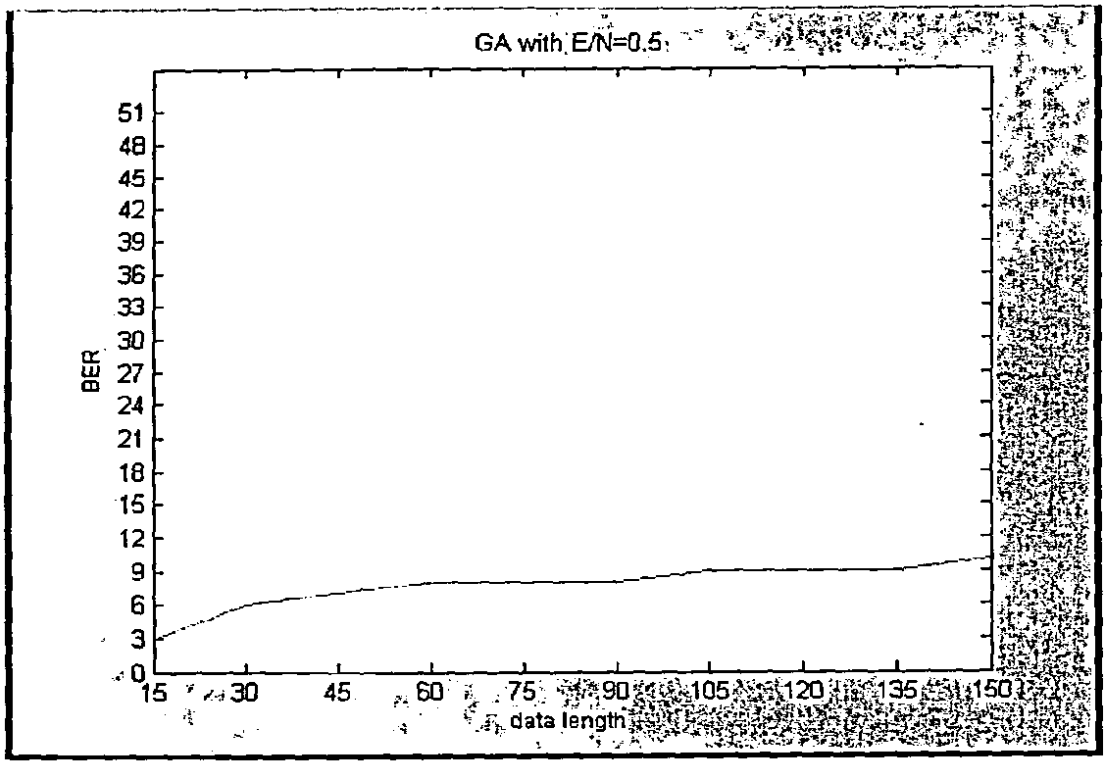
4.4 Results



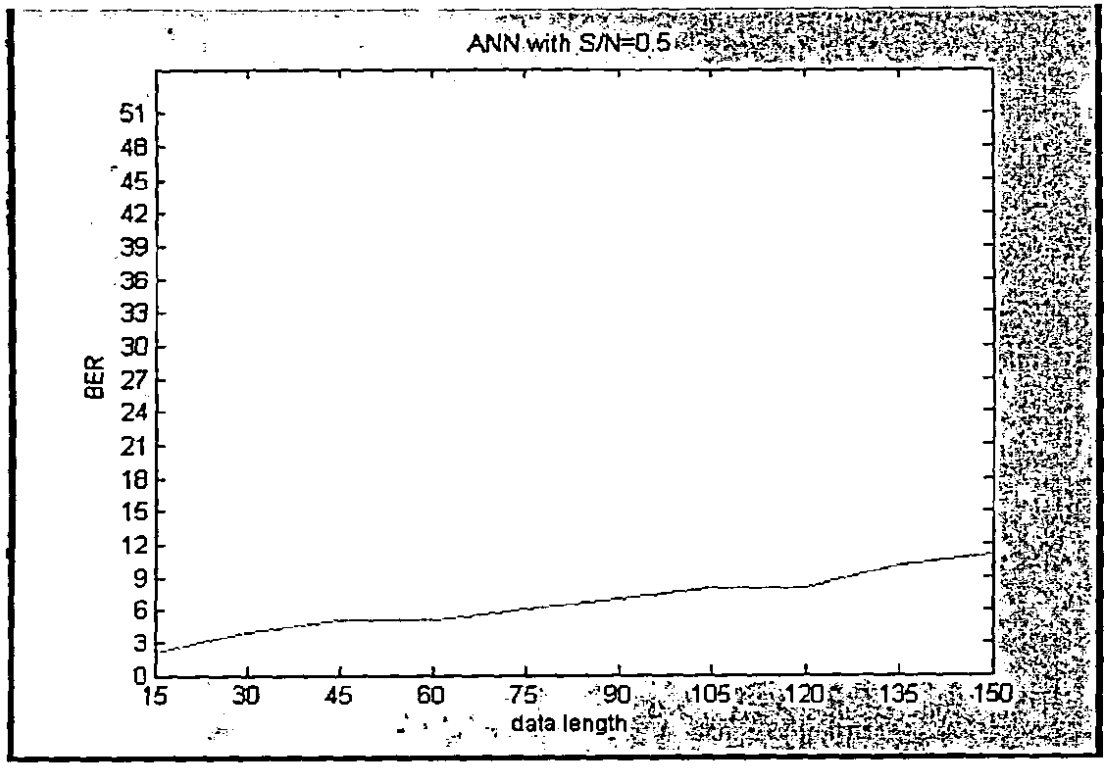
Graph 4.1 GA with E/N=0.0



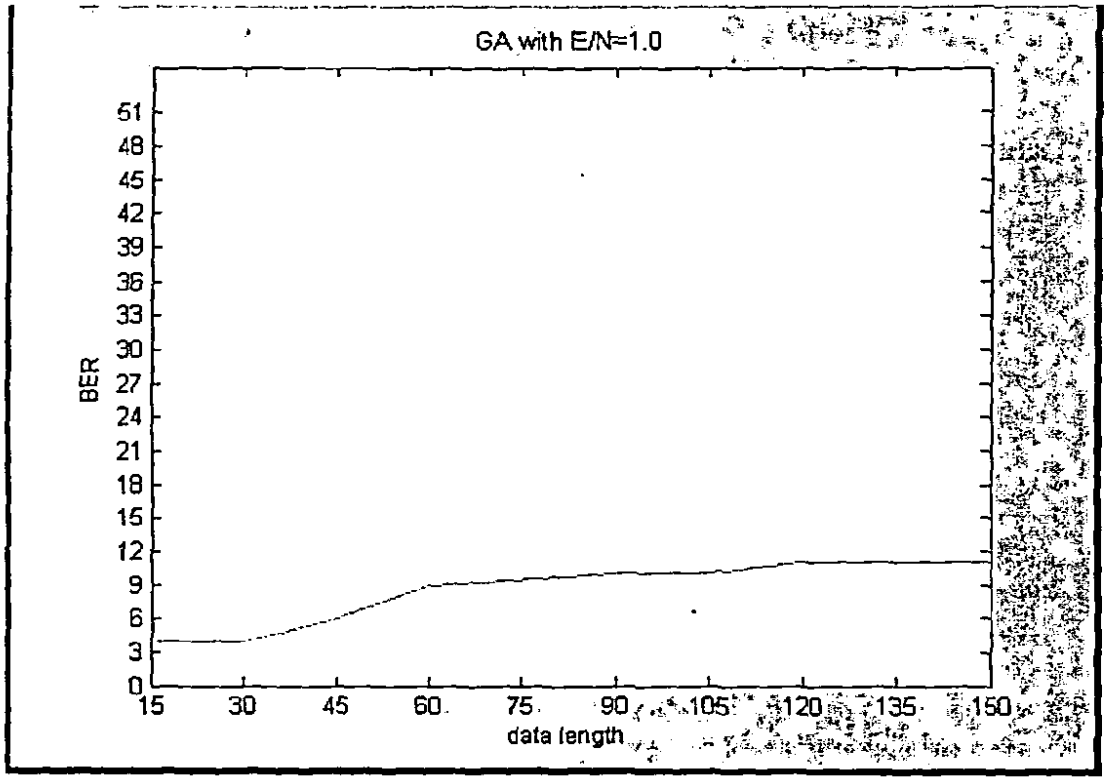
Graph 4.2 ANN with E/N=0.0



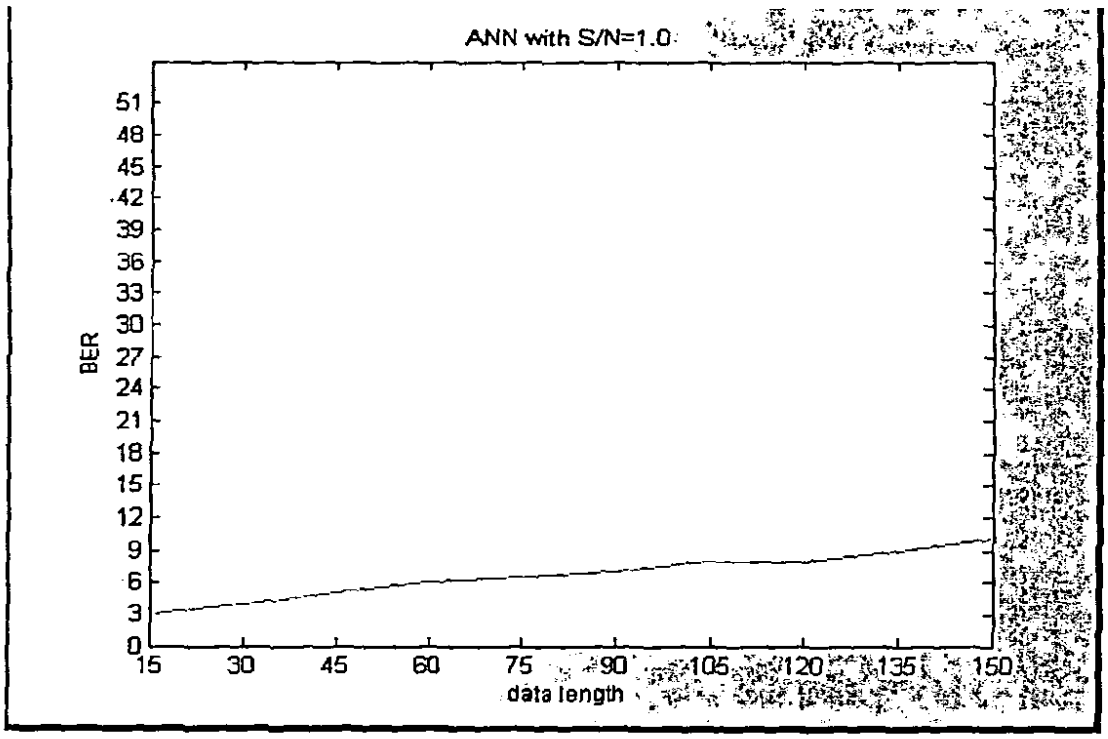
Graph 4.3 GA with E/N=0.5



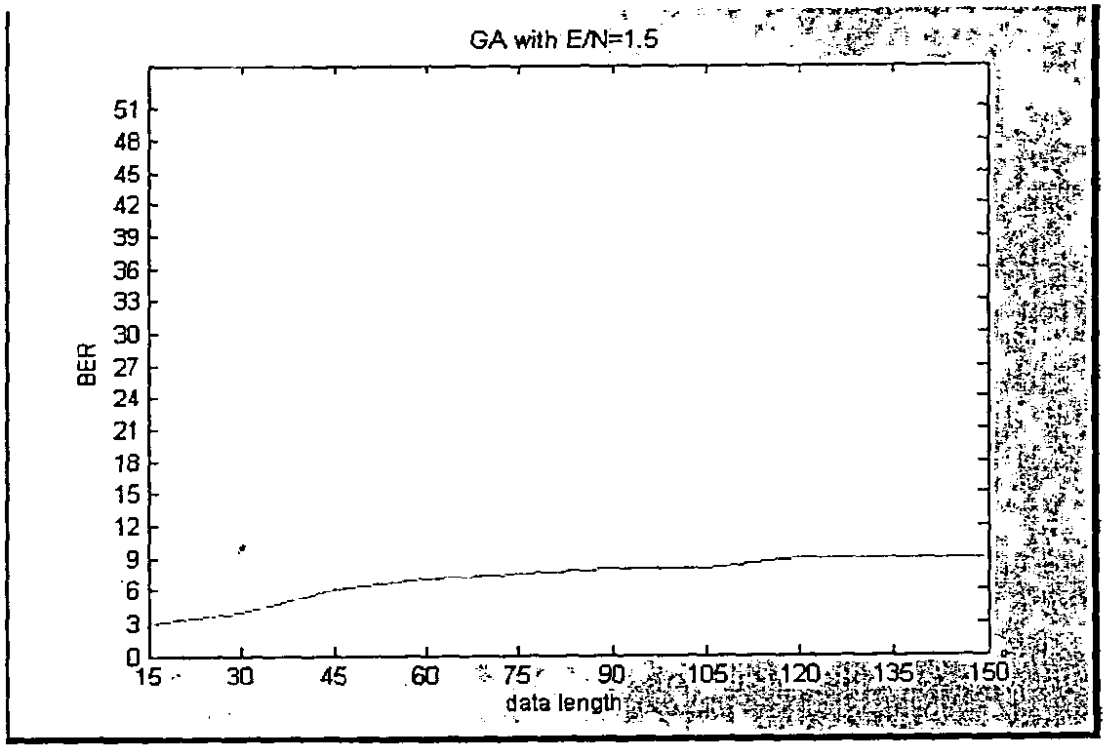
Graph 4.4 ANN with E/N=0.5



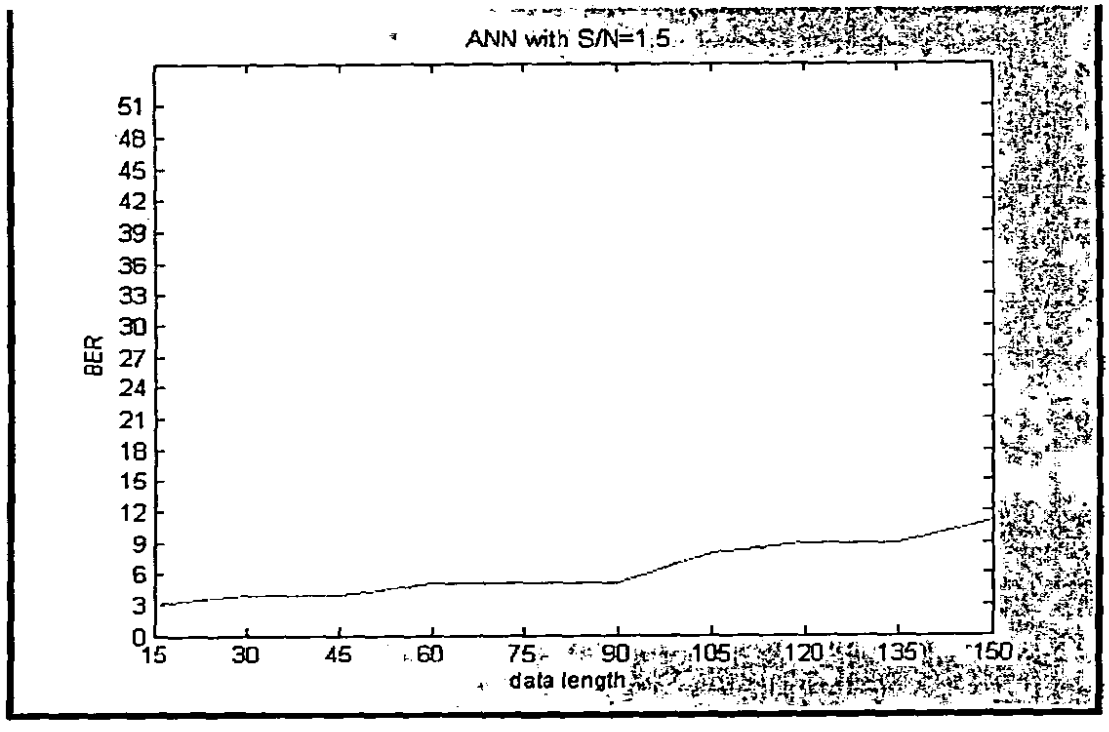
Graph 4.5 GA with E/N=1.0



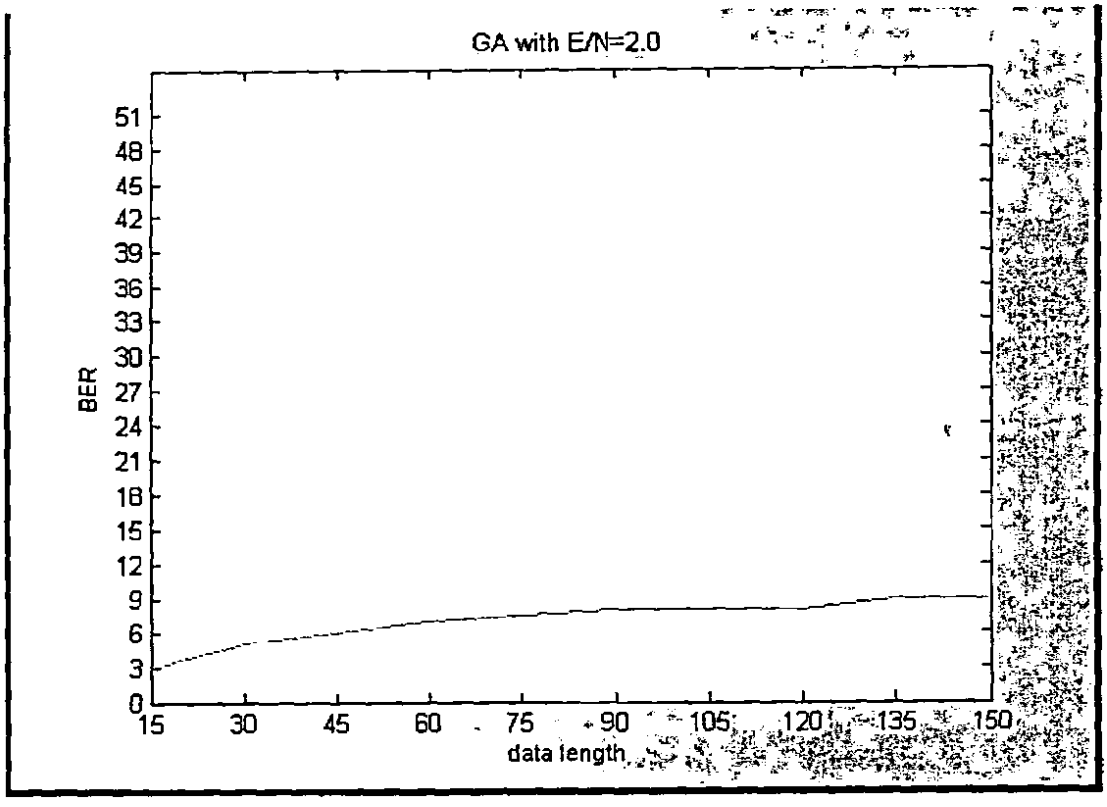
Graph 4.6 ANN with E/N=1.0



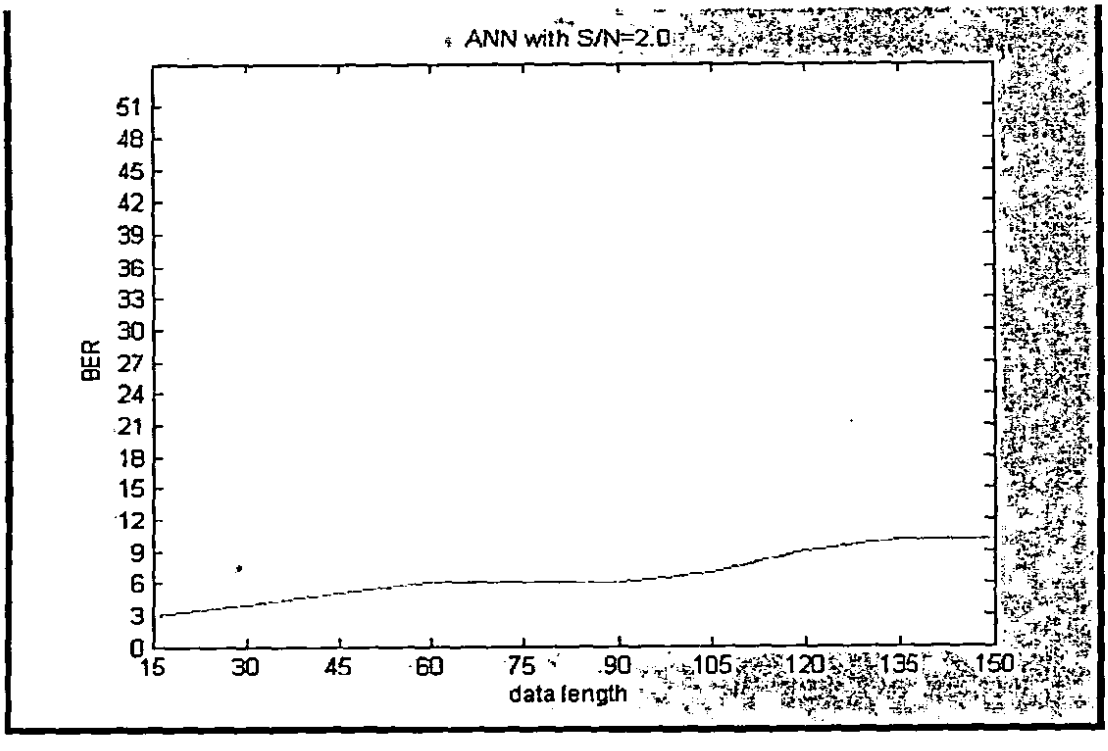
Graph 4.7 GA with E/N=1.5



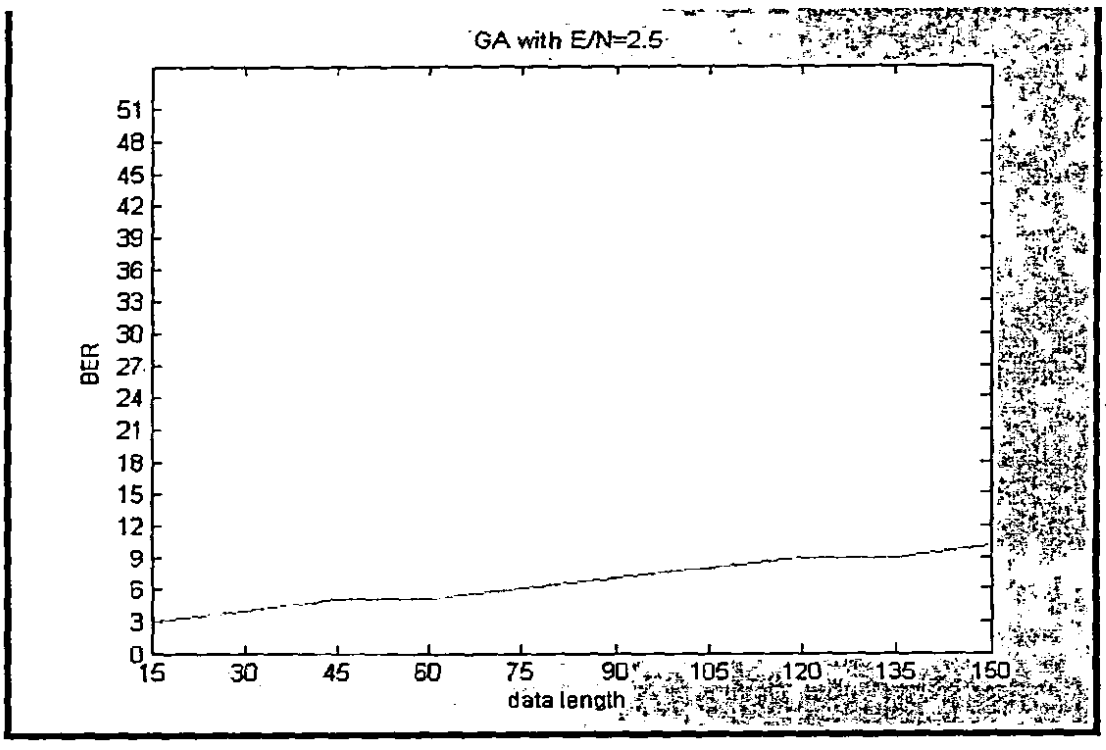
Graph 4.8 ANN with E/N=1.5



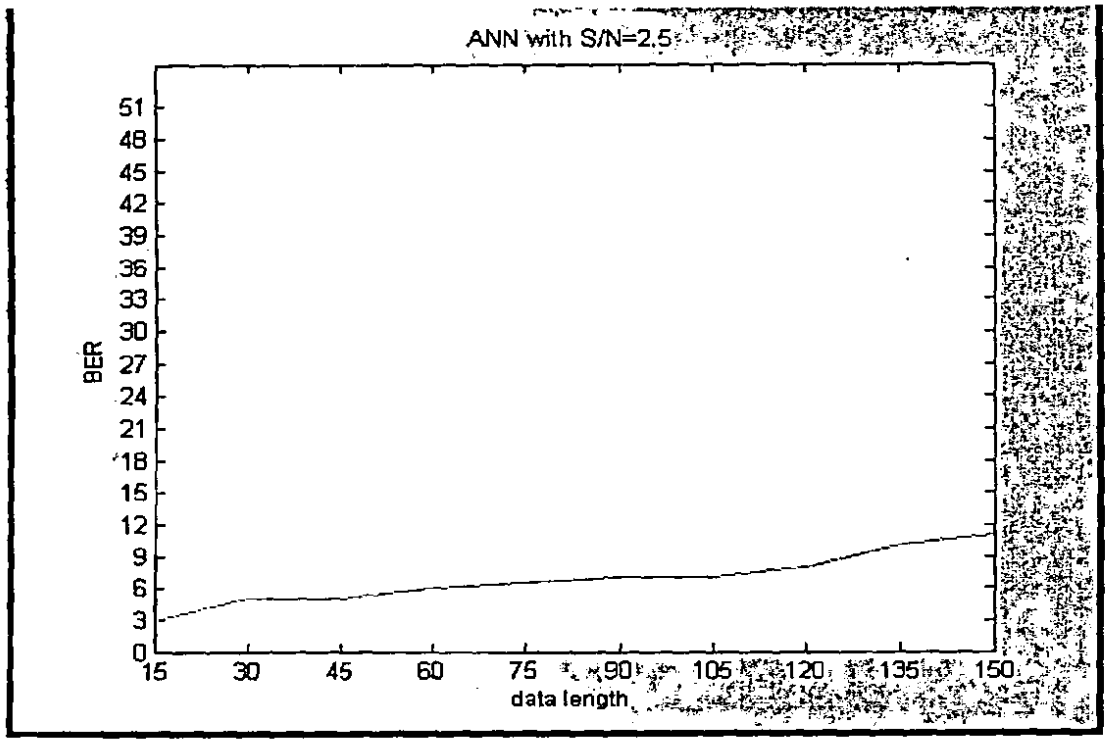
Graph 4.9 GA with E/N=2.0



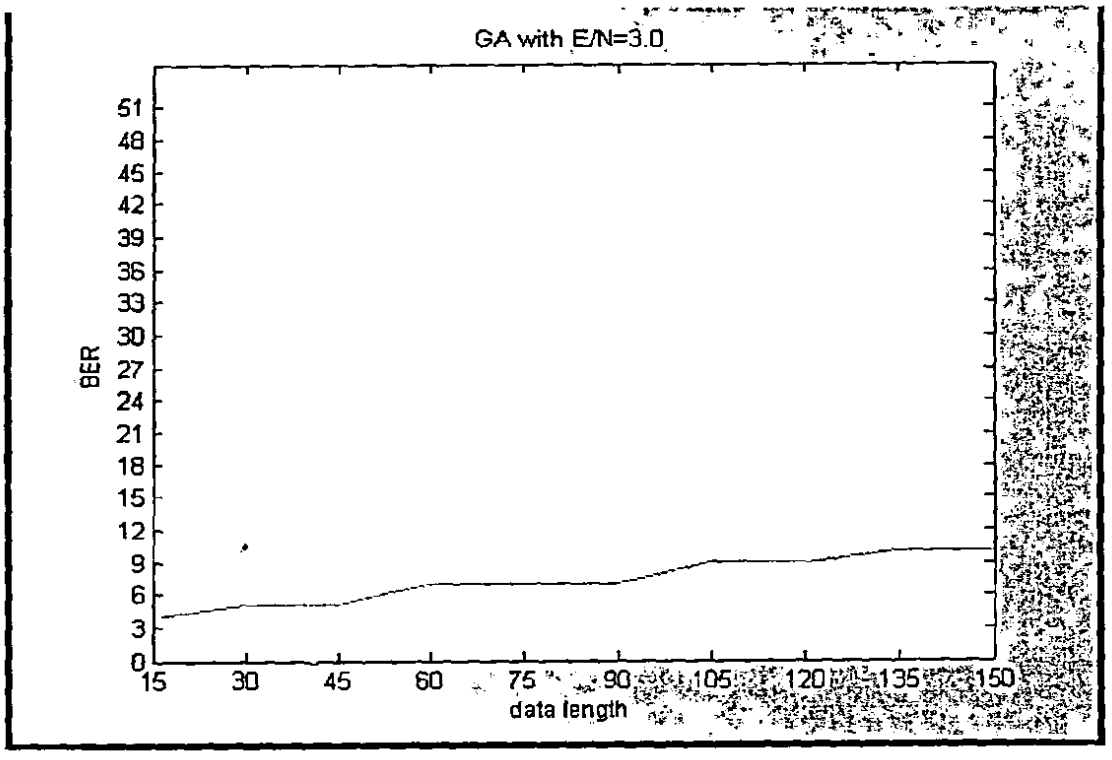
Graph 4.10 ANN with E/N=2.0



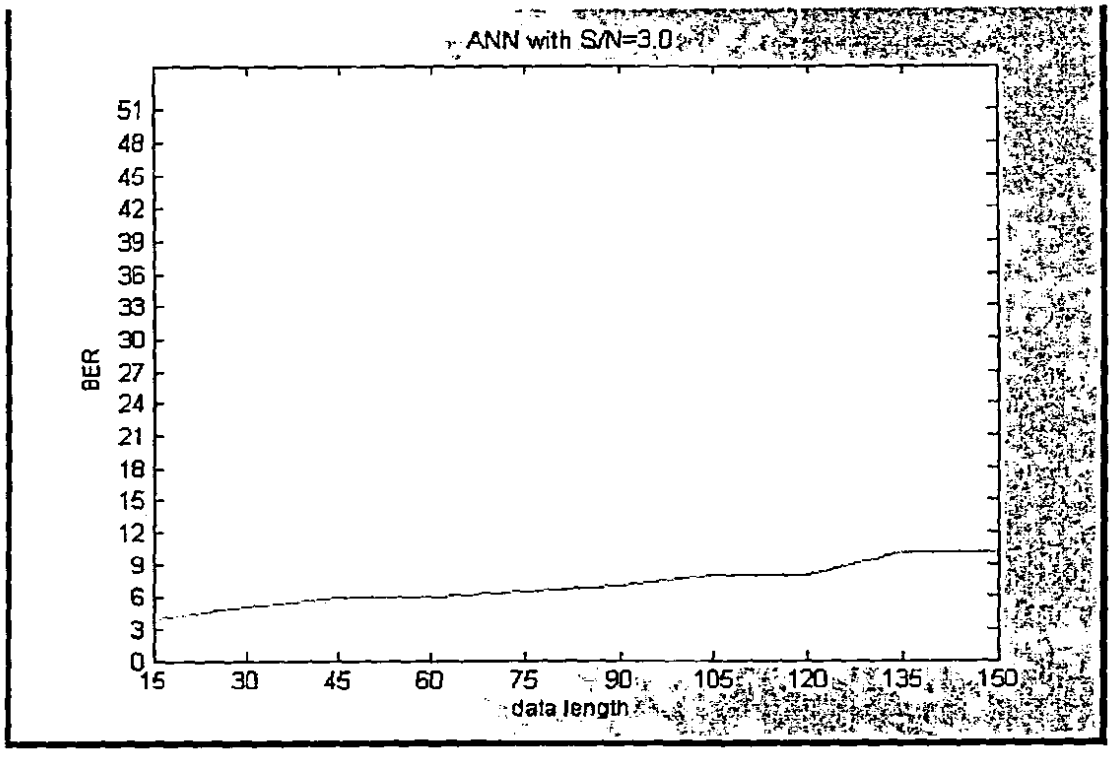
Graph 4.11 GA with E/N=2.5



Graph 4.12 ANN with E/N=2.5



Graph 4.13 GA with E/N=3.0



Graph 4.14 ANN with E/N=3.0

4.5 Application Implementation

This is the graphical interface of the implemented system. Now the functionalities and procedures will be defined.

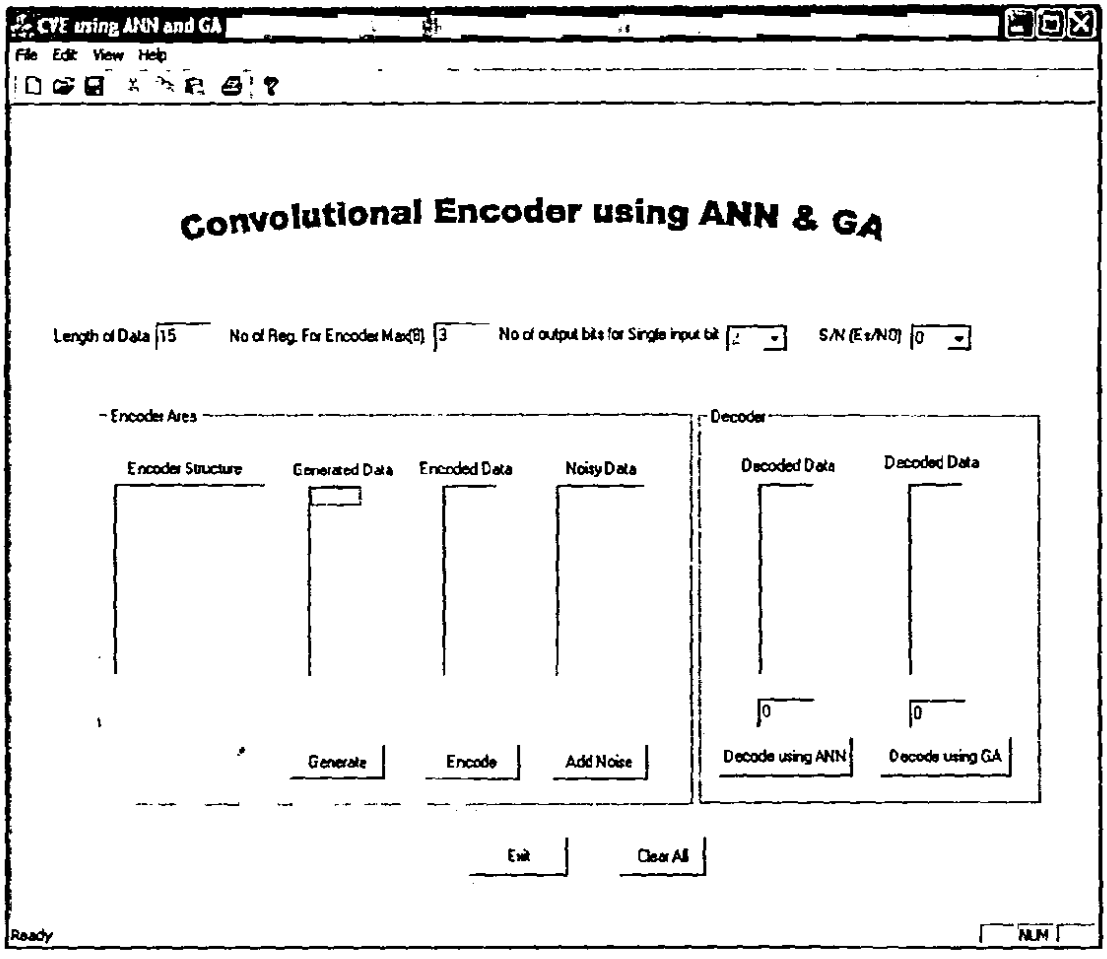


Figure 4.1 Application View

Firstly just look at the parameters which are taken as input from the user:

1. **Data Length** is the length of the data which is to be generated by a generator class which generates the data with the specified length in the form of '0','1'.
2. **No of Reg. used in Encoder** is the parameter which defines the registers used in the encoder.

3. **No of outputs for Single input** is the parameter which defines the 2 module adders in the system.
4. **Signal to Noise Ratio (Es/N0)** is the parameter which defines that at which ratio error should be induced at the encoded data.

4.6 Function of Application

Now we shall define the functionalities provided by the application:

1. **Generate** event initializes the class DataGenerator and calls its function generateData i.e., generateData(long data_len, int *out_array)is called.this function randomly generates the '0','1' data.
2. **Encode** event initializes the class convolutionEncoder and calss its function Encode i.e., Encode(int g[2][K], long input_len, int *in_array, int *out_array) is called. This function takes takes four parameters and returns the encoded data acoording to supplied encoder specification.
3. **Add Noise** is the event which Intializes the class Noise and its function addNosie i.e., addnoise(float es_ovr_n0, long channel_len, int *in_array, float *out_array) is called.in this function four aprametrs are passed and in this function given the desired Es/No standard deviation of the additive white gaussian noise (AWGN) is calculated. The standard deviation of the AWGN is then used to generate Gaussian random variables simulating the noise that is added to the signal.
4. **Decode with ANN** event initializes the class ANN and calls the function Decode i.e., Decode(int g[2][K], float es_ovr_n0, long channel_length,float *channel_output_vector, int *decoder_output_matrix) and in this class there is functionalities of ANN which are to train the network , weights are supplied with the file name data file and this function computes as a whole the optimum decoded data according to the provided weights after the training of the network. Here also the state diagram of the encoder is calculated and with the help of that state diagram the results are produced.

5. **Decode with GA** event initializes the class GA and calls the function Decode i.e., Decode(int g[2][K], float es_ovr_n0, long channel_length,float *channel_output_vector, int *decoder_output_matrix) and in this class there is functionalities of GA which is to gain the optimum solution after the given no of population the fitness value here we used the population size of 25. Here also the state diagram of the encoder is calculated and with the help of that state diagram the results are produced.

CHAPTER 5
TESTING

5. Testing

System testing is an essential step for the development of a reliable and error-free system. Testing is the process of executing a program with the explicit intension of finding errors i.e., making the program fail and test cases are devised with the purpose in mind. A test case is a set of data items that the system processes as normal input. A successful test is the one that finds an error.

5.1 Testing Strategies

The basic strategies that were used for testing were following

1. Specification Testing
2. Black Box Testing
3. White Box Testing
4. Regression Testing
5. Acceptance Testing
6. Assertion Testing
7. Unit Testing
8. System Testing

Each of the strategy are discussed as following

5.1.1 Specification Testing

Even if the code testing is performed exclusively, it doesn't ensure against program failure. Code testing doesn't answer whether the code meets the agreed specifications document. It doesn't also determine whether all aspects of the design are implemented.

Therefore, examining specifications stating what program should do and how it should behave under various conditions performs specification testing. Test cases are developed to test the range of values expected including both valid and invalid data. It helps in finding discrepancies between the system and its original objective. During this testing phase, all efforts were made to remove programming bugs and minor design faults.

5.1.2 Black Box Testing

In Black Box testing only the functionality was tested without any regard to the code written. If the functionality, which was expected from a component, is provided then black box testing is completed.

5.1.3 White Box Testing

In White Box testing internal code written in every component was tested and it was checked that the code written is efficient in utilizing various resources of the system like memory or the utilizing of input output

5.1.4 Regression Testing

In Regression testing the software was tested against the boundary conditions. Various input fields were tested against abnormal values and it was tested that the software does not behave abnormally at any time.

5.1.5 Acceptance Testing

In acceptance testing the software was checked for completeness that it is ready.

Normally the quality assurance department performs the acceptance testing that the software is ready and can be exported.

5.1.6 Assertion Testing

In assertion testing the software is tested against the possible assertions. Assertions are used to check the program and various locations that whether the state of the program at a particular point is the same as expected or not.

5.1.7 Unit Testing

In unit testing we checked that all the individual components were working properly. Before integration of the entire components unit testing is essential because it gives a confidence that all the components individually are working fine and ready to be integrated with the other ones.

5.1.8 System Testing

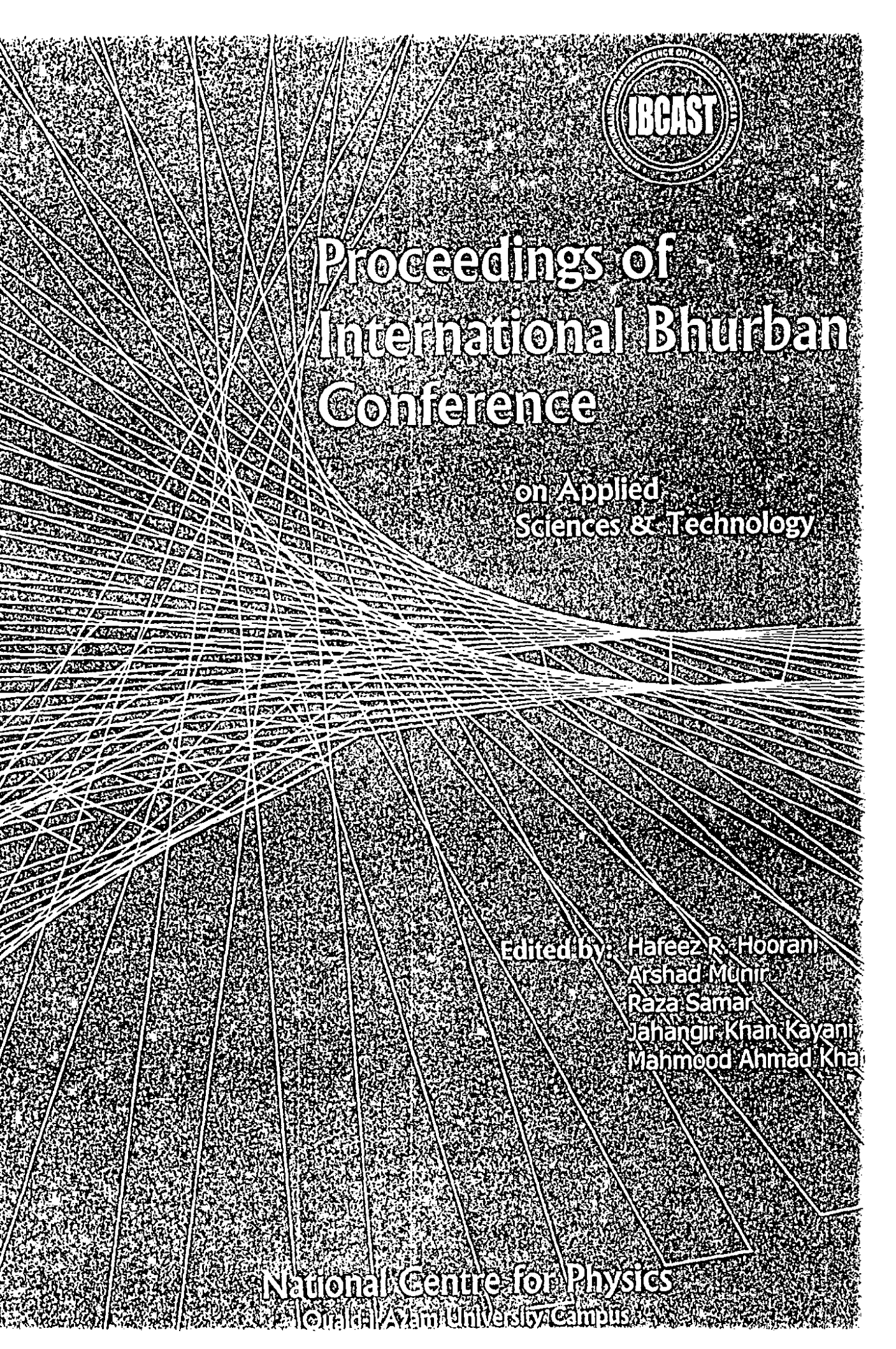
When all the units were working properly and unit testing was performed then comes the time for system testing where we checked all the integrated components as a whole and looked for possible discrepancies, which could have arisen after the integration.

REFERENCES AND BIBLIOGRAPHY

References and Bibliography

1. S. B. Wicker, Error Control Systems for Digital Communication and storage. Englewood Cliffs, NJ: Prentice Hall, 1995.
2. www.rennard.org/alife/english/gavintrgb.html
3. www.geneticprogramming.com
4. www.cs.gmu.edu/research/gag/
5. www.doc.ic.ac.uk/~nd/surprise_96/ journal/vol4/cs11/report.html
6. www.dacs.dtic.mil/techs/neural/neural.title.html
7. www.openclinical.org/neuralnetworks.html
8. www.cisco.com/warp/public/ 788/signalling/waveform_coding.pdf
9. cnx.rice.edu/content/m0097/latest/
10. www.math.uri.edu/~thoma/teaching/
11. www.cs.berkeley.edu/~luca/pacc/lecture14.pdf
12. www.scholar.lib.vt.edu/theses/available/etd-71897-15815/unrestricted/chap2.pdf
13. ww.ee.unb.ca/tervo/ee4253/convolution1.htm
14. www.s2.chalmers.se/graduate/ courses/errctrlcoding/convcode_decoding.pdf
15. mth391_fall2004/introduction-to-cyclic-codes.html
16. www.math.uic.edu/~fields/ DecodingGolayHTML/introduction.html

PUBLICATION


IBCAST
Proceedings of International Bhurban Conference
on Applied Sciences & Technology
Edited by: Hafeez R. Hoorani
Arshad Munir
Raza Samar
Jahangir Khan Kayani
Mahmood Ahmad Khan
National Centre for Physics
Quaid-i-Azam University Campus

3rd International Bhurban Conference on Applied Sciences and Technology, Bhurban, Pakistan. June 07-12, 2004

Channel Encoding of Wireless Medium Based on Convolutional codes using Neural Networks and Genetic Algorithm

Syed Muhammad Saqlain, Imran Khan
Dr.Muhammad Sher, Dr.Sikandar Hayat Khayal
Department of Computer Sciences, International Islamic University, Islamabad Pakistan.

Abstract

Channel coding refers to the class of signal transformations designed to improve communications performance by enabling the transmitted signals to better withstand the effects of various channel impairments, such as noise, interference, and fading. Convolutional coding is one of the schemes of channel coding and most of the research work in this contest is done in the area of decoders for Convolutional encoders. This paper presents the idea algorithms which are constructed for the purpose of decoding the Convolutional codes. The idea behind presenting these algorithms is not based on the old algorithms used for the decoding of Convolutional codes but by using the implementation of Artificial Intelligence in the form of ANN and GA. These algorithms try to minimize the errors while decoding the code and the new idea is to use the state diagram for decoding in new fashion i.e. state diagrams values are used as the patterns which are to be matched.

Keywords: Convolutional Codes, State Diagram, Artificial Intelligence, Artificial Neural Network (ANN), Genetic Algorithm (GA).

Introduction:

Digital communication has become necessity of this era and the most important issue of the digital communication is the improved communication performance by enabling the transmitted signals to better withstand the effects of various channel impairment, such as noise, interference, and fading. For this improvement purpose the phenomena of Channel Coding is used that can be portioned into two study area, waveform coding and structured sequence. Our paper lies in the later one which deals with transforming data sequences into better sequences having structured redundancy. Structured sequences are partitioned into three subcategories i.e. Block, Convolutional and Turbo Coding.

References and Bibliography

1. S. B. Wicker, Error Control Systems for Digital Communication and storage. Englewood Cliffs, NJ: Prentice Hall, 1995.
2. www.rennard.org/alife/english/gavintrgb.html
3. www.geneticprogramming.com
4. www.cs.gmu.edu/research/gag/
5. www.doc.ic.ac.uk/~nd/surprise_96/ journal/vol4/cs11/report.html
6. www.dacs.dtic.mil/techs/neural/neural.title.html
7. www.openclinical.org/neuralnetworks.html
8. www.cisco.com/warp/public/ 788/signalling/waveform_coding.pdf
9. cnx.rice.edu/content/m0097/latest/
10. www.math.uri.edu/~thoma/teaching/
11. www.cs.berkeley.edu/~luca/pacc/lecture14.pdf
12. www.scholar.lib.vt.edu/theses/available/etd-71897-15815/unrestricted/chap2.pdf
13. ww.ee.unb.ca/tervo/ee4253/convolution1.htm
14. www.s2.chalmers.se/graduate/ courses/errctrlcoding/convcode_decoding.pdf
15. mth391_fall2004/introduction-to-cyclic-codes.html
16. www.math.uic.edu/~fields/ DecodingGolayHTML/introduction.html

There are three ways in which we can look at the encoder graphically to gain better understanding of its operation. These are State diagram, Tree diagram and Trellis Diagram. We can see state diagram for an (2, 1, 4) Code along with the Encoder itself. Different Algorithms have been devised not only for decoding the Convolutional codes but also for error controlling by using these diagrams, e.g. Viterbi algorithm uses Trellis diagram for decoding purpose while sequential decoding and Feedback decoding schemes use Tree diagram for this. There is no well known Algorithm which is using the state diagram for decoding and these algorithms will use state diagram to Decode the data provided. Before we go into the algorithms firstly there is introduction of state diagram, Genetic Algorithm and Neural Network.

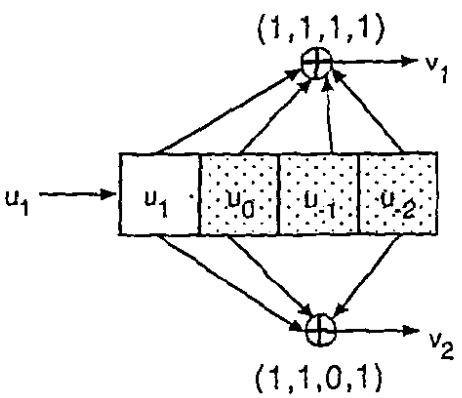


Figure 1- Encoder of (2,1,4) code

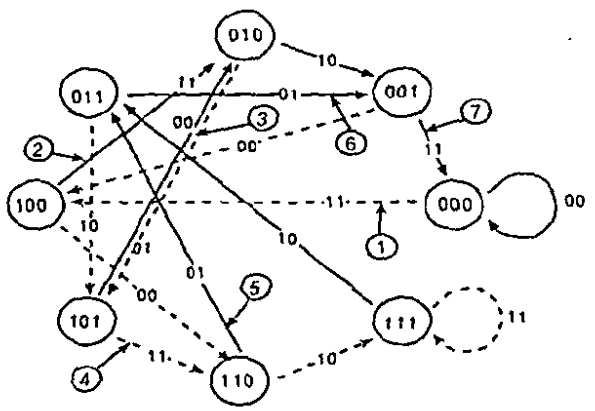


Figure 2- State diagram of (2,1,4) code

In State diagram as in figure2 each circle represents a state. At any one time, the encoder resides in one of these states. The lines to and from it show state transitions that are possible as bits arrive. Only two events can happen at each time, arrival of a 1 bit or arrival of a 0 bit. Each of these two events allows the encoder to jump into a different state. To use the state diagram for the purpose of decoding we have to compare it with the Lookup table of the Encoder. The Lookup table for encoder shown in figure1 can be drawn as i.e.,

Table 1 – Look-up Table for the encoder of code (2,1,4)

Input Bit	Input State			Output Bits		Output State		
I_1	s_1	s_2	s_3	O_1	O_2	s_1	s_2	s_3
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	0	0
0	0	0	1	1	1	0	0	0
1	0	0	1	0	0	1	0	0
0	0	1	0	1	0	0	0	1
1	0	1	0	0	1	1	0	1
0	0	1	1	0	1	0	0	1
1	0	1	1	1	0	1	0	1
0	1	0	0	1	1	0	1	0
1	1	0	0	0	0	1	1	0
0	1	0	1	0	0	0	1	0
1	1	0	1	1	1	1	1	0
0	1	1	0	0	1	0	1	1
1	1	1	0	1	0	1	1	1
0	1	1	1	1	0	0	1	1
1	1	1	1	0	1	1	1	1

Genetic

Algorithms (GAs) are adaptive and robust computational procedures modeled on the mechanics of natural genetic systems. GAs act as a biological metaphor and try to emulate some of the processes observed in natural evolution. They are viewed as randomized, yet structured, search and optimization techniques. They express their ability by efficiently exploiting the historical information to speculate on new offspring with expected improved performance. GAs are executed iteratively on a set of coded solutions, called a population, with three basic operators: selection/reproduction, crossover, and mutation. In order to find out the optimum solution of a problem, a GA starts from a set of assumed solutions (chromosomes) and evolves different but better set (of solutions) over a sequence of iterations. In each generation (iteration) the objective function (fitness measuring criterion) determines the suitability of each solution and, based on these values, some of them (which are called parent chromosomes) are used for reproduction. The number of copies reproduced by an individual parent is expected to be directly proportional to it's fitness value, thereby embodying the natural selection procedure, to some extent. The procedure thus selects better (highly fitted) chromosomes and worst one are eliminated. Hence, the performance of a GA depends on the fitness evaluation criterion, to a large extent. Genetic operators are applied on these (selected) parent chromosomes and new chromosomes (offspring) are generated. Conventional genetic algorithms (CGAs) consider only the fitness value of the chromosome under consideration for measuring its suitability for selection for the next generation, i.e., the fitness of a chromosome is function of functional value of the objective function. Fitness of a chromosome $x=g[f(x)]$, where $f(x)$ is the objective function and g is another function which by operating on $f(x)$ gives the fitness value. Hence, a CGA does not discriminate between two identical offspring, one produced from better (highly fit) parents and other from comparatively weaker (low fit) parents. In nature normally an offspring is more fit (suitable) if its ancestors (parents) are better, i.e., an offspring possesses some extra facility to exist in its environment if it belongs to a better family (ancestors are highly fitted). In other words, the fitness of an individual depends also on the fitness of its

ancestors in addition to its own fitness. This I perhaps the basic intuition to give more weightage to highly fitted chromosomes (due to better ancestors) to produce more for the next generation.

In general a typical GA consists of following components:

- A population of strings or coded possible solutions (biologically referred to as chromosomes)
- A mechanism to encode a possible solution (mostly as a binary string)
- Objective function and associated fitness evaluation techniques
- Selection/reproduction procedure
- Genetic operators
- Probabilities to perform genetic operations.

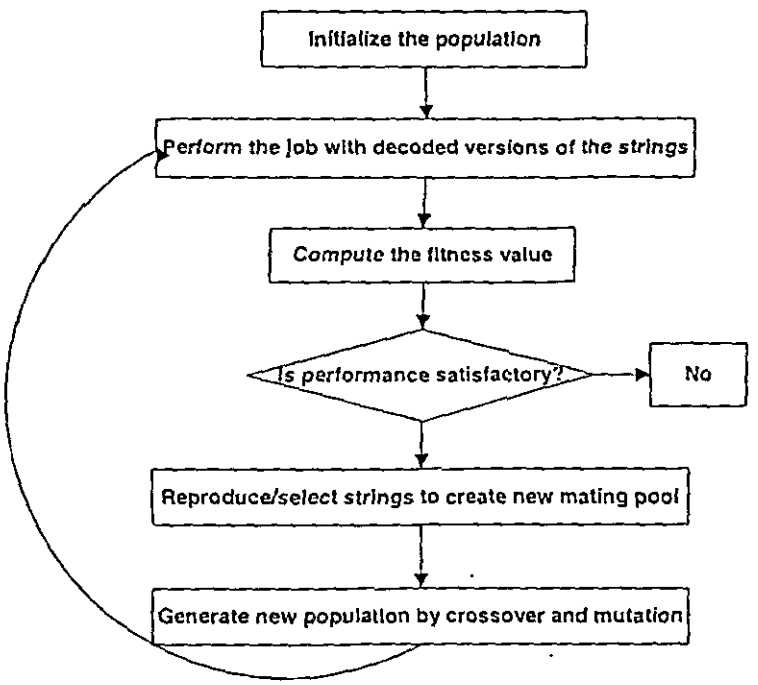


Figure 3- Basic steps of genetic Algorithm

Neural Network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and can carry out localized information processing operations) interconnected via unidirectional signal channels called connections. Each processing element has a single output connection that branches ("fans out") into as many collateral connections as desired; each carries the same signal — the processing element output signal. The processing element output signal can be of any mathematical type desired. The information processing that goes on within each processing element can be defined arbitrarily with the restriction that it must be completely local; that is it must depend only on the current values of the input signals arriving at the processing element via impinging connections and on values stored in the processing element's local memory.

artificial neural systems are physical cellular systems which can acquire, store, and utilize experiential knowledge. The following characteristics of neural networks have played an important role in a wide variety of applications:

1. **Adaptiveness** — Powerful learning algorithms and self-organizing rules allow it to self-adapt as per the requirements in a continually changing environment.
2. **Nonlinear processing** — Ability to perform tasks involving nonlinear relationships and noise-immunity make it a good candidate for classification and prediction.
3. **Parallel processing** — Architectures with a large number of processing units enhanced by extensive interconnectivity provide for concurrent processing as well as parallel distributed information storage.

Multilayer Perceptron (MLP) networks trained by back propagation [Rumelhart et al., 1986] are among the most popular and versatile forms of neural network classifiers. It has been shown that multilayer perceptron networks with a single hidden layer and a nonlinear activation function are universal classifiers [Funahashi, 1989; Cybenko, 1989, Hartman et al., 1990; Hornik et al., 1989]. That is, such networks can approximate decision boundaries of arbitrary complexity. Thus given an input vector consisting of a set of features representing an input pattern, this network is known to have the inherent discriminative power necessary to serve as a strong recognition engine. This does not mean that we can achieve zero error. Classes are often overlapping so that classification by minimizing error due to misclassification cannot produce zero error.

The input vector representing the pattern to be recognized is incident on the input layer and distributed to subsequent hidden layers and finally to the output layer via weighted connections. Each neuron in the network operates by taking the sum of its weighted inputs and passing the result through a nonlinear activation function. This is shown mathematically as:

$$out_i = f(net_i) = f\left(\sum_j W_{ij} out_j + \theta_i\right) \qquad (1)$$

Here out_i is the output of the i^{th} neuron in the layer under consideration; out_j is the output of the j^{th} neuron in the preceding layer. There are several conventionally used choices for the nonlinear activation function, f. One of those most frequently employed is the sigmoid:

$$f(net_i) = \frac{1}{1 + e^{\frac{-net_i}{Q_0}}} \qquad (2)$$

The term Q_0 in equation 2 is referred to as the temperature of the neuron. The higher the temperature the more gently the sigmoid changes. At very low temperatures it approaches a step function.

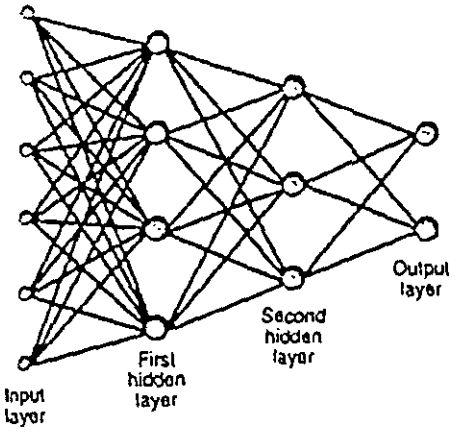


Figure 4- Feed-forward multilayer perceptron architecture

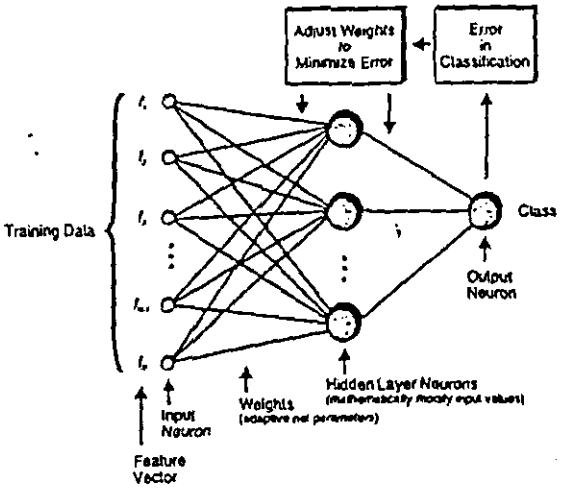


Figure 5- Feed-forward Multilayer perceptron with Backpropogation

Experimental Setup:

Actually we have two algorithms developed using Genetic algorithm and Neural Network. Both these have less and more same steps except these two techniques. Here are the steps of these algorithms:

a. Decoding using Artificial Neural Network

1. We will assume the case that before the first values enters into the encoder all the registers are with the value '0',e.g., we are at state 00 for k=3 encoder

2. From code rate (k/n) provided we determined the no of module 2-adders, this will determine the no of output values for one value entered into the encoder.

3. Pad the k-n zeros to the output sequence obtained at i^{th} time so that length(u_i) =k say it is InRegVal.

4. Provide this InRegVal to Genetic algorithm system which determines the optimum register contents at the encoder for which the output was generated say it OutRegVal.

5. Take first k-n bit values of OutRegVal.
6. Compare Values obtained from step 5 with the states shown in state diagram, If there exist in state diagram's transition states from previous step find whether there would be '1' or '0'.If there is no match it means there have occurred an error.

b. Decoding using Artificial Neural Network

1. We will assume the case that before the first values enters into the encoder all the registers are with the value '0',e.g., we are at state 00 for k=3 encoder

2. From code rate(k/n) provided we determined the no of module 2-adders, this will determine the no of output values for one value entered into the encoder.

3. Pad the k-n zeros to the output sequence obtained at i^{th} time so that length(u_i)=k say it is InRegVal.

4. Provide this InRegVal to Feed Forward with Back propagation network which has k neurons at it's input layer and k neurons at it's output layer. The optimum register contents at the encoder for which the output was generated say it OutRegVal.

5. Take first k-n bit values of OutRegVal.
6. Compare Values obtained from step 5 with the states shown in state diagram, If there exist in state diagram's transition states from previous step find whether there would be '1' or '0'.If there is no match it means there have occurred an error.

Error Correction is done with the idea of interest that when we have an error we randomly take bit '0' or '1' but when we decode the next sequence of data and get the decoded bit, firstly we see the value at second position which we got from the 5^{th} point and also see that from the previous state this state comes or not if not we move back and change the previous bit so the error is corrected.

Coding gain is defined as the reduction in the required $E_b N_0$ to achieve a specified error probability of the coded system over an uncoded system with the same modulation and channel characteristics. Table 2 shows the Basic Coding Gain (db);

UnCoded	Code Rate	1/3		1/2			2/3		3/4	
$E_b N_0$ (db)	P_n K	7	8	5	6	7	6	8	6	9
6.8	10^{-3}	4.2	4.4	3.3	3.5	3.8	2.9	3.1	2.6	2.6
	10^{-5}	5.7	5.9	4.3	4.6	5.1	4.2	4.6	3.6	4.2
	10^{-7}	6.2	6.5	4.9	5.3	5.8	4.7	5.2	3.9	5.7

Table 2 - Basic Coding Gain (db)

Comparison of two or more things is quite difficult; it needs complete statistics of all of the contesters. One problem with Viterbi decoding was that it as the no of registers grow mean as k grows it comes slow and its performance degrades badly when it increases to

double digits. It requires the computation of 2^k k metrics at each node of the trellis and the storage of $2^k(k-1)$ surviving sequences, each of which may be 5^kK bits long. The computation burden and the storage required to implement the Viterbi algorithm make it impractical for Convolutional codes with large constraint length. Whereas there is no such problem with these algorithms. Some people might give an argue in against our these algorithm about their slow nature but it is very reliable and not very slow system if we select proper weights for neural network and mutation rate, fitness values and population size properly.

Conclusion:

This paper presents the two algorithms which are based on the application of Artificial Intelligence. Artificial Intelligence is becoming hot topic these days and the purpose of these algorithms was not to follow the traditional techniques but we have adopted the basics of Convolutional encoders, the state diagram and used this diagram with Neural network and Genetic algorithm for decoding purpose. The important thing at the time of implementation is choose properly some parameters like weights (means you have to train your system so that it chooses the proper weight), mutation value, fitness function etc., so that you can get better results from these algorithms. One thing which we like to mention is that currently we are working on the process of error correction in these algorithms and after completion it would be better than any algorithm.

References:

1. S. Lin and D. J. Costello, Error Control Coding. Englewood Cliffs, NJ: Prentice Hall, 1982.
2. M. Michelson and A. H. Levesque, Error Control Techniques for Digital Communication. New York: John Wiley& Sons, 1985.
3. W. W. Peterson and E. J. Weldon, Jr., Error Correcting Codes, 2nd ed. Cambridge, MA: The MIT Press, 1972.
4. V. Pless, Introduction to the Theory of Error-Correcting Codes, 3rd ed. New York: John Wiley & Sons, 1998.
5. C. Schlegel and L. Perez, Trellis Coding. Piscataway, NJ: IEEE Press, 1997
6. S. B. Wicker, Error Control Systems for Digital Communication and Storage. Englewood Cliffs, NJ: Prentice Hall,1995.
7. Analog to Digital Communications, Error Control Coding, Chapter 9 Channel Encoding, Page 393